MITIGATING FORGETTING IN LLM SUPERVISED FINE-TUNING AND PREFERENCE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training of pre-trained LLMs, which typically consists of the supervised fine-tuning (SFT) stage and the preference learning (RLHF or DPO) stage, is crucial to effective and safe LLM applications. The widely adopted approach in post-training popular open-source LLMs is to sequentially perform SFT and RLHF/DPO. However, sequential training is sub-optimal in terms of SFT and RLHF/DPO trade-off: the LLM gradually forgets about the first stage's training when undergoing the second stage's training. We theoretically prove the sub-optimality of sequential post-training. Furthermore, we propose a practical joint post-training framework that has theoretical convergence guarantees and empirically outperforms sequential post-training framework, while having similar computational cost.

1 INTRODUCTION

Recent years have witnessed the great capabilities of large language models (LLMs) trained on a large corpus of datasets (OpenAI, 2022; Dubey et al., 2024; Abdin et al., 2024). These models have been applied to a wide range of tasks including virtual assistant (OpenAI, 2022), code development (Roziere et al., 2023), and education/research (Achiam et al., 2023). Typically LLMs undergo the pre-training phase and the post-training phase. The post-training phase adapts the pre-trained LLM to specific tasks, thus is crucial to its successful applications.

The post-training phase of LLMs often has two stages (Abdin et al., 2024; Dubey et al., 2024): the Supervised Fine-Tuning (SFT) stage and the preference learning stage. Typical methods for preference learning include Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022), and Direct Preference Optimization (DPO) (Rafailov et al., 2024). Given this two-stage process, a natural approach is to performing *sequential training*, e.g., first perform DPO then SFT or vice verse. For example, the instruct variant of popular open-source models like PHI-3 (Abdin et al., 2024) or LLAMA-3 (Dubey et al., 2024) sequentially undergo SFT and DPO training. Or in other scenarios like continual learning of an aligned model, the process can be interpreted as sequentially performing DPO/RLHF followed by SFT (Tang et al., 2020; Qi et al., 2023; Fang et al., 2024).

However, sequential training of RLHF and SFT is sub-optimal in terms of the trade-off between preference learning and SFT. When the model is undergoing the second stage of training, it gradually and inevitably *forgets about* the first stage's training. In this case, we argue that even regularization like KL divergence used in RLHF/DPO cannot avoid forgetting due to the data distribution shift from the SFT dataset to the preference dataset. An illustration of the sub-optimality of sequential post-training is shown in Figure 1 (left), where we observe that sequential training leads to the increase of the DPO objective during SFT, resulting in a worse trade-off between the two objectives than the method to be introduced in this work. Similar issue has also been observed in, e.g., (Qi et al., 2023). To overcome this issue and achieve a better trade-off, a naive thought is to mix the preference objective and SFT objective by minimizing their linear scalarization. However, the naive mixing method is computationally inefficient in practice, since the optimization objectives are different with different formats of data. The increased computational complexity can be observed in Figure 1 (right), where mixing significantly increases the cost. The cost is prohibitively high in LLM training due to the size of the model. Therefore, in this work, we aim to answer the following question:

Can we design a post-training framework that achieves better trade-off than the sequential training method, while having reduced computational cost than naive mixing?

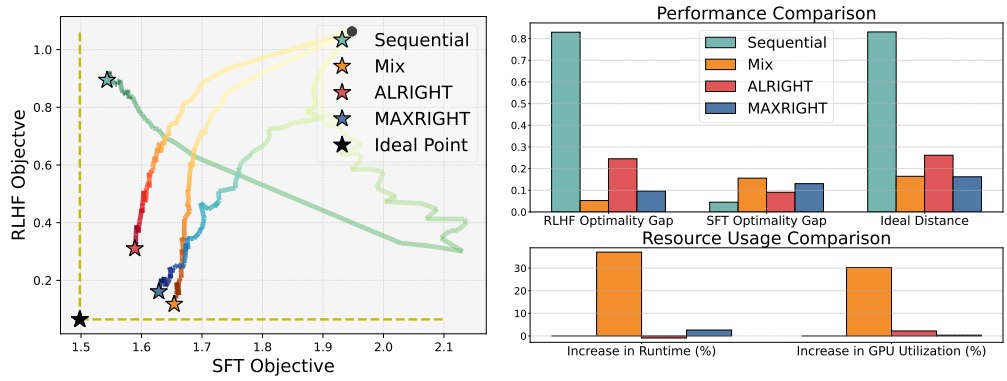

Figure 1: **Efficient Trade-off in RLHF and SFT Optimization.** **Sequential** optimization, commonly used to align and fine-tune pre-trained models, often biases the model towards the last objective it was optimized on, as illustrated by the optimization trajectories in the objective space (left) and the performance comparison (top right, lower the better). In contrast, simultaneous optimization of a **Mix** of RLHF and SFT objectives achieves a more balanced performance but requires significantly more resources (bottom right, lower the better). We propose **ALRIGHT** and **MAXRIGHT** methods for simultaneous RLHF and SFT optimization, offering an improved trade-off with minimal extra cost.

Our contributions. To this end, we propose a joint SFT and DPO training framework with the ALRIGHT and MAXRIGHT variants. The contributions of this work can be summarized as follows:

C1) Insights into the forgetting issue of two-stage sequential training. We provide theoretical results on the forgetting issue of sequential method, which is further supported by empirical evidence. Specifically, we prove that sequentially doing DPO and SFT can have a non-diminishing optimality gap. To our best knowledge, this is the first theoretical result on the suboptimality of sequentially learning the SFT and DPO objectives. We further conduct experiments on LLAMA-3-8B and PYTHIA-1B, empirical results support our claims.

C2) Principled post-training methods with almost no extra cost. We propose post-training algorithms with theoretical guarantees, which outperform the sequential method while enjoying lower computational cost than the mixing method. Specifically, we propose 1) ALternating supeRvised fIne-tuninG and Human preference alignmenT (ALRIGHT) method, which provably converges to any desired trade-off between DPO and SFT objectives; and 2) MAXimum supeRvised fIne-tuninG and Human preference alignmenT (MAXRIGHT) method, which adaptively alternates between optimizing RLHF and SFT objectives.

C3) Strong empirical performance on standard benchmarks. Using the LLAMA3-8B model, our methods outperform the sequential approach by up to 3% on the MMLU (1-shot) benchmark (Hendrycks et al., 2020) and achieve up to a 31% increase in win rate on the RLHF dataset (evaluated by GPT-4-TURBO), with minimal additional computation.

Technical challenges. The main technical challenge lies in theoretically proving the issue of forgetting when optimizing SFT and DPO losses (two log-likelihood functions). Existing lower bounds for continual learning in *non-LLM* settings using SGD rely on quadratic objectives (Ding et al., 2024), where iterate updates are more tractable due to the linear structure of the gradient.

However, the problem we address involves negative log-likelihood objectives, whose gradients are non-linear with respect to the parameters. This non-linearity complicates the analysis, requiring careful construction of an example that is guaranteed to exhibit undesirable performance. Additionally, we must analyze the behavior of the updated parameters throughout both stages of optimization and derive conditions that relate this behavior to the lower bound of the performance. We successfully overcome these challenges, and the details of this analysis are provided in Appendix A.2.

2 PRELIMINARIES

In this section, we formally introduce the notations and problem setup for DPO and SFT.

Model. We denote the LLM parameter to be optimized for either RLHF or SFT by θ, and we use $\pi_\theta(y \mid x)$ to denote the LLM that generates an output y given an input x for an SFT or RLHF task.

DPO. We consider using DPO (Rafailov et al., 2024) to align θ with a preference dataset, given by $\mathcal{D}_{\text{DPO}} = \{x^{(i)}, y_w^{(i)}, y_\ell^{(i)}\}_{i=1}^{N_1}$, where N_1 is the number of data, $x^{(i)}$ are the inputs for the LLM, $y_w^{(i)}$ and $y_\ell^{(i)}$ are the chosen (preferred) and rejected (dispreferred) responses to $x^{(i)}$, respectively, for all $i \in \{1, \ldots, N_1\}$. The DPO objective is given by

$$f_{\text{DPO}}(\theta; \mathcal{D}_{\text{DPO}}, \pi_{\text{ref}}, \beta) := -\tfrac{1}{N_1} \textstyle\sum_{x, y_w, y_\ell \in \mathcal{D}_{\text{DPO}}} \left[\log \left(\sigma \left(\beta \log \left(\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} \right) - \beta \log \left(\frac{\pi_\theta(y_\ell \mid x)}{\pi_{\text{ref}}(y_\ell \mid x)} \right) \right) \right) \right], \tag{1}$$

where σ is the sigmoid function, π_{ref} is a given reference model, and β is a regularization constant that penalize the objective when $\pi_\theta(y \mid x)$ is diverging too much from $\pi_{\text{ref}}(y \mid x)$ on $x \sim \mathcal{D}_{\text{DPO}}$. In sequential training, when DPO is performed before SFT, we use the model trained on the chosen responses in $\mathcal{D}_{\text{DPO}}$ as π_{ref}. When SFT is performed before DPO, the model obtained after the SFT phase is used as the π_{ref}. Given a data point (x, y_w, y_ℓ), the gradient estimate of f_{DPO} is given as

$$g_{\text{DPO}}(\theta; x, y_w, y_\ell, \pi_{\text{ref}}, \beta) := -\left(1 - \sigma(h_\beta(\theta; x, y_w, y_\ell, \pi_{\text{ref}}))\right) \nabla_\theta h_\beta(\theta; x, y_w, y_\ell, \pi_{\text{ref}}), \tag{2}$$

where

$$h_\beta(\theta; x, y_w, y_\ell, \pi_{\text{ref}}) := \beta \log \left(\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} \right) - \beta \log \left(\frac{\pi_\theta(y_\ell \mid x)}{\pi_{\text{ref}}(y_\ell \mid x)} \right). \tag{3}$$

For brevity, in the rest of the paper we will use $f_{\text{DPO}}(\theta)$ for $f_{\text{DPO}}(\theta; \mathcal{D}_{\text{DPO}}, \pi_{\text{ref}}, \beta)$, $g_{\text{DPO}}(\theta; x, y_w, y_\ell)$ for $g_{\text{DPO}}(\theta; x, y_w, y_\ell, \pi_{\text{ref}}, \beta)$, and $h_\beta(\theta; x, y_w, y_\ell)$ for $h_\beta(\theta; x, y_w, y_\ell, \pi_{\text{ref}})$. Note that $\mathbb{E}_{x, y_w, y_\ell \sim \mathcal{D}_{\text{DPO}}} \left[g_{\text{DPO}}(\theta; x, y_w, y_\ell) \right] = \nabla f_{\text{DPO}}(\theta)$.

SFT. We denote the dataset used for SFT as $\mathcal{D}_{\text{SFT}} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N_2}$, where N_2 is the number of data points. The SFT dataset consists of input $x^{(i)}$ and corresponding target outputs $y^{(i)}$ for all $i \in \{1, \ldots, N_2\}$. The objective used for fine-tuning θ for $\mathcal{D}_{\text{SFT}}$ can be given as

$$f_{\text{SFT}}(\theta; \mathcal{D}_{\text{SFT}}) := -\tfrac{1}{N_2} \textstyle\sum_{x, y \in \mathcal{D}_{\text{SFT}}} \log(\pi_\theta(y \mid x)). \tag{4}$$

Given a data point (x, y), the gradient estimate for the objective f_{SFT} is given as

$$g_{\text{SFT}}(\theta; x, y) := -\nabla_\theta \pi_\theta(y \mid x) / \pi_\theta(y \mid x). \tag{5}$$

From this point, we will use $f_{\text{SFT}}(\theta)$ for $f_{\text{SFT}}(\theta; \mathcal{D}_{\text{SFT}})$. Note that $\mathbb{E}_{x, y \sim \mathcal{D}_{\text{SFT}}} \left[g_{\text{SFT}}(\theta; x, y) \right] = \nabla f_{\text{SFT}}(\theta)$.

Performance metric and trade-off. In this work we investigate different methods for their performance on both DPO and SFT tasks, simultaneously. Thus, to evaluate the performance of a model θ on f_{DPO} and f_{SFT}, we define the optimality gap of a mixture of objectives as

$$G_{\text{Mix},\lambda}(\theta) := f_{\text{Mix},\lambda}(\theta) - f^*_{\text{Mix},\lambda}, \tag{6}$$

where $\lambda \in [0, 1]$, $f_{\text{Mix},\lambda}(\theta) := \lambda f_{\text{DPO}}(\theta) + (1 - \lambda) f_{\text{SFT}}(\theta)$, and $f^*_{\text{Mix},\lambda} = \min_{\theta \in \Theta} f_{\text{Mix},\lambda}(\theta)$. Here λ defines a trade-off between the DPO and SFT objective: a larger λ results in more emphasis on the DPO performance compared to SFT. We say a model parameter θ achieves optimal trade-off defined by λ when $G_{\text{Mix},\lambda}(\theta) = 0$. We chose this metric because, as established in multi-objective optimization literature (Miettinen, 1999), the optimizer of $G_{\text{Mix},\lambda}(\theta)$ for any $\lambda \in [0, 1]$ will be 'Pareto optimal'. This means that no other solution can optimize both objectives simultaneously, and the solution can be viewed as one of the optimal trade-off points for the problem of optimizing f_{DPO} and f_{SFT}. Additionally, $G_{\text{Mix},\lambda}(\theta)$ is differentiable when both f_{DPO} and f_{SFT} are differentiable, which facilitates the theoretical analysis of gradient-based methods.

3 Why Sequential DPO and SFT is Suboptimal?

This section studies the sequential DPO and SFT method commonly used in the continual training of aligned LLMs (see, e.g., Tang et al. (2020); Qi et al. (2023); Fang et al. (2024)). We give insights on why such a sequential training framework is suboptimal in terms of DPO and SFT trade-offs.

3.1 Sequential Training Algorithm and Its Suboptimality

Following Rafailov et al. (2024), we first obtain a reference model π_{ref} by performing SFT on the target outputs in the preference dataset $\mathcal{D}_{\text{DPO}}$. Given π_{ref}, we iteratively perform the DPO update as

$$\theta^1_{t+1} = \Pi_\Theta \left(\theta^1_t - \alpha_{1,t} g_{\text{DPO}}(\theta^1_t; x^t, y^t_w, y^t_\ell) \right), \quad \text{for } t = 1, 2, \ldots, T_{\text{DPO}} - 1 \tag{7}$$

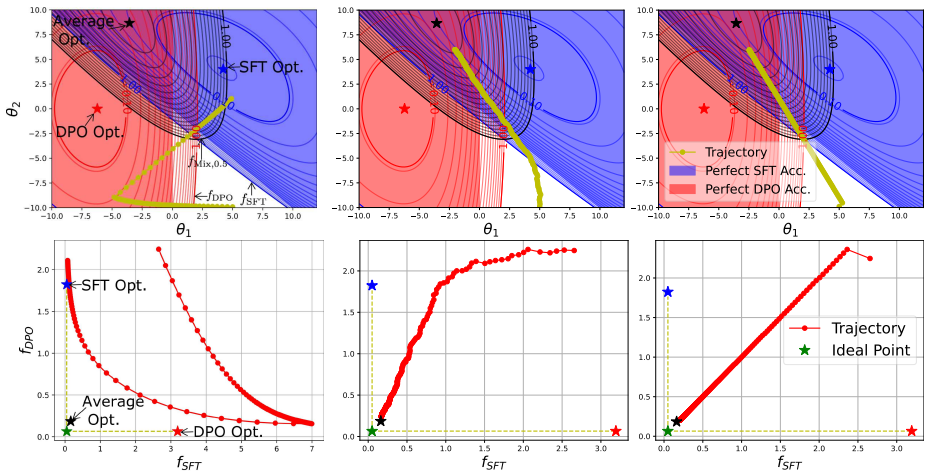

Figure 2: Consider a two-dimensional model and one data each for DPO and SFT optimization setting. **Sequential DPO and SFT (Left):** The model oscillates between the optima of DPO and SFT losses in weight space, resulting in a final trade-off that is significantly distant from the ideal point in loss space, where both DPO and SFT loss values are optimal. **ALRIGHT (Middle) / MAXRIGHT (Right):** The model directly navigates towards a point in weight space that is reasonably optimal for both DPO and SFT objectives (average optimum), achieving a final trade-off of DPO and SFT losses much closer to the ideal point compared to sequential DPO and SFT.

where $\alpha_{1,t}$ is the step size, $x^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$, g_{DPO} is defined in (2), and T_{DPO} is the max iteration for DPO training. Given the aligned model parameter $\theta^1_{T_{\text{DPO}}}$, we next perform SFT updates as follows:

$$\theta^2_{t+1} = \Pi_\Theta\left(\theta^2_t - \alpha_{2,t} g_{\text{SFT}}(\theta^2_t; x^t, y^t)\right), \quad \text{for } t = 1, 2, \ldots, T_{\text{SFT}} - 1 \tag{8}$$

where θ^2_1 is initialized as $\theta^1_{T_{\text{DPO}}}$, $x^t, y^t \sim \mathcal{D}_{\text{SFT}}$, g_{SFT} is defined in (5), and T_{SFT} is the max iteration for SFT training. The whole process is summarized in Algorithm 1. We next study why the sequential training framework is suboptimal.

A toy illustration of suboptimality. At a given phase of Algorithm 1, the algorithm only focuses on optimizing one objective (either f_{DPO} or f_{SFT}) and ignores the other. This results in the model oscillating between the optimums of two objectives, without converging to a point that is 'reasonably optimal' for both objectives f_{DPO} and f_{SFT}. We first illustrate this on a toy example (see example details in Appendix D.1). The results are depicted in Figure 2. For the weight space trajectory (Figure 2 upper-left), although there is a region that is optimal for both DPO and SFT, the sequential DPO and SFT method fails to reach this region due to its focus on one objective at a given phase. Furthermore, from the loss space trajectory (Figure 2 lower-left), the model oscillates between extreme trade-offs for DPO and SFT, and ends up at a point far away from the ideal point.

Algorithm 1 Sequential DPO and SFT

Input $\mathcal{D}_{\text{DPO}}$, $\mathcal{D}_{\text{SFT}}$, $\{\alpha_{1,t}\}_{t=1}^{T_{\text{DPO}}}$, $\{\alpha_{2,t}\}_{t=1}^{T_{\text{SFT}}}$
Phase 1: Optimize for f_{DPO}
Initialize $\theta^1_1 := \theta_1 \in \Theta$
for $t = 1, \ldots, T_{\text{DPO}} - 1$ **do**
  Sample $x^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$
  Update $\theta^1_{t+1} = \Pi_\Theta\left(\theta^1_t - \alpha_{1,t} g_{\text{DPO}}(\theta^1_t; x^t, y_w^t, y_\ell^t)\right)$
end for
Set $\hat{\theta}_{\text{DPO}} := \theta^1_{T_{\text{DPO}}}$
Phase 2: Optimize for f_{SFT}
Initialize $\theta^2_1 := \hat{\theta}_{\text{DPO}}$
for $t = 1, \ldots, T_{\text{SFT}} - 1$ **do**
  Sample $x^t, y^t \sim \mathcal{D}_{\text{SFT}}$
  Update $\theta^2_{t+1} = \Pi_\Theta\left(\theta^2_t - \alpha_{2,t} g_{\text{SFT}}(\theta^2_t; x^t, y^t)\right)$
end for
Output $\hat{\theta}_{\text{Seq}} := \theta^2_{T_{\text{SFT}}}$

3.2 Theoretical Analysis of Suboptimality in Sequential Method

In this section, we provide a theoretical result on the suboptimal trade-offs between DPO and SFT in sequential training. We view the LLM as a policy π_θ that is characterized by a softmax:

$$\pi_\theta(y \mid x) := \frac{\exp(\theta^\top \phi_{y,x})}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top \phi_{y',x})},$$

where $\phi_{y,x}$ is a feature vector corresponding to the input x and the target output y. Furthermore, reference policy π_{ref} is similarly parameterized by a fixed parameter θ_{ref}.

Remark 3.1. The softmax characterization is also used in previous theoretical works on RLHF; see, e.g., Zhu et al. (2023a). When the trainable parameter is the output projection weights, the LLM is fully characterized by the softmax. In other scenarios like performing LoRA Hu et al. (2021) on attention matrices or full-parameter training, we believe this characterization still provides valuable insights and our result will be verified empirically later in the experimental section.

We make the following mild assumption on the features.

Assumption 3.1 (Bounded feature)**.** For all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, there exists $\Phi > 0$ such that $\|\phi_{y,x}\| \leq \Phi$.

We can then have the following result for the sub-optimality of the output of Algorithm 1 to the optimum of some combination of functions f_{DPO} and f_{SFT} in terms of $G_{\text{Mix},\lambda}$.

Theorem 3.1 (Lower bound for sequential method performance)**.** *Consider Algorithm 1 with $T_{DPO} = T_{SFT} = T$ under Assumption 3.1. Then there exists data $\mathcal{D}_{DPO}$ and $\mathcal{D}_{SFT}$ such that given any $\lambda \in (0,1)$, Algorithm 1 with any sufficiently large T has non-diminishing performance gap:*

$$\mathbb{E}\Big[\lambda f_{DPO}(\hat{\theta}_{Seq}) + (1-\lambda) f_{SFT}(\hat{\theta}_{Seq}) - \min_{\theta \in \Theta} \big(\lambda f_{DPO}(\theta) + (1-\lambda) f_{SFT}(\theta)\big)\Big] = \Omega(1), \tag{9}$$

where the expectation $\mathbb{E}[\,\cdot\,]$ is taken over the randomness of Algorithm 1.

The above result suggests that there exists DPO and SFT optimization problems such that given any trade-off between DPO and SFT defined by $\lambda \in (0,1)$, the sequential method suffers from constant suboptimality gap, even when optimized for a large number of iterations. The reason for the constant suboptimality gap is the sequential method described in Algorithm 1 suffers from forgetting, and cannot appropriately optimize both the DPO and SFT objectives. In the next section, we explore alternatives to the sequential method that can resolve this issue.

4 Proposed Alternating Training Methods

In this section we introduce new algorithms with theoretically convergence guarantees, which also outperforms sequential DPO and SFT empirically.

4.1 ALRIGHT for joint DPO and SFT

The main disadvantage of using Algorithm 1 for DPO and SFT optimization is that at a given phase of the algorithm, the model is updated with respective to only one objective. In contrast, it is computationally intensive, if not prohibitive, to optimize a linear combination of both DPO and SFT objectives. This is because, although the objectives share a single parameter, constructing two computational graphs (one per objective) in standard machine learning libraries requires significant additional memory, particularly for LLMs. To alleviate these problems, we propose to alternate between optimizing for DPO and SFT objectives, based on a given preference for each objective. For this purpose, we can define a modified objective

$$f_{\text{Alt},\lambda}(\theta; i) = \mathbb{I}_{i=1} f_{\text{DPO}}(\theta) + \mathbb{I}_{i=0} f_{\text{SFT}}(\theta), \tag{10}$$

where $i \sim \text{Bern}(\lambda)$, $\text{Bern}(\lambda)$ is the Bernoulli distribution parameterized by $\lambda \in [0,1]$, and $\mathbb{I}_A$ is the indicator function of event A. Hence, the objective in (10) behaves as $f_{\text{Mix},\lambda}$ in expectation, i.e.

$$\mathbb{E}_{i \sim \text{Bern}(\lambda)}\left[f_{\text{Alt},\lambda}(\theta; i)\right] = \lambda f_{\text{DPO}}(\theta) + (1-\lambda) f_{\text{SFT}}(\theta) = f_{\text{Mix},\lambda}(\theta). \tag{11}$$

For optimizing $\mathbb{E}_{i \sim \text{Bern}(\lambda)}\left[f_{\text{Alt},\lambda}(\theta; i)\right]$, we first sample $i_t \sim \text{Bern}(\lambda)$ per iteration, which determines the objective to be updated. Specifically, if $i_t = 1$, we update θ with respect to DPO objective as

$$\theta_{t+1} = \Pi_\Theta\left(\theta_t - \alpha_t g_{\text{DPO}}(\theta_t; x^t, y_w^t, y_\ell^t)\right), \tag{12}$$

where $x^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$, and α_t is the learning rate. Similarly, if $i_t = 0$, we update θ with respect to SFT objective as

$$\theta_{t+1} = \Pi_\Theta\left(\theta_t - \alpha_t g_{\text{SFT}}(\theta_t; x^t, y^t)\right), \tag{13}$$

Algorithm 2 ALRIGHT

Input $\mathcal{D}_{\text{DPO}}$, $\mathcal{D}_{\text{SFT}}$, $\{\alpha_t\}_{t=1}^{T}$, $\lambda \in [0,1]$
Initialize $\theta_1 \in \Theta$
for $t = 1, \ldots, T-1$ **do**
  Sample $i_t \sim \text{Bern}(\lambda)$
  if $i_t = 1$ **then**
    Sample $x^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$
    $\theta_{t+1} = \Pi_\Theta(\theta_t - \alpha_t g_{\text{DPO}}(\theta_t; x^t, y_w^t, y_\ell^t))$
  else
    Sample $x^t, y^t \sim \mathcal{D}_{\text{SFT}}$
    $\theta_{t+1} = \Pi_\Theta\left(\theta_t - \alpha_t g_{\text{SFT}}(\theta_t; x^t, y^t)\right)$
  end if
end for
Output $\hat{\theta}_{\text{AL}} := \theta_T$

Algorithm 3 MAXRIGHT

Input $\mathcal{D}_{\text{DPO}}, \mathcal{D}_{\text{SFT}}, \{\alpha_t\}, \lambda \in [0,1]$. Initialize $\theta_1 \in \Theta$
for $t = 1, \ldots, T-1$ **do**
  Sample $x_1^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$ and $x_2^t, y^t \sim \mathcal{D}_{\text{SFT}}$
  Evaluate
    $\bar{f}_{1,\lambda}(\theta_t) := \lambda\left(f_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t) - f_{\text{DPO}}^*\right)$
    $\bar{f}_{2,\lambda}(\theta_t) := (1-\lambda)\left(f_{\text{SFT}}(\theta_t; x_2^t, y^t) - f_{\text{SFT}}^*\right)$
  if $\bar{f}_{1,\lambda}(\theta_t) \geq \bar{f}_{2,\lambda}(\theta_t)$ **then**
    $\theta_{t+1} = \Pi_\Theta\left(\theta_t - \alpha_t g_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t)\right)$
  else
    $\theta_{t+1} = \Pi_\Theta\left(\theta_t - \alpha_t g_{\text{SFT}}(\theta_t; x_2^t, y^t)\right)$
  end if
end for
Output $\hat{\theta}_{\text{MAX}} := \theta_T$

where $x^t, y^t \sim \mathcal{D}_{\text{SFT}}$. This process is summarized in Algorithm 2. Unlike the sequential method that focuses on optimizing one objective at a time, the ALRIGHT approach integrates both objectives simultaneously, allowing the model to balance alignment and fine-tuning performance. In Figure 2 (Middle), we can see how this alternating navigates the model to a point where the trade-off between DPO and SFT is significantly better than the sequential approach.

Next, we provide the convergence guarantee of Algorithm 2 to a given DPO-SFT trade-off.

Theorem 4.1 (Upper bound for alternating method performance)**.** *Consider Algorithm 2 with $\alpha_t = \alpha_0/\sqrt{T}$ for all $t \in \{1, \ldots, T\}$ and $\alpha_0 > 0$. Then, under Assumption 3.1, for any $\lambda \in [0,1]$, we have*

$$\mathbb{E}\Big[\lambda f_{DPO}(\hat{\theta}_{AL}) + (1-\lambda) f_{SFT}(\hat{\theta}_{AL}) - \min_{\theta\in\Theta}\left(\lambda f_{DPO}(\theta) + (1-\lambda) f_{SFT}(\theta)\right)\Big] = \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right). \tag{14}$$

Remark 4.1. The above result implies that the performance metric diminishes with increasing T, thus we can achieve an arbitrary trade-off between DPO and SFT defined by λ up to arbitrary optimality, by increasing the number of iterations T. This is in contrast to the lower bound result in Theorem 3.1 established for sequential training: there exist data sets such that the sequential method never approaches optimal trade-off, even when trained for larger number of iterations.

While ALRIGHT offers theoretical convergence guarantees for any arbitrary trade-off in expectation, the alternation between optimizing DPO and SFT objectives occurs randomly based on a predetermined probability, which may introduce additional noise in the updates. This raises the natural question: Can we design a performance-aware, *adaptive alternating* optimization method with minimal additional computational resource usage compared to ALRIGHT? In the next section, we will propose an alternative post-training method that adaptively selects the objective to optimize.

4.2 MAXRIGHT FOR JOINT DPO AND SFT

In this section, we introduce a method that can adaptively choose which objective to optimize based on the current performance of θ, which can lead to faster convergence to a point that can perform well for both DPO and SFT objectives. When determining which objective to optimize at a given iteration, we first compare the current model's performance on f_{DPO} and f_{SFT}. Define the maximum (weighted) sub-optimality gap as

$$f_{\text{Max},\lambda}(\theta) = \max\left(\lambda(f_{\text{DPO}}(\theta) - f_{\text{DPO}}^*), (1-\lambda)(f_{\text{SFT}}(\theta) - f_{\text{SFT}}^*)\right), \tag{15}$$

where $f_{\text{DPO}}^* = \min_{\theta\in\Theta} f_{\text{DPO}}(\theta)$ (similarly for $f_{\text{SFT}}(\theta)$), and $\lambda \in [0,1]$. The idea is to optimize this maximum sub-optimality to reach a balance between the two λ-scaled objectives. Define the index of the objective with maximum (weighted) sub-optimality gap as $i_t = \text{argmax}_i\, \bar{f}_i(\theta_t)$, where

$$\bar{f}_{1,\lambda}(\theta_t) := \lambda\left(f_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t) - f_{\text{DPO}}^*\right), \text{ and} \tag{16}$$

$$\bar{f}_{2,\lambda}(\theta_t) := (1-\lambda)\left(f_{\text{SFT}}(\theta_t; x_2^t, y^t) - f_{\text{SFT}}^*\right), \tag{17}$$

where $x_1^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$ and $x_2^t, y^t \sim \mathcal{D}_{\text{SFT}}$. Accordingly, we can update θ with respect to DPO objective using update (12) when $i_t = 1$ (or equivalently, when $\bar{f}_{1,\lambda}(\theta_t) \geq \bar{f}_{2,\lambda}(\theta_t)$), and update θ

with respect to DPO objective using update (13) otherwise. This process is summarized in Algorithm 3. We can see in the toy illustration (Figure 2 Right), that MAXRIGHT can converge closer to the ideal point more directly compared to the alternating method, due to its ability to adaptively choose the worse performing objective to optimize.

Remark 4.2. While we do not provide any theoretical convergence guarantees for Algorithm 3, it is a well-known fact in multi-objective optimization literature (Miettinen, 1999) that under some assumptions on the problem setup, the solution of problem (15) for any $\lambda \in [0, 1]$ is guaranteed to be Pareto optimal (i.e. no other solution can further optimize both the objectives simultaneously). Furthermore, unlike the ALRIGHT algorithm, MAXRIGHT requires prior knowledge or computation of f^*_{DPO} and f^*_{SFT}, adding to its overall computational budget. However, this computation is performed once and can be reused across different implementations with varying λ. Details on approximating f^*_{DPO} and f^*_{SFT} are provided in Appendix D.

Even though MAXRIGHT allows one to compute the index needed for selecting the objective with a maximum (weighted) sub-optimality gap, in practice evaluating both objectives can be memory intensive, and only one objective is updated at a given iteration. To alleviate this issue, we propose to do simultaneous evaluations only every k steps. We call a time step that simultaneous evaluation is done as a 'max evaluation step'. At such time step $t = t_0$, we compute i_{t_0}, and update the corresponding objective as in Algorithm 3. After the update, we store the computed (weighted) sub-optimality gap as $\bar{f}^{\text{stale},t_0}_{1,\lambda} = \bar{f}_{1,\lambda}(\theta_{t_0})$ and $\bar{f}^{\text{stale},t_0}_{2,\lambda} = \bar{f}_{2,\lambda}(\theta_{t_0})$. Then, for every iteration before the next max evaluation step $t_0 + k$, we choose the index of the objective to be optimized as

$$i_{t_0+k'} = \operatorname*{argmax}_i \bar{f}^{\text{stale},t_0}_{i,\lambda}, \tag{18}$$

where $k' < k$. Once the index is computed, we update the corresponding objective following (12) or (13), and update the stale (weighted) sub-optimality gap as

$$\bar{f}^{\text{stale},t_0}_{i,\lambda} = \bar{f}_{i,\lambda}(\theta_{t_0+k'}), \quad \text{if } i_{t_0+k'} = i, \tag{19}$$

where $i \in \{1, 2\}$. This process is summarized in Appendix B. With this modification, we can match the evaluation and gradient computation complexity of Algorithm 2 in most iterations, at the expense of degraded accuracy in choosing i_t.

5 Related Work

RLHF. The most fundamental form of RLHF was introduced by Christiano et al. (2017) and has been successfully used for aligning LLMs in many works such as OpenAI (2022); Ouyang et al. (2022); Bai et al. (2022a;b); Sun et al. (2024). There have been many works on RLHF for LLM alignment, including the more efficient direct preference optimization (Rafailov et al., 2024; Xu et al., 2024; Lee et al., 2024; Zhong et al., 2024), reference model free preference alignment (Meng et al., 2024; Hong et al., 2024b), generalized RLHF (Azar et al., 2024; Munos et al., 2023), safe RLHF (Dai et al., 2023), group preference learning (Zhao et al., 2023; Chakraborty et al., 2024), and theory or understanding of RLHF (Zhu et al., 2023a; Shen et al., 2024; Xiong et al., 2024; Wang et al., 2023; Kirk et al., 2023). In this work, we consider DPO (Rafailov et al., 2024) which has been used in training many popular open-source LLMs (Abdin et al., 2024; Dubey et al., 2024).

SFT. Another important step before using a pre-trained LLM in downstream applications is SFT (Howard & Ruder, 2018; Devlin, 2018; Wei et al., 2021; Zhu et al., 2023b; Zhang et al., 2023b). In recent years, there have been a large body of work on efficient LLM SFT; see, e.g., zeroth-order fine-tuning (Malladi et al., 2023a), quantized fine-tuning (Kim et al., 2024; Li et al., 2023b), parameter-efficient fine-tuning (Chen et al., 2023a; Zhang et al., 2023a; Shi & Lipani, 2023; Chen et al., 2023b; Nikdan et al., 2024), truthful fine-tuning (Tian et al., 2023), robust fine-tuning (Tian et al., 2024), SFT with data selection (Lu et al., 2023; Kang et al., 2024; Zhao et al., 2024), self-play fine-tuning (Chen et al., 2024) and understanding of LLM fine-tuning (Malladi et al., 2023b).

Sequential RLHF and SFT Issues. Given the importance of both RLHF and SFT in LLM post-training, they are implemented as a sequential recipe (one after the other) on a pre-trained LLM. However, recent studies show that this either hinders the alignment performance (Qi et al., 2023), or fine-tuning performance (Ouyang et al., 2022), depending on the order of applying RLHF and SFT.

Reconciling RLHF and SFT. The challenges of sequential RLHF and SFT have prompted several approaches to improve the trade-off between alignment and fine-tuning. Hong et al. (2024a) introduce

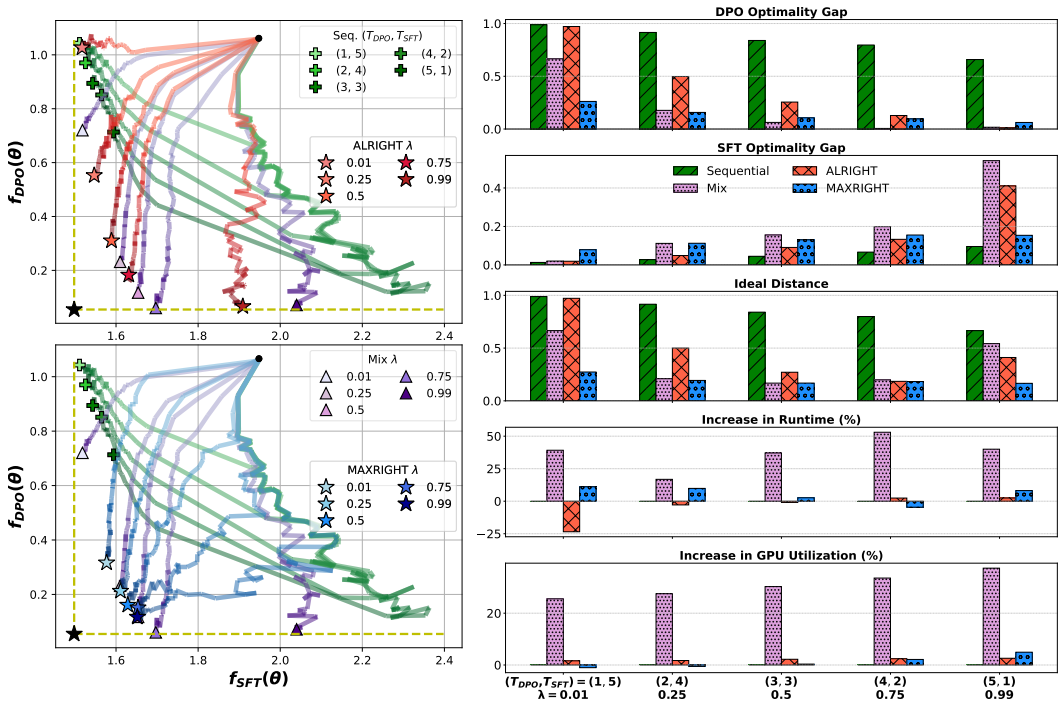

Figure 3: Comparison of proposed methods with first DPO then SFT using PYTHIA-1B model. **Left:** Training trajectories in the objective space (e.g., DPO and SFT objectives). **Right:** Performance comparison across multiple evaluation metrics, including optimality gap for DPO and SFT objectives, ideal distance, runtime, and GPU utilization. The bar charts highlight the trade-offs and resource efficiency of each method for different choices of $(T_{\text{DPO}}, T_{\text{SFT}})$ or λ.

odds ratio preference optimization, which integrates a regularization term into the SFT objective, removing the need for a separate RLHF phase. Similarly, Hua et al. (2024) propose intuitive fine-tuning, reformulating SFT to align on positive responses without requiring distinct RLHF and SFT phases. Li et al. (2024) propose to train on preference data both with demonstration and RLHF simultaneously. However, these methods restrict the use of different datasets for RLHF and SFT, limiting flexibility. Lin et al. (2023) suggest optimizing SFT and RLHF separately with adaptive model averaging (AMA) to balance alignment and fine-tuning, but the original AMA is computationally intensive, requiring three sets of LLM parameters. A memory-efficient AMA version exists, but its reliance on imitation learning from RLHF-trained data weakens its connection to the original objectives, and the link between AMA weights and optimal trade-offs is unclear.

6 EXPERIMENTS

In this section we compare the proposed methods with some existing baselines, in terms of their Pareto-front performance, and resource consumption such as computation time and memory usage.

6.1 EXPERIMENTAL SETUP

In this section, we introduce models, datasets, baselines, and evaluation metrics used to evaluate the performance of our proposed methods. Additional experiment details are provided in Appendix D.

Models. We employ two architectures. The first is ELEUTHERAI/PYTHIA-1B[1], a widely used model balancing computational efficiency and performance. Despite not being designed for downstream tasks, it matches or exceeds models like OPT and GPT-Neo of similar size. We use it to assess optimization dynamics, performance trade-offs, and resource usage of proposed methods. The second is META-LLAMA/META-LLAMA-3-8B[2], a larger model suited for fine-tuning and downstream real-world tasks. Both models are fine-tuned using Low-Rank Adaptation (LoRA) (Hu et al., 2021).

[1] https://huggingface.co/EleutherAI/pythia-1b
[2] https://huggingface.co/meta-llama/Meta-Llama-3-8B

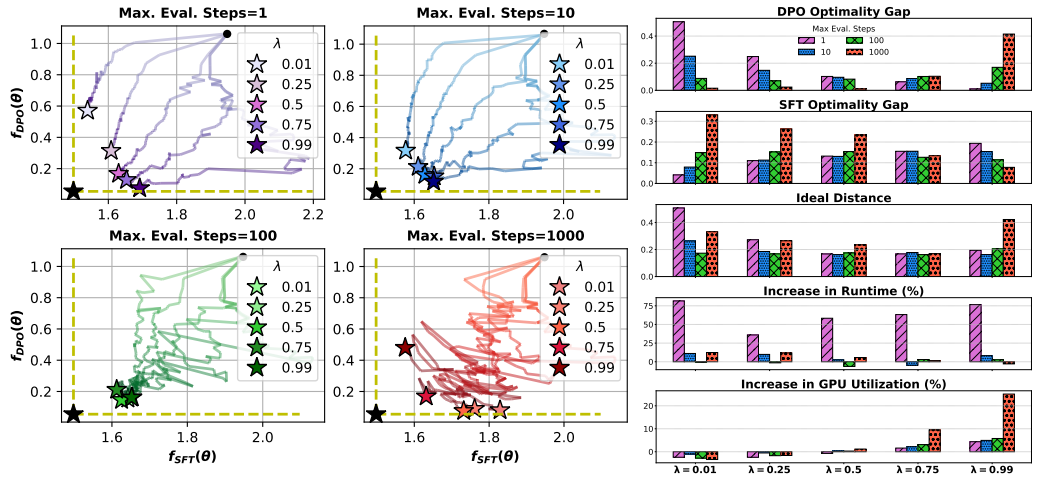

Figure 4: Comparison of different choices of evaluation steps for MAXRIGHT with Pythia-1b model.

Datasets. For the DPO dataset, we use the DAHOAS/RM-HH-RLHF dataset, which contains human feedback data designed to align models to human preference. For the SFT phase, we use the VICGALLE/ALPACA-GPT4 dataset, which consists of English instruction-following data generated by GPT-4 using Alpaca prompts, designed for fine-tuning LLMs.

Baseline Methods. Comparing the performance of ALRIGHT and MAXRIGHT, we use the following baselines: Mix of DPO and SFT ('Mix'), which simultaneously optimizes both DPO and SFT objectives by optimizing a convex combination of the objectives, and Sequential DPO and SFT ('Sequential'), where DPO and SFT objectives are optimized one after the other.

Evaluation Metrics. To assess the performance of each method with respect to the DPO and SFT objectives, we utilize several evaluation metrics. For evaluating the DPO objective, we measure the **optimality gap** as $f_{\text{DPO}}(\theta) - f^*_{\text{DPO}}$, where f^*_{DPO} is approximated by independently optimizing the DPO objective for the same number of iterations as used for the baselines and proposed methods. The optimality gap for the SFT objective is similarly defined using the optimal value f^*_{SFT}, obtained by separately optimizing the SFT objective. To evaluate overall performance, we use the **ideal distance** metric, which represents the Euclidean distance between the final iterate produced by the method and the point corresponding to optimal values for both DPO and SFT objectives: $\sqrt{(f_{\text{DPO}}(\theta) - f^*_{\text{DPO}})^2 + (f_{\text{SFT}}(\theta) - f^*_{\text{SFT}})^2}$. For resource efficiency, we compare the **percentage increase in runtime** relative to the corresponding sequential implementation, e.g., the percentage runtime increase of Alternating DPO and SFT with $\lambda = 0.01$ compared to Sequential DPO and SFT with $(T_{\text{DPO}}, T_{\text{SFT}}) = (1, 5)$. Additionally, we compute the **percentage increase in GPU utilization** for each method relative to the baselines. Further details on these metrics are provided in Appendix D due to space constraints. Furthermore, for evaluating the real-world performance of proposed methods compared to baselines, we use the following benchmarks: **MMLU**(Hendrycks et al., 2020), a benchmark with multiple-choice questions across 57 diverse tasks; **Win rate**, which is calculated as the proportion of times a model's response is preferred by an evaluator over a baseline in head-to-head comparisons. For this purpose, we use ALPACAEVAL(Li et al., 2023a) framework with GPT-4-TURBO as the evaluator and DAHOAS/RM-HH-RLHF test data as the baseline.

6.2 EXPERIMENT RESULTS

In this section we illustrate and discuss the empirical results obtained under the experiment setup introduced in the previous section.

ALRIGHT provides better control over the trade-off compared to Sequential. As shown in the top left plot of Figure 3, the optimization trajectories for DPO followed by SFT illustrate that the set of final models produced by ALRIGHT, for various values of λ, is more evenly distributed in the objective space. This distribution forms a Pareto front, indicating that no model is strictly worse than another with respect to both objectives. Moreover, the spread of these models is comparable to that of the Mix method. In contrast, Sequential tends to produce models that are biased towards the SFT objective, even when the number of DPO epochs is significantly higher than the number of SFT epochs (e.g., $(T_{\text{DPO}}, T_{\text{SFT}}) = (5, 1)$).

	MMLU (1-shot) (%)			Win rate (%)		
$\lambda/(T_{\text{SFT}}, T_{\text{DPO}})$	$0.25/(3,1)$	$0.5/(2,2)$	$0.75/(1,3)$	$0.25/(3,1)$	$0.5/(2,2)$	$0.75/(1,3)$
Sequential	73.18	72.80	72.68	57.19	65.62	59.38
Mix	73.45	73.40	72.29	81.88	84.22	**88.42**
ALRIGHT	**74.66**	72.65	**75.50**	**88.28**	85.78	87.34
MAXRIGHT	72.35	**73.42**	74.24	86.56	**86.09**	83.75

Table 1: Comparison of Win rate and MMLU (1-shot) for different methods using LLAMA3-8B.

MAXRIGHT achieves near-ideal performance compared to other methods. As illustrated in the top left plot of Figure 3, the optimization trajectories for DPO followed by SFT show that the set of final models produced by MAXRIGHT, for different values of λ, converge closer to the ideal point compared to other methods. This behavior is further supported by the Ideal Distance comparison in the right plot of Figure 3, where MAXRIGHT consistently achieves the best ideal distance performance across all λ values. We attribute this advantage to the adaptive nature of MAXRIGHT, which dynamically selects the objective to update based on performance, rather than adhering to a fixed schedule like ALRIGHT. This adaptability is particularly beneficial in heavily over-parameterized settings, where models have the capacity to approach ideal performance.

ALRIGHT and MAXRIGHT require minimal additional resources compared to Sequential and significantly lower than Mix. As shown in Figure 3 (right, Increase in Runtime (%) and Increase in GPU Utilization (%)), the additional computational resources required by different implementations of ALRIGHT and MAXRIGHT are minimal (or even negative) relative to their Sequential counterparts. In contrast, Mix incurs substantial additional resource usage, with increases of over 50% in runtime and more than 35% in GPU utilization, despite achieving similar performance metrics to ALRIGHT and MAXRIGHT.

Effect of maximum evaluation step for memory-efficient MAXRIGHT. Figure 4 illustrates the influence of maximum evaluation step choices in memory efficient MAXRIGHT on optimization trajectories and resource usage. For low values (e.g., 1), the algorithm closely follows the trade-off determined by λ, keeping the solutions concentrated near the ideal point (e.g., compared to ALRIGHT), but incurs high runtime due to frequent evaluations. In contrast, high values (e.g., 1000) cause significant oscillations in the objective space, failing to maintain the desired trade-off and resulting in increased GPU utilization from excessive SFT updates. The aforementioned oscillation leads to poor ideal distance performance as the model drifts away from the ideal point.

ALRIGHT and MAXRIGHT significantly outperform Sequential on real-world tasks with minimal additional resources. Table 1 presents a performance comparison of Sequential, Mix, ALRIGHT, and MAXRIGHT using the MMLU benchmark and win rate. ALRIGHT and MAXRIGHT consistently surpass the baselines on the MMLU benchmark and achieve a significantly higher win rate than Sequential across all λ values considered. Moreover, on both evaluation metrics, either ALRIGHT or MAXRIGHT performs on par with or better than Mix. On the other hand, Figure 7 in Appendix D shows that ALRIGHT and MAXRIGHT require minimal additional resources compared to Sequential, whereas Mix exhibits a substantial increase in resource usage (up to a 58% increase in runtime and a 7% increase in GPU utilization).

7 Conclusions and Discussion

In this paper, we have shown both theoretically and empirically that the widely adopted sequential approach to post-training LLMs with RLHF and SFT is sub-optimal, as the model gradually forgets the effects of the initial stage during the second stage of training. Our proposed ALRIGHT and MAXRIGHT methods address this issue, with ALRIGHT providing theoretical convergence guarantees and MAXRIGHT demonstrating strong empirical performance. Notably, this improvement is achieved with minimal additional computational cost, making these methods practical and efficient alternatives for enhancing both the performance and preference alignment of LLMs.

8 REPRODUCIBILITY STATEMENT

For the theoretical results given in the main text, we provide proofs in Appendix A. For all the experiment results provided in the main text and in the Appendix, we provide the implementation details in Appendix D. We will provide the code for implementing the main experiments in the discussion forums once they are open when the paper is under review, and the code will be released in GitHub upon acceptance of the paper.

REFERENCES

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Furong Huang, Dinesh Manocha, Amrit Singh Bedi, and Mengdi Wang. Maxmin-rlhf: Towards equitable alignment of large language models with diverse human preferences. *arXiv preprint arXiv:2402.08925*, 2024.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821*, 2023a.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023b.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*, 2023.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Meng Ding, Kaiyi Ji, Di Wang, and Jinhui Xu. Understanding forgetting in continual learning with linear regression. *arXiv preprint arXiv:2405.17583*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. Llama-omni: Seamless speech interaction with large language models. *arXiv preprint arXiv:2409.06666*, 2024. URL `https://huggingface.co/ICTNLP/Llama-3.1-8B-Omni`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2(4):5, 2024a.

Jiwoo Hong, Noah Lee, and James Thorne. Reference-free monolithic preference optimization with odds ratio. *arXiv e-prints*, pp. arXiv–2403, 2024b.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

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.

Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Ermo Hua, Biqing Qi, Kaiyan Zhang, Yue Yu, Ning Ding, Xingtai Lv, Kai Tian, and Bowen Zhou. Intuitive fine-tuning: Towards unifying sft and rlhf into a single process. *arXiv preprint arXiv:2405.11870*, 2024.

Feiyang Kang, Hoang Anh Just, Yifan Sun, Himanshu Jahagirdar, Yuanzhi Zhang, Rongxing Du, Anit Kumar Sahu, and Ruoxi Jia. Get more for less: Principled data selection for warming up fine-tuning in llms. *arXiv preprint arXiv:2405.02774*, 2024.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems*, 36, 2024.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*, 2023.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *arXiv preprint arXiv:2401.01967*, 2024.

Chenliang Li, Siliang Zeng, Zeyi Liao, Jiaxiang Li, Dongyeop Kang, Alfredo Garcia, and Mingyi Hong. Joint demonstration and preference learning improves policy alignment with human feedback. *arXiv preprint arXiv:2406.06874*, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023a.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023b.

Yong Lin, Lu Tan, Hangyu Lin, Zeming Zheng, Renjie Pi, Jipeng Zhang, Shizhe Diao, Haoxiang Wang, Han Zhao, Yuan Yao, et al. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *arXiv preprint arXiv:2309.06256*, 2023.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023a.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023b.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*, 2024.

OpenAI. Chatgpt: Optimizing language models for dialogue, November 2022. URL `https://openai.com/blog/chatgpt/`. Accessed: 2024-08-28.

Francesco Orabona. Last iterate of sgd converges even in unbounded domains, 2020. URL `https://parameterfree.com/2020/08/07/last-iterate-of-sgd-converges-even-in-unbounded-domains/`. Accessed: 2024-09-10.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Ralph Tyrell Rockafellar. Convex analysis, 1970.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Han Shen, Zhuoran Yang, and Tianyi Chen. Principled penalty-based methods for bilevel reinforcement learning and rlhf. *arXiv preprint arXiv:2402.06886*, 2024.

Zhengxiang Shi and Aldo Lipani. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. *arXiv preprint arXiv:2309.05173*, 2023.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*, 2020. URL `https://huggingface.co/SnypzZz/Llama2-13b-Language-translate`.

Junjiao Tian, Yen-Cheng Liu, James S Smith, and Zsolt Kira. Fast trainable projection for robust fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Fine-tuning language models for factuality. *arXiv preprint arXiv:2311.08401*, 2023.

Yuanhao Wang, Qinghua Liu, and Chi Jin. Is rlhf more difficult than standard rl? a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:76006–76032, 2023.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Forty-first International Conference on Machine Learning*, 2024.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023a.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023b.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *arXiv preprint arXiv:2402.04833*, 2024.

Siyan Zhao, John Dang, and Aditya Grover. Group preference optimization: Few-shot alignment of large language models. *arXiv preprint arXiv:2310.11523*, 2023.

Han Zhong, Guhao Feng, Wei Xiong, Li Zhao, Di He, Jiang Bian, and Liwei Wang. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*, 2024.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *International Conference on Machine Learning*, pp. 43037–43067. PMLR, 2023a.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023b.

Supplementary Material for "Mitigating Forgetting in LLM Supervised Fine-Tuning and Preference Learning"

Table of Contents

A Proofs for Theoretical Results **15**
- A.1 Basic claims on the problem setup . . . 15
- A.2 Proof of Theorem 3.1 . . . 17
- A.3 Proof of Theorem 4.1 . . . 19

B Memory Efficient MAXRIGHT Implementation. **21**

C Additional Experiment Results **21**
- C.1 Additional experiments with PYTHIA-1B . . . 21
- C.2 Additional experiments with LLAMA3-8B . . . 22

D Experiment Details **22**
- D.1 Experiments details for toy illustration in Figure 2 . . . 23
- D.2 Additional details for experiments with PYTHIA-1B . . . 23
- D.3 Additional details for experiments with LLAMA3-8B . . . 24
- D.4 Evaluation metrics used for measuring resource usage . . . 25

A PROOFS FOR THEORETICAL RESULTS

In this section, we provide the proofs for the main theoretical results of the paper, Theorems 3.1 and 4.1. The proof is organized as follows: In Appendix A.1, we establish some fundamental claims regarding the problem setup, such as the convexity of the objectives and the boundedness of the second moment of the stochastic gradients. Then, in Appendix A.2, we prove the lower bound stated in Theorem 3.1 along with several supporting lemmas. Finally, in Appendix A.3, we prove the upper bound stated in Theorem 4.1, accompanied by additional supporting lemmas.

A.1 BASIC CLAIMS ON THE PROBLEM SETUP

Before going to the proofs of the main results, we can have the following basic claims on the problem setup.

Proposition A.1 (Bounded second moment of stochastic gradient)**.** *For all $\theta \in \Theta$, there exist $M_1, M_2 > 0$ such that*

$$\mathbb{E}_{x,y_w,y_\ell\sim\mathcal{D}_{DPO}}[\|g_{DPO}(\theta;x,y_w,y_\ell)\|^2] \le M_1^2, \tag{20}$$

$$\mathbb{E}_{x,y\sim\mathcal{D}_{SFT}}[\|g_{SFT}(\theta;x,y)\|^2] \le M_2^2. \tag{21}$$

Proof. Considering g_{DPO}, we can first simplify h_β defined in (3) under softmax parameterization of π_θ and π_{ref} as

$$\begin{aligned} h_\beta(\theta;x,y_w,y_\ell) &= \beta\log\left(\frac{\pi_\theta(y_w\mid x)}{\pi_{\text{ref}}(y_w\mid x)}\right) - \beta\log\left(\frac{\pi_\theta(y_\ell\mid x)}{\pi_{\text{ref}}(y_\ell\mid x)}\right) \\ &= \beta(\theta-\theta_{\text{ref}})^\top(\phi_{y_w,x}-\phi_{y_\ell,x}). \end{aligned} \tag{22}$$

Then we can simplify g_{DPO} as

$$\begin{aligned} g_{\text{DPO}}(\theta;x,y_w,y_\ell) &= -\left(1-\sigma(h_\beta(\theta;x,y_w,y_\ell,\pi_{\text{ref}}))\right)\nabla_\theta h_\beta(\theta;x,y_w,y_\ell,\pi_{\text{ref}}) \\ &= -\beta(1-\sigma\left(\beta(\theta-\theta_{\text{ref}})^\top(\phi_{y_w,x}-\phi_{y_\ell,x})\right)(\phi_{y_w,x}-\phi_{y_\ell,x}). \end{aligned} \tag{23}$$

We can then bound the norm of g_{DPO} as

$$\begin{aligned}\|g_{\text{DPO}}(\theta; x, y_w, y_\ell)\|^2 &= \beta^2(1-\sigma\left(\beta(\theta-\theta_{\text{ref}})^\top(\phi_{y_w,x}-\phi_{y_\ell,x})\right)^2\|\phi_{y_w,x}-\phi_{y_\ell,x}\|^2 \\ &\le \beta^2\|\phi_{y_w,x}-\phi_{y_\ell,x}\|^2 \\ &\le 4\beta^2\Phi^2 =: M_1, \end{aligned} \tag{24}$$

where the first inequality is due to the fact that $0 \le \sigma(z) \le 1$ for all $z \in \mathbb{R}$, second inequality is due to Cauchy-Schwarz inequality and Assumption 3.1. Taking expectation (average) over the dataset $\mathcal{D}_{\text{DPO}}$ in (24) proves the first part of Proposition A.1. For proving the second part of the proposition, we start by simplifying the gradient of π_θ under softmax-parameteriation, given by

$$\begin{aligned}\nabla_\theta \pi_\theta(y \mid x) &= \nabla_\theta \frac{\exp(\theta^\top \phi_{y,x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})} \\ &= \frac{\exp(\theta^\top\phi_{y,x})\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})\phi_{y,x} - \exp(\theta^\top\phi_{y,x})\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})\phi_{y',x}}{\left(\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})\right)^2} \\ &= \left(\phi_{y,x} - \bar{\phi}_x(\theta)\right)\pi_\theta(y \mid x), \end{aligned} \tag{25}$$

where

$$\bar{\phi}_x(\theta) := \frac{\sum_{y'\in\mathcal{Y}}\phi_{y',x}\exp(\theta^\top\phi_{y',x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})}. \tag{26}$$

Then we can simplify g_{SFT} as

$$\begin{aligned} g_{\text{SFT}}(\theta; x, y) &= -\frac{\nabla_\theta\pi_\theta(y \mid x)}{\pi_\theta(y \mid x)} \\ &= -(\phi_{y,x} - \bar{\phi}_x(\theta)). \end{aligned} \tag{27}$$

We can then bound the norm of g_{SFT} as

$$\begin{aligned}\|g_{\text{SFT}}(\theta; x, y)\|^2 &= \|\phi_{y,x} - \bar{\phi}_x(\theta)\|^2 \\ &\le 4\Phi^2 =: M_2, \end{aligned} \tag{28}$$

where the inequality is due to the Cauchy-Schwarz inequality and Jensen's inequality. Taking expectation (average) over the dataset $\mathcal{D}_{\text{SFT}}$ in (28) proves the second part of Proposition A.1. □

Proposition A.2 (Convexity of objectives)**.** *Under Assumption 3.1, the objectives f_{DPO} and f_{SFT} (defined in (1) and (4), respectively) are convex.*

Proof. The goal of the proof is to show the Hessians of the objectives f_{DPO} and f_{SFT} are semi-positive definite, under Assumption 3.1 and LLMs (both trainable and reference) modeled using softmax parameterization. First, considering f_{DPO}, we can have

$$\begin{aligned}\nabla_\theta g_{\text{DPO}}(\theta; x, y_w, y_\ell) &= -\nabla_\theta\beta(1-\sigma\left(\beta(\theta-\theta_{\text{ref}})^\top(\phi_{y_w,x}-\phi_{y_\ell,x})\right)(\phi_{y_w,x}-\phi_{y_\ell,x}) \\ &= \beta^2\sigma\left(h_\beta(\theta; x, y_w, y_\ell)\right)(\phi_{y_w,x}-\phi_{y_\ell,x})(\phi_{y_w,x}-\phi_{y_\ell,x})^\top \succeq 0, \end{aligned} \tag{29}$$

where first equality is due to (23), and the semi-positive definiteness is due to the fact that $\beta > 0$, $0 \le \sigma(z) \le 1$ for all $z \in \mathbb{R}$, and $(\phi_{y_w,x}-\phi_{y_\ell,x})(\phi_{y_w,x}-\phi_{y_\ell,x})^\top \succeq 0$ for all $\phi_{y_w,x}, \phi_{y_\ell,x}$. The convexity of f_{DPO} follows from the fact that

$$\nabla^2 f_{\text{DPO}}(\theta) = \mathbb{E}_{x,y_w,y_\ell\sim\mathcal{D}_{\text{DPO}}}\left[\nabla_\theta g_{\text{DPO}}(\theta; x, y_w, y_\ell)\right]. \tag{30}$$

Similarly, we can compute $\nabla_\theta g_{\text{SFT}}$ as

$$\begin{aligned}\nabla_\theta g_{\text{SFT}}(\theta; x, y) &= -\nabla_\theta(\phi_{y,x} - \bar{\phi}_x(\theta)) \\ &= \nabla_\theta\frac{\sum_{y'\in\mathcal{Y}}\phi_{y',x}\exp(\theta^\top\phi_{y',x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})} \\ &= \sum_{y'\in\mathcal{Y}}\frac{\phi_{y',x}\phi_{y',x}^\top\exp(\theta^\top\phi_{y',x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})} \\ &\quad - \left(\frac{\sum_{y'\in\mathcal{Y}}\phi_{y',x}\exp(\theta^\top\phi_{y',x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})}\right)\left(\frac{\sum_{y'\in\mathcal{Y}}\phi_{y',x}\exp(\theta^\top\phi_{y',x})}{\sum_{y'\in\mathcal{Y}}\exp(\theta^\top\phi_{y',x})}\right)^\top, \end{aligned} \tag{31}$$

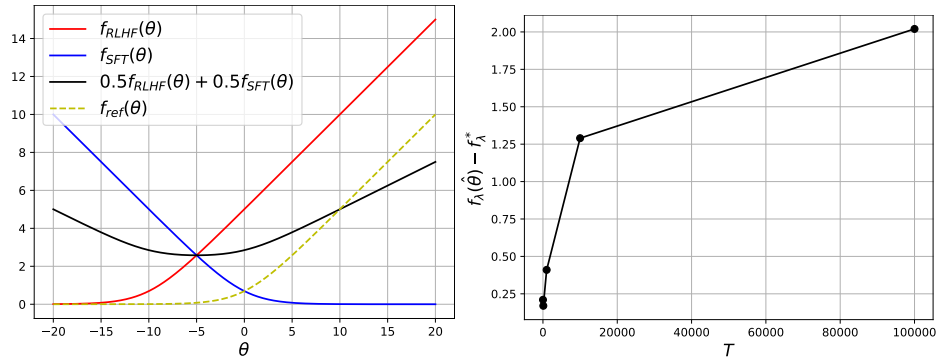

Figure 5: An Illustration of the example used for lower bound derivation in Theorem 3.1

where the first equality is due to (27). To establish the semi-positivedefiniteness of $\nabla_\theta g_{\text{SFT}}$, cosider any v with same dimension as θ, and let $p_{y,x} = \frac{\exp(\theta^\top \phi_{y,x})}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top \phi_{y',x})}$. Note that $p_{y,x} \geq 0$ for all x, y, and $\sum_{y \in \mathcal{Y}} p_{y,x} = 1$ for all x. Then we can have

$$
\begin{aligned}
& v^\top \Bigg(\sum_{y' \in \mathcal{Y}} \frac{\phi_{y',x} \phi_{y',x}^\top \exp(\theta^\top \phi_{y',x})}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top \phi_{y',x})} \\
& - \left(\frac{\sum_{y' \in \mathcal{Y}} \phi_{y',x} \exp(\theta^\top \phi_{y',x})}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top \phi_{y',x})} \right) \left(\frac{\sum_{y' \in \mathcal{Y}} \phi_{y',x} \exp(\theta^\top \phi_{y',x})}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top \phi_{y',x})} \right)^\top \Bigg) v \\
& = \sum_{y' \in \mathcal{Y}} (v^\top \phi_{y',x})^2 p_{y',x} - \left(\sum_{y' \in \mathcal{Y}} v^\top \phi_{y',x} p_{y',x} \right)^2 \geq 0, \qquad (32)
\end{aligned}
$$

where the last inequality is due to Jensen's inequality. This suggests that $\nabla_\theta g_{\text{SFT}}(\theta; x, y) \succeq 0$, and the convexity of of f_{SFT} follows from the fact that

$$
\nabla^2 f_{\text{SFT}}(\theta) = \mathbb{E}_{x,y \sim \mathcal{D}_{\text{SFT}}} \left[\nabla_\theta g_{\text{SFT}}(\theta; x, y) \right]. \qquad (33)
$$

Note that when f_{DPO} and f_{SFT} are convex, $f_{\text{Mix},\lambda}$ is also convex for all $\lambda \in [0, 1]$. □

A.2 Proof of Theorem 3.1

Proof. For deriving the lower bound given in 3.1, we consider the following problem setup that satisfies Assumption 3.1. Let Θ be $\mathbb{R}$, and let $\mathcal{D}_{\text{DPO}} = \{(x_1, y_w, y_\ell)\}$ and $\mathcal{D}_{\text{SFT}} = \{(x_2, y)\}$ and consider only two possible outputs exist (i.e. binary classification) such that we have the specification in Table 2. Note that the data point y' is not used in training explicitly, and it is specified to enable the

Input	Output	Feature $\phi_{y,x}$
x_1	$y_w = 1$	-1.0
x_1	$y_\ell = 0$	-0.5
x_2	$y = 0$	1.0
x_2	$y' = 1$	0.5

Table 2: Data set specification for lower bound analysis example

calculation of the output of π_θ with softmax parameterization in the SFT optimization phase. Based on this dataset, we can also define the dataset for reference policy objective as $\mathcal{D}_{\text{ref}} = \{(x_1, y_w)\}$, which has a similar optimization procedure as SFT. Before moving forward, we have to choose the reference policy π_{ref}, or equivalently θ_{ref}. The objective to determine θ_{ref} is given by

$$
\theta_{\text{ref}} \in \operatorname*{argmin}_\theta f_{\text{ref}}(\theta) := -\log \pi_\theta(y_w \mid x_1), \qquad (34)
$$

which is graphically illustrated in Figure 5. We choose $\theta_{\text{ref}} = -5$ since this choice reasonably optimizes the objective.

With this problem setup and choice of θ_{ref}, we can then derive the objectives and corresponding gradients using (1), (4), (23), and (27) as

$$f_{\text{DPO}}(\theta) = \log\left(1 + c\frac{1+\exp(\theta/2)}{1+\exp(-\theta/2)}\right) \tag{35}$$

$$f_{\text{SFT}}(\theta) = \log(1+\exp(-\theta/2)) \tag{36}$$

$$g_{\text{DPO}}(\theta) = \frac{1}{2}\cdot\frac{1}{1+\exp(-(\theta/2+5))} \tag{37}$$

$$g_{\text{SFT}}(\theta) = -\frac{1}{2}\cdot\frac{1}{1+\exp(\theta/2)}, \tag{38}$$

where $c = \frac{1+\exp(5)}{1+\exp(-5)}$. Choosing $\lambda = 0.5$, we can numerically find an upper bound to $f^*_{\text{Mix},\lambda}$ such that $f^*_{\text{Mix},\lambda} \leq \frac{1}{2}\log c^*$ where $c^* = 173.78$. With this, we can derive a lower bound to $G_{\text{Mix},\lambda}(\theta)$ as

$$\begin{aligned} G_{\text{Mix},\lambda}(\theta) &= f_{\text{Mix},\lambda}(\theta) - f^*_{\text{Mix},\lambda} \\ &\geq \frac{1}{2}\cdot\log\left(1 + c\frac{1+\exp(\theta/2)}{1+\exp(-\theta/2)}\right) + \frac{1}{2}\cdot\log(1+\exp(-\theta/2)) - \frac{1}{2}\log c^* \\ &= \frac{1}{2}\log\left(\frac{1}{c^*}(1+\exp(-\theta/2)) + \frac{c}{c^*}(1+\exp(\theta/2))\right). \end{aligned} \tag{39}$$

When $\theta = 0$, the right hand side of (39) approximately equals 0.256843. Since it is monotonically increasing for $\theta \in [0,\infty)$, we have $G_{\text{Mix},\lambda}(\theta) \gtrsim 0.256843 = \Omega(1)$ when $\theta \geq 0$. Thus to prove the result, it is sufficient to show Algorithm 1 will generate a $\hat{\theta}_{\text{Seq}}$ that is greater than 0.

We have the first stage iterates:

$$\theta^1_{t+1} = \theta^1_t - \alpha_t\frac{1}{1+\exp(-\frac{\theta^1_t}{2}-5)}, \quad \text{for } t = 1,...,T-1. \tag{40}$$

Using the first stage's last iterate as initial point, we have the second stage iterates:

$$\theta^{2,T}_{t+1} = \theta^{2,T}_t + \alpha_t\frac{1}{1+\exp(\frac{\theta^{2,T}_t}{2})}, \quad \text{for } t = 1,...,T-1. \tag{41}$$

where $\theta^{2,T}_1 = \theta^1_T$ and the superscript T in $\theta^{2,T}_t$ indicates the max iteration index in the first stage.

Without loss of generality, we initialize $\theta^1_1 = 0$. Then by (40) and (41), we have

$$\hat{\theta}_{\text{Seq}} = \theta^{2,T}_T = -\alpha_t\sum_{t=1}^{T-1}\frac{1}{1+\exp(-\frac{\theta^1_t+10}{2})} + \alpha_t\sum_{t=1}^{T-1}\frac{1}{1+\exp(\frac{\theta^{2,T}_t}{2})} \tag{42}$$

We first prove the following lemma.

Lemma A.1. *Given a positive constant c and a sequence $\{\alpha_t\}$, consider the iterates generated by*

$$\theta_{t+1} = \theta_t + \alpha_t\frac{1}{1+\exp(c\theta_t)}, \quad \theta'_{t+1} = \theta'_t + \alpha_t\frac{1}{1+\exp(c\theta'_t)} \tag{43}$$

If $\theta_1 - \theta'_1 \geq 0$ and $\frac{c\alpha_t}{4} \leq 1$ for any t, then we have

$$\theta_t - \theta'_t \geq (\theta_1 - \theta'_1)\Pi_{i=1}^{t-1}\Big(1 - \frac{c\alpha_i}{4}\Big), \quad \forall t.$$

Proof. We prove the result by induction. Assume $\theta_t - \theta'_t \geq 0$ for some t. We first have

$$\theta_{t+1} - \theta'_{t+1} = \theta_t - \theta'_t + \alpha_t\Big(\frac{1}{1+\exp(c\theta_t)} - \frac{1}{1+\exp(c\theta'_t)}\Big) \tag{44}$$

With $\nabla_\theta \frac{1}{1+\exp(c\theta)} = -c\sigma(-c\bar{\theta})\big(1-\sigma(-c\theta)\big)$ where σ is the sigmoid function, using the mean value theorem in (44), we have for some $\bar{\theta}_t$ in between θ_t and θ'_t that

$$\begin{aligned}\theta_{t+1} - \theta'_{t+1} &= \theta_t - \theta'_t - c\alpha_t\sigma(-c\bar{\theta}_t)\big(1-\sigma(-c\bar{\theta}_t)\big)(\theta_t - \theta'_t)\\ &\geq \theta_t - \theta'_t - \frac{1}{4}c\alpha_t(\theta_t - \theta'_t)\\ &= \Big(1 - \frac{1}{4}c\alpha_t\Big)(\theta_t - \theta'_t)\end{aligned}$$

where the inequality follows from $\sigma(-c\bar{\theta}_t)\big(1-\sigma(-c\bar{\theta}_t)\big) \leq \max_{x\in[0,1]} x(1-x) = \frac{1}{4}$ and the assumption that $\theta_t - \theta'_t \geq 0$. Since $\theta_1 - \theta'_1 \geq 0$, it follows from induction that $\theta_t - \theta'_t \geq 0$ for any t. Then recursively applying the last inequality completes the proof. □

Now we consider (42). Rewriting (40) gives that θ^1_t is generated by

$$-(\theta^1_{t+1} + 10) = -(\theta^1_t + 10) + \alpha_t \frac{1}{1+\exp(-\frac{\theta^1_t+10}{2})}, \quad \text{for } t = 1, ..., T-1,$$

and $\theta^{2,T}_t$ is generated by (41). Thus $\theta^{2,T}_t$ and $-(\theta^1_t + 10)$ are generated by (43) with $c = 1/2$. Assuming the step size α_t is proper that $\sum_{t=1}^{\infty} \alpha_t = \infty$, then there always exists T^* that for $T \geq T^*$, we have θ^1_T is small enough such that $-(\theta^1_1 + 10) - \theta^{2,T}_1 = -10 - \theta^1_T \geq 0$. If $\alpha_t/8 \leq 1$, we have Lemma A.1 holds for sequences $\theta^{2,T}_t$ and $-(\theta^1_{t+1} + 10)$, and we have

$$-(\theta^1_t + 10) \geq \theta^{2,T}_t \Rightarrow \frac{1}{1+\exp(\theta^{2,T}_t/2)} - \frac{1}{1+\exp(-(\theta^1_t+10)/2)} \geq 0. \tag{45}$$

Using (45) in (42), we have $\hat{\theta}_{\text{Seq}} \geq 0$. This completes the proof. □

A.3 Proof of Theorem 4.1

In this section, we provide the proof for Theorem 4.1. First, we provide some useful results that will later be used in the main proof. For conciseness, we denote $g_{\text{DPO}}(\theta_t; x^t, y^t_w, y^t_\ell)$ and $g_{\text{SFT}}(\theta_t; x^t, y^t)$ by $g_{\text{DPO},t}$ and $g_{\text{SFT},t}$, respectively.

Lemma A.2 ((Rockafellar, 1970) Theorem 25.1 and Corollary 25.11)**.** *Consider any convex differentiable function f and let $\theta \in \Theta$. Then, for any $\theta' \in \Theta$, we have*

$$f(\theta') \geq f(\theta) + \nabla f(\theta)^\top(\theta' - \theta). \tag{46}$$

Lemma A.3 ((Orabona, 2020) Lemma 1.)**.** *Let $\{\eta_t\}_{t=1}^T$ be a non-increasing sequence of positive numbers and $q_t \geq 0$ for all $t = 1, \ldots, T$. Then*

$$n_T q_T \leq \frac{1}{T}\sum_{t=1}^{T} \eta_t q_t + \sum_{k=1}^{T-1} \frac{1}{k(k+1)} \sum_{t=T-k+1}^{T} \eta_t(q_t - q_{T-k}) \tag{47}$$

Lemma A.4. *Consider iterates θ_t and θ_{t+1} generated by Algorithm 2 for $t \in \{1, \ldots, T-1\}$. Then for any $\theta' \in \Theta$ and $\lambda \in [0,1]$, we have*

$$\mathbb{E}\left[f_{Mix,\lambda}(\theta_t) - f_{Mix,\lambda}(\theta')\right] \leq \frac{1}{2\alpha_t}\mathbb{E}\left[\|\theta_t - \theta'\|^2\right] - \frac{1}{2\alpha_t}\mathbb{E}\left[\|\theta_{t+1} - \theta'\|^2\right] + \frac{\alpha_t}{2}M^2_\lambda, \tag{48}$$

where $M_\lambda = \lambda M_1 + (1-\lambda)M_2$, and M_1, M_2 are as defined in Proposition A.1.

Proof. Considering Algorothm 2, for any $\theta' \in \Theta$, we can have

$$\begin{aligned}\|\theta_{t+1} - \theta'\|^2 - \|\theta_t - \theta'\|^2 &\leq \|\theta_t - \alpha_t(\mathbb{I}_{i_t=1}g_{\text{DPO},t} + \mathbb{I}_{i_t=0}g_{\text{SFT},t}) - \theta'\|^2 - \|\theta_t - \theta'\|^2\\ &= -2\alpha_t(\mathbb{I}_{i_t=1}g_{\text{DPO},t} + \mathbb{I}_{i_t=0}g_{\text{SFT},t})^\top(\theta_t - \theta')\\ &\quad + \alpha_t^2\|\mathbb{I}_{i_t=1}g_{\text{DPO},t} + \mathbb{I}_{i_t=0}g_{\text{SFT},t}\|^2\\ &= -2\alpha_t(\mathbb{I}_{i_t=1}g_{\text{DPO},t} + \mathbb{I}_{i_t=0}g_{\text{SFT},t})^\top(\theta_t - \theta')\\ &\quad + \alpha_t^2\left(\mathbb{I}_{i_t=1}\|g_{\text{DPO},t}\|^2 + \mathbb{I}_{i_t=0}\|g_{\text{SFT},t}\|^2\right)\end{aligned} \tag{49}$$

Taking conditional expectation $\mathbb{E}[\cdot \mid \theta_t]$ (over randomness of datapoints and i_t) in both sides of above inequality, we obtain

$$\begin{aligned}\mathbb{E}\left[\|\theta_{t+1}-\theta'\|^2 \mid \theta_t\right] - \|\theta_t-\theta'\|^2 &\le -2\alpha_t(\lambda\nabla f_{\text{DPO}}(\theta_t)+(1-\lambda)\nabla f_{\text{SFT}}(\theta))^\top(\theta_t-\theta')+\alpha_t^2 M_\lambda^2 \\ &= -2\alpha_t\nabla f_{\text{Mix},\lambda}(\theta_t)^\top(\theta_t-\theta')+\alpha_t^2 M_\lambda^2 \\ &\le -2\alpha_t\left(f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta')\right)+\alpha_t^2 M_\lambda^2, \end{aligned} \tag{50}$$

where the first inequality is using the definitions of $g_{\text{DPO},t}$,$g_{\text{SFT},t}$, and M_λ, equality is by the defintion of $f_{\text{Mix},\lambda}$, and the last inequality is due to Lemma A.2. The result follows from taking total expectation in both sides and rearranging the inequality (50). □

With the above results, we are ready to prove Theorem 4.1.

Theorem A.1 (Theorem 4.1 Restated with Additional Details)**.** *Consider Algorithm 2 with $\alpha_t = \alpha_0/\sqrt{T}$ for all $t \in \{1,\dots,T\}$ and $\alpha_0 > 0$. Then, under Assumption 3.1, for any $\lambda \in [0,1]$, we have*

$$\mathbb{E}[G_{Mix,\lambda}(\theta_T)] \le \frac{\alpha_0}{2\sqrt{T}}\|\theta_1-\theta^*_{Mix,\lambda}\|^2 + \frac{(2+\log(T-1))M_\lambda^2\alpha_0}{2\sqrt{T}} \tag{51}$$

where $\theta^*_{Mix,\lambda} \in \operatorname{argmin}_{\theta\in\Theta} f_{Mix,\lambda}(\theta)$, and $M_\lambda = \lambda M_1 + (1-\lambda)M_2$ with M_1, M_2 as defined in Assumption A.1.

Proof. Substituting $\eta_t = \alpha$ and $q_t = G_{\text{Mix},\lambda}(\theta_t)$ in (47) (Lemma A.3), we have

$$\begin{aligned}G_{\text{Mix},\lambda}(\theta_T) &\le \frac{1}{T}\sum_{t=1}^{T} G_{\text{Mix},\lambda}(\theta_t) + \sum_{k=1}^{T-1}\frac{1}{k(k+1)}\sum_{t=T-k+1}^{T}\left(G_{\text{Mix},\lambda}(\theta_t)-G_{\text{Mix},\lambda}(\theta_{T-k})\right) \\ &= \frac{1}{T}\sum_{t=1}^{T} G_{\text{Mix},\lambda}(\theta_t) + \sum_{k=1}^{T-1}\frac{1}{k(k+1)}\sum_{t=T-k+1}^{T}\left(f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta_{T-k})\right), \end{aligned} \tag{52}$$

where we have used the definition of $G_{\text{Mix},\lambda}$ in the equality. Taking total expectation on both sides of (52), we get

$$\mathbb{E}[G_{\text{Mix},\lambda}(\theta_T)] \le \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[G_{\text{Mix},\lambda}(\theta_t)] + \sum_{k=1}^{T-1}\frac{1}{k(k+1)}\sum_{t=T-k+1}^{T}\mathbb{E}\left[f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta_{T-k})\right] \tag{53}$$

The first term in the right hand side of (53) can be bounded by choosing $\theta' = \theta^*_{\text{Mix},\lambda}$ and $\alpha_t = \alpha$ in (48) from Lemma A.4 and then taking a telescoping sum:

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}[G_{\text{Mix},\lambda}(\theta_t)] &\le \frac{1}{2\alpha}\|\theta_1-\theta^*_{\text{Mix},\lambda}\|^2 - \frac{1}{2\alpha}\|\theta_{T+1}-\theta^*_{\text{Mix},\lambda}\|^2 + T\frac{M_\lambda^2\alpha}{2} \\ &\le \frac{1}{2\alpha}\|\theta_1-\theta^*_{\text{Mix},\lambda}\|^2 + T\frac{M_\lambda^2\alpha}{2} \end{aligned} \tag{54}$$

Now we consider the second term in the right hand side of (53). We first have

$$\begin{aligned}\sum_{t=T-k+1}^{T}\mathbb{E}\left[f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta_{T-k})\right] &= \sum_{t=T-k}^{T}\mathbb{E}\left[f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta_{T-k})\right] \\ &\le (k+1)\frac{M_\lambda^2\alpha}{2} \end{aligned} \tag{55}$$

where the inequality follows from setting $\theta' = \theta_{T-k}$, $\alpha_t = \alpha$ in (48) from Lemma A.4 and then taking a telescoping sum from $t = T-k$ to T. Substituting the last inequality to the second term in the right hand side of (53) yields

$$\begin{aligned}\sum_{k=1}^{T-1}\frac{1}{k(k+1)}\sum_{t=T-k+1}^{T}\mathbb{E}\left[f_{\text{Mix},\lambda}(\theta_t)-f_{\text{Mix},\lambda}(\theta_{T-k})\right] &\le \sum_{k=1}^{T-1}\frac{1}{k}\frac{M_\lambda^2\alpha}{2} \\ &\le (1+\log(T-1))\frac{M_\lambda^2\alpha}{2}. \end{aligned} \tag{56}$$

Algorithm 4 Memory Efficient MAXRIGHT

Input $\mathcal{D}_{\text{DPO}}$, $\mathcal{D}_{\text{SFT}}$, $\{\alpha_t\}_{t=1}^{T}$,$\lambda \in [0,1]$, max evaluation steps k
Initialize $\theta_1 \in \Theta$
for $t = 1, \ldots, T-1$ **do**
  Sample $x_1^t, y_w^t, y_\ell^t \sim \mathcal{D}_{\text{DPO}}$
  Sample $x_2^t, y^t \sim \mathcal{D}_{\text{SFT}}$
  if $t \bmod k = 0 \parallel t = 1$ **then**
    Evaluate (without generating computational graph)
      $\bar{f}_{1,\lambda}(\theta_t) \coloneqq \lambda \left(f_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t) - f^*_{\text{DPO}}\right)$ and
      $\bar{f}_{2,\lambda}(\theta_t) \coloneqq (1-\lambda) \left(f_{\text{SFT}}(\theta_t; x_2^t, y^t) - f^*_{\text{SFT}}\right)$
    Set
      $t_0 = t$
      $\bar{f}_{1,\lambda}^{\text{stale},t_0} = \bar{f}_{1,\lambda}(\theta_t)$
      $\bar{f}_{2,\lambda}^{\text{stale},t_0} = \bar{f}_{2,\lambda}(\theta_t)$
  end if
  Set $i_t = \operatorname{argmax}_i \bar{f}_{1,\lambda}^{\text{stale},t_0}$
  if $i_t = 1$ **then**
    Set $\bar{f}_{1,\lambda}^{\text{stale},t_0} = \lambda \left(f_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t) - f^*_{\text{DPO}}\right)$
    Update $\theta_{t+1} = \Pi_\Theta \left(\theta_t - \alpha_t g_{\text{DPO}}(\theta_t; x_1^t, y_w^t, y_\ell^t)\right)$
  else
    Set $\bar{f}_{2,\lambda}^{\text{stale},t_0} = (1-\lambda) \left(f_{\text{SFT}}(\theta_t; x_2^t, y^t) - f^*_{\text{SFT}}\right))$
    Update $\theta_{t+1} = \Pi_\Theta \left(\theta_t - \alpha_t g_{\text{SFT}}(\theta_t; x_2^t, y^t)\right)$
  end if
end for
Output $\hat{\theta}_{\text{MAX}} \coloneqq \theta_T$

Choosing $\alpha = \frac{\alpha_0}{\sqrt{T}}$, and substituting (54) and (56) in (53) yields

$$\mathbb{E}[G_{\text{Mix},\lambda}(\theta_T)] \leq \frac{\alpha_0}{2\sqrt{T}} \|\theta_1 - \theta^*_{\text{Mix},\lambda}\|^2 + \frac{(2 + \log(T-1))M_\lambda^2 \alpha_0}{2\sqrt{T}}$$

which completes the proof. □

B MEMORY EFFICIENT MAXRIGHT IMPLEMENTATION.

In this section we summarize the memory-efficient implementation of MAXRIGHT in Section 4.2.

C ADDITIONAL EXPERIMENT RESULTS

In this section, we provide additional experiment results using PYTHIA-1B and LLAMA3-8B models.

C.1 ADDITIONAL EXPERIMENTS WITH PYTHIA-1B

ALRIGHT and MAXRIGHT significantly outperform Sequential. In Figure 6 (left), it can be seen that the final models obtained by ALRIGHT and MAXRIGHT achieve better trade-off in DPO and SFT objective values in general compared to Sequential. Furthermore, ALRIGHT and MAXRIGHT perform comparably or significantly better in terms of SFT optimality gap and ideal distance metrics (Figure 6 (right)), while Sequential demonstrates a better performance in RLHF optimality gap. This is because, in this experiment setup, Sequential is implemented by optimizing for SFT first then DPO.

ALRIGHT and MAXRIGHT require minimal additional resources compared to Sequential and significantly lower than Mix. In Figure 6 (right), the additional computational resources required by different implementations of ALRIGHT and MAXRIGHT are minimal (or even negative) relative to their Sequential counterparts. In contrast, Mix incurs substantial additional resource usage, with

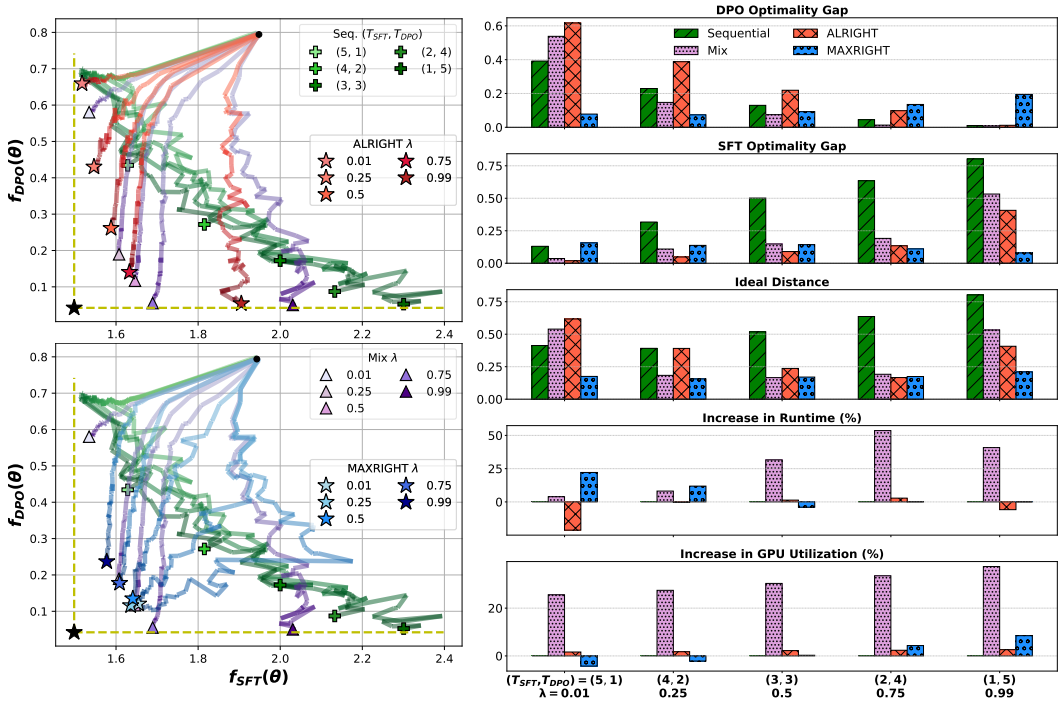

Figure 6: Comparison of proposed methods with baselines (SFT first then DPO for Sequential). **Left:** Training trajectories for various methods in the objective space, visualizing their convergence with respect to the DPO and SFT objectives. **Right:** Performance comparison across multiple evaluation metrics, including optimality gap for DPO and SFT objectives, ideal distance, runtime, and GPU utilization. The bar charts highlight the trade-offs and resource efficiency of each method for different choices of $(T_{\text{SFT}}, T_{\text{DPO}})$ or λ.

increases of upto 53% in runtime and upto 37% in GPU utilization, despite achieving comparable performance metrics to ALRIGHT and MAXRIGHT.

C.2 Additional experiments with Llama3-8b

ALRIGHT and MAXRIGHT perform comparably or better than Sequential. In Figure 7 (left), it can be seen that the final models obtained by ALRIGHT and MAXRIGHT achieve better or comparable trade-off in DPO and SFT objective values in general compared to Sequential. Furthermore, MAXRIGHT performs consistently better in terms of ideal distance metric (Figure 7 (right)), which is consistent with Pythia-1b experiments.

ALRIGHT and MAXRIGHT require minimal additional resources compared to Sequential and significantly lower than Mix. In Figure 7 (right), the additional computational resources required by different implementations of ALRIGHT and MAXRIGHT are minimal (or even negative) relative to their Sequential counterparts. In contrast, Mix incurs substantial additional resource usage, with increases of upto 58% in runtime and upto 7% in GPU utilization, despite achieving comparable performance metrics to ALRIGHT and MAXRIGHT. Note that, unlike in Pythia-1b experiments, the increase in GPU utilization for Mix is lower. We believe this is because we implement gradient checkpointing when training Llama3-8b, and gradient checkpointing improves GPU utilization at the cost of increased runtime due to duplicate activation computations in the backward pass.

D Experiment Details

In this section, we provide experiment details for the experiments in Sections 6 and C. We build upon OpenRLHF (Hu et al., 2024) framework to implement the experiments in Section 6, C.1, and C.2.

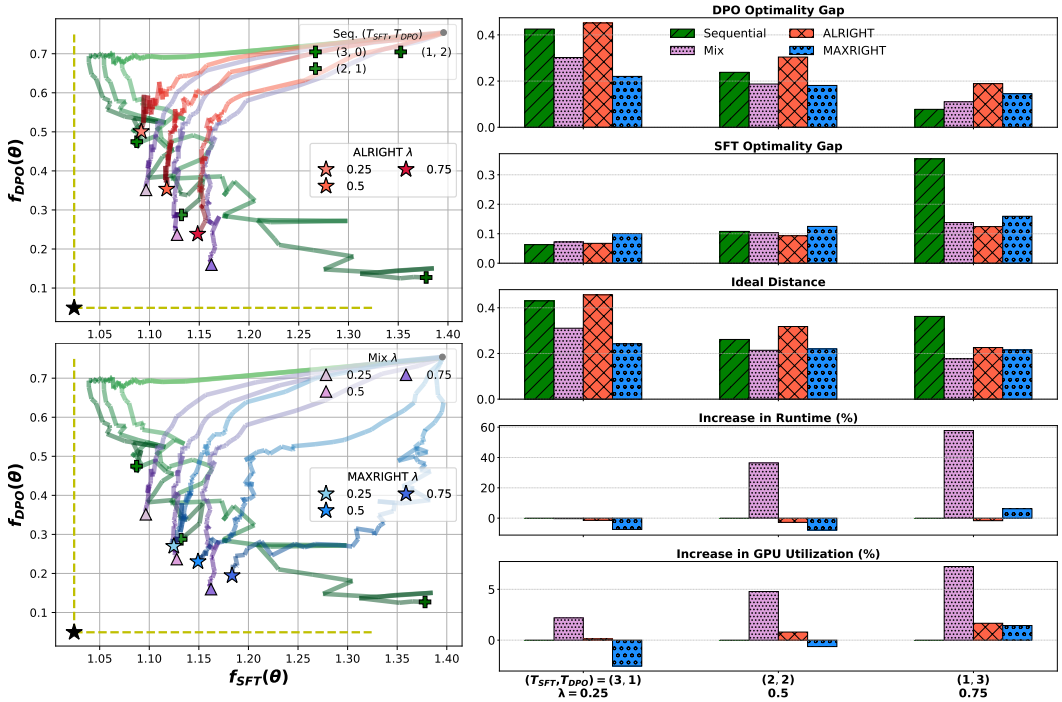

Figure 7: Comparison of proposed methods with baselines (SFT first then DPO for Sequential) using LLAMA3-8B. **Left:** Training trajectories for various methods in the objective space, visualizing their convergence with respect to the DPO and SFT objectives. **Right:** Performance comparison across multiple evaluation metrics, including optimality gap for DPO and SFT objectives, ideal distance, runtime, and GPU utilization. The bar charts highlight the trade-offs and resource efficiency of each method for different choices of $(T_{\text{SFT}}, T_{\text{DPO}})$ or λ.

D.1 EXPERIMENTS DETAILS FOR TOY ILLUSTRATION IN FIGURE 2

In this section we provide details for the experiment results given in Figure 2. We consider Θ be $\mathbb{R}^2$,$\mathcal{D}_{\text{DPO}} = \{(x_1, y_w, y_\ell)\}$ and $\mathcal{D}_{\text{SFT}} = \{(x_2, y)\}$ and setting where only two possible outputs exist (i.e. binary classification) such that we have the specification in Table 2. Note that the data point y' is not used in training explicitly, and it is specified to enable the calculation of the output of π_θ with softmax parameterization in the SFT optimization phase. Based on this dataset, we can also define the dataset for reference policy objective as $\mathcal{D}_{\text{ref}} = \{(x_1, y_w)\}$, which has a similar optimization procedure as SFT.

To obtain θ_{ref}, we train a parameter initialized at $[5.0; -9.9]^\top$ for 1000 epochs with a learning rate of 0.01. This parameter initialization and learning rate are also used to train the model θ using the Sequential, ALRIGHT, and MAXRIGHT methods. Furthermore, for illustration purposes, we use a weight decay of 0.001 in optimization. The resulting π_{ref} is then used as the reference policy for the DPO objective in all three methods.

For the sequential method, we train θ for 10,000 epochs per objective (DPO first, then SFT). A threshold of 0.05 for the objective value is applied to stop training for a given objective, preventing excessive overfitting to that objective.

For the ALRIGHT and MAXRIGHT methods, we train θ for 20,000 epochs, while keeping other training configurations identical to the Sequential method.

D.2 ADDITIONAL DETAILS FOR EXPERIMENTS WITH PYTHIA-1B

We conducted three sets of experiments using the PYTHIA-1B model:

Input	Output	Feature $\phi_{y,x}$
x_1	$y_w = 1$	$[1.0; 1.0]^\top$
x_1	$y_\ell = 0$	$[0.5; 0.5]^\top$
x_2	$y = 0$	$[1.0; 0.5]^\top$
x_2	$y' = 1$	$[0.5; 0.5]^\top$

Table 3: Data set specification for toy illustration in Figure 2

(1) Comparison of baselines and proposed methods for sequential training with DPO first, followed by SFT (Figure 3),

(2) Comparison of baselines and proposed methods for sequential training with SFT first, followed by DPO (Figure 6), and

(3) Ablation study on the choice of maximum evaluation steps for memory-efficient MAXRIGHT (Figure 4). The primary difference between the first two experiments is the π_{ref} used (and thus the DPO objective), as described in Section 2.

For training the models (both θ and θ_{ref}) in experiments (1), (2), and (3), we use LoRA with rank 32 and $\alpha = 32$. The QUERY_KEY_VALUE layers are the target modules to which LoRA is applied. No gradient checkpointing is used for PYTHIA-1B training. The learning rate is set to 5×10^{-5} for all model training with PYTHIA-1B.

To obtain θ_{ref} for experiments (1) and (3), we train the model for 6 epochs using $24,000$ input-response pairs from the RM-HH-RLHF dataset, with a batch size of 12 and a learning rate of 5×10^{-5}. For experiment (2), we train the model for 6 epochs using $24,000$ samples from the ALPACA-GPT4 dataset, with a batch size of 24.

To compute f^*_{DPO} and f^*_{SFT}, which are required for calculating the optimality gap, ideal distance metrics, and implementing the MAXRIGHT method, we perform independent optimization of the DPO and SFT objectives for 6 epochs. For the SFT objective, we use $24,000$ samples from the ALPACA-GPT4 dataset with a batch size of 24, and for the DPO objective, we use $8,000$ samples from the RM-HH-RLHF dataset with a batch size of 8. Additionally, we run ALRIGHT for 6 epochs to establish a reference Pareto front, which, along with the optimal objective values, is used as a stopping criterion for joint optimization. No stopping criterion is applied for the sequential method.

Finally, all methods are trained for 6 epochs, using the corresponding λ for joint optimization methods or a combination of T_{DPO} and T_{SFT} for the sequential method, until the stopping criterion is reached. For the memory-efficient MAXRIGHT implementation in experiments (1) and (2), the maximum evaluation step is set to 10.

D.3 Additional details for experiments with Llama3-8b

We present the experiment results for LLAMA3-8B training in Table 1 and Figure 7. Both result sets share the same training configuration, which is described below.

For training the models (both θ and θ_{ref}), we use LoRA with rank 16 and $\alpha = 16$. The Q_PROJ and V_PROJ layers are the target modules for LoRA application. Gradient checkpointing is enabled during training with LLAMA3-8B. The learning rate is set to 5×10^{-5} for all model training with LLAMA3-8B. To obtain θ_{ref}, we train the model for 4 epochs using $24,000$ samples from the ALPACA-GPT4 dataset, with a batch size of 16.

To compute f^*_{DPO} and f^*_{SFT}, required for calculating the optimality gap, ideal distance metrics, and implementing MAXRIGHT, we independently optimize the DPO and SFT objectives for 4 epochs. We use $24,000$ samples from the ALPACA-GPT4 dataset for the SFT objective with a batch size of 16, and $6,000$ samples from the RM-HH-RLHF dataset for the DPO objective with a batch size of 4. Additionally, we run ALRIGHT for 4 epochs to establish a reference Pareto front, which, along with the optimal objective values, serves as a stopping criterion for joint optimization. No stopping criterion is applied for the sequential method.

Finally, all methods are trained for 4 epochs, using the corresponding λ for joint optimization methods or a combination of T_{DPO} and T_{SFT} for the sequential method, until the stopping criterion is reached. For memory-efficient MAXRIGHT implementation, the maximum evaluation step is set to 10.

D.4 Evaluation metrics used for measuring resource usage

In this section we give the formula for computing the resource usage metrics used in Section 6; percentage increase in runtime and percentage increase in GPU utilization.

Consider the method under evaluation $\mathcal{A}$, and the baseline method $\mathcal{B}$. Then, percentage increase in runtime is given by

$$\text{percentage increase in runtime for } \mathcal{A} = \frac{\text{runtime of } \mathcal{A} - \text{runtime of } \mathcal{B}}{\text{runtime of } \mathcal{B}} \times 100\%. \tag{57}$$

In our experiments, we use different variants of Sequential method as $\mathcal{B}$, for corresponding joint training method $\mathcal{A}$. For example, in PYTHIA-1B experiments we use Sequential with $(T_{\text{DPO}}, T_{\text{SFT}}) = (5, 1)$ configuration as the baseline for Mix, ALRIGHT, and MAXRIGHT with $\lambda = 0.99$. We can similarly define the percentage increase of GPU utilization as

$$\text{percentage increase in GPU utilization for } \mathcal{A} = \frac{\text{GPU utilization of } \mathcal{A} - \text{GPU utilization of } \mathcal{B}}{\text{GPU utilization of } \mathcal{B}} \times 100\%. \tag{58}$$

Here, the GPU utilization is computes as the median GPU utilzation throughout the runtime of a given method.