

LoRA vs. LoReFT: Parameter-Efficient Fine-Tuning of GPT-2

Stanford CS224N Default Project

Andrew Chung

Department of Computer Science
Stanford University
awchung@stanford.edu

Ryan Suh

Department of Computer Science
Stanford University
ryansuh@stanford.edu

Allen Yuan

Department of Computer Science
Stanford University
ayuan06@stanford.edu

Abstract

We implement GPT-2 from scratch and explore parameter-efficient fine-tuning (PEFT) methods across three downstream tasks: sentiment analysis, paraphrase detection, and sonnet generation. We systematically compare Low-Rank Adaptation (LoRA) and Low-Rank Linear Subspace Representation Fine-Tuning (LoReFT) for paraphrase detection on the Quora Question Pairs dataset using GPT-2 medium. For our initial LoReFT configuration, we intervene only on the last token with an orthogonal constraint at rank 4 and achieve 0.741 dev accuracy. We show that this score, which is much lower than the full fine-tuning baseline of 0.882, is a result of specific design choices rather than a fundamental limitation of representation interventions. Through controlled experiments varying token scope, rank, and the orthogonality constraint, we find that removing the orthogonal constraint (free LoReFT, 0.886 dev) matters far more than increasing rank (0.756 at rank 16), and that LoRA alone (0.893 dev) is a strong PEFT baseline that surpasses full fine-tuning. Our best results combine LoRA with all-token orthogonal LoReFT, reaching 0.894 dev accuracy and 0.893 test accuracy with 1.77M trainable parameters (0.51% of the model). For sonnet generation, we achieve a CHRF score of 40.7 using LoRA fine-tuning with nucleus sampling.

1 Key Information to include

- Mentor: Simon Kim
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Late Days (if you are using any):

Team member	Late days remaining
<Andrew Chung>	<0>
<Ryan Suh>	<0>
<Allen Yuan>	<0>
Total pooled	<0>
Team size	<3>
Late days applied for the report	<0>

2 Introduction

Large pretrained language models such as GPT-2 [1] have demonstrated strong performance across an array of NLP tasks when fine-tuned on task-specific data. However, updating model parameters can become extremely computationally expensive and memory intensive, especially for larger models with hundreds of millions or billions of parameters. To address this issue, parameter-efficient fine tuning (PEFT) methods keep the pretrained weights frozen and use only a small subset of additional or modified parameters to train [2, 3].

Among PEFT methods, Low-Rank Adaptation (LoRA) [2] and Low-Rank Linear Subspace Representation Fine-Tuning (LoReFT) [3] operate at different stages of computation. LoRA introduces trainable low-rank updates into the weight matrices, modifying how the model computes attention projections across all token positions. In contrast, LoReFT acts directly on hidden states by learning targeted perturbations within low-rank subspaces of the activation space. Although both methods are designed to adapt pretrained models with limited parameter overhead, they differ significantly in where and how they modify the model. This in turn makes it important to compare their behavior on specific tasks using empirical evidence.

In our explorations, we first build GPT-2 from scratch and evaluate it on sentiment analysis using the Stanford Sentiment Treebank (SST) [4] and CFIMDB datasets. We then study PEFT methods on paraphrase detection on Quora Question Pairs, as well as sonnet generation using a corpus of Shakespearean sonnets. For paraphrase detection, we perform a systematic comparison of LoRA and LoReFT. Here, we vary the token scope of the intervention (last-token versus all-token), the orthogonality constraint on the rotation matrix, and the intervention rank. Our results show that the orthogonality constraint is the main factor limiting LoReFT performance on this task and that LoRA provides an especially strong PEFT baseline that outperforms full fine-tuning. Additionally, we observed that the best overall performance comes from combining LoRA with all-token LoReFT. For sonnet generation, LoRA fine-tuning with nucleus sampling and tweaked temperature achieves a CHRF score of 40.7. Here, the temperature is critical for capturing the lexical diversity of Shakespearean verses.

3 Related Work

Pretrained Language Models. The GPT family of models [1] demonstrated that large-scale autoregressive pretraining on diverse text gives us representations that can be transferred well to downstream tasks. We use GPT-2 medium, a 345M parameter decoder-only transformer [5] with 24 layers, 16 attention heads, and a hidden dimension of 1024.

PEFT. Numerous parameter-efficient fine-tuning approaches have been developed to address the high cost of fine-tuning models. One approach, known as adapter methods [6], inserts small bottleneck modules between transformer layers. Another method, prefix tuning [7], attaches trainable continuous prefix vectors to the key and value sequences at each layer. Aghajanyan et al. [8] showed that pretrained models have a low intrinsic dimensionality, providing theoretical support for low-rank adaptation.

LoRA. In LoRA [2], weight updates are expressed as low-rank matrix products in the form $\Delta W = \frac{\alpha}{r} BA$, where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times d}$. Applied to the query and value projection matrices, these updates alter the attention computation across all token positions. As a result, LoRA provides a flexible form of adaptation while keeping the number of trainable parameters small.

ReFT and LoReFT. Instead of modifying the weight matrices, representation fine-tuning [3] learns interventions on hidden representations. One specific version, LoReFT, constrains these interventions to low-rank orthogonal subspaces. Building on causal abstraction and interchange intervention approaches [9], this method offers a principled method for targeted edits to a model’s internal representations. While Wu et al. report competitive results with very few parameters on several benchmarks, the extent to which LoReFT’s design choices (including token scope, the orthogonality constraint, and intervention rank) depend on task-specific settings remains underexplored.

Our work contributes an empirical comparison of LoRA and LoReFT on paraphrase detection, systematically isolating the effect of each LoReFT design choice and exploring hybrid combinations of both methods.

4 Approach

4.1 GPT-2 Implementation and Sentiment Analysis

We implement GPT-2 from scratch, including masked multi-head self-attention with causal masking, pre-norm transformer layers with GELU activations, and AdamW optimization [10]. Our implementation is initialized with pretrained weights from HuggingFace and is sanity-checked against the reference model. For sentiment analysis, we add a linear classification head on top of the last token’s hidden state from our custom GPT-2 (small, 124M parameters). We then evaluate under two configurations: training only the classification head (last-linear-layer) and fine-tuning all parameters (full-model).

4.2 Paraphrase Detection

For all paraphrase detection experiments, we use GPT-2 medium (345M parameters) from HuggingFace with all pretrained weights frozen. Input pairs are formatted as a cloze-style prompt:

Is "{s1}" a paraphrase of "{s2}"? Answer "yes" or "no":

The hidden state of the last non-padding token is extracted from the final layer; it is passed through a dropout layer where $p = 0.1$ and a linear classification head to produce a 2-class prediction.

4.2.1 LoReFT

Low-Rank Linear Subspace ReFT [3] learns an intervention on hidden representations at each transformer layer. Given a hidden state $\mathbf{h} \in \mathbb{R}^d$ at layer ℓ , the intervention computes:

$$\mathbf{h}' = \mathbf{h} + R^\top (W_s \mathbf{h} - R \mathbf{h}) \quad (1)$$

where $R \in \mathbb{R}^{r \times d}$ is a rotation matrix constrained to have orthonormal rows (enforced by `torch.nn.utils.parametrizations.orthogonal`) and $W_s \in \mathbb{R}^{r \times d}$ is a learned projection with a bias term. The intervention projects the hidden state into a rank- r subspace via R , computes a learned target representation via W_s , and applies the difference back to the full space via $R^\top$.

In our initial setup, we apply rank $r = 4$ interventions across all 24 layers, but only to the last token’s hidden state at each layer. This was motivated by the idea that the last token position in a causal model carries the most complete representation of the input.

4.2.2 LoRA

Low-Rank Adaptation [2] modifies the frozen pretrained weight matrices by adding trainable low-rank decompositions. For each attention layer, we add adapters to the query (W_Q) and value (W_V) projections:

$$W' = W + \frac{\alpha}{r} B A \quad (2)$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times d}$ are low-rank matrices. B is initialized to zeros and A is initialized with Kaiming uniform initialization, so the adapter’s initial output is zero. We apply rank-16 adapters with $\alpha = 16$ (so the scaling factor is $\alpha/r = 1.0$) across all 24 layers. These adapters are implemented through forward hooks on the combined QKV projection and affect only the query and value portions.

Unlike LoReFT, LoRA’s perturbation to the weight matrices propagates through all token positions at every layer through the attention mechanism, enabling the model to reshape cross-sentence representations.

4.2.3 LoReFT Design Variations

Due to our initial LoReFT result (0.741 dev) significantly underperforming the full fine-tuning baseline (0.882 dev), we conduct a systematic investigation of its design choices, isolating three main factors:

Token scope. We compare two options, intervening on only the last token or on all tokens at each layer. All-token intervention allows the model to reshape representations across the entire input sequence, not just at the classification position.

Rank. We compare the original setting, rank $r = 4$, with $r = 16$, which quadruples the intervention capacity and the number of trainable parameters.

Orthogonality constraint. In orthogonal LoReFT, R is required to have orthonormal rows, restricting the intervention within a fixed r -dimensional orthogonal subspace of $\mathbb{R}^d$. In the free unconstrained variant, R is an unconstrained learned matrix, allowing the back-projected delta $R^\top(\cdot)$ to span directions outside the subspace defined by R ’s rows.

4.2.4 Hybrid LoRA + LoReFT

We also investigate a hybrid approach that combines the two methods by applying LoRA adapters to the query and value projections and LoReFT interventions to the layer outputs across all 24 layers. This lets us test whether representation-level interventions provide complementary information on top of LoRA’s weight-level adaptation. In the hybrid model, we evaluate LoReFT under both last-token and all-token settings.

4.3 Sonnet Generation

For sonnet generation, the model is given the first three lines of a Shakespearean sonnet and its task is to generate the remainder. We fine-tune GPT-2 medium using LoRA (rank 16, $\alpha = 16$, Q+V projections, 24 layers) as an autoregressive language model. All pretrained weights are frozen and training updates only the LoRA adapter parameters. The model is trained with next-token prediction loss (cross-entropy) on 143 training sonnets, and we tie the output projection weights to the input word embedding matrix by using its transpose.

At generation time, we use nucleus sampling (top- p) with $p = 0.9$ and temperature $\tau = 1.2$. The elevated temperature encourages lexical diversity that better matches the writing style and variability of Shakespearean verse. We found that lower temperatures, such as $\tau = 0.2$, produced overly conservative outputs that failed to capture the Shakespeare’s poetic style.

5 Experiments

5.1 Data

Sentiment Analysis. We use the Stanford Sentiment Treebank (SST; [4]) with 5 sentiment classes (8,544 train / 1,101 dev / 2,210 test examples) and CFIMDB with binary sentiment labels (1,701 train / 245 dev).

Paraphrase Detection. We use the Quora Question Pairs dataset, which contains pairs of questions labeled as semantically equivalent or not. We use 141,506 training pairs, 20,215 dev pairs, and 40,431 test pairs.

Sonnet Generation. We use 143 Shakespeare sonnets for training and 12 held-out sonnets for test evaluation, with an additional dev split for model selection. Each prompt provides only the first 3 lines of a sonnet, and the model must generate the remaining lines.

5.2 Evaluation Method

For sentiment analysis and paraphrase detection, we use classification accuracy on the dev set, selecting the best checkpoint across epochs. For sonnet generation, we evaluate using the CHRF score [11], a character-level n -gram F-score that measures surface-level similarity to the ground-truth sonnets. The best checkpoint is selected based on dev CHRF, with early stopping at a patience of 10 epochs.

5.3 Experimental Details

For all paraphrase detection experiments, we use the following configuration: GPT-2 medium (345M parameters, frozen), AdamW optimizer [10] with weight decay 0.01, cosine annealing learning rate schedule with 10% linear warmup, batch size 16, and 5 training epochs. For pure LoReFT experiments, we use a learning rate of 6×10^{-4} , while for LoRA and hybrid LoRA+LoReFT

experiments, we use 2×10^{-4} . These learning rates were tuned independently for each method family in our milestone and were held fixed across all subsequent variants.

For sonnet generation, we use learning rate 1×10^{-4} , batch size 8, 20 epochs with early stopping at a patience of 10 epochs, and the same optimizer and schedule configuration.

We implemented the LoRA adapters, LoReFT interventions, hook-based injection into the transformer layers, and the learning rate schedule. We used the provided CS224N starter code for GPT-2 model loading, tokenization, dataset classes, and training scaffold. All experiments were run on NVIDIA GPUs (B200 for paraphrase detection, A10G for sonnet generation) using Modal cloud infrastructure.

5.4 Results

Paraphrase Detection. Table 1 reports the results of all paraphrase detection experiments in the order they were conducted. Each row varies one design choice relative to prior experiments.

Table 1: Paraphrase detection results on Quora Question Pairs. All methods use GPT-2 medium. PEFT methods keep all pretrained weights frozen. “Orth.” = orthogonal constraint on R ; “Free” = unconstrained R . Trainable parameter counts are approximate.

Method	Configuration	Params	Dev	Test
Full fine-tuning	All parameters	345M	0.882	0.856
LoReFT	Last-token, orth., $r = 4$	$\sim 199K$	0.741	–
LoRA	Q+V, $r = 16$	$\sim 1.57M$	0.893	0.891
LoReFT	All-token, orth., $r = 4$	$\sim 199K$	0.748	–
LoReFT	All-token, orth., $r = 16$	$\sim 789K$	0.756	–
LoReFT	All-token, free, $r = 4$	$\sim 199K$	0.886	–
LoRA + LoReFT	Q+V $r = 16$; last-tok orth. $r = 4$	$\sim 1.77M$	0.744	–
LoRA + LoReFT	Q+V $r = 16$; all-tok orth. $r = 4$	$\sim 1.77M$	0.894	0.893

The full fine-tuning baseline achieves 0.882 dev / 0.856 test accuracy. Our initial LoReFT configuration (last-token, orthogonal, $r = 4$) significantly underperforms at 0.741 dev. LoRA with rank 16 on Q and V projections achieves 0.893 dev / 0.891 test, surpassing full fine-tuning with only 1.57M trainable parameters (0.43% of the model).

Among the LoReFT variants, switching from last-token to all-token scope provides only a marginal gain (0.741 $\rightarrow$ 0.748), and increasing rank from 4 to 16 helps slightly further (0.748 $\rightarrow$ 0.756). In contrast, removing the orthogonal constraint while keeping rank 4 produces a dramatic improvement (0.748 $\rightarrow$ 0.886), nearly matching the full fine-tuning baseline.

For the hybrid methods, adding last-token LoReFT to LoRA worsens performance (0.893 $\rightarrow$ 0.744), while adding all-token LoReFT improves it slightly (0.893 $\rightarrow$ 0.894 dev, 0.893 test). The best system, LoRA combined with all-token orthogonal LoReFT, achieves the highest dev and test accuracy with 1.77M trainable parameters (0.51% of GPT-2 medium).

Sentiment Analysis. Using our GPT-2 implemented from scratch (small, 124M parameters), the last-linear-layer approach achieves 0.452 dev / 0.481 test on SST-5 and 0.849 dev on CFIMDB, while full-model fine-tuning achieves 0.502 dev / 0.528 test on SST-5 and 0.971 dev on CFIMDB. These results confirm that our GPT-2 implementation is functional and that full fine-tuning provides meaningful gains over frozen representations.

Sonnet Generation. LoRA fine-tuning with nucleus sampling ($\tau = 1.2$, top- $p = 0.9$) achieves a CHRF score of 40.74 on dev and 40.64 on test. These scores reflect moderate surface-level similarity to Shakespeare’s language, which is expected given the small training set of 143 sonnets and the inherent difficulty of generating stylized sonnets.

6 Analysis

The orthogonality constraint is the primary bottleneck for LoReFT. The most striking finding in our experiments is the gap between orthogonal and free LoReFT. With the same rank $r = 4$, token

scope (all-token), and training hyperparameters, removing the orthogonal constraint improves dev accuracy from 0.748 to 0.886. Under the orthogonal constraint, the intervention $R^\top(W_s\mathbf{h} - R\mathbf{h})$ can only modify the hidden state within the r -dimensional subspace spanned by R 's rows. With $r = 4$ and $d = 1024$, the perturbation is limited to a 4-dimensional subspace within the full 1024-dimensional hidden space. When the orthogonality constraint is removed, the back-projection $R^\top$ can distribute changes across all d dimensions even through an r -dimensional bottleneck, giving the model significantly more expressive freedom.

Increasing rank has limited effect under the orthogonal constraint. Quadrupling the rank from 4 to 16, which increases the parameter size from $\sim 199\text{K}$ to $\sim 789\text{K}$, only slightly improves all-token orthogonal LoReFT (0.748 $\rightarrow$ 0.756). This suggests that the bottleneck is not the subspace dimensionality, but instead the rigid nature of the orthogonal parametrization. Even when the subspace dimension is increased to 16, the intervention is still restricted to a fixed collection of orthonormal directions, which limits its capacity to capture the complex task-specific changes needed for paraphrase detection.

Token scope matters, but only in combination with other factors. Switching from last-token to all-token intervention improves orthogonal LoReFT only marginally (0.741 $\rightarrow$ 0.748), suggesting that expanding token scope alone is insufficient under the orthogonal constraint. However, token scope becomes critical in the hybrid setting: adding last-token LoReFT to LoRA actually hurts performance (0.893 $\rightarrow$ 0.744), whereas all-token LoReFT provides a tiny improvement (0.893 $\rightarrow$ 0.894). This indicates that last-token interventions can actively interfere with LoRA's learned representations at the classification position, while all-token interventions can sometimes complement LoRA by providing distributed refinements.

LoRA is a strong PEFT backbone for pairwise classification. LoRA alone surpasses full fine-tuning (0.893 vs. 0.882 dev) with less than 0.5% of the parameters, consistent with findings that pretrained models admit effective low-rank adaptations [8]. LoRA's strength on paraphrase detection likely stems from its ability to modify the attention mechanism across all token positions simultaneously, enabling richer cross-sentence comparisons. In contrast, LoReFT's localized representation edits must capture the same pairwise relationships through a narrower channel.

The best system combines both approaches. The hybrid combining LoRA and all-token LoReFT achieves the best performance overall (0.894 dev / 0.893 test), suggesting that representation-level interventions can contribute helpful information beyond what weight-level adaptation captures. At the same time, the small improvement over LoRA alone (0.1 point) indicates that LoRA already captures most of the necessary adaptation and LoReFT adds only a very marginal bonus.

Sonnet generation benefits from high temperature and LoRA. For sonnet generation, we found that LoRA rank 16 provided a good balance between capacity and overfitting (rank 64 caused severe overfitting with dev CHRF plateauing at ~ 38). A temperature of 1.2 was critical for generating text that captures the lexical diversity of Shakespearean verse, especially since lower temperatures produced conservative, repetitive outputs. The CHRF metric is inherently noisy for creative text, which motivated using a more lenient patience of 10 epochs for early stopping rather than the more typical 3–5.

7 Conclusion

We present a systematic comparison of Low-Rank Adaptation (LoRA) and Low-Rank Linear Subspace Representation Fine Tuning (LoReFT) for parameter-efficient fine-tuning (PEFT) of GPT-2 medium on paraphrase detection. Our key finding is that LoReFT's initial poor performance (0.741 dev) was not an inherent flaw of representation interventions. Instead, it was a consequence of specific design choices, particularly last-token intervention together with an orthogonality constraint at low rank. Across the LoReFT's variants we examined, the orthogonality constraint proved to be the most important factor: removing it yielded a 13.8 point improvement with no other increase in parameters. We also find that LoRA emerged as a strong standalone baseline (0.893 dev) that surpasses full fine-tuning. Ultimately, the best system combined LoRA with all-token LoReFT, achieving 0.894 dev and 0.893 test accuracy. For sonnet generation, LoRA fine-tuning with tuned generation parameters achieved a CHRF score of 40.7.

These results suggest that between PEFT methods for pairwise classification tasks, LoRA should be favored for its simplicity and strong baseline performance, while hybrid approaches should be considered when marginal gains matter. The sensitivity of LoReFT to its design choices, particularly the orthogonality constraint, highlights the importance of careful ablation when applying representation intervention methods to new tasks.

Team contributions

Andrew Chung: Implemented and tuned the sonnet generation pipeline, including the LoRA-based language model, nucleus sampling with temperature tuning, and CHRF evaluation.

Ryan Suh: Implemented and ran paraphrase detection experiments, including the LoRA module, the hybrid LoRA+LoReFT architecture, and the systematic ablation study.

Allen Yuan: Implemented the base GPT-2 model (the sentiment analysis classifier), the LoReFT module, the hook-based injection, and the nucleus sampling with temperature tuning.

All group members contributed equally to the writeup.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [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. *arXiv preprint arXiv:2106.09685*, 2021.
- [3] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*, 2024.
- [4] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [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. *International Conference on Machine Learning*, 2019.
- [7] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- [8] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2021.
- [9] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D Goodman. Finding alignments between interpretable causal variables and distributed neural representations. *arXiv preprint arXiv:2303.02536*, 2024.
- [10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.
- [11] Maja Popović. chrF: Character n-gram f-score for automatic mt evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.

A Appendix

A.1 Parameter Count Derivations

For GPT-2 medium ($d = 1024$, $L = 24$ layers):

LoReFT ($r = 4$): Each layer has a rotation matrix R ($r \times d = 4,096$ params) and a learned source W_s with bias ($r \times d + r = 4,100$ params), totaling 8,196 per layer. Across 24 layers: 196,704. With the classification head ($1,024 \times 2 + 2 = 2,050$): $\sim 199\text{K}$ total.

LoReFT ($r = 16$): Per layer: $16,384 + 16,400 = 32,784$. Across 24 layers plus head: $\sim 789\text{K}$.

LoRA ($r = 16$, Q+V): Each adapter has A ($d \times r$) and B ($r \times d$), totaling $2 \times d \times r = 32,768$ per adapter. Two adapters per layer: 65,536. Across 24 layers plus head: $\sim 1.57\text{M}$.

LoRA + LoReFT: Sum of LoRA and LoReFT ($r = 4$) parameters plus head: $\sim 1.77\text{M}$.

A.2 Hyperparameter Summary

Table 2 summarizes the training hyperparameters for each task and method family.

Table 2: Training hyperparameters across tasks and methods.

Hyperparameter	LoReFT variants	LoRA / Hybrid	Sonnet (LoRA)
Base model	GPT-2 medium	GPT-2 medium	GPT-2 medium
Optimizer	AdamW	AdamW	AdamW
Weight decay	0.01	0.01	0.01
LR schedule	Cosine + warmup	Cosine + warmup	Cosine + warmup
Warmup ratio	0.1	0.1	0.1
Learning rate	6×10^{-4}	2×10^{-4}	1×10^{-4}
Batch size	16	16	8
Epochs	5	5	20
Dropout	0.1	0.1	0.1
LoRA rank / α	—	16 / 16	16 / 16
ReFT rank	4 or 16	4	—
Temperature	—	—	1.2
Top- p	—	—	0.9
Early stop patience	—	—	10