Recovering the Pre-Fine-Tuning Weights of Generative Models

Eliahu Horwitz [1] **Jonathan Kahana** [1] **Yedid Hoshen** [1]

Abstract

The dominant paradigm in generative modeling consists of two steps: i) pre-training on a large-scale but unsafe dataset, ii) aligning the pre-trained model with human values via fine-tuning. This practice is considered safe, as no current method can recover the unsafe, *pre-fine-tuning* model weights. In this paper, we demonstrate that this assumption is often false. Concretely, we present *Spectral DeTuning*, a method that can recover the weights of the pre-fine-tuning model using a few low-rank (LoRA) fine-tuned models. In contrast to previous attacks that attempt to recover pre-fine-tuning capabilities, our method aims to recover the exact pre-fine-tuning weights. Our approach exploits this new vulnerability against large-scale models such as a personalized Stable Diffusion and an aligned Mistral. The code is available at https://vision.huji.ac.il/spectral_detuning/.

1. Introduction

A key paradigm in deep learning is to first pre-train a foundation model (Touvron et al., 2023; Roziere et al., 2023) on a large, general-purpose dataset and then fine-tune the model for a specific task. Fine-tuning is used for critical applications including model safety (Perez et al., 2022), alignment to human preferences and values (Ouyang et al., 2022; Christiano et al., 2017; Rafailov et al., 2023), providing privacy guarantees (Yu et al., 2021), personalization (Ruiz et al., 2023a), and more (Burns et al., 2023; Zhang et al., 2023a). In this paper, we identify a vulnerability in fine-tuned models, wherein the pre-fine-tuning (Pre-FT) weights, i.e., the model weights before the fine-tuning stage, can be recovered using a small number of models fine-tuned via low-rank adaptation (LoRA) (Hu et al., 2021).

[1]School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel. Correspondence to: Eliahu Horwitz <eliahu.horwitz@mail.huji.ac.il>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

To illustrate our setting, let us consider a Large Language Model (LLM). While the pre-trained version of the LLM exhibits advanced language understanding and generation capabilities, it is unaligned with human preference and is often deemed unsafe (Ouyang et al., 2022; Touvron et al., 2023). These unsafe models can be used for example to get instructions for building a bomb or other malicious activities. To improve instruction following and enhance safety, model creators perform an alignment fine-tuning stage. Usually, only the aligned version of the LLM is published, and the recovery of the original Pre-FT unsafe weights, is implicitly assumed to be impossible. While for existing models the recovery of the Pre-FT weights poses a security and safety vulnerability; for future superhuman models, it may lead to catastrophic consequences.

Motivated by the above, we propose the task of *Pre-Fine-Tuning Weight Recovery*. In this paper, we tackle this task in cases where multiple LoRA fine-tuned flavors of the same source model are available. We present an overview of our setting in Figure 1. This task is particularly timely due to two trends: i) Popular foundation models come in multiple flavors. E.g., LLaMA 2, Code LLaMA, Code LLaMA-Python, Code LLaMA-Instruct. ii) LoRA is becoming a key component for creating SoTA models (Lin et al., 2024; Sidahmed et al., 2024). These two trends have not yet merged, i.e, we are not aware of multi-flavored foundational models that use LoRA alignment fine-tuning. Here, we bring to the attention of the community the risks and perils involved in merging these trends.

We present *Spectral DeTuning*, a method that recovers the Pre-FT weights with remarkably high precision using iterative low-rank matrix factorization. To enhance optimization stability and accelerate convergence, we introduce a *rank scheduler* that progressively increases the rank of the factorized matrices during optimization. A key distinction from prior attacks on model alignment (Carlini et al., 2023; Wei et al., 2023; Zou et al., 2023) is that Spectral DeTuning prioritizes restoring the exact Pre-FT *weights* over Pre-FT *functionalities*. It also does not require running inference through the model. This is advantageous as i) it does not require training data ii) it is highly parallelizable, e.g., on a cluster of desktop GPUs such as RTX2080 our method can recover the Pre-FT weights of a Mistral-7B model in under five minutes.

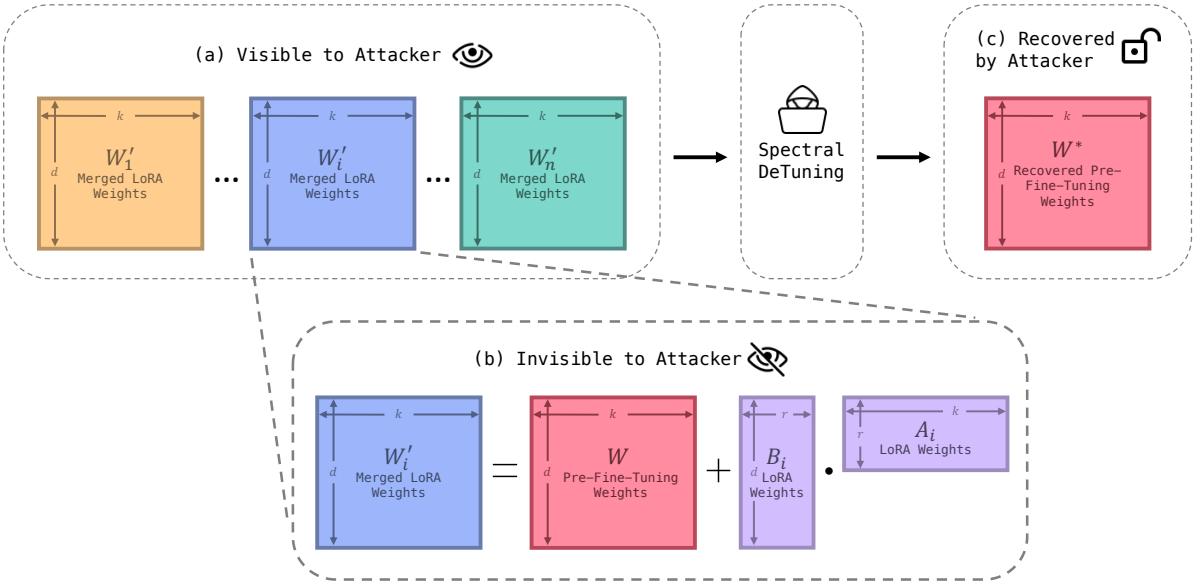

Figure 1. ***Pre-Fine-Tuning Weight Recovery Attack Setting:*** We uncover a vulnerability in LoRA fine-tuned models wherein an attacker is able to undo the fine-tuning process and recover the weights of the original pre-trained model. The setting for the vulnerability is as follows: (a) The attacker only has access to n different LoRA fine-tuned models. (b) The attacker assumes that all n models originated from the same source model. **Note: The attacker has no access to the low-rank decomposition of the fine-tuned models.** (c) Using only the n visible models, the attacker attempts to recover the original source model. Our method, *Spectral DeTuning*, can perform the attack in an unsupervised and data-free manner on real models such as Stable Diffusion and Mistral. For simplicity, we illustrate the attack on a single layer, in reality, the attack is carried out independently on all the fine-tuned layers. Best viewed in color

We demonstrate the effectiveness of our method by uncovering the vulnerability of real and widely used NLP and Vision models. Our approach achieves remarkable precision on an aligned Mistral model, effectively reversing the alignment training and restoring the original model (See Figure 2). Similarly, on Stable-Diffusion, we recover the original model's weights with a vanishingly small error, showcasing almost perfect reconstruction of the original generation capabilities (See Figure 3).

This work aims to stimulate research into preventing Pre-FT weight leakage and the associated risks in terms of model safety and alignment. To facilitate this research, we introduce *LoWRA Bench*, a comprehensive benchmark comprising datasets and evaluation metrics, designed for assessing Pre-FT weight recovery methods.

To summarize, our main contributions are:

1. Introducing the task of *Pre-Fine-Tuning Weight Recovery*, a new attack vector against fine-tuned models.
2. Presenting *Spectral DeTuning*, a highly effective method for pre-fine-tuning weight recovery attacks against state-of-the-art models.
3. Providing *LoWRA Bench*, a comprehensive suite of datasets and metrics designed for the evaluation of pre-fine-tuning weight recovery methods.

2. Related Works

2.1. Model Fine-tuning

Model fine-tuning, crucial in deep learning research (Zhang et al., 2023a; Zhai et al., 2022; Avrahami et al., 2023b), can be resource-intensive. Parameter-Efficient Fine-tuning (PEFT) methods (Hu et al., 2021; Dettmers et al., 2023; Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Liu et al., 2023; He et al., 2021; Liu et al., 2022; Jia et al., 2022; Zhang et al., 2023b; Wang et al., 2023b; Hyeon-Woo et al., 2021) aim to economize and broaden access to fine-tuning. These methods approximate full fine-tuning with fewer parameters. Some recent works combine multiple PEFT models (Yadav et al., 2023; Gu et al., 2023; Shah et al., 2023; Po et al., 2023; Huang et al., 2023), hoping to leverage the strengths of individual models. LoRA (Hu et al., 2021) is perhaps the most popular PEFT method and is known for its effectiveness across various tasks and modalities (Wang et al., 2023a; Ye et al., 2023; Ruiz et al., 2023b; Avrahami et al., 2023a), sometimes even outperforming full fine-tuning. Given its popularity, in this paper, we focus on recovering Pre-FT weights of LoRA fine-tuned models.

2.2. Model Safety and Security

Deep learning models have various safety and security vulnerabilities. Membership inference attacks aim to detect if

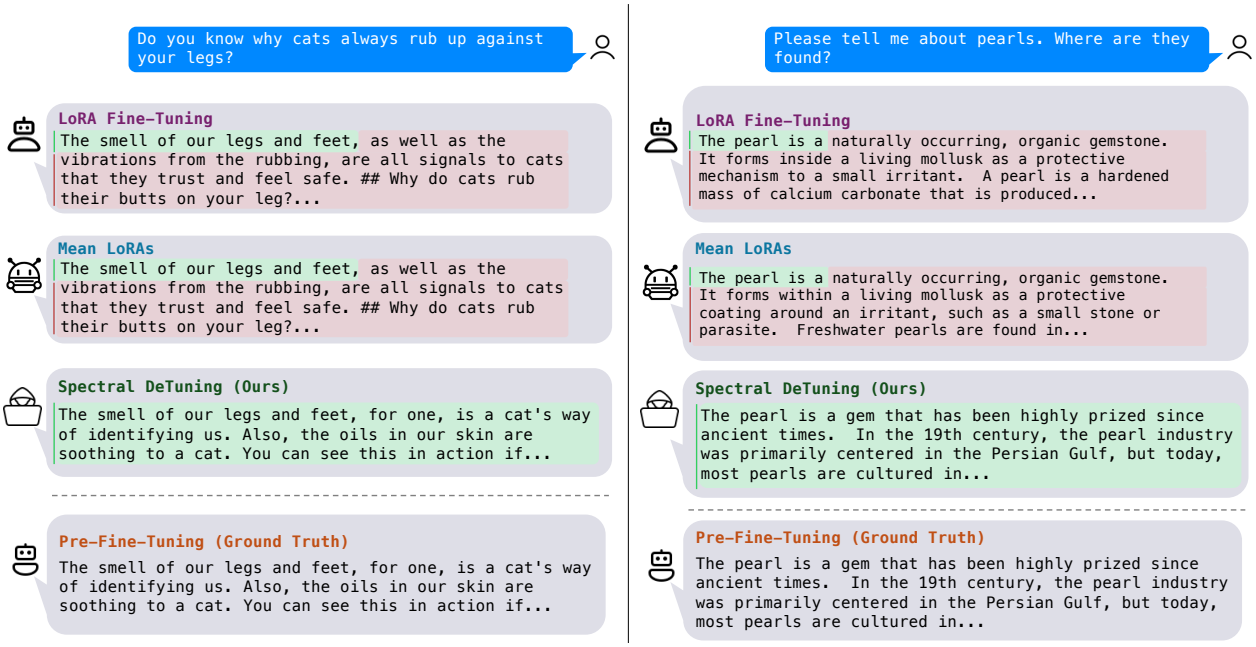

Figure 2. ***Mistral DPO Results:*** Our method, Spectral DeTuning, recovers the pre-fine-tuning generation capabilities with high precision, essentially undoing the DPO alignment LoRA fine-tuning. In green exact recovery, in red unrecovered words. Best viewed in color

specific data samples were used in training (Shokri et al., 2017; Shafran et al., 2021). Model inversion attempts to generate the samples used during training (Fredrikson et al., 2015; 2014). Machine unlearning protects against attacks by removing the effect of specific training samples without retraining the entire model (Bourtoule et al., 2021). Model extraction, or model stealing, involves stealing a target model hidden behind an API by querying it multiple times (Tramèr et al., 2016; Shafran et al., 2023). In contrast, Pre-FT weight recovery aims to recover the *exact weights* of the pre-trained model, compromising the entire model rather than just a subset of capabilities. Additionally, our method, Spectral DeTuning, operates in an unsupervised and data-free manner.

2.3. Model Red-Teaming and Adversarial Attacks

One of the primary methods for ensuring model safety involves incorporating human feedback through a reward model trained on annotator preferences, followed by reinforcement learning to fine-tune the model (Rafailov et al., 2023; Christiano et al., 2017; Perez et al., 2022; Ganguli et al., 2022; Segev et al., 2023; Sun et al., 2023). However, Wolf et al. (2023) argue that these alignment processes may leave undesired behavior partially intact and are thus vulnerable to adversarial prompting attacks. This has been demonstrated by red teaming (Perez et al., 2022; Ganguli et al., 2022) and adversarial attacks (Carlini et al., 2023; Wei et al., 2023; Zou et al., 2023) approaches. Unlike targeted attacks, Pre-FT weight recovery compromises the entire model by restoring the pre-trained weights. Moreover, our method, Spectral DeTuning, does not require running inference through the model.

3. Preliminaries - LoRA

Fine-tuning deep networks traditionally consisted of training all the network weights initialized by a pre-trained model. As this is costly for large-scale models, Hu et al. (2021) recently introduced Low Rank Adaptation (LoRA). The authors postulate that the change in weights during fine-tuning often has a "low intrinsic rank". They therefore introduced LoRA, which transforms each parameter matrix by the addition of a low-rank matrix. To create this low-rank matrix they multiply two full-rank matrices with suitable dimensions. This reparametrization drastically reduces the number of parameters being optimized. Specifically, for a pre-trained weight matrix $W_{\mathcal{P}} \in \mathbb{R}^{d\times k}$, the update ΔW can be decomposed into a rank r decomposition $\Delta W = BA$ where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$ and $r \ll min(d, k)$. During fine-tuning, $W_{\mathcal{P}}$ is frozen and only A and B are fine-tuned. This results in the following forward pass $W_{\mathcal{P}}x + \Delta W x = W_{\mathcal{P}}x + BAx$, where x is the outcome of the previous layer. Since LoRA is linear by design, it is possible to merge the fine-tuned matrices back into the original matrix

$$W' = W_{\mathcal{P}} + BA \tag{1}$$

, thus introducing no additional parameters or inference latency to the original model. Originally, LoRA was applied

to the query and value layers of attention blocks; however, it has been demonstrated that LoRA can be effectively extended to additional layers. Once merged, current models implicitly assume that recovering $W_{\mathcal{P}}$ and BA from W' is impossible. **Throughout the paper, whenever we refer to the weights of a LoRA fine-tuned model, we assume the weights have been merged back as seen in Equation (1).**

4. Problem Definition

We introduce the task of *Pre-Fine-Tuning Weight Recovery*. Its goal is to recover the Pre-FT weights of a given model, i.e., the weights of the original, pre-trained model. Specifically, in this work we assume that the fine-tuning was performed using LoRA.

Notation. Formally, consider a model $\mathcal{F}_{\mathcal{P}}$ with m fine-tuned layers that were fine-tuned via a rank r LoRA and originated from the source model $\mathcal{P}$. We denote the weight matrices of $\mathcal{F}_{\mathcal{P}}$ by $\{W'^{(j)}\}_{j=1}^{m}$ and those of $\mathcal{P}$ by $\{W_{\mathcal{P}}^{(j)}\}_{j=1}^{m}$ where both $W'^{(j)}$ and $W_{\mathcal{P}}^{(j)}$ are $\in \mathbb{R}^{d\times k}$. **Throughout the paper we assume the attacker does *not* have access to $\mathcal{P}$ (nor to its weights $\{W_{\mathcal{P}}^{(j)}\}_{j=1}^{m}$).**

Attack setting. The attacker has access to the weights of n different $\mathcal{F}_{\mathcal{P}}$ models, all LoRA fine-tuned from the same pre-trained source model $\mathcal{P}$. The attack succeeds with precision ϵ if the attacker can accurately recover the weights of the pre-trained source model $\mathcal{P}$ up to an ϵ precision. Formally, given $\left\{\{W_i'^{(j)}\}_{j=1}^{m}\right\}_{i=1}^{n}$, the attacker needs find $\{W^{*(j)}\}_{j=1}^{m}$ such that

$$\sum_{j=1}^{m}\left\|W_{\mathcal{P}}^{(j)}-W^{*(j)}\right\|<\epsilon \tag{2}$$

We present an overview of this setting in Figure 1.

Success criteria. We measure the success of the attack by the distance between the recovered weights and the original weights, in addition, in Section 6 we discuss a number of ways to measure the success of the attack semantically.

5. Spectral DeTuning

We now describe our method for carrying out a Pre-FT weight recovery attack. We start by introducing our optimization objective, followed by our optimization method and finally, a rank scheduler that stabilizes the optimization and results in better convergence. For simplicity, assume for now that all n LoRA fine-tuned models used the same rank r, and that the value of r is known to the attacker, in Sections 5.3 and 5.4 we relax these assumptions. For brevity, we omit the layer index superscript (j) and perform the same optimization across all layers independently.

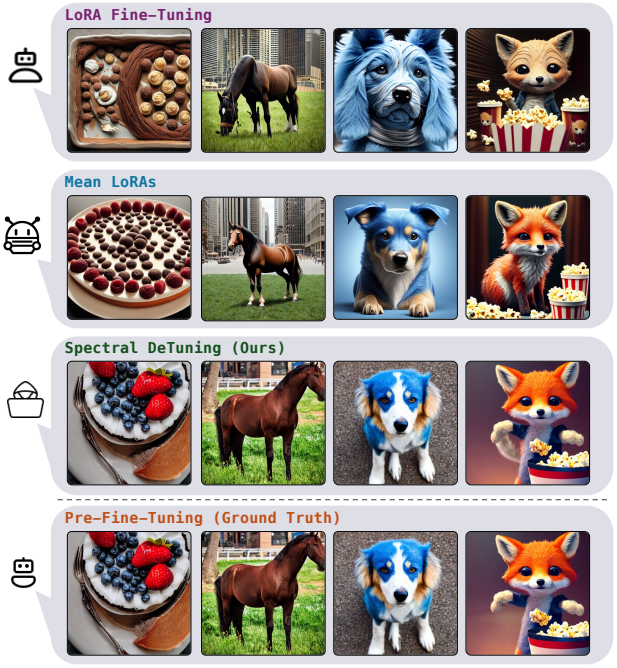

Figure 3. ***Stable Diffusion Results:*** Spectral DeTuning recovers the Pre-Fine-Tuning images with high precision, even when using "in the wild" LoRAs, essentially reversing the personalization fine-tuning of the LoRA model

5.1. Optimization Objective

To recover the Pre-FT weights, we need to predict $W_{\mathcal{P}}$ given n fine-tuned weight matrices $\{W_i'\}_{i=1}^{n}$. Leveraging their difference of up to r principal components, we formulate the task as an optimization problem, where each LoRA provides additional constraints on $W_{\mathcal{P}}$. Specifically, recall that according to Equation (1), W_i' can be decomposed into $W\in\mathbb{R}^{d\times k}$ and a rank r matrix which we will denote by $M_i\in\mathbb{R}^{d\times k}$. Taking into account all n different LoRA weights, we define the following objective

$$arg\min_{\substack{W,M_i\\1\le i\le n}}\sum_{i=1}^{n}\|W_i'-(W+M_i)\|_2^2 \quad s.t. \quad rank\, M_i\le r \tag{3}$$

Where $W\in\mathbb{R}^{d\times k}$ is the matrix we are optimizing to estimate $W_{\mathcal{P}}$. Intuitively, the objective optimizes the decomposition of each fine-tuned weight matrix into a *shared* weight matrix which is the approximated Pre-FT matrix and an independent *low rank* residual matrix.

This objective exhibits desirable properties for an attacker. First, it is training-free, meaning, it requires no data, nor does it make any assumptions with regards to the data used to train the model. Moreover, the optimization is performed on a per-layer basis, enabling high parallelization of the

attack. Finally, the objective is unsupervised, allowing an attacker to recover a model even when they have no prior knowledge regarding the source model.

5.2. Pre-FT Weight Recovery Algorithm

We propose Spectral DeTuning, an iterative, gradient-free algorithm for Pre-FT weight recovery. The method is fast (even on CPU) and is easily parallelizable. The core idea is that while the optimization problem in Equation (3) is non-convex, it can be iteratively broken down into a set of simple sub-problems which have closed-form solutions. Our procedure has three major components: initialization, M-step and W-step. Note, solving Equation (3) requires optimizing $n+1$ matrices, i.e., W and $M_1, M_2, ..., M_n$.

Initialization. At iteration 0, we set W^* as the average of all the fine-tuned matrices, i.e., $W^* = \frac{1}{n}\sum_{i=1}^{n} W_i'$.

M-step. We solve the optimization problem by coordinate descent (Wright, 2015). We first fix W^* and solve for $\{M_i\}_{i=1}^n$. Note that when W^* is given, the optimization problems for each $M_1, .., M_n$ are decoupled. Specifically, at each iteration, the optimization problem for M_i is:

$$M_i^* = arg\min_{M_i} \|(W_i' - W^*) - M_i\|_2^2 \quad s.t. \quad rank\, M_i \leq r \tag{4}$$

Luckily, the solution to this optimization problem is available in closed-form and is given by the "Singular Value Decomposition" (SVD) of $W_i' - W^*$. The optimal value of M_i is:

$$\begin{aligned} U_i, \Sigma_i, V_i^T &= \text{SVD}(W_i' - W^*) \\ M_i^* &= U_i \Sigma_{i|r} V_i^T \end{aligned} \tag{5}$$

Where $\Sigma_{i|r}$ represents the top r singular values of Σ_i.

W-step. By fixing the values of $M_1^*, .., M_n^*$, we can easily compute the optimal value of W. The optimization problem is given by:

$$W^* = arg\min_{W} \sum_{i=1}^{n} \|(W_i' - M_i^*) - W\|_2^2 \tag{6}$$

By simple calculus, the closed-form solution is:

$$W^* = \frac{1}{n}\sum_{i=1}^{n} (W_i' - M_i^*) \tag{7}$$

We iterate between the M-step and W-step until convergence. As shown in Algorithm 1, the algorithm can be easily implemented in as little as 8 lines of python.

Algorithm 1 PyTorch Pseudocode for Spectral DeTuning

```
# W_ps: List of n fine-tuned weight matrices
# steps: Number of optimization steps
# r: LoRA rank

# Initialize W_star
W_s = torch.mean(torch.stack(W_ps), axis=0)

# Perform optimization
for step in range(steps):
    # M-step
    # Approximate each M^*_i (Eq. 5)
    M_s = [W_p - W_s for W_p in W_ps]

    # Truncate each M^*_i to rank <= r (Eq. 5)
    for i in range(len(M_s)):
        (U, S, V) = torch.svd_lowrank(M_s[i], q=r)
        M_s[i] = (U @ torch.diag_embed(S)) @ V.T


    # W-step
    # Approximate W_star (Eq. 7)
    W_s = [W_p - M_si for (W_p, M_si) in zip(W_ps, M_s)]
    W_s = torch.mean(torch.stack(W_s), axis=0)
```

5.3. Rank Scheduler

The algorithm proposed in Section 5.2 tends to perform well in general. However, we empirically found that solving the optimization problem with high ranks can result in slow and inaccurate convergence. We therefore introduce a rank scheduler. The idea of the rank scheduler is to start by forcing M_i to be of rank $r^* < r$, allowing Spectral DeTuning to focus on the most significant principal components first. r^* is increased according to a schedule until finally $r^* = r$. Specifically, we use an "Increase on Plateau" type of scheduler where the rank is increased whenever the loss term from Equation (3) plateaus. When not all LoRAs have the same rank, we assign a distinct rank scheduler to each LoRA. The rank scheduler requires knowing the LoRA rank; we show how to estimate it in Section 5.4.For more details see Appendix F. We show empirically in Section 8 that there are cases where the rank scheduler improves the rate and quality of convergence significantly.

5.4. LoRA Rank Estimation

We propose an effective heuristic for estimating LoRA rank. Assume we have two LoRA fine-tuned models $W_i' = W + M_i$ and $W_j' = W + M_j$, where the ranks of M_i, M_j are r_i, r_j respectively. While it is not trivial to recover the rank of M_i solely by observing W_i', there is a trick. Subtracting the two fine-tuned models obtains $W_i' - W_j' = M_i - M_j$. Importantly, the rank $W_i' - W_j'$ is upper bounded by $r_i + r_j$, i.e., $rank(W_i' - W_j') \leq r_i + r_j$. Given n LoRAs, there are $\frac{n(n-1)}{2}$ distinct inequalities for the n unknown ranks $r_1, r_2, .., r_n$.

We can formulate this as a linear programming problem as

follows:

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & \mathbf{1}^T\mathbf{r} \\ \text{subject to} \quad & \mathbf{A}\mathbf{r} \geq \mathbf{b} \\ & r_i \geq 1, \quad \forall i \end{aligned}$$

where:

- $\mathbf{A} \in \{0,1\}^{n^2,n}$ so that $A_{ni+j,i} = 1$ and $A_{ni+j,j} = 1$ and 0 elsewhere.
- $\mathbf{b} \in R^{n^2}$ so that $b_{ni+j} = rank(W'_i - W'_j)$.

In practice, we populate b using a *numerical rank* computed via the multiplicative gap following a similar protocol to (Carlini et al., 2024). Using an off-the-shelf linear programming solver accurately retrieves the ranks. We demonstrate the accuracy of this method in Section 8, the unknown ranks were recovered perfectly in all cases.

6. LoWRA Bench

We present ***Lo**RA **W**eight **R**ecovery **A**ttack* (LoWRA) Bench, a comprehensive benchmark designed to evaluate Pre-FT weight recovery methods.

6.1. Dataset

Our dataset encompasses three pre-trained representative source models: a Vision Transformer (ViT) (Dosovitskiy et al., 2020) trained on ImageNet-1K (Russakovsky et al., 2015), Mistral-7B-v0.1 (Jiang et al., 2023), and Stable Diffusion 1.5 (Rombach et al., 2022). These models collectively cover supervised and self-supervised objectives, spanning both vision and natural language processing (NLP) domains, as well as generative and discriminative tasks. Notably, these models are widely used and deployed in numerous production systems. See Table 1 for an overview of the dataset.

For each source model, we curate 15 LoRA models fine-tuned on diverse datasets, tasks, and objectives. The dataset comprises a diverse array of layer types, including self-attention, cross-attention, and MLPs. This diversity enables us to assess the generalization capabilities of Pre-FT methods. The evaluation can be conducted on a per-model basis, per layer type, or per layer depth, allowing for a comprehensive analysis of Pre-FT methods. Overall, our dataset includes 544 source model layers. When taking into account the fine-tuned LoRA layers, the dataset includes over $8,000$ layers. For further details see Appendix E.

6.2. Numeric Evaluation Metrics

Weight Error (W-Error). We measure numeric convergence by the mean squared weight error (as defined in Equation (2)) and average across all layers in log space:

$$\frac{1}{m}\sum_{j=1}^{m}\left(\log_{10}\left(MSE(W_{\mathcal{P}}^{(j)} - W^{*(j)}\right)\right) \tag{8}$$

We use log-space as when errors are very small, the average mean squared weight error is determined by outliers, e.g., a single non-converging layer when all other layers converge. Log transforming the mean squared error is robust to such outliers. We visualize this in Figure 4 where Spectral DeTuning clearly converges to a much better solution. Despite the outstanding convergence, the small number of outliers create a false impression where the MSE shows a significantly higher error. In Appendix C we show that the W-Error is strongly correlated with the recovery of the Pre-FT semantic capabilities ($\rho = 0.880$ for W-Error vs. LPIPS).

Table 1. ***LoWRA Bench Dataset Summary:*** The dataset covers widely used models spanning vision and language modalities. It includes over 540 Pre-FT layers and over $8,000$ fine-tuned layers

Pre-FT Model	Task	Fine-tuning Task	# Pre-FT Layers	# FT Layers
ViT	Classific.	VTAB-1K	24	360
SD1.5	T2I Gen.	Personalization	264	3960
Mistral	Text Gen.	UltraChat SFT	128	1920
Mistral	Text Gen.	UltraFeedback DPO	128	1920

6.3. Semantic Evaluation Metrics

We design model specific metrics focusing on the Pre-FT task from a semantic perspective.

ViT Activation Distance (Act.-Dist.). We take the cosine distance between the activations of the Pre-FT model and those of the recovered one. Specifically, we take the mean of all transformer tokens at the end of the last transformer block. We use a subset of 5000 images from the ImageNet validation set.

Stable Diffusion LPIPS (LPIPS). The LPIPS (Zhang et al., 2018) distance between images generated by the Pre-FT model and by the recovered model. We report the mean LPIPS for the first 100 prompts of the COCO Captions validation dataset (Chen et al., 2015).

Mistral SBERT (SBERT). The log cosine distance between the Sentence-BERT (Reimers & Gurevych, 2019) (SBERT) textual embeddings of text generated by the Pre-FT model and by the recovered model. We report the mean log cosine for the first 100 prompts of the Alpaca Farm evaluation benchmark (Dubois et al., 2023).

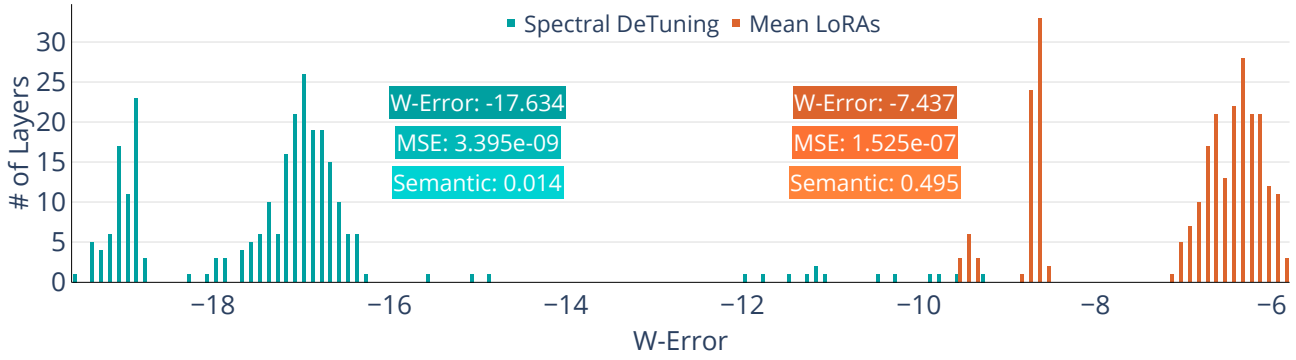

Figure 4. ***Motivation for the Log in W-Error:*** We visualize the convergence of all layers using Spectral DeTuning and the Mean LoRAs baselines. Spectral DeTuning clearly converges to a much better solution for almost all layers. Note that MSE does not summarize the convergence well as it yields the value of the poorly converging outlier layers. The W-Error better conveys the actual convergence by working in log-space. Results for a random subset of 5 Stable Diffusion LoRAs

6.4. Experimental Setup

Subsets. In each experiment, we specify a number of LoRA fine-tuned models L, which is often lower than the total number of LoRAs available in the datasets. We do this by randomly sampling a set of L models from the datasets. We then perform the Pre-FT weight recovery method on this subset. We repeat this experiment (including subset sampling) 10 times. The reported performance metrics are the average and standard deviation over the experiments.

Baselines. The two baseline methods are i) using one of the fine-tuned LoRA models; we average the results over all models in the sampled subset. ii) averaging the weights across all LoRA fine-tuned models in the sampled subset and reporting the results of the weight averaged model. The motivation behind the mean LoRA baseline, is the assumption that the mean of the residuals is the zero matrix, i.e., $\frac{1}{n}\sum_{i=1}^{n} M_i = 0$. In this case the optimum of Equation (3) becomes the average of all the weights.

7. Experiments

7.1. Preliminary Investigation on ViT

We begin our exploration of Pre-FT weight recovery using ViT, due to its simple architecture with consistent weight dimensions and relatively small model size. While this is our simplest task, it is not a "toy example" but a real model that is widely used and deployed in countless production settings. In Table 2 we show the results for $n = 5$ fine-tuned LoRAs. As expected, the LoRA fine-tuned models are indeed different from the Pre-FT model. Averaging over several LoRA models slightly improves the results, but is still far from recovering the Pre-FT activations. Our method, Spectral DeTuning, performs much better and attains an almost perfect semantic convergence, outperforming the baselines by a wide margin.

Table 2. ***ViT Results:*** As expected, the LoRA fine-tuned models have drifted away from the initial weights and activations. The mean of the LoRAs is slightly better, but is still far from the Pre-FT model. In contrast, Spectral DeTuning achieves an almost perfect semantic convergence. Reported results use $n = 5$ fine-tuned LoRAs

Method	W-Error ↓	Act.-Dist. ↓
LoRA FT	-4.602 ±0.110	1e-1 ±9e−2
Mean LoRA	-5.214 ±0.114	5e-2 ±1e−2
Spectral DeTuning	**-15.942** ±1.889	**1e-6** ±3e-6

7.2. In the Wild Weight Recovery of Stable Diffusion

Having shown the vulnerability of an image classification model, we now test the vulnerability of Stable Diffusion, a multi-modal text-to-image model. To this end, we used publicly fine-tuned LoRAs found on *civitai*, allowing us to validate our method "in the wild". As in the case of ViT, the baselines perform poorly on all metrics. In contrast, Spectral DeTuning recovers the Pre-FT weights with high precision. This results in a significant improvement of the recovered semantic capabilities of the Pre-FT model while using as little as $n = 5$ fine-tuned LoRAs (See Table 3 and Figure 3).

Implication: SoTA personalization methods using LoRA are vulnerable to Pre-FT weight recovery attacks.

7.3. Pre-FT Weight Recovery of an Aligned LLM

Having achieved success with mid-sized image models, we now investigate the ability of our method to scale up to a large-scale aligned LLM. Specifically, we use Mistral-7B, a

Table 3. ***Stable Diffusion Results:*** Spectral DeTuning is almost three times better than the baselines, recovering a large portion of the semantic capabilities of the pre-fine-tuning Stable Diffusion. Reported results use $n = 5$ fine-tuned LoRAs taken from an online LoRA marketplace

Method	W-Error ↓	LPIPS ↓
LoRA FT	-6.921 ±1.080	0.514 ±0.047
Mean LoRA	-7.540 ±1.099	0.482 ±0.012
Spectral DeTuning	**-17.816** ±2.126	**0.009** ±0.006

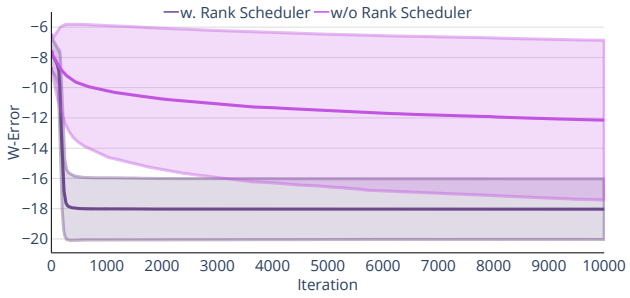

Figure 5. ***Rank Scheduler Convergence Speed:*** Using the rank scheduler has three benefits, i) accelerated convergence , ii) less variance between layers, and iii) higher precision convergence. Here we visualize i), see Figure 6 for a layer-wise visualization

top performing open-source LLM. Following common practice, we fine-tune the model in two stages, first performing supervised fine-tuning (SFT) followed by a direct preference optimization (DPO) alignment fine-tuning stage (Rafailov et al., 2023). We report the results of both stages in Table 4, as we can see, Spectral DeTuning successfully recovers the weights with high precision. This high quality recovery is also expressed in recovering the semantic capabilities of the Pre-FT model. I.e., the estimated weights yield a model which provides responses that are very similar to the Pre-FT model and much more so than the LoRA fine-tuned model (See Figure 2).

Implication: SoTA LLMs that use LoRA for alignment fine-tuning are vulnerable to Pre-FT weight recovery attacks.

8. Ablations

8.1. Rank Scheduler Ablation

We ablate the rank scheduler introduced in Section 5.3 using the Stable Diffusion experiment. Based on Figure 5 we observe three phenomena, i) The rank scheduler drastically *accelerates* the convergence, ii) When using the rank scheduler, there is much *less variance* between the convergence of different layers, and iii) Using the rank scheduler results in a *higher precision* convergence. Figure 6 visualizes phenomena (ii) and (iii) by showing the cumulative percent

Table 4. ***Mistral Results:*** Spectral DeTuning recovers the Pre-FT weights and semantic capabilities with high precision, both in the supervised fine-tuning (SFT) stage and the alignment fine-tuning stage (DPO). Reported results use $n = 12$ fine-tuned LoRAs for SFT and $n = 8$ fine-tuned LoRAs for DPO

	Method	W-Error ↓	SBERT ↓
SFT	LoRA FT	-8.677 ±0.153	-0.994 ±0.731
SFT	Mean LoRA	-9.299 ±0.222	-1.007 ±0.726
SFT	Spectral DeTuning	**-16.502** ±1.855	**-9.324** ±6.942
DPO	LoRA FT	-9.903 ±0.166	-3.058 ±4.763
DPO	Mean LoRA	-10.757 ±0.178	-3.455 ±5.171
DPO	Spectral DeTuning	**-22.062** ±1.180	**-14.708** ±3.123

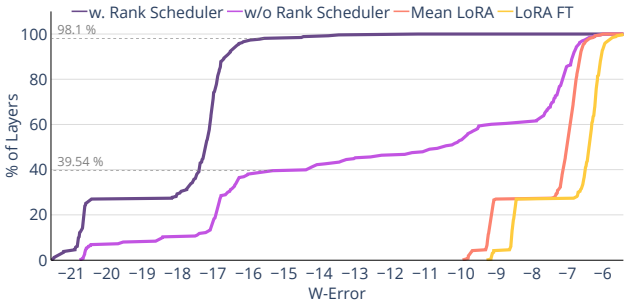

Figure 6. ***Rank Scheduler Convergence Quality:*** When using the rank scheduler, over 95% of the layers converge with a precision of at least -16, in contrast to less than 40% without the scheduler

of layers (y axis) that converge to a given W-Error level (x axis). When using the rank scheduler, over 95% of the layers converge with a precision of at least -16, in contrast to less than 40% when not using the scheduler. Moreover, by using the rank scheduler, some layers converge to a more precise solution.

8.2. Robustness to Unknown and Varying Ranks

We tested the LoRA rank estimation heuristic presented in Section 5.4 on hundreds of combinations of LoRAs with different ranks. The heuristic achieved an accuracy of 100%. We further tested the idea of using a dedicated rank scheduler for each LoRA model as described in Sections 5.3 and 5.4. We use $n = 6$ fine-tuned LoRAs with ranks $[8, 32, 32, 32, 64, 100]$ taken from CivitAI. Spectral DeTuning is robust to the varying ranks, exhibiting only a minor decrease in performance despite the higher rank of the LoRAs (See Table 5).

8.3. Robustness to Different Models

We demonstrate the robustness of Spectral DeTuning to cases where a fine-tuned LoRA from a different Pre-FT model (with the same architecture) was mixed into the set

Table 5. ***Robustness to Unknown and Varying Ranks Results:*** We test the robustness to LoRAs with varying ranks. Spectral DeTuning is robust to varying ranks, exhibiting only a minor decrease in performance. We use $n = 6$ fine-tuned LoRAs with ranks $[8, 32, 32, 32, 64, 100]$ taken from an online LoRA marketplace

Method	W-Error ↓	LPIPS ↓
LoRA FT	-5.882	0.462
Mean LoRA	-6.969	0.307
Spectral DeTuning	**-14.453**	**0.073**

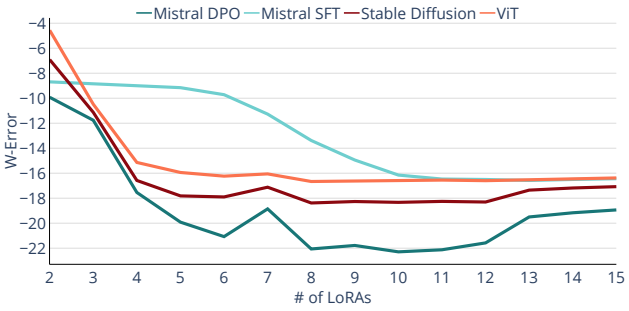

Figure 7. ***Effect of the Number of LoRAs on W-Error Convergence:*** For the semantic equivalent see Appendix A

of fine-tuned LoRAs. Using the same heuristic presented in Section 5.4, the difference between the mixed model weights and any other LoRA should be of full rank (since the Pre-FT model is different) and trivial to detect.

We validated this solution using Stable Diffusion. We added to the set of fine-tuned LoRA models a model that originated from Stable Diffusion 1.4 (all the others originated from Stable Diffusion 1.5). Indeed, the above steps indicated the LoRA that originated from Stable Diffusion 1.4 has a full rank difference from any other LoRA (while the pairwise rank between the LoRAs that used the same Pre-FT model were low rank, as expected). This allows us to detect the LoRA that got mixed up into the set and remove it.

8.4. W-Error vs. Loss

In reality an attacker has no access to the error and can only measure the loss in Equation (3). To show the loss accurately reflects the error defined in Equation (2), we measure their relation and find they are almost perfectly correlated ($\rho = 0.994$). For further details see Appendix B.

9. Discussion and Limitations

9.1. Number of LoRAs

Spectral DeTuning requires several LoRAs to recover the Pre-FT weights. In Figure 7 we illustrate the impact of the number of fine-tuned LoRA models on the W-Error convergence. Note that different W-Error values are not comparable across models, e.g., Mistral DPO obtains a lowest W-Error but only semantically converges when using 8 LoRAs (See Figure 11). In Appendix A we study the effects of the number of LoRAs on the semantic convergence for all LoWRA Bench subsets. We anticipate that future methods will incorporate additional constraints to reduce the required number of LoRAs.

9.2. Public Availability of LoRA Fine-tuned Models

We assume the availability of multiple LoRA fine-tuned models originating from the same pre-fine-tuning model. This is a reasonable assumption as there are model "marketplaces" such as *Hugging Face* and *civitai*, where many LoRA fine-tuned models are publicly available. These LoRA models often share the same source Pre-FT model, which fits our proposed setting perfectly.

9.3. Other Types of Fine-tuning

While our focus has been on exposing the vulnerability of LoRA fine-tuned models, numerous other parameter-efficient fine-tuning methods exist. The general case of Pre-FT weight recovery of fully fine-tuned models is the most general and probably hardest case. Extending the scope of our attack to encompass these methods presents an exciting avenue for research.

9.4. Pre-FT Weight Recovery Defense

We do not know of a defense against this attack. Also, as this attack targets publicly available models, once a vulnerability is identified, there is no option to retract the model. However, we remain optimistic that a defense will be discovered in the future. For instance, modifying training such that an infeasible high number of LoRAs will be required for accurate recovery.

10. Conclusion

In this paper, we unveiled a new vulnerability in LoRA fine-tuned models, allowing attackers to recover the Pre-FT weights using multiple models. Our method, Spectral DeTuning, demonstrates this vulnerability on large-scale models like Mistral and Stable Diffusion. We introduced LoWRA Bench and discussed future directions to promote further research. By highlighting this vulnerability, we hope to encourage the research community to develop better defenses against such attacks.

Acknowledgements

This work was supported in part by the "Israel Science Foundation" (ISF), the "Council for Higher Education" (Vatat), and the "Center for Interdisciplinary Data Science Research" (CIDR).

Impact Statement

This work uncovers a significant vulnerability in fine-tuned models, allowing attackers to access pre-fine-tuning weights. While this discovery reveals potential security risks, our primary objective is to advance the field of Machine Learning and raise awareness within the research community about the existing vulnerabilities in current models.

Instead of using the findings of this study to execute attacks, we advocate for their use by model creators to enhance the safety and security of their models. By acknowledging and addressing vulnerabilities, creators can proactively safeguard against potential threats.

Furthermore, in the discussion section, we outline potential future directions and mitigation strategies. Following established practices in the cyber security community, we emphasize the importance of open discussion and encourage the reporting of vulnerabilities. By fostering transparency and collaboration, we can collectively create a safer environment for deploying machine learning models.

References

Avrahami, O., Aberman, K., Fried, O., Cohen-Or, D., and Lischinski, D. Break-a-scene: Extracting multiple concepts from a single image. In *SIGGRAPH Asia 2023 Conference Papers*. Association for Computing Machinery, 2023a. 2

Avrahami, O., Hayes, T., Gafni, O., Gupta, S., Taigman, Y., Parikh, D., Lischinski, D., Fried, O., and Yin, X. Spatext: Spatio-textual representation for controllable image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b. 2

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021. 3

Burns, C., Izmailov, P., Kirchner, J. H., Baker, B., Gao, L., Aschenbrenner, L., Chen, Y., Ecoffet, A., Joglekar, M., Leike, J., et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023. 1

Carlini, N., Nasr, M., Choquette-Choo, C. A., Jagielski, M., Gao, I., Awadalla, A., Koh, P. W., Ippolito, D., Lee, K., Tramer, F., et al. Are aligned neural networks adversarially aligned? *arXiv preprint arXiv:2306.15447*, 2023. 1, 3

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024. 6

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 6, 16

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017. 1, 3

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023. 16

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. 2

Ding, N., Chen, Y., Xu, B., Qin, Y., Zheng, Z., Hu, S., Liu, Z., Sun, M., and Zhou, B. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023. 16

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6

Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023. 6, 16

Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., and Ristenpart, T. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfar in dosing. In *23rd USENIX security symposium (USENIX Security 14)*, pp. 17–32, 2014. 3

Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015. 3

Ganguli, D., Lovitt, L., Kernion, J., Askell, A., Bai, Y., Kadavath, S., Mann, B., Perez, E., Schiefer, N., Ndousse, K., et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022. 3

Gu, Y., Wang, X., Wu, J. Z., Shi, Y., Chen, Y., Fan, Z., Xiao, W., Zhao, R., Chang, S., Wu, W., et al. Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models. *arXiv preprint arXiv:2305.18292*, 2023. 2

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366, 2021. URL https://api.semanticscholar.org/CorpusID:238583580. 2

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019. 2

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1, 2, 3

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023. 2

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021. 2

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *European Conference on Computer Vision*, pp. 709–727. Springer, 2022. 2, 15

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 6

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 2

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 2

Lin, P., Ji, S., Tiedemann, J., Martins, A. F., and Schütze, H. Mala-500: Massive language adaptation of large language models. *arXiv preprint arXiv:2401.13303*, 2024. 1

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965, 2022. 2

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. Gpt understands, too. *AI Open*, 2023. 2

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022. 15

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. 1

Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., and Irving, G. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022. 1, 3

Po, R., Yang, G., Aberman, K., and Wetzstein, G. Orthogonal adaptation for modular customization of diffusion models. *arXiv preprint arXiv:2312.02432*, 2023. 2

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. 1, 3, 8

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084. 6

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022. 6

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023. 1

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023a. 1

Ruiz, N., Li, Y., Jampani, V., Wei, W., Hou, T., Pritch, Y., Wadhwa, N., Rubinstein, M., and Aberman, K. Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models. *arXiv preprint arXiv:2307.06949*, 2023b. 2

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015. 6, 15

Segev, E., Alroy, M., Katsir, R., Wies, N., Shenhav, A., Ben-Oren, Y., Zar, D., Tadmor, O., Bitterman, J., Shashua, A., et al. Align with purpose: Optimize desired properties in ctc models with a general plug-and-play framework. *arXiv preprint arXiv:2307.01715*, 2023. 3

Shafran, A., Peleg, S., and Hoshen, Y. Membership inference attacks are easier on difficult problems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14820–14829, October 2021. 3

Shafran, A., Shumailov, I., Erdogdu, M. A., and Papernot, N. Beyond labeling oracles: What does it mean to steal ml models? *arXiv preprint arXiv:2310.01959*, 2023. 3

Shah, V., Ruiz, N., Cole, F., Lu, E., Lazebnik, S., Li, Y., and Jampani, V. Ziplora: Any subject in any style by effectively merging loras. 2023. 2

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017. 3

Sidahmed, H., Phatale, S., Hutcheson, A., Lin, Z., Chen, Z., Yu, Z., Jin, J., Komarytsia, R., Ahlheim, C., Zhu, Y., et al. Perl: Parameter efficient reinforcement learning from human feedback. *arXiv preprint arXiv:2403.10704*, 2024. 1

Sun, S., Gupta, D., and Iyyer, M. Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of rlhf. *arXiv preprint arXiv:2309.09055*, 2023. 3

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pp. 601–618, 2016. 3

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023. 15, 16

Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., and Zhu, J. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *ArXiv*, abs/2305.16213, 2023a. URL https://api.semanticscholar.org/CorpusID:258887357. 2

Wang, Z., Panda, R., Karlinsky, L., Feris, R., Sun, H., and Kim, Y. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*, 2023b. 2

Wei, A., Haghtalab, N., and Steinhardt, J. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023. 1, 3

Wolf, Y., Wies, N., Levine, Y., and Shashua, A. Fundamental limitations of alignment in large language models. *arXiv preprint arXiv:2304.11082*, 2023. 3

Wright, S. J. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015. 5

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023. 2

Ye, Q., Xu, H., Xu, G., Ye, J., Yan, M., Zhou, Y., Wang, J., Hu, A., Shi, P., Shi, Y., et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023. 2

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., Kulkarni, J., Lee, Y. T., Manoel, A., Wutschitz, L., et al. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*, 2021. 1

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 15

Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A., and Beyer, L. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18123–18133, 2022. 2

Zhang, L., Rao, A., and Agrawala, M. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3836–3847, October 2023a. 1, 2

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b. 2

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018. 6

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. 1, 3

A. The Effect of the Number of LoRAs on Semantic Convergence

We visualize the effect of the number of LoRAs on the semantic convergence for each of the LoWRA Bench subsets, results are shown in Figures 8 to 11.

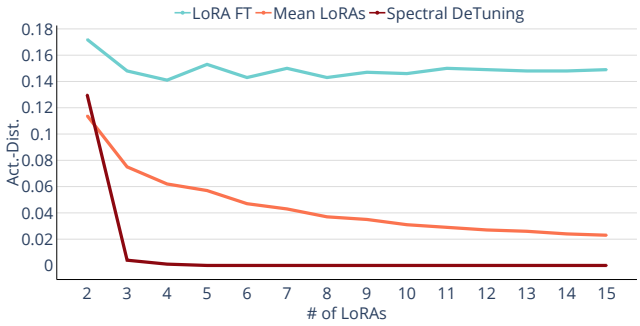

Figure 8. ***Number of LoRAs vs. Semantic Convergence - ViT***

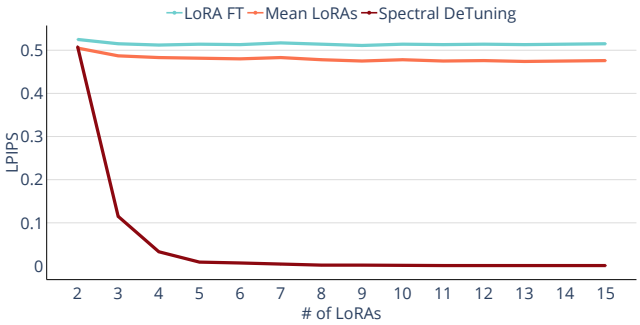

Figure 9. ***Number of LoRAs vs. Semantic Convergence - Stable Diffusion***

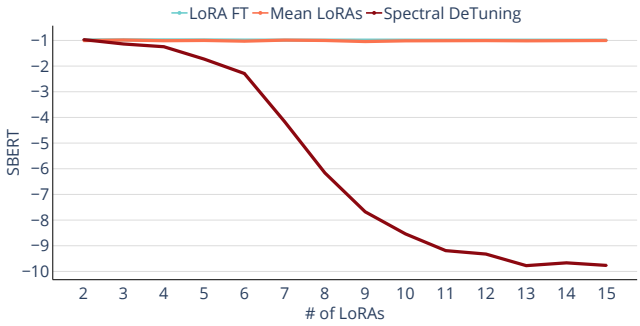

Figure 10. ***Number of LoRAs vs. Semantic Convergence - Mistral SFT***

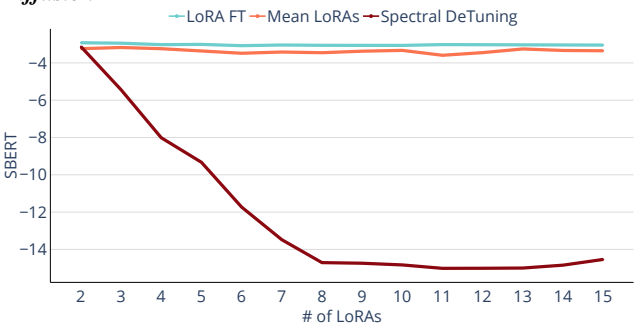

Figure 11. ***Number of LoRAs vs. Semantic Convergence - Mistral DPO***

B. W-Error vs. Loss

We visualize the relation between the W-Error and the log loss and find they are almost perfectly correlated ($\rho = 0.994$), see Figure 12 for a visualization over 200 iterations using Stable Diffusion.

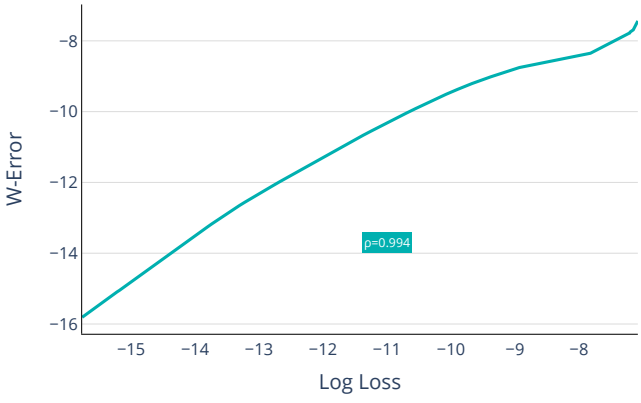

Figure 12. ***W-Error vs. Loss - Stable Diffusion***

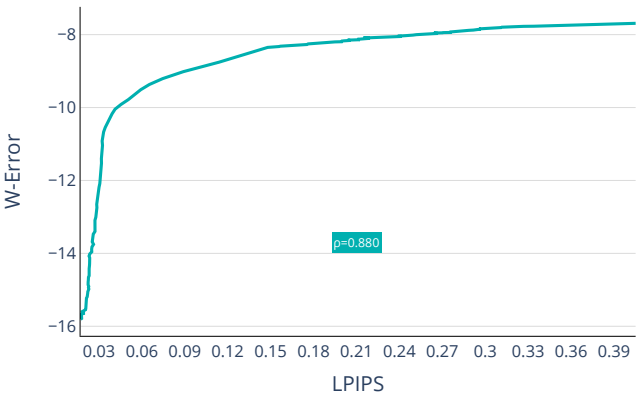

Figure 13. ***W-Error vs. LPIPS***

C. W-Error vs. LPIPS

We visualize the relation between the W-Error and LPIPS and find they are strongly correlated ($\rho = 0.880$), see Figure 13 for a visualization over 200 iterations using Stable Diffusion.

D. LoRA Rank vs. W-Error

In Tables 6 and 7 we show the results for the ViT model when using different LoRA ranks and fixing the number of LoRAs.

Table 6. ***Using*** 5 ***LoRAs***

Rank	W-Error
8	-15.636
12	-15.550
16	-13.480
32	-4.817

Table 7. ***Using*** 5 ***LoRAs***

Rank	W-Error
8	-15.822
12	-15.773
16	-15.258
32	-9.639

E. LoWRA Bench Dataset

We now elaborate on the implementation details of the LoWRA Bench dataset.

E.1. ViT Models

As the Pre-FT model we use "vit-base-patch16-224" found on hugging face (https://huggingface.co/google/vit-base-patch16-224). We fine-tune the model using the PEFT library (Mangrulkar et al., 2022). For each LoRA we use a different VTAB-1k (Zhai et al., 2019) dataset, the datasets we use are: `cifar100`, `caltech101`, `dtd`, `flower102`, `pet37`, `svhn`, `patch_camelyon`, `clevr-count`, `clevr-distance`, `dmlab`, `kitti`, `dsprites-location`, `dsprites-orientation`, `smallnorb-azimuth`, `smallnorb-elevation`. We pre-process the datasets according to the protocol of Jia et al. (2022) found on their github page https://github.com/KMnP/vpt/blob/main/VTAB_SETUP.md. We use an $80/20$ train/validation split and choose the checkpoint with the best validation loss.

We use a rank $r = 16$ and LoRA fine-tune the `query` and `value` layers. This protocol results in 24 Pre-FT model layers and a total of $24 \cdot 15 = 360$ LoRA fine-tuned layers. See Table 8 for the fine-tuning hyper-parameters.

For semantic evaluation we use a subset of the ImageNet-1K (Russakovsky et al., 2015) validation set. We construct the subset by taking the first 5 images of each class, resulting in a subset of 5000.

Table 8. ***ViT Hyper-parameters***

Name	Value
`lora_rank` (r)	16
`lora_alpha` (α)	16
`lr`	$9e-3$
`batch_size`	128
`epochs`	20
`datasets`	cifar100, caltech101, dtd, flower102, pet37, svhn, patch_camelyon, clevr-count, clevr-distance, dmlab, kitti, dsprites-location, dsprites-orientation, smallnorb-azimuth, smallnorb-elevation

E.2. Mistral Models

As the Pre-FT model we use "Mistral-7B-v0.1" found on hugging face (https://huggingface.co/mistralai/Mistral-7B-v0.1). We fine-tune the model following the protocol of Tunstall et al. (2023), note that unlike Tunstall et al. (2023), we perform LoRA fine-tuning as found on their official github repo https://github.com/huggingface/

alignment-handbook. Following the original LoRA setting, we make a minor adjustment to the original hyper-parameters of the repo and use a LoRA alpha of 64 instead of 16 (i.e. $\alpha = 64$), this leads to faster and better convergence. To fine-tune 15 different models, we use different *random* subsets of 80% of the fine-tuning dataset. We use seeds of $0 - 14$ for the different fine-tuned models.

We follow this protocol for both the supervised fine-tuning stage (SFT) and the direct preference optimization (DPO) alignment stage. Following Tunstall et al. (2023), the SFT stage uses the UltraChat dataset (Ding et al., 2023) and the DPO stage uses the UltraFeedback dataset (Cui et al., 2023). We first fine-tune the 15 SFT models, and then fine-tune the 15 DPO models, where each DPO model continues the training of the SFT model with the corresponding seed.

Following the original setup, use a rank $r = 64$ and LoRA fine-tune the q_proj, k_proj, v_proj, and o_proj layers. This protocol results in 128 Pre-FT model layers and a total of $128 \cdot 15 = 1920$ LoRA fine-tuned layers for both the SFT and DPO stages. See Tables 9 and 10 for the fine-tuning hyper-parameters. For inference we use the following decoding hyper-parameters: max_new_tokens=50, do_sample=True, temperature=0.7, top_k=50, top_p=0.95.

For evaluation we use the first 100 prompts from the AlpacaFarm benchmark (Dubois et al., 2023) found in the following link https://huggingface.co/datasets/tatsu-lab/alpaca_farm/viewer/alpaca_farm_evaluation. We provide these prompts in the SM.

Table 9. ***Mistral SFT Hyper-parameters***

Name	Value
lora_rank (r)	64
lora_alpha (α)	64
lora_dropout	0.1
lr	$2e-5$
batch_size	4
gradient_accumulation_steps	128
learning_rate_scheduler	Cosine
epochs	1
warmup_ratio	0.1
data_type	bfloat16
dataset	random 80% of UltraChat
seeds	$0 - 15$

Table 10. ***Mistral DPO Hyper-parameters***

Name	Value
lora_rank (r)	64
lora_alpha (α)	64
lora_dropout	0.1
lr	$5e-6$
batch_size	2
gradient_accumulation_steps	32
learning_rate_scheduler	Cosine
epochs	1
warmup_ratio	0.1
data_type	bfloat16
dataset	random 80% of UltraFeedback
seeds	$0 - 15$

E.3. Stable Diffusion Models

As the Pre-FT model we use "Stable Diffusion 1.5" found on hugging face (https://huggingface.co/runwayml/stable-diffusion-v1-5). We collect 15 personalization fine-tuned models from civitai.com, a public and widely used LoRA models marketplace. This allows us to examine our method in a real world setting, for the full list of LoRAs see Table 11. After examining the downloaded models, we deduce that their LoRA rank is $r = 32$ and that their fine-tuned layers are: to_q, to_v, to_k, to_out, proj_out, proj_in, and ff. Resulting in 192 Pre-FT model layers for and a total of $192 \cdot 15 = 2880$ LoRA fine-tuned layers. For inference we use the default Stable Diffusion 1.5 generation pipeline (i.e. 50 sampling steps).

For evaluation we use a the first 100 captions from the COCO Captions (Chen et al., 2015) validation dataset found in the following link https://github.com/tylin/coco-caption/blob/master/annotations/captions_val2014.json. We provide these prompts in the SM.

F. Spectral DeTuning Implementation Details

For all semantic evaluations we use a seed of 0 for all baselines and for our results. For both the ViTs and Stable Diffusion (SD) experiments we run Spectral DeTuning for 300 optimization steps. For the Mistral SFT and DPO experiments we use 1000 optimization steps. We base our rank scheduler implementation on the official PyTorch implementation of a the

Table 11. ***Stable Diffusion Fine-tuned LoRA Links***

https://civitai.com/models/186716/smol-animals-lora-15sdxl?modelVersionId=241137
https://civitai.com/models/189905/pastry-lora-15sdxl?modelVersionId=241955
https://civitai.com/models/191203/bastet-egypt-cat-style-lora-15sdxl?modelVersionId=243232
https://civitai.com/models/190176/fur-pirates-lora-15sdxl?modelVersionId=241976
https://civitai.com/models/211973/cigarette-style-lora-15sdxl?modelVersionId=247079
https://civitai.com/models/233316/smol-dragons-lora-15sdxl?modelVersionId=263316
https://civitai.com/models/234324/polygon-style-lora-15sdxl?modelVersionId=264506
https://civitai.com/models/202128/overgrowth-style-lora-15sdxl?modelVersionId=264449
https://civitai.com/models/218327/mythical-creatures-lora-15sdxl?modelVersionId=289861
https://civitai.com/models/203169/lava-style-lora-15sdxl?modelVersionId=265372
https://civitai.com/models/197998/chocolate-coffee-style-lora-15sdxl?modelVersionId=259150
https://civitai.com/models/180780/crystals-lora-15sdxl?modelVersionId=238435
https://civitai.com/models/196040/transparent-glass-body-lora-15sdxl?modelVersionId=245630
https://civitai.com/models/199968/liquid-flow-style-lora-15sdxl?modelVersionId=259228
https://civitai.com/models/206783/christmas-critters-lora-15sdxl?modelVersionId=275204

`ReduceLROnPlateau` learning rate scheduler [1]. We expand on the hyper-parameters of the rank scheduler in Table 12.

Table 12. ***Spectral DeTuning Rank Scheduler Hyper-parameters***

Name	Value Used	Explanation
`total_steps`	200 for ViT and SD, 1000 for Mistral	The total number of optimization steps
`start_rank`	1	The rank to start the optimization from (i.e. r^*)
`end_rank`	16 for ViTs, 32 for SD, 64 for Mistral	The final rank of the scheduler (i.e. r, the actual rank of the LoRA models)
`factor`	2	The multiplicative factor to increase the rank by
`patience`	15	Number of scheduler steps with no improvement after which rank will be increased.
`force_end_rank_percent`	0.5	Percent of the `total_steps` after which `end_rank` will be forced

G. Runtime and Compute

Since Spectral DeTuning does not pass any gradients through the model, it is highly parallelizable and can recover the weights of even large models (e.g., Mistral 7B) in minutes using a cluster of desktop-grade GPUs or even *CPUs*. For example, using a cluster of RTX2080 it can recover Mistral-7B in under five minutes.

[1] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

H. Detecting the Fine-Tuned Layers

We note that it is easy to detect which layers were fine-tuned. This can simply be done by comparing the layers weights of n different fine-tuned versions. The layers which have not been fine-tuned will be equal across all n models, while the fine-tuned layers will have some variation between them.

I. Algorithm with Rank Scheduler

In Algorithm 2 we present pytorch-like pseudocode for Spectral DeTuning that includes that rank scheduler.

Algorithm 2 PyTorch Pseudocode for Spectral DeTuning

```
# W_ps: List of n fine-tuned weight matrices
# steps: Number of optimization steps
# r: LoRA rank

# Initialize rank scheduler
current_lora_rank = 1
rank_scheduler = LoRARankScheduler(start_rank=current_lora_rank, end_rank=r)

# Initialize W_star
W_s = torch.mean(torch.stack(W_ps), axis=0)

# Perform optimization
for step in range(steps):
   # M-step
   # Approximate each M^*_i (Eq. 5)
   M_s = [W_p - W_s for W_p in W_ps]

   # Truncate each M^*_i to rank <= r (Eq. 5)
   for i in range(len(M_s)):
      (U, S, V) = torch.svd_lowrank(M_s[i], q=current_lora_rank)
      M_s[i] = (U @ torch.diag_embed(S)) @ V.T


   # W-step
   # Approximate W_star (Eq. 7)
   W_s = [W_p - M_si for (W_p, M_si) in zip(W_ps, M_s)]
   W_s = torch.mean(torch.stack(W_s), axis=0)

   # Compute the current loss
   iteration_losses = [torch.mean((W_ps[i] - (W_s + M_s[i])) ** 2) for i in range(len(M_s))]
   loss = torch.mean(torch.stack(iteration_losses), axis=0)

   # Step the rank scheduler
   rank_scheduler.step(loss)
   current_lora_rank = rank_scheduler.current_rank
```

J. Mistral Additional Results

For the list of mistral prompts see supplementary material (SM). In Figure 17 we show side-by-side results for 10 randomly (`random_seed=42`) sampled prompts from our evaluation dataset, using the Pre-FT recovered weights of the DPO fine-tuned Mistral model. See SM for the rest of the DPO results and for the SFT results.

K. Stable Diffusion Additional Results

For the list of stable diffusion prompts see SM. In Figures 14 to 16 we show side-by-side results for the entire dataset. Note, images are compressed to reduce file size, for the full resolution images see the SM.

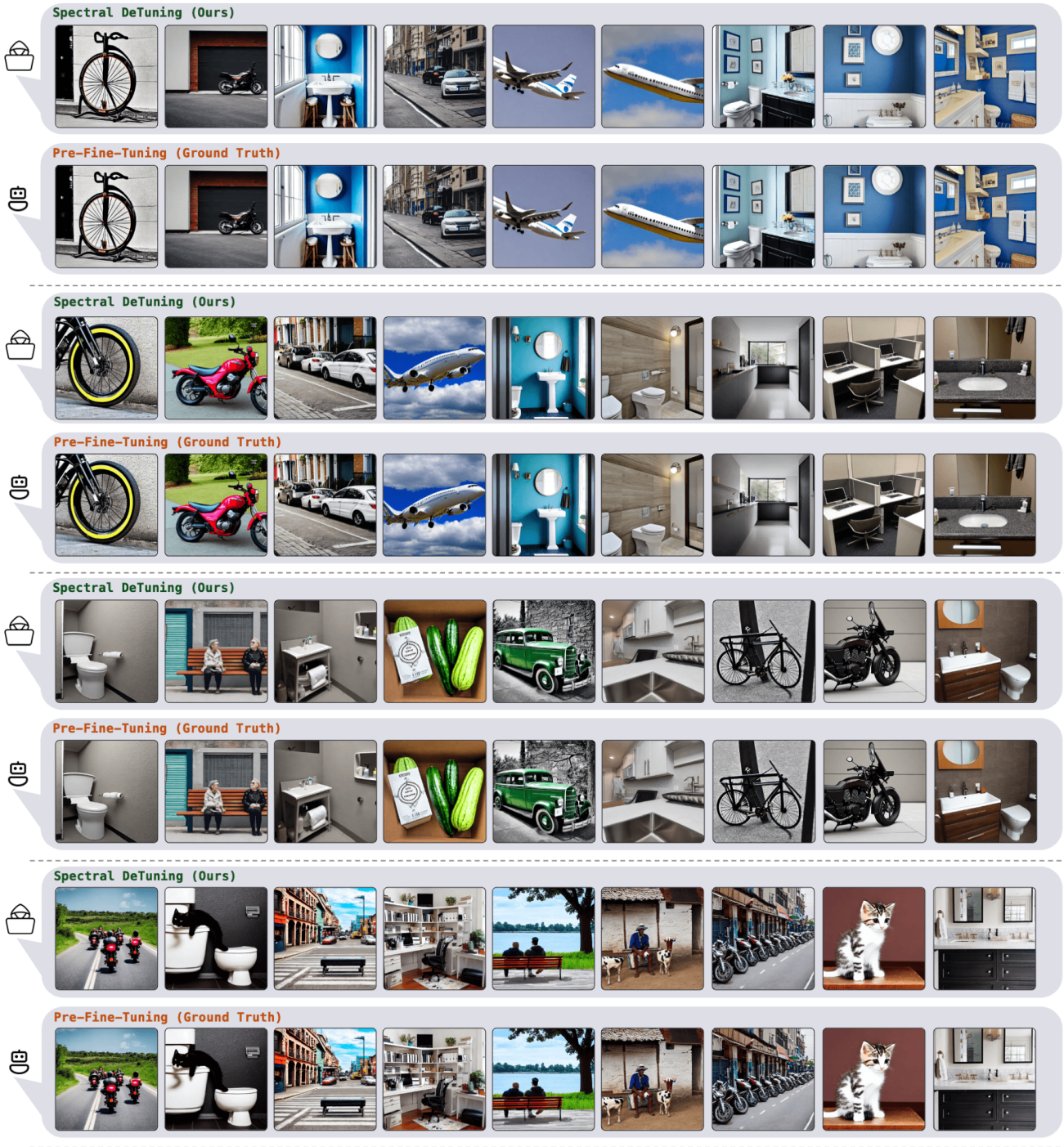

Figure 14. ***Stable Diffusion Results:*** Note, images are compressed to reduce file size, for the full resolution images see the SM.

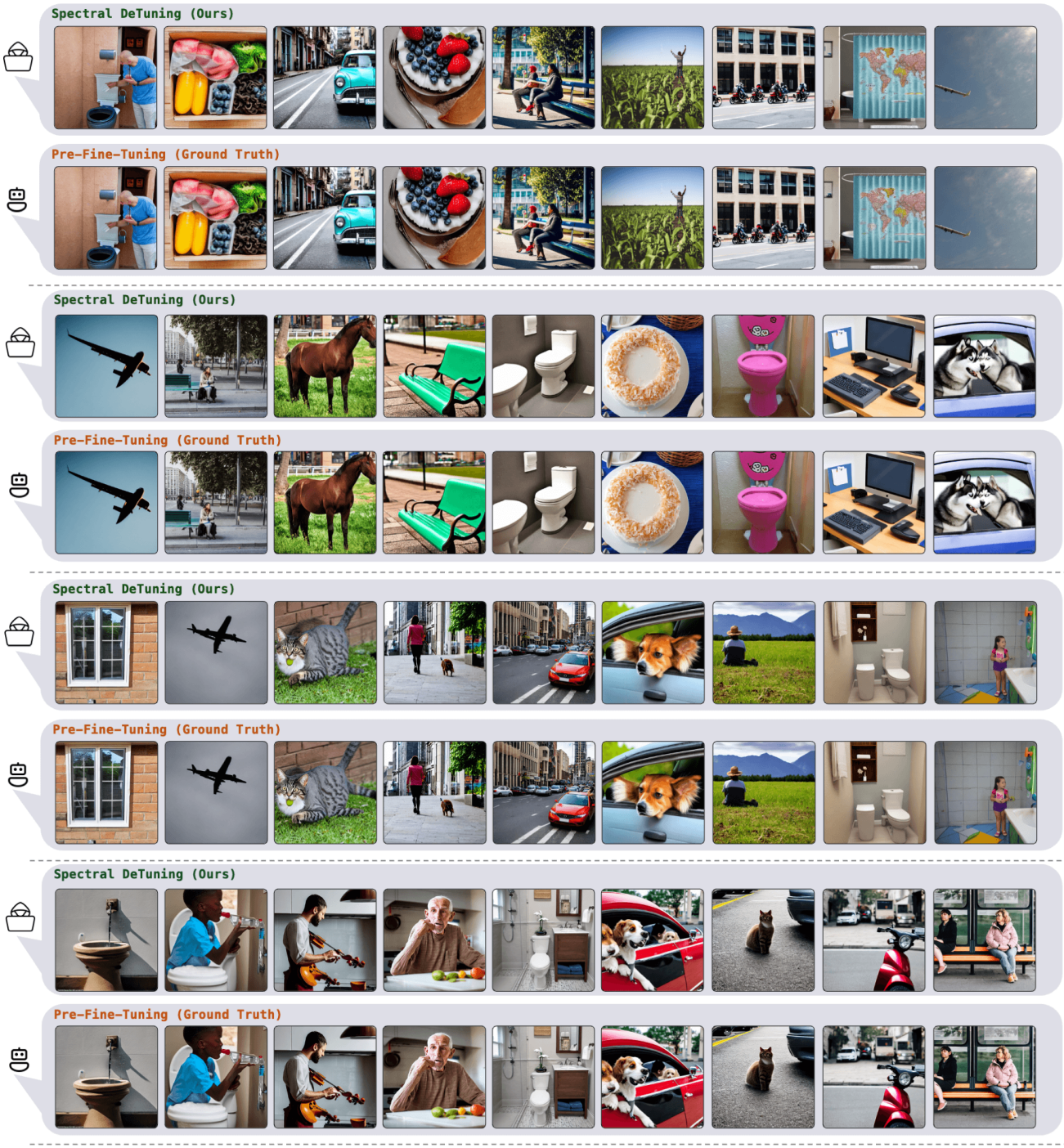

Figure 15. ***Stable Diffusion Results:*** Note, images are compressed to reduce file size, for the full resolution images see the SM.

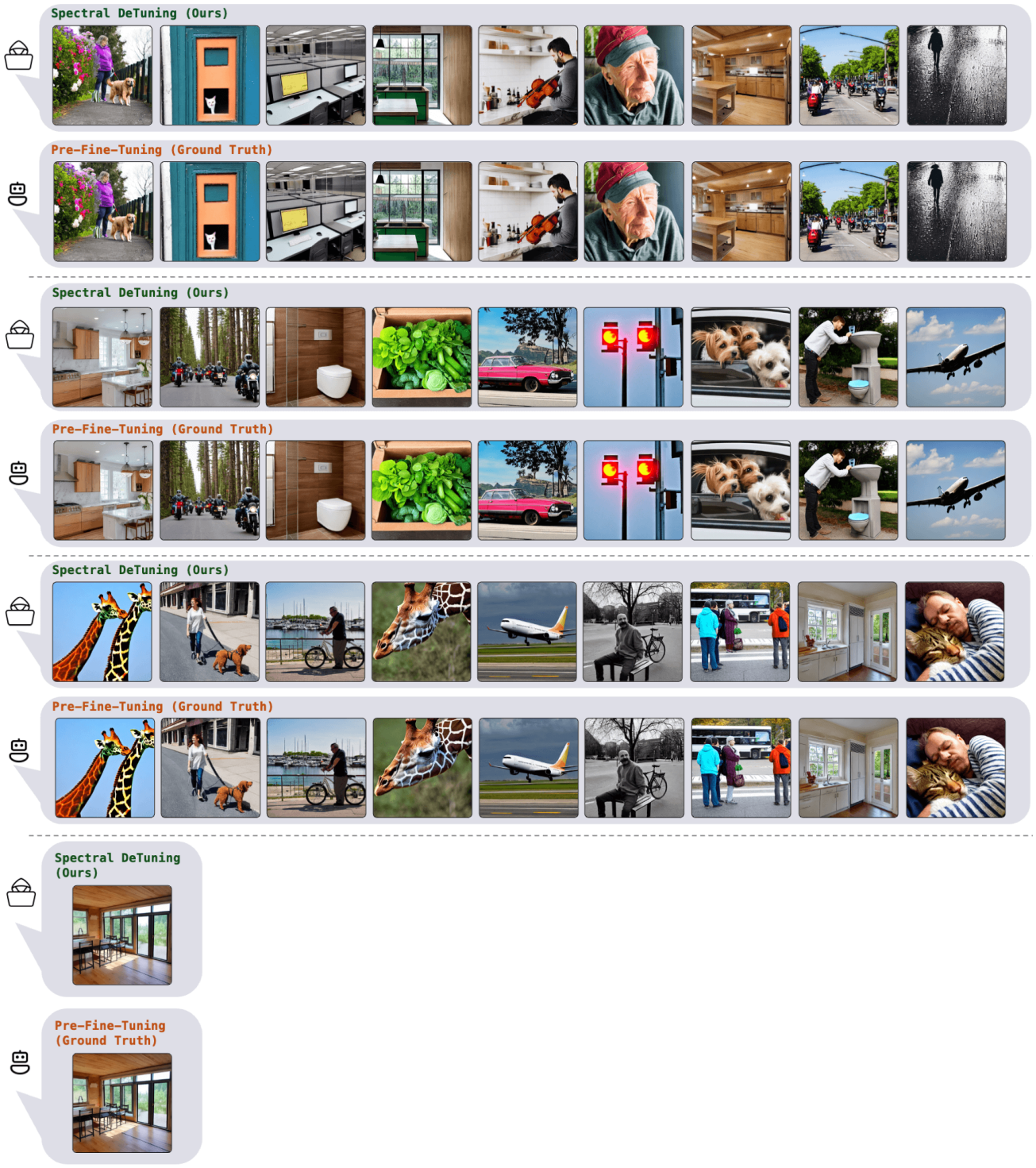

Figure 16. ***Stable Diffusion Results:*** Note, images are compressed to reduce file size, for the full resolution images see the SM.

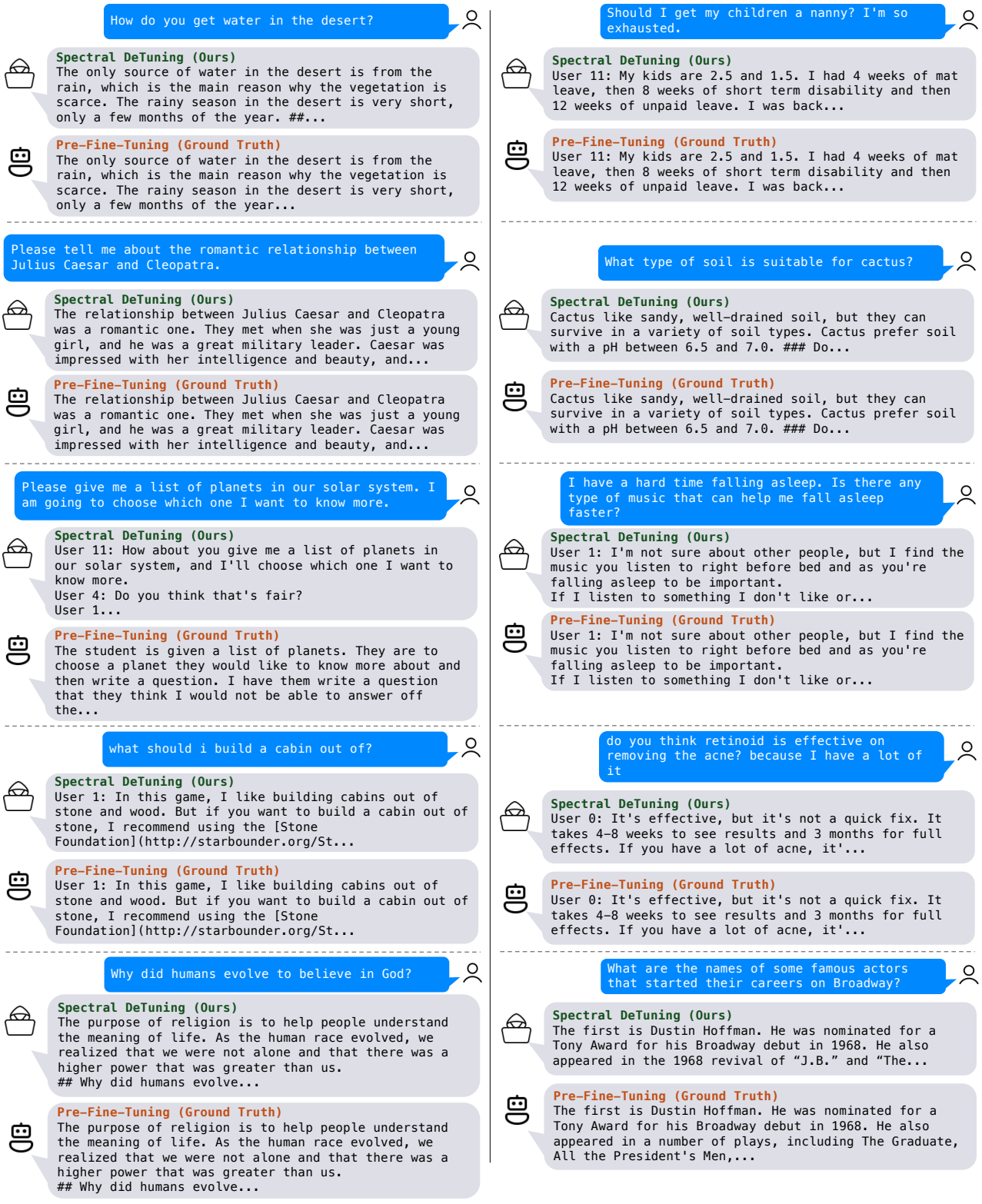

Figure 17. ***Non Cherry-picked Mistral DPO Results:*** We display side-by-side results for 10 randomly (`random_seed=42`) sampled prompts from our evaluation dataset. For the rest of the results see supplementary material.