

Weight Decomposition Matters: DoRA vs. LoRA for Small GPT-2 Task Adaptation

Stanford CS224N {Default} Project

Svea Drekhshagen

Department of Computational and Mathematical Engineering

Stanford University

svea@stanford.edu

TA mentor: Tristan Thrush

External collaborators : No

External mentor: No

Sharing project: No

Abstract

This project presents a three-part framework built on a custom GPT-2 model: sentiment classification, paraphrase detection, and Shakespearean sonnet generation. Motivated by the high cost of full fine-tuning, the core focus is on comparing full and partial fine-tuning with parameter-efficient methods (LoRA and DoRA) for sonnet generation under limited compute. I implement a modular PEFT setup that injects adapters into attention query and value projections and supports controlled hyperparameter experiments. Performance is evaluated using chrF on a held-out development set. The results suggest that parameter-efficient methods can match or surpass full fine-tuning on small datasets while requiring significantly fewer trainable parameters.

1 Introduction

Large pretrained language models such as GPT-2 have become a standard approach for a wide range of natural language processing tasks. These models are typically adapted to downstream tasks through full fine-tuning, where all model parameters are updated on task-specific data. Although this approach often achieves strong performance, it can and is mostly computationally expensive. These limitations become especially problematic in low-resource settings or when people are experimenting with many configurations.

To address these challenges, parameter-efficient fine-tuning methods have recently gained attention. Instead of updating all model parameters, PEFT methods introduce small sets of additional trainable parameters while keeping the pretrained weights frozen. Methods such as LoRA [1] model weight updates as low-rank matrices added to existing weight projections, significantly reducing the number of trainable parameters. More recently, DoRA [2] extends this idea by decomposing pretrained weights into magnitude and directional components and applying low-rank adaptation only to the directional component, allowing more and finer control of the update.

Despite the growing popularity of PEFT methods, it remains unclear how these approaches compare to traditional full fine-tuning in smaller experimental settings on different tasks with limited data. In particular, it is not obvious whether the additional structure imposed by PEFT methods helps or hurts adaptation quality when training data is scarce and applied to a new task.

In this project, I study this question in the context of GPT-2 adaptation for Shakespearean sonnet generation, on top of implementing a GPT-2 baseline, sentiment classification and paraphrase detection. Using the CS224N default project framework, I implement a modular GPT-2 pipeline supporting full fine-tuning, partial fine-tuning, LoRA, and DoRA. I conduct controlled hyperparameter

sweeps over rank, scaling factor, and learning rate, and evaluate generated sonnets using the chrF metric against held-out reference continuations.

My experiments show that parameter-efficient methods perform competitively with full fine-tuning in this setting. These results suggest that PEFT methods can act as an effective regularization mechanism when adapting large language models on small datasets, allowing strong performance with significantly fewer trainable parameters.

2 Related Work

The two most related works are these papers:[1] and[2].

In their experiments, Hu et al. [1] show that LoRA can compete and sometimes surpass full fine-tuning and several PEFT baselines across a range of NLU and NLG tasks, including GLUE, GPT-2 generation benchmarks, and GPT-3 adaptation settings. They further find that LoRA remains highly effective even at very small ranks, suggesting that downstream adaptation can often be captured by low-rank updates. In addition, their results highlight that LoRA offers strong scalability and parameter efficiency, achieving competitive performance with substantially lower training and storage costs than full fine-tuning.

DoRA extends LoRA by decomposing pretrained weights into magnitude and direction, then applying low-rank adaptation only to the directional component while learning the magnitude separately. In their experiments, Liu et al.[2] show that DoRA can outperform LoRA across a wide range of tasks, including commonsense reasoning, multimodal understanding, and visual instruction tuning. They further also report that DoRA remains also effective even when using fewer trainable parameters than LoRA. In addition, their studies indicate that DoRA can be more robust across different rank settings and can better match the learning behavior of full fine-tuning while retaining the efficiency advantages of PEFT.

3 Approach

I will tackle four different tasks as follows: I implement a GPT-2 baseline with the standard transformer components, establishing a reference for comparing full, partial, and parameter-efficient fine-tuning methods [3]. Then I implement sentiment classification, where I add a dropout and linear classification head on top of GPT-2 sentence representations [3]. Furthermore, I implement paraphrase detection, where I use a cloze-style formulation with a binary decision head on the final token [3]. Finally, I integrate sonnet generation, where I first fine-tune GPT-2 end to end and implement token-level prediction.[3] I then fix the output decoding to return only the generated continuation, rather than a hardcoded, character-trimmed string. For numerical stability, I constrain the top-p, added safeguards for temperature scaling, and filtered NaN and infinite logits. Generation quality is further improved by adding repetition penalties, blocking repeated n-grams, applying top-k filtering, and using a more robust top-p filtering with renormalization and fallbacks. Default settings (temperature = 0.8, top-p = 0.9) reduce repetition and incomplete outputs. The goal is to establish a strong baseline with good sonnets for fair comparison against PEFT methods. Moreover, I introduce parameter efficient adaptation in three stages: partial fine-tuning (updating only top GPT-2 blocks plus normalization and embeddings), LoRA (adding low-rank trainable updates to attention projections while freezing base weights, [1]), and DoRA (decomposing weights into magnitude and direction and applying low-rank adaptation to the directional component, [2]), enabling a good comparison of quality efficiency trade-offs in sonnet generation. For LoRA and DoRA, I apply parameter-efficient adapters only to the GPT-2 self-attention query and value projection layers in each transformer block. In full fine-tuning, all GPT-2 parameters are trainable. In partial fine-tuning, I train only the top two transformer blocks, the final layer normalization, and the token embedding layer, while keeping the remaining parameters frozen.

To ensure fair and reproducible comparisons, I built an experiment management pipeline that runs full fine-tuning, partial fine-tuning, LoRA, and DoRA across different ranks, alphas, and learning rates, logging all runs and aggregating results into summary tables and plots.

The primary focus is on sonnet generation; therefore, the baseline is full fine-tuning of GPT-2 for sonnet generation with the aforementioned modifications: token-level prediction, continuation-

only decoding, numerical safeguards (top-p, temperature, NaN/Inf), and stronger decoding controls (repetition penalty, no-repeat n-grams, top-k, robust top-p) using temperature = 0.8 and top-p = 0.9. Training uses a learning rate of 1e-5 for 8 epochs with batch size 8 and a chrF score of 38.729.

4 Experiments

4.1 Data

For the different task, I use the standard CS224N Default Project datasets. [3]

- Stanford Sentiment Treebank (SST) with a size of 11,855 sentences for sentiment classification. Beyond the full sentences, the dataset also includes 215,154 unique phrases from parse trees, each annotated by three human judges.
- CFIMDB with 2,434 highly polar movie reviews also for the sentiment classification. Each movie review has a binary label of negative or positive.
- Quora Question Pairs (QQP) with 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. This is being used for paraphrase detection.
- Sonnet Dataset for Sonnet Generation, which is a collection of 154 sonnets written by Shakespeare. Each sonnet consists of 14 lines of text following the rhyme scheme ABAB CDCD EFEF GG.

4.2 Evaluation method

I used the following evaluation methods, based on its standard metric in the CS224N default project setup.[3]

- For SST and CFIMDB sentiment classification, I report accuracy on the dev set.
- For QQP paraphrase detection, I report accuracy (and track macro-F1 during development for class-balance robustness).
- For sonnet generation, I use chrF between generated sonnets and held-out reference sonnets to measure character-level similarity to Shakespearean style.

Important to know all model selection and hyperparameter tuning are done on train/dev data only.

The novelty of this approach is using DoRA and LoRA comparison to FT and partial FT on this specific dataset, with hyperparameter tuning. More studies with DoRA and LoRA were evaluated in [2] and [1].

4.3 Experimental details

For SST and CFIMDB sentiment classification and QQP paraphrase detection I use the standard experiment approach which was given by the default project.[3]

For sonnet generation I use a structured experiment pipeline. It automatically runs many configurations with different training modes and hyperparameters. Each of these runs uses the same data paths and decoding settings and is saved in its own, for the task created, directory. For every run, the pipeline stores full training logs and generated sonnets. Furthermore, it also extracts key results such as final and best training loss, chrF score, number of trainable parameters, runtime, and number of checkpoints, not all since this was too much space, and writes them to a summary file. After all of the runs are completed, the pipeline combines the results into two summary tables (CSV and JSON). Then it also creates comparison plots, such as chrF versus rank, alpha, or learning rate, and shows performance differences between training modes (see appendix). The best configuration is selected based on the chrF score.

First, I created the baseline as described above in the approach. I then run a controlled study of parameter-efficient fine-tuning methods, comparing full fine-tuning, partial fine-tuning, LoRA, and DoRA using the same training setup. The hyperparameter, I use for the first comparison are; ranks $r \in 8, 16$, scaling factors $\alpha \in 8, 16$, and learning rates of 5×10^{-6} or 1×10^{-5} for LoRA and DoRA. Partial fine-tuning used a learning rate of 1e-5, so the same as the baseline. This resulted in 18 total runs (1 full, 1 partial, 8 LoRA, and 8 DoRA) to ensure fair comparison across methods. Furthermore,

to keep generation quality consistent, all models use the same decoding settings: temperature = 0.8 and top-p = 0.9, along with repetition penalties, n-gram blocking, top-k filtering, and a top-p renormalization step to reduce repetitive and unstable outputs, which was a problem I had in the beginning.

After this initial controlled comparison, I conducted an additional larger $4 \times 4 \times 4$ sweep. The search space was expanded to ranks $r \in \{4, 12, 24, 32\}$, scaling factors $\alpha \in \{4, 12, 24, 48\}$ for LoRA and $\alpha \in \{2, 6, 12, 24\}$ for DoRA as well as learning rates $\{2 \times 10^{-6}, 3 \times 10^{-6}, 2 \times 10^{-5}, 3 \times 10^{-5}\}$. I used different ranks, based on the first sweep. This configuration produced a total of 136 runs: 64 LoRA runs, 64 DoRA runs, 4 full fine-tuning runs, and 4 partial fine-tuning runs.

Based on the results of this larger sweep, the strongest configurations were selected for a final run.

Before this run I tested some different temperature and top-p parameter but stayed with temperature = 0.8 and top-p = 0.9. Moreover, I implemented several practical improvements. I added early stopping support with configurable metric, patience, and minimum delta, and used this together with best-checkpoint tracking so that the model now saves a dedicated best checkpoint whenever performance improves. I also changed checkpoint loading so generation first tries to load this best checkpoint and only falls back to the most recent epoch checkpoint if needed.

In generation, I changed the no-repeat n-gram logic bug by ensuring that continuation tokens are always added to the n-gram map for repeated prefixes, which makes repetition blocking behave correctly. I then refined decoding to improve stability and output quality by increasing the generation length budget (max length 240), lowering the repetition penalty, switching to 4-gram blocking, and preventing EOS from being selected too early by enforcing a minimum number of newly generated tokens (min 120). I also normalized output formatting by removing drifting line indentation in generated text.

For experiment management and reproducibility, I introduced a configurable checkpoint directory so runs can be isolated and do not overwrite one another, and I added a predict only mode so I can generate predictions from existing checkpoints without retraining. Finally, I made output writing more robust by ensuring the prediction output directory is created automatically before saving files.

In this final four-run comparison, I evaluated one selected configuration per method based on the earlier sweeps.

4.4 Results

My results for the sentiment classification are as follows:

- Last Linear Layer for SST: Dev Accuracy: 0.457
- Full Model for SST: Dev Accuracy: 0.508
- Last Linear Layer for CFIMDB: Dev Accuracy: 0.865
- Full Model for CFIMDB: Dev Accuracy: 0.988

My result for the paraphrase detection on the test set is 0.872.

For sonnet generation my result on the test set is 41.715 on GPT-2 medium.

For the comparison of the method in sonnet generation I got the following results: In the final four-run comparison, I decided to run based on the results in the analysis section, I evaluated one selected configuration per method based on the earlier sweeps: DoRA with $r = 8$, $\alpha = 16$, and $\text{lr} = 1 \times 10^{-5}$; LoRA with $r = 24$, $\alpha = 12$, and $\text{lr} = 2 \times 10^{-5}$; full fine-tuning (FT) with $\text{lr} = 3 \times 10^{-5}$; and partial fine-tuning with $\text{lr} = 2 \times 10^{-6}$. The resulting chrF scores were 40.729 for DoRA, 39.386 for LoRA, 40.540 for FT, and 40.040 for partial fine-tuning. This final comparison confirms that the selected DoRA setup remains the strongest among the PEFT variants in this setting, while FT is competitive but still below the best peak scores observed in the larger sweeps. I do think DoRa performs so well, due to acting as a form of regularization. Results are a bit lower than the baseline submitted, due to being trained only on GPT-2 and not GPT-2 medium.

But, if I do a qualitative comparison, full fine-tuning sounds enormously better than all the others: it is less repetitive, smoother, and doesn't include weird artifacts in the middle.

5 Analysis

My preliminary findings are summarized in Tables 1 and 2. Somewhat surprisingly, the DoRA and LoRA models outperform sometimes full fine-tuning. A possible explanation is that these parameter-efficient methods act as a form of regularization on the relatively small dataset, allowing the model to adapt to the task while preserving most of the pretrained knowledge.

In the scatter plot for the scaling factor α , moderate values generally yielded the strongest performance across configurations (see Figure 3 and 4).

For LoRA, the best-performing runs were typically observed at $\alpha = 12$ and $\alpha = 24$. Smaller values such as $\alpha = 4$ produced competitive but slightly more variable results, while very large values such as $\alpha = 48$ occasionally led to unstable training with several lower-scoring runs.

For DoRA, strong results were obtained across a wider range of α values, highest at 16, although the highest-scoring runs were most frequently observed around $\alpha = 6$ and $\alpha = 12$. Similar to the learning-rate results, larger α values tended to increase variability without consistently improving peak performance.

Overall, the results suggest that moderate scaling factors provide the best balance between stable optimization and strong peak performance.

For the learning rate, the strongest single run was obtained with a learning rate of 1×10^{-5} , achieving the highest chrF score observed in the experiments (see Figure 7 and 8).

However, learning rates around 5×10^{-6} produced more consistently strong results across different configurations. In this range, most runs achieved chrF scores between 39 and 41, indicating stable optimization behavior.

In contrast, the higher learning rate of 1×10^{-5} showed greater variability. While it enabled the best-performing run, it also resulted in several lower-scoring runs, suggesting that training becomes less stable at this level.

Overall, the results indicate that moderate learning rates around 5×10^{-6} provide the most reliable performance, while 1×10^{-5} can yield the highest peak performance at the cost of increased variability.

Regarding the ranks, very small ranks, particularly $r = 4$, consistently produced weaker results. Many runs at this rank achieved chrF scores below 38, indicating that the adapter capacity was insufficient to effectively adapt the model (see Figure 6 and 5).

Increasing the rank to $r = 8$ and $r = 12$ led to a substantial improvement in the performance. In this range, most configurations achieved chrF scores around or above 39, and several of the highest-scoring runs were observed at these ranks.

Beyond $r = 12$, performance gains largely plateaued. Higher ranks such as $r = 24$ and $r = 32$ produced results comparable to those obtained with $r = 12$, but did not consistently yield further improvements. This suggests that increasing the rank beyond this point provides limited additional benefit.

Overall, the results indicate that moderate ranks offer the best trade-off between adaptation capacity and efficiency.

6 Conclusion

In this project, I worked across all required tasks: sentiment analysis, paraphrase detection, and sonnet generation, using a shared GPT-2-based setup with AdamW optimization and task-specific heads/adaptation strategies. For generation in particular, I implemented and compared full fine-tuning, partial fine-tuning, LoRA, and DoRA for GPT-2-based sonnet continuation, and evaluated them using chrF on held-out sonnet completions. My main finding is that PEFT methods, especially DoRA, provided the strongest and most consistent results under our setup. In a controlled $2 \times 2 \times 2$ study and a larger $4 \times 4 \times 4$ sweep, DoRA repeatedly reached the top-performing region, with our best overall dev result at chrF = 41.338. A final head-to-head run with one selected configuration per

method confirmed the same trend, with DoRA outperforming the other PEFT variant and remaining competitive against full fine-tuning.

Beyond raw scores, an important achievement was improving generation stability through targeted debugging and analysis. I identified and fixed decoding issues (including no-repeat n-gram bookkeeping and checkpoint selection behavior), added best-checkpoint loading, and improved run management for reproducible experiments. These changes reduced common failure modes such as repetitive loops and unstable endings, and made comparisons across methods more reliable.

The primary limitation is that output quality still varies noticeably across reruns, and final rerun scores can be lower than peak sweep values, indicating sensitivity to training and decoding randomness. In addition, while chrF is useful for overlap-based evaluation, it does not fully capture poetic quality, rhythm, or semantic coherence. Future work could include stronger decoding strategies (e.g., beam reranking or contrastive decoding), multi-seed model selection, and richer evaluation that combines automatic metrics with structured human judgment of sonnet form and style.

References

- [1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.
- [2] 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, 2024.
- [3] CS224N Teaching Team Winter 25/26. Cs 224n: Default final project: Build gpt-2. 2026.

A Appendix (optional)

Figure 1: chrF ranking across runs. The table reports the run name, adaptation method (mode), hyperparameters (rank r , scaling factor α , learning rate), and the resulting chrF score.

rank	run_name	mode	r	α	lr	chrF
1	dora_r8_a16.0_lr1e-05	dora	8	16.0	1e-05	41.338
2	lora_r16_a8.0_lr5e-06	lora	16	8.0	5e-06	41.156
3	lora_r8_a8.0_lr1e-05	lora	8	8.0	1e-05	40.769
4	dora_r16_a8.0_lr1e-05	dora	16	8.0	1e-05	40.733
5	lora_r16_a16.0_lr5e-06	lora	16	16.0	5e-06	40.628
6	lora_r8_a16.0_lr1e-05	lora	8	16.0	1e-05	40.448
7	lora_r16_a8.0_lr1e-05	lora	16	8.0	1e-05	40.379
8	dora_r16_a16.0_lr5e-06	dora	16	16.0	5e-06	40.279
9	dora_r16_a16.0_lr1e-05	dora	16	16.0	1e-05	40.216
10	dora_r8_a8.0_lr5e-06	dora	8	8.0	5e-06	40.031
11	dora_r8_a8.0_lr1e-05	dora	8	8.0	1e-05	40.010
12	lora_r8_a8.0_lr5e-06	lora	8	8.0	5e-06	39.889
13	lora_r8_a16.0_lr5e-06	lora	8	16.0	5e-06	39.722
14	lora_r16_a16.0_lr1e-05	lora	16	16.0	1e-05	39.454
15	dora_r8_a16.0_lr5e-06	dora	8	16.0	5e-06	39.047
16	ft_r0_a0.0_lr1e-05	ft	0	0.0	1e-05	38.729
17	dora_r16_a8.0_lr5e-06	dora	16	8.0	5e-06	38.274
18	partial_r0_a0.0_lr1e-05	partial	0	0.0	1e-05	33.530

chrF Banking Across Rate

row	id	name	type	z	alpha	beta	gamma
1	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
2	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
3	WGLC-02-01-00000	WGA	20	12.0	0.00	45.000	0.000
4	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
5	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
6	WGLC-02-01-00000	WGA	4	20.0	0.00	45.000	0.000
7	WGLC-02-01-00000	WGA	4	6.8	0.00	45.000	0.000
8	WGLC-02-01-00000	WGA	4	6.8	0.00	45.000	0.000
9	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
10	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
11	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
12	WGLC-02-01-00000	WGA	20	6.8	0.00	45.000	0.000
13	WGLC-02-01-00000	WGA	20	10.0	0.00	45.000	0.000
14	WGLC-02-01-00000	WGA	4	6.8	0.00	45.000	0.000
15	WGLC-02-01-00000	WGA	4	6.8	0.00	45.000	0.000
16	WGLC-02-01-00000	WGA	12	20.0	0.00	45.000	0.000
17	WGLC-02-01-00000	WGA	4	20.0	0.00	45.000	0.000
18	WGLC-02-01-00000	WGA	4	20.0	0.00	45.000	0.000
19	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
20	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
21	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
22	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
23	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
24	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
25	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
26	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
27	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
28	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
29	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
30	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
31	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
32	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
33	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
34	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
35	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
36	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
37	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
38	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
39	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
40	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
41	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
42	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
43	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
44	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
45	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
46	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
47	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
48	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
49	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
50	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
51	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
52	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
53	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
54	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
55	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
56	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
57	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
58	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
59	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
60	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
61	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
62	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
63	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
64	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
65	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
66	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
67	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
68	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
69	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
70	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
71	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
72	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
73	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
74	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
75	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
76	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
77	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
78	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
79	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
80	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
81	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
82	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
83	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
84	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
85	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
86	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
87	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
88	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
89	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
90	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
91	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
92	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
93	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
94	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
95	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
96	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
97	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
98	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
99	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000
100	WGLC-02-01-00000	WGA	12	6.8	0.00	45.000	0.000

Figure 2: chrF ranking across runs. The table reports the run name, adaptation method (mode), hyperparameters (rank r , scaling factor α , learning rate), and the resulting chrF score.

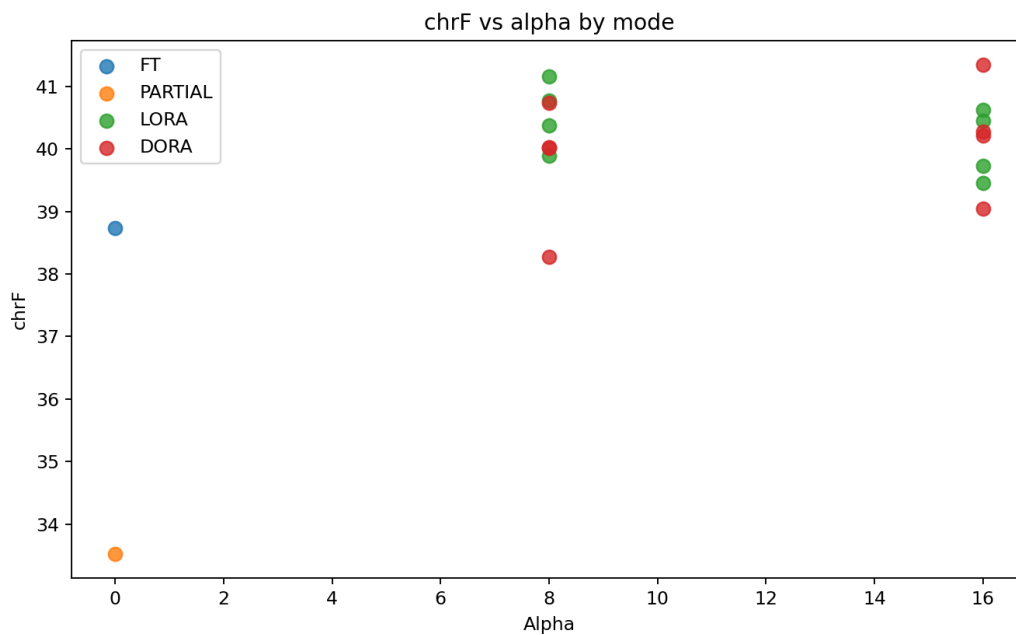


Figure 3: chrF score as a function of the scaling factor α , grouped by adaptation method (FT, Partial FT, LoRA, DoRA).

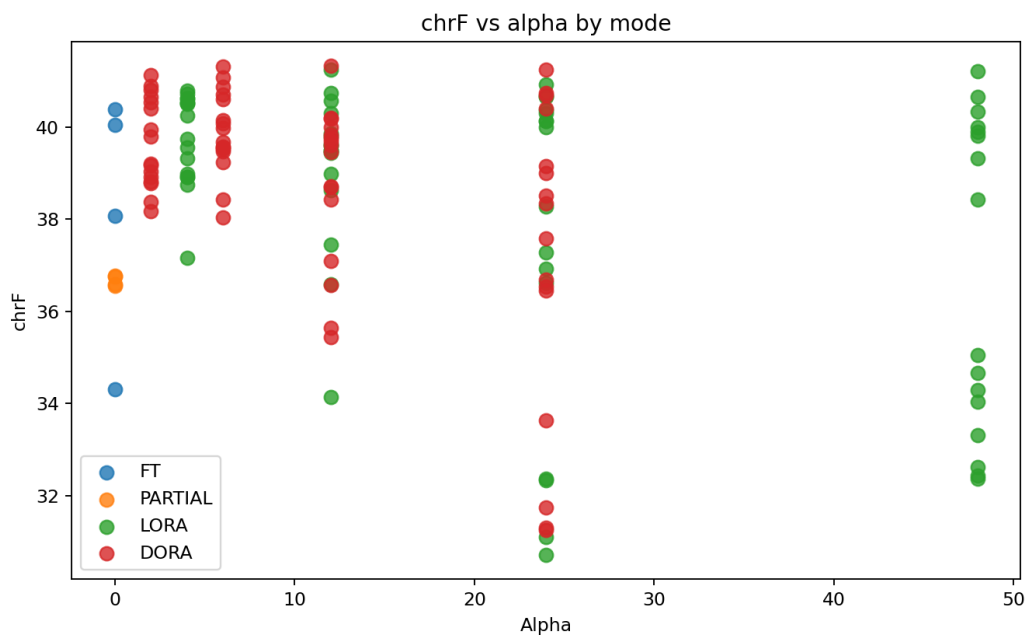


Figure 4: chrF score as a function of the scaling factor α , grouped by adaptation method (FT, Partial FT, LoRA, DoRA).

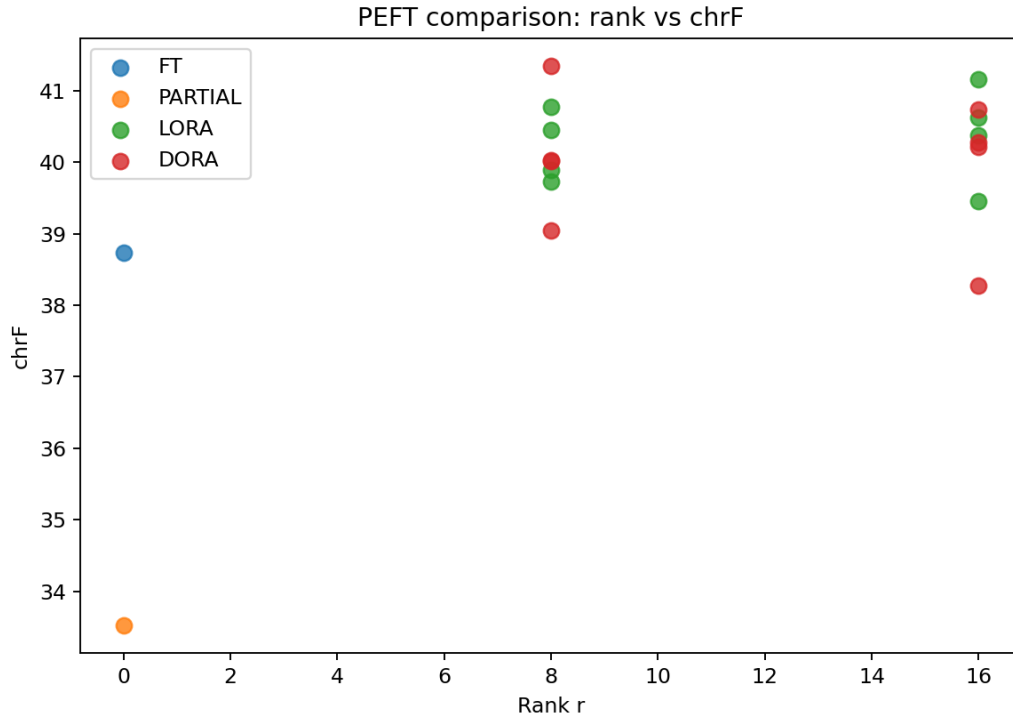


Figure 5: PEFT comparison: chrF score as a function of the adapter rank r , grouped by adaptation method (FT, Partial FT, LoRA, DoRA).

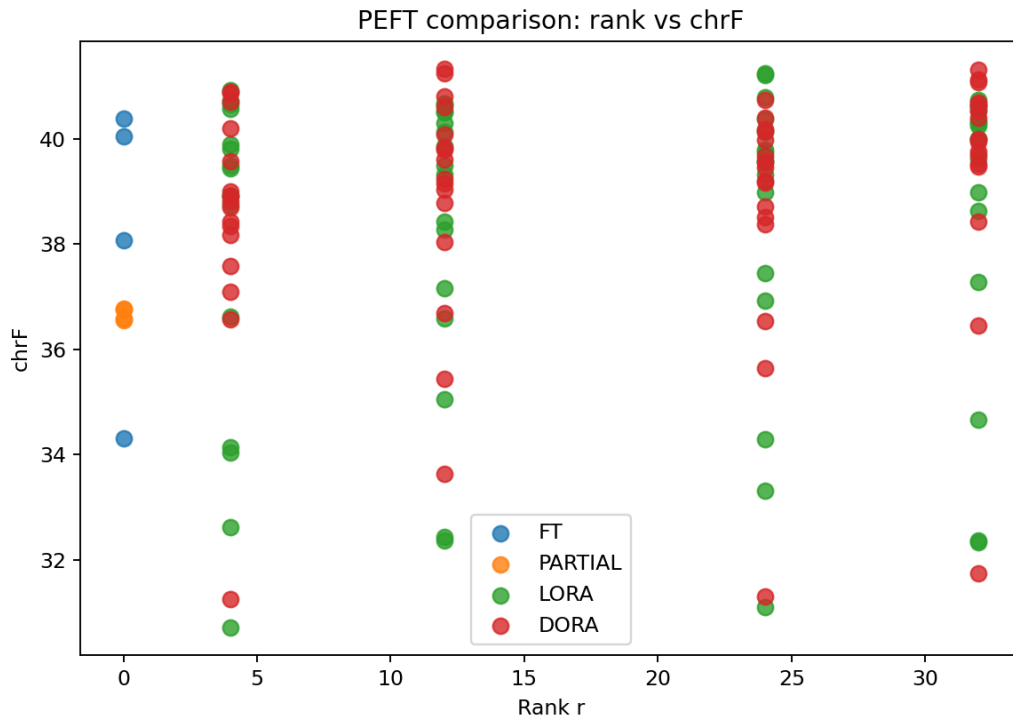


Figure 6: PEFT comparison: chrF score as a function of the adapter rank r , grouped by adaptation method (FT, Partial FT, LoRA, DoRA).

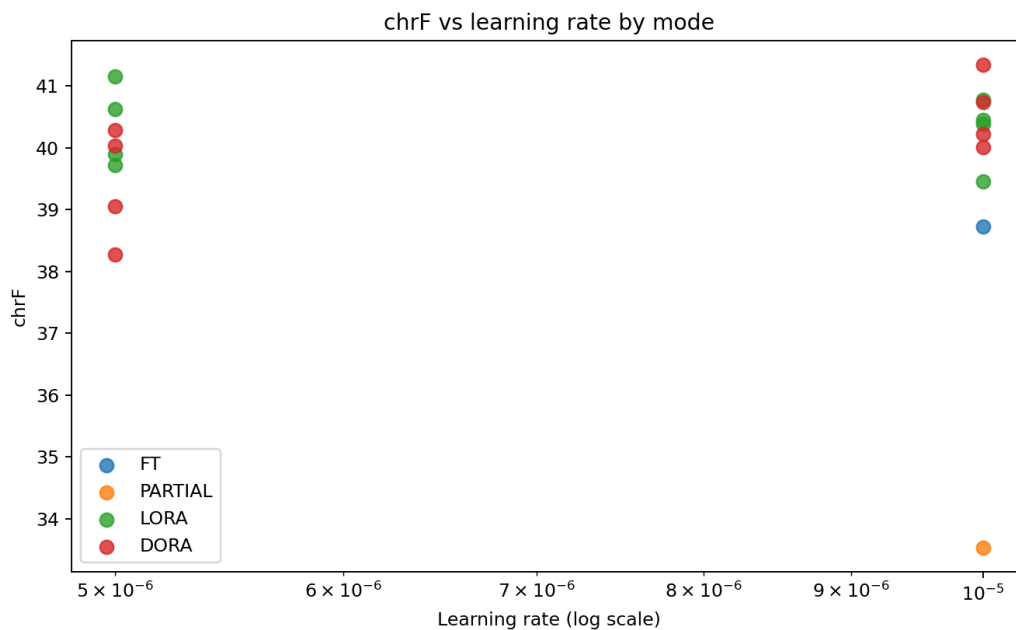


Figure 7: chrF score as a function of the learning rate (log scale), grouped by adaptation method (FT, Partial FT, LoRA, DoRA).

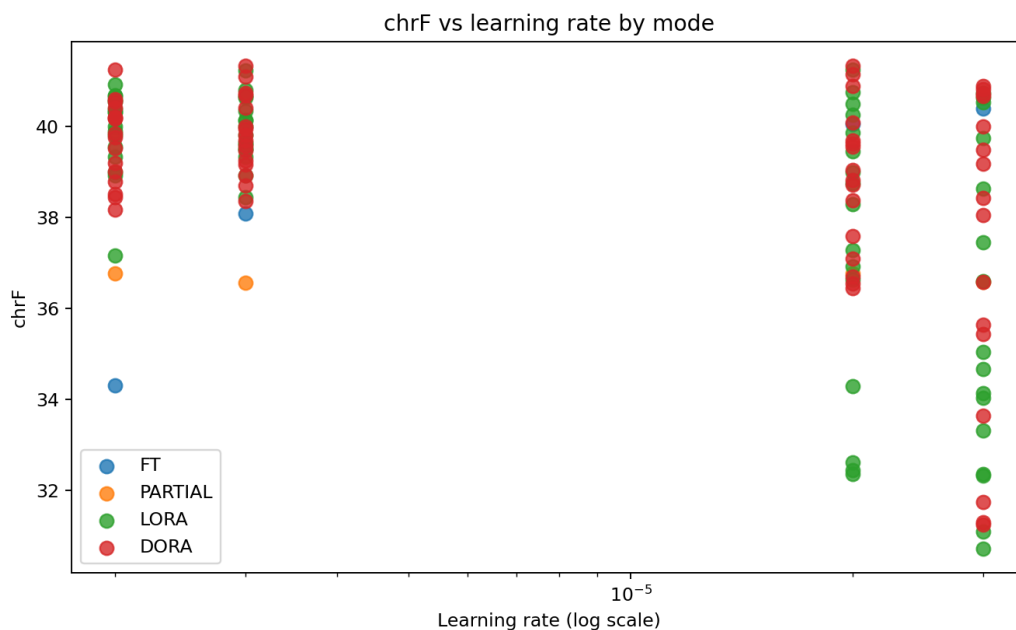


Figure 8: chrF score as a function of the learning rate (log scale), grouped by adaptation method (FT, Partial FT, LoRA, DoRA).