Mitigating the Alignment Tax of RLHF

**Yong Lin[1]*, Hangyu Lin[2]*, Wei Xiong[3]*, Shizhe Diao[4]*, Jianmeng Liu[2], Jipeng Zhang[2], Rui Pan[3],
Haoxiang Wang[3], Wenbin Hu[2], Hanning Zhang[2], Hanze Dong[2], Renjie Pi[2],
Han Zhao[3], Nan Jiang[3], Heng Ji[3], Yuan Yao[2], Tong Zhang[3]**

[1] Princeton University, Princeton Language and Intelligence
[2] The Hong Kong University of Science and Technology
[3] University of Illinois Urbana-Champaign, [4] NVIDIA

Abstract

LLMs acquire a wide range of abilities during pre-training, but aligning LLMs under Reinforcement Learning with Human Feedback (RLHF) can lead to forgetting pretrained abilities, which is also known as the alignment tax. To investigate alignment tax, we conducted experiments with existing RLHF algorithms using OpenLLaMA-3B, which revealed a pronounced alignment tax in NLP tasks. Whereas, despite various techniques to mitigate forgetting, they are often at odds with the RLHF performance, leading to a trade-off between alignment performance and forgetting mitigation, leading to an alignment-forgetting trade-off.

In this paper we show that model averaging, which simply interpolates between pre and post RLHF model weights, surprisingly achieves the most strongest alignment-forgetting Pareto front among a wide range of competing methods. To understand its effectiveness, we offer theoretical insights into model averaging, revealing that it enhances performance Pareto front by increasing feature diversity on the layers where tasks share overlapped feature spaces. Empirical evidence corroborates our analysis by showing the benefits of averaging low-level transformer layers. Building on the analysis and the observation that averaging different layers of the transformer leads to significantly different alignment-forgetting trade-offs, we propose Heterogeneous Model Averaging (HMA) to Heterogeneously find various combination ratios of model layers. HMA seeks to maximize the alignment performance while incurring minimal alignment tax. Moreover, we validate HMA's performance across a range of RLHF algorithms over OpenLLaMA-3B and further extend our findings to Mistral-7B which is evaluated by open-sourced preference model and GPT4. Code available here[1].

---

*indicates equal contributions, random order. Correspond to <hlinbh@connect.ust.hk>

[1] https://github.com/avalonstrel/Mitigating-the-Alignment-Tax-of-RLHF.git

1 Introduction

Large Language Models (LLMs), such as GPT4 (OpenAI, 2023), Bard (Google, 2023), and Claude (Anthropic, 2023), have attracted widespread attention due to their remarkable achievements. LLMs are pre-trained on vast datasets, which equip them with the ability to effectively handle diverse tasks, e.g., GPT-3 showcases its prowess in various tasks such as reasoning, common sense question-answering (QA), translation, and so on.

While LLMs exhibit strong abilities among various benchmarks, they still require alignment with human preferences, including the principles of being helpful, honest, and harmless as outlined by (Askell et al., 2021). The goal is to ensure that LLMs are designed to assist users in completing tasks, provide truthful information without deception, and avoid causing harm, whether physical, psychological, or social, to individuals or the environment. The process of aligning LLMs with human preferences often involves the application of Reinforcement Learning with Human Feedback (RLHF) (Ouyang et al., 2022), as shown in Figure 1. Although RLHF allows LLMs to align with human expectations, prior studies (Askell et al., 2021; OpenAI, 2023; Song et al., 2023) have found that this approach can lead to forgetting in the diverse abilities that the LLMs have already acquired, as illustrated in Figure 1. This phenomenon, also known as the "alignment tax" in the literature, has accumulated substantial attention from both academia and industry (Ouyang et al., 2022; Anthropic, 2023; Askell et al., 2021; Tu et al., 2023; Noukhovitch et al., 2023).

Investigating alignment tax. In this paper, we first conduct a comprehensive investigation on alignment tax and develop methods to reduce alignment tax while maintaining the alignment performance. In particular, we followed the approach presented by (Ouyang et al., 2022) and evaluated align-
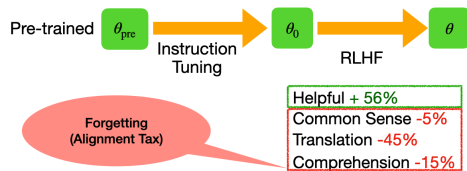
Figure 1: Illustration of RLHF procedure and the alignment tax.

ment tax using multiple NLP benchmarks from common sense QA, such as ARC Easy and Challenge (Clark et al., 2018), Race (Lai et al., 2017), and PIQA (Bisk et al., 2020), reading comprehension benchmarks including SQuAD (Rajpurkar et al., 2018) and DROP (Dua et al., 2019), and translation tasks, including WMT 2014 French to English translation (Bojar et al., 2014) (c.f. Section 3). Our primary focus is on aligning the OpenLLaMA-3B on the helpfulness and harmlessness dataset (Bai et al., 2022) using Rejection Sampling Fine-tuning methods (Dong et al., 2023) (also known as the best-of-n algorithm). In the later part, we extend our experiments to Mistral-7B and Direct Preference Optimization (DPO, (Rafailov et al., 2023)). We mainly focus on RSF and DPO since they are popular and nearly all of the latest open-sourced LLMs on the leaderboards are aligned by these two methods[2]. Indeed, we observed a substantial alignment tax on these benchmarks consistently, confirming the findings of (Ouyang et al., 2022; Gao et al., 2023). Specifically, as we gained a higher reward during RLHF, indicating better alignment with human preference, the alignment tax also increased simultaneously, clearly inducing a *alignment-forgetting trade-off*.

Surprising effectiveness of model averaging over. We then compare various methods developed in different communities as potential rescues to alleviate the alignment tax. This includes the model averaging method (Wortsman et al., 2022b,a; Lin et al., 2023) from out-of-distribution (OOD) generalization literature, regularization-based techniques from the continual learning literature (Panigrahi et al., 2023; Xuhong et al., 2018; Buzzega et al., 2020; Huang et al., 2021), low-rank adaptation (LoRA) (Huang et al., 2021) from the parameter-efficient fine-tuning literature, as well as the utilization of reward penalty from the reinforcement learning literature (Ziegler et al., 2019; Wu et al., 2021a; Ouyang et al., 2022; Yuan et al., 2023). Interestingly, we found that model averaging, which

simply interpolates between the weights of models before and after RLHF, achieves the most efficient alignment-forgetting Pareto front. In Appendix C.1, we further show and discuss the in-effectiveness of Experience Reply (Rebuffi et al.) method compared with MA.

Understanding the effectiveness of model averaging. To understand the effectiveness of model averaging, we provide theoretical insights based on the framework of (Lin et al., 2023). In particular, we show that the method can enhance Pareto front by increasing feature diversity on layers where two tasks share similar feature spaces. Empirical evidence also indicates that averaging the low-level layers of Transformers consistently improves both alignment reward and NLP task performance. This aligns with our theoretical insights, as tasks could share similar lower-level features, e.g., better word representation on low-level layers benefits both NLP and alignment tasks.

Heterogeneous model averaging. We noticed that averaging different layers of the Transformers unveiled notably distinct patterns of alignment-forgetting trade-off, aligning with our earlier analysis that tasks may exhibit varying overlapping feature spaces in different layers. Motivated by this observation, we propose Heterogeneous Model Averaging (HMA), which adaptively averages different parts of the models during model averaging. We start by dividing the transformer into K parts and assigning unique averaging ratios for each part, represented as $\alpha_i \in [0, 1]$ for the ith part. HMA aims to maximize alignment reward by optimizing the averaging ratios $(\alpha_1, \ldots, \alpha_K)$ while maintaining the overall alignment tax, thus consistently improve the alignment-forgetting Pareto front. To demonstrate the efficiency of HMA, we also contrasted our method with other RLHF techniques, including Direct Preference Optimization (DPO). (Rafailov et al., 2023) We further substantiate our findings on Mistral-7B where evaluations conducted by open sourced perference model and GPT4, which further corroborates our empirical findings on OpenLLaMA-3B.

We summarize our contributions as follows:

- We provide a comprehensive investigation of the alignment tax challenge in RLHF on NLP tasks. We systematically compare a wide range of methods to alleviate alignment tax and highlight model averaging as a particularly effective approach.

- We provide theoretical insights into the efficiency of model averaging in enhancing the alignment-forgetting trade-off, demonstrating that both NLP and alignment tasks can benefit from the increased feature diversity from model averaging in the shared feature space.

- Motivated by our analysis, we introduce Heterogeneous Model Averaging (HMA), which optimizes the averaging ratios of different model layers to maximize alignment performance. HMA consistently improves the Pareto front across different benchmarks, and it also generalizes well across various RLHF algorithms and different model types, such as OpenLLaMA-3B and Mistral-7B, evaluated by open-sourced preference model and GPT4.

The paper is structured as follows: we conduct a systematic investigation of existing methods in Section 3-4. In Section 5, we provide insights into the effectiveness of model averaging. Subsequently, we propose Heterogeneous Model Averaging in Section 6. We conclude the paper in Section 7.

2 Discussion with existing works.

In this section, we provide comparison of this work with existing works to highlight the novelty of our findings. We defer more comprehensive related works to Appendix A.

Existing works of model averaging for LLMs. Previous research has covered certain aspects of model averaging. (Ramé et al., 2024) demonstrate the utilization of model averaging to construct a more resilient reward model for reinforcement learning with human feedback (RLHF). In a similar vein, (Rame et al., 2024) employ model averaging to merge policy models trained for distinct objectives, facilitating multi-objective RLHF. (Sanyal et al., 2023) introduce the integration of moving averaging to enhance pre-training. However, none of these studies investigate the alignment tax, and their findings are independent of our research.

Existing works on finding adaptive combinations for model merging. Previous studies (Yang et al., 2023; Akiba et al., 2024) have also discussed the idea of dynamically assigning different weights to different layers when merging models, aiming to maximize performance on a specific task (e.g., $\mathcal{T}_i$). These approaches assume access to the task-specific data $\mathcal{T}_i$. However, considering the nature of alleviating alignment tax, which aims to mitigate forgetting across a extremely wide range of

tasks ($\mathcal{T}_{j_1}...\mathcal{T}_{j_K}$), these methods fail to effectively optimize performance for multiple tasks simultaneously. In the Appendix E.4, we demonstrate that using the method proposed by (Yang et al., 2023), which optimizes for a single task, does not effectively address forgetting on the other tasks. Furthermore, our work is the first to provide an explanation for the surprising effectiveness of model averaging in alleviating forgetting, as well why we should assign heterogeneous combination ratios.

Existing works on the forgetting of language models. Most research on forgetting in language models focuses on sequentially pre-training (Chen et al., 2023; Gong et al., 2022; Jin et al., 2021; Qin et al., 2022; Liu et al., 2021) or fine-tuning tasks (Sun et al., 2019; Razdaibiedina et al., 2023; Wu et al., 2021b; Zhang et al., 2022; Madotto et al., 2020), e.g., sequentially training on task $\mathcal{T}_i$ and then task $\mathcal{T}_j$. They evaluate forgetting by measuring the model's performance on a task (e.g., task $\mathcal{T}_i$) after training it on another task (e.g., task $\mathcal{T}_j$). However, these methods have not explored the effectiveness of model averaging. In our case, we demonstrate the significant power of model averaging which outperform a wide range of existing methods. Furthermore, existing works assume that the data size of each task is comparable (i.e., the dataset size of $\mathcal{T}_i$ and $\mathcal{T}_j$ is similar), allowing for a subset (e.g., 10%) of old task data replay, which is shown to effective alleviate the forgetting without excessive computation overhead in their settings. However, in our alignment tax situation, we aim to preserve a wide range of abilities gained during pre-training, which is challenging since pre-training datasets are often not publicly available. In Appendix C.1, we show that even when we have access to the pre-training data and replay a subset up to four times larger than the RLHF data (which costs significant computation overhead), experience replay still under-performs model averaging in two out of three benchmarks. This is likely due to the vast size of the pre-training data, where the subset only covers a small fraction of it (e.g., only covers ~0.01% of the pre-training data). So replay methods are less practical for alleviating alignment tax.

3 Experimental Settings

Basic Setting. We chose the OpenLLaMA-3B model (Geng and Liu, 2023) because (1) it is computational friendly compared with 7B models (2) it has openly available pre-training dataset,

which is convenient to investigate Experience Replay in Appendix C.1. Furthermore, we extend the experiments to Mistral-7B in Sec. 6. Following the standard procedure outlined in (Ouyang et al., 2022), we initially conducted instruction tuning, followed by RLHF. Here, θ represents an LLM with parameters θ, with the pre-trained model denoted as θ_{pre}. We commenced with instruction fine-tuning for θ_{pre} on ShareGPT [3], which yielded θ_0. Subsequently, RLHF was performed on θ_0 to obtain θ. Similar to the methodology proposed in (Ouyang et al., 2022), the alignment tax was evaluated by comparing the performance regression of θ with θ_0 across various NLP tasks. The whole procedure and notations are illustrated in Fig. 1.

Datasets for Evaluating Alignment Tax. Following the approach in (Ouyang et al., 2022), our evaluation of alignment tax encompasses various NLP benchmarks: (a) Common Sense QA: This includes ARC Easy and Challenge (Clark et al., 2018), Race (Lai et al., 2017), and PIQA (Bisk et al., 2020), with the performance being assessed using accuracy. (b) Reading Comprehension: we employ SQuAD (Rajpurkar et al., 2018) and DROP (Dua et al., 2019) to gauge reading comprehension ability, with evaluation based on the F1 score for both datasets. (c) Translation: Our evaluation utilizes WMT 2014 French to English translation (Bojar et al., 2014), with performance measured using BLEU (Papineni et al., 2002) scoring.

RLHF Basics. In our notation, π_θ denotes the policy induced by the LLM θ. Additionally, x represents the input prompt and a denotes the output (which is also referred to as an action in RL literature (Schulman et al., 2017)). Drawing from (Ouyang et al., 2022; Bai et al., 2022; Dong et al., 2023; Touvron et al., 2023; Rafailov et al., 2023), we assume the existence of a ground-truth reward function $r^*(x, a) : \mathcal{X} \times \mathcal{A} \to [0, 1]$, where $\mathcal{X}$ and $\mathcal{A}$ denote the spaces of x and a respectively. The primary objective of RLHF is to maximize:

$$\max_\theta \mathbb{E}_x \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}[r^*(x, a)]. \tag{1}$$

RLHF Algorithm. We adopt Rejection Sampling Finetuning (RSF) for our main experiments (Dong et al., 2023; Touvron et al., 2023; Yuan et al., 2023; Gulcehre et al., 2023) and also further verify our findings on Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Direct Preference Optimization (DPO) (Rafailov et al., 2023)
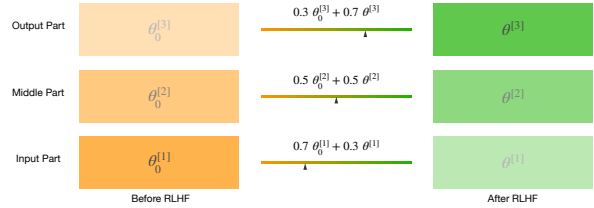
Figure 2: Illustration of Heterogeneous Model Averaging (HMA) when $K = 3$.

in Sec. 6. Essentially, the RSF learns from the best-of-n policy (Nakano et al., 2021), which samples n responses for each prompt query and returns the one with the highest reward. As suggested by (Dong et al., 2023; Touvron et al., 2023; Gulcehre et al., 2023), we adopt an iterative training set-up for the implementation instead of always sampling the samples from the starting checkpoint because we find that the iterative training is far more sample-efficient. Specifically, for each iteration, we first sample a batch of prompts and generate n responses for each prompt from the current model. Then, we use the reward model to compute the rewards for each prompt-response pair, and for each prompt, we select the one with the highest reward into a small subset. By this process, we collect a batch of samples from the best-of-n policy that are with high reward. We simply fine-tune the current model on this subset to get the next model and the next iteration begins.

4 Evaluating Existing Methods

In Figure 12 of Appendix E.1, we visualize the training procedure in terms of the alignment-forgetting trade-off during RLHF. Specifically, we can clearly see that as the RLHF proceeds, the reward begins to increase while the translation and reading comprehension ability continues to drop. Interestingly, we observe that the performance of common sense increases first and then drops. Given that alignment tax is inherently a catastrophic forgetting issue, we then proceed to explore methods to reduce alignment tax. Research focused on reducing forgetting is mainly classified into two main categories, depending on the availability of the pre-training dataset. We also investigate the reward penalty method developed in RL community in Appendix C.2.

4.1 Basic Methods

To explore methods for alleviating alignment tax, we initially examine solutions that do not rely on pre-training datasets. These methods encompass

the following:(a) Early stopping. (b) Regularization towards θ_0 in the weight space as follows:

$$\max_\theta \mathbb{E}_x \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}[r^*(x,a)] + \lambda\|\theta - \theta_0\|_\alpha, \quad (2)$$

where we use $\alpha = 1, 2$ which corresponds to the L1 and L2 (Xuhong et al., 2018) penalties, respectively. (c) Low-Rank Adaptation (LoRA) (Hu et al., 2021). It introduces trainable rank decomposition matrices into linear layers to update $\theta - \theta_0$ during RLHF. (d) Knowledge distillation (Buzzega et al., 2020; Huang et al., 2021). We use π_{θ_0} serves as the teacher and π_θ as the student, with a penalty imposed as:

$$\max_\theta \mathbb{E}_x \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}[r^*(x,a)] + \lambda\|\pi_\theta(\boldsymbol{x}) - \pi_{\theta_0}(\boldsymbol{x})\|_2^2.$$

(e) Model Averaging (MA) (Wortsman et al., 2022a,b). This involves simply interpolating between θ_0 and θ to yield the policy $\pi_{(1-\alpha)\theta_0+\alpha\theta}$, where α is a hyper-parameter ranging from 0 to 1. (f) Stochastic Moving Averaging (SMA) (Noukhovitch et al., 2024). More implementation details are provided in the appendix.

Results. Figure 3 depicts the performance of each aforementioned method. The results demonstrate that these approaches effectively alleviate the alignment tax; however, they also result in a reduction in the RLHF reward, indicating a clear trade-off between reward and alignment tax. Notably, despite its simplicity, the Pareto-front of model averaging supersedes nearly all other methods across various hyper-parameters. In Appendix C.1 and C.2, we compared model averaging with Experience Replay (ER) and KL reward penalty methods for Proximal policy optimization (Schulman et al., 2017) algorithms, the conclusions are similar.

5 Unravelling the Mysteries of Model Averaging for Alleviating Alignment Tax

Given the promising performance of model averaging, we try to understand the efficacy of model averaging in this Section and motivate our method to improve it. We utilize the theoretical framework proposed by (Lin et al., 2023) to gain insights into its effectiveness in alignment tax. While the framework addresses classification problems, the insights derived can aid our understanding of model averaging. We also conduct empirical analysis using a generative model (Openllama-3B) to verify these theoretical insights. Analyzing the performance

of model averaging in alignment tax is more intricate compared to the work of the study by (Lin et al., 2023) focuses on out-of-distribution (OOD) scenarios, where the same task is performed under different distributions. In contrast, our focus in alignment tax is to comprehend the performance trade-offs among different tasks. To illustrate, consider the entire feature space $\mathcal{Y}$ and two tasks with label spaces $\mathcal{Y}_a \subset \mathcal{Y}$ and $\mathcal{Y}_b \subset \mathcal{Y}$, with the simplifying assumption that $|\mathcal{Y}_a| = |\mathcal{Y}_b| = K$. While (Lin et al., 2023) only considers the case where $\mathcal{Y}_a = \mathcal{Y}_b$, we extend these results to encompass the case where $\mathcal{Y}_a \neq \mathcal{Y}_b$.

Theoretical Settings. Suppose we have many features $\mathcal{S}_x = \{\boldsymbol{x}_i\}_{i=1}^D$ where each feature $\boldsymbol{x}_i \in \mathbb{R}^d$ and the observed feature $\boldsymbol{x} \in \mathbb{R}^{d\times D}$ is a concatenation of $\boldsymbol{x}_1, ..., \boldsymbol{x}_D$. Following (Lin et al., 2023), we adopt a simplified model $f(\boldsymbol{x}) = w\Phi(\boldsymbol{x})$ where $w \in \mathbb{R}^{d\times K}$, $\Phi(\boldsymbol{x}) = \sum_{i=1}^D \Phi_i \boldsymbol{x}_i$ and $\Phi_i \in \{0,1\}, \forall i$. Suppose we have two models $f_a(\cdot) = w_a\Phi_a(\cdot)$ and $f_b = w_b\Phi_b(\cdot)$ for tasks $\mathcal{T}_a$ and $\mathcal{T}_b$, respectively, relying on feature sets $\mathcal{S}_{x,a} \subset \mathcal{S}_x$ and $\mathcal{S}_{x,b} \subset \mathcal{S}_x$, with $|\mathcal{S}_{x,a}| = |\mathcal{S}_{x,b}| = n$, and $|\mathcal{S}_{x,a} \cap \mathcal{S}_{x,a}| = n_o$ overlapped features. The averaged model of f_a and f_b is $f_{\text{avg}}(\cdot) = w_{\text{avg}}\Phi_{\text{avg}}(\cdot)$, where $w_{\text{avg}} = (w_a + w_b)/2$ and $\Phi_{\text{avg},i} = (\Phi_{a,i} + \Phi_{b,i})/2, \forall i$ (Lin et al., 2023). To gain an intuitive understanding, we compare model averaging in two cases: Case (1) when the tasks are quite similar ($|\mathcal{Y}_A \cap \mathcal{Y}_B| = K$) and Case (2) when the tasks are independent ($|\mathcal{Y}_A \cap \mathcal{Y}_B| = 0$). [4] Furthermore, even if the tasks are very similar, fitting two models on them can rely on different features due to randomness in data or training procedures (Lin et al., 2023; Allen-Zhu and Li, 2020). We will investigate the performance of model averaging in Case (1) and (2) to gain insights on when it works. Following (Lin et al., 2023), we assume each feature is weak, failing with probability p. The effectiveness of model averaging is given by

$$\xi = \frac{1}{2}\left(\mathcal{A}_a(f_{\text{avg}}) - \mathcal{A}_a(f_a) + \mathcal{A}_b(f_{\text{avg}}) - \mathcal{A}_b(f_b)\right),$$

where $\mathcal{A}_a(f)$ and $\mathcal{A}_b(f)$ denote the accuracy of f on task a and b, respectively. We use $\xi^{(1)}$ to denote the effective averaging robustness for Case (1) and similarly define $\xi^{(2)}$ for Case (2).

---

[4] Notably, the overlap in features is independent of the overlap in label space. For instance, when classifying a dog, we can use either the animal shape or the texture (overlapped label space, non-overlapped feature); when classifying a dog or a cat, we can both use the animal shape (non-overlapped label space, overlapped feature).
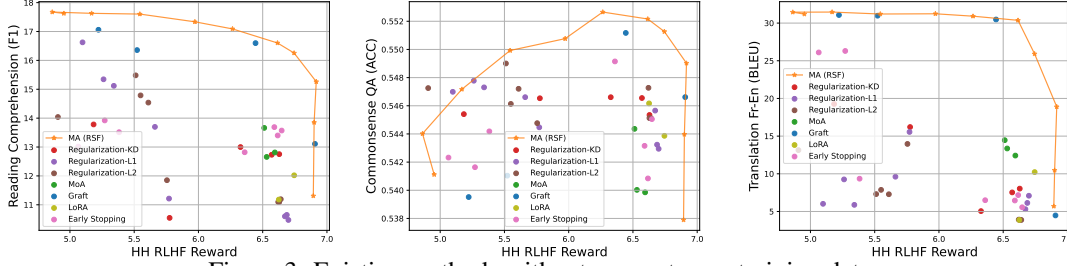
Figure 3: Existing methods without access to pre-training data

Proposition 5.1. *Consider the assumptions specified in the appendix. We have:*

$$\xi^{(1)} - \xi^{(2)} = F_p\left(\frac{\sqrt{2}(1-p)n}{\sqrt{n+n_o}}\right) - F_p\left((1-p)\sqrt{n}\right) \geq 0,$$

where the equality holds when $n_o = n$ and $F_p(x)$ is a cumulative density function in Appendix F.4.

Implications. Proposition 5.1 demonstrates that when $\mathcal{T}_a$ and $\mathcal{T}_b$ are more similar, the averaging of models (f_a and f_b) yields greater improvement. However, this improvement is reduced if f_a and f_b use more overlapping features. Recall that each weak feature can fail with probability p. If $\mathcal{T}_a$ and $\mathcal{T}_b$ are similar, the feature utilized by the two models would be projected into a shared space, allowing model averaging to take advantage of a more diverse set of features. This diversity reduces the probability of model failure because a diverse set of features is less likely to fail together simultaneously (Lin et al., 2023). However, if $\mathcal{T}_a$ and $\mathcal{T}_b$ are dissimilar, for example, if $|\mathcal{Y}_a \cap \mathcal{Y}_b| = 0$ and the feature spaces corresponding to $\mathcal{Y}_a$ and $\mathcal{Y}_b$ are disjoint, then the features in the space of $\mathcal{Y}_a$ would not provide any information for predicting $\mathcal{Y}_b$. Therefore, averaging f_a and f_b would not improve the prediction of either task in this case. Refer to Appendix F.3 for a detailed discussion.

Notably, the model θ_0 excels in NLP abilities before RLHF, while the model θ excels in alignment reward after RLHF. Using an analogy, we can equate NLP tasks with $\mathcal{T}_a$, alignment with $\mathcal{T}_b$, θ_0 to f_a, and θ to f_b. Recall that we adopt a simplified model for theoretical analysis by considering only one layer feature learner, although, in practice, we average a deep Transformer with 26 layers. Research has shown that different layers in deep neural networks capture varying levels of features (Yosinski et al., 2015; Zeiler and Fergus, 2014; Simonyan and Zisserman, 2014). For instance, low-level layers capture low-level features.

Furthermore, tasks share similar feature space at a low level (alternatively, from the perspective of low-level layers, tasks look more similar). For example, improving the low-level features such as better word representation could enhance both RLHF reward and NLP tasks. Therefore, according to Proposition 5.1, averaging the low-level layers could potentially elicit more improvements in both $\mathcal{T}_a$ (NLP tasks) and $\mathcal{T}_b$ (alignment reward) than higher layers.

Empirical Validation. We categorize the 26 transformer layers of Openllama into three parts: the input part (layers 1-8), the middle part (layers 9-17), and the output part (layers 18-26). This division is depicted in Figure 4. We use the superscripts [1], [2], and [3] to denote the input, middle, and output parts, respectively. For instance, $\theta^{[2]}$ represents the middle layers (9-18) of θ. Here, θ_0 and θ respectively refer to the models before and after RLHF. We investigate the impact of averaging one part instead of the whole Transformer: given a combination ratio $\alpha \in [0, 1]$, we average the i-th part of θ (i.e., $\theta^{[i]}$) with the corresponding part of θ_0 (i.e., $\theta_0^{[i]}$), while keeping the remaining two parts of θ unchanged. So when we average the input part, the j-th part of the averaged model is:

$$j\text{th part} = \begin{cases} \alpha\theta^{[j]} + (1-\alpha)\theta_0^{[j]}, & \text{if } j = 1, \\ \theta^{[j]}, & \text{if } j = 2, 3. \end{cases}$$

The results of the above scheme are denoted as "Input Part MA". "Middle Part MA" and "Output Part MA" represent that we average the middle and output parts, respectively. Figure 4 illustrates that the alignment-forgetting trade-off varies distinctly when different parts of the transformers are averaged. Specifically, when we average the low-level layers, we observe a "magical" improvement in both the NLP tasks and alignment rewards, which is consistent with our previous analysis. Furthermore, we show results in Appendix E.2 that the magical improvement in averaging the low-level parts is consistent among DPO and PPO models.
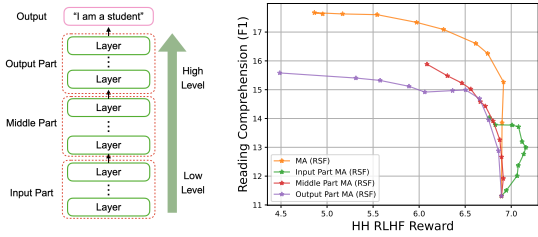
Figure 4: (Left) Illustration of proof of concept experiments. We divide the Transformer into 3 parts. We only average one part each time. (Right) Merging different parts of the transformers.

6 Heterogeneous Model Averaging

We have already shown that averaging different layers results in diverse patterns of alignment-forgetting trade-off (Wu et al., 2022; Lee et al., 2022b). Therefore, different layers should not be equally treated during averaging. This leads to a natural question: can we enhance the alignment-forgetting trade-off by using adaptive weights for different layers? Consequently, we conduct proof-of-concept experiments to provide affirmative answers to this question and subsequently propose a practical algorithm.

Proof of Concept. The following proof of concept experiments provide insights into average different layers with various ratios. We use different averaging ratio, i.e., $\alpha_1, \alpha_2, \alpha_3$, for the three parts. Specifically, the ith part of the averaged model is simply $\alpha_i \theta^{[i]} + (1 - \alpha_i)\theta_0^{[i]}$. We try three patterns experiment given a base $\alpha \in \{0.2, 0.3, 0.4\}$: (a) $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$; (b) $\alpha_1 = \alpha_2 = \alpha$, $\alpha_3 = \alpha - 0.1$, and (c) $\alpha_1 = \alpha$, $\alpha_2 = \alpha_3 = \alpha - 0.1$. We use $(\alpha|\alpha|\alpha)$, $(\alpha|\alpha|\alpha-0.1)$ and $(\alpha|\alpha-0.1|\alpha-0.1)$ to denote these three patterns, respectively. These results confirm that certain ratio combinations exceed the trade-off curve of vanilla model averaging, as displayed in Figure 9 in Appendix C.3. Notably, some combination ratios consistently outperform the equal ratio across various benchmarks. This affirms the potential to identify consistent combination ratios that demonstrate superior performance across a broad spectrum of benchmarks in terms of alignment-forgetting trade-off.

Heterogeneous Model Averaging. Upon dividing the Transformer into K parts, our objective is to adaptively determine a combination ratio for different layers that consistently perform well across an extensive range of tasks. The conventional averaging method uses a shared α for all layers, playing a crucial role in defining the trade-off between re-

ward and tax. We aim to identify an optimized combination of $(\alpha_1, ..., \alpha_K)$ to replace a uniform α. Let $\theta(K)$ represent the model merged by $(\alpha_1, ..., \alpha_K)$. In particular, the kth component of the merged model $\theta(K)$ is given by

$$\theta^{[k]}(K) := \alpha_k \theta^{[k]} + (1 - \alpha_k)\theta_0^{[k]}, \forall k \in 1, ..., K.$$

To optimize the Pareto-front influenced by α, we identify combination ratios corresponding to each α. Subsequently, we establish the mean of $(\alpha_1, ..., \alpha_K)$ as α and ascertain the best combination of $(\alpha_1, ..., \alpha_K)$ to maximize the reward. Specifically, denoting $\Omega := \left\{ \frac{1}{K} \sum_k \alpha_k = \alpha, \alpha_1, ..., \alpha_K \in [0, 1] \right\}$, we solve:

$$\max_{(\alpha_1, ..., \alpha_K) \in \Omega} \mathbb{E}_x \mathbb{E}_{a \sim \pi_{\theta(K)}(\cdot|x)} \left[r^*(x, a) \right]. \quad (3)$$

The intuition behind HMA is outlined as follows: (1) When maintaining the mean, i.e., $\frac{1}{K} \sum_k \alpha_k$, as α, we can compare HMA performance with the performance of vanilla model averaging with the same α. (b) We only optimize K parameters, where K is typically small. For example, we adopt $K = 3$ by default and also include results with varying K to the ablation study. This helps to ensure that the forgetting level of $(\alpha_1, ..., \alpha_K)$ remains close to α. Intuitively, if we optimize a large number of parameters, it could easily lead to over-fitting in the in-domain (RLHF reward) and may also result in more significant forgetting. The whole algorithm is summarized Algorithm 1 in appendix.

Results. The results of HMA are shown in Figure 5. We can see that HMA can consistently push forward the Perato-front of the vanilla model averaging. Furthermore, such improvement is consistent over various RLHF algorithms. More detailed results (e.g., on Commonsense QA and Translation with different RLHF algorithms) of HMA can be found in Appendix E.5.

Ablation results on different K. We tested different values of K with $\alpha = 0.2, 0.4, 0.6$ as illustrated in Figure 5 (Right). The trade-off curve shows a slight decrease as we increase K from 3 to 6 and 9, but still consistently improves over the vanilla model averaging. This decrease is likely due to overfitting. Specifically, comparing the performance of HMA with different K for the same mean ratio, we observe that as the alignment reward increases with an increase in K from 3 to 9, the reading comprehension performance drops.

How to choose the averaging ratio. In practice, we determine the averaging ratio α for adopt-
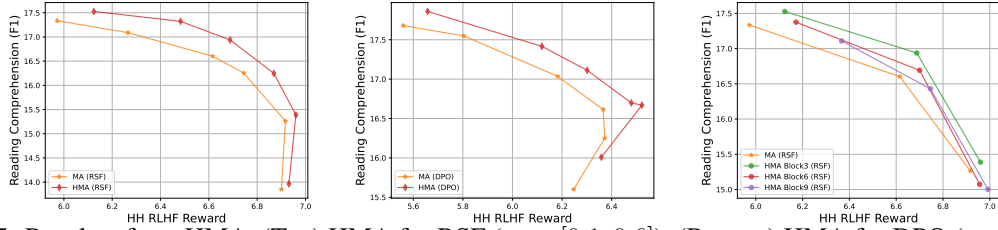
Figure 5: Results of our HMA. (Top) HMA for RSF ($\alpha \in [0.1, 0.6]$), (Bottom) HMA for DPO ($\alpha \in [0.1, 0.6]$). (Right) HMA for RSF with different choices of K. Refer to Appendix E.5 for more results.
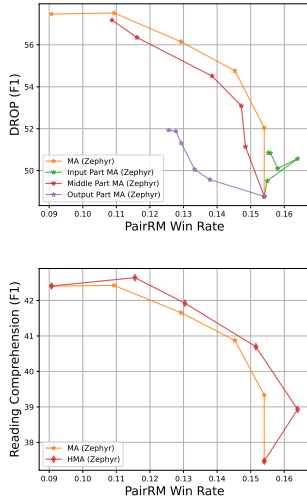

Figure 6: Results of Zephyr-7B-β evaluated by open sourced preference model. (Top) Similar trends evaluated by PairRM when we average different blocks. (Bottom) Our HMA consistently improve over MA.

ing vanilla MA or our HMA. Changing the averaging ratio for MA and HMA is convenient as these methods are applied after training the vanilla RLHF checkpoint. The comprehensive results in Figures 3, 5, and 16 (details in Appendix C.4) show that $\alpha = 0.2$ can consistently alleviate the alignment tax without hurting alignment performance. Further results of Zephyr-7B are shown in Figure 6. Additionally, the performance of the averaging ratio on different benchmarks (Figure 9) exhibits similar trends. Hence, we believe $\alpha = 0.2$ is a suitable choice that can generalize to more tasks.

Model	Win-Rate	Reading	CommonSense	Trans
Zephyr-7B-β	8.10%	37.47	66.34	36.55
HMA (Ours)	**9.32%**	**38.93**	**66.55**	**37.23**
Zephyr-7B-Gemma	11.3%	41.15	66.3	38.09
HMA (Ours)	**11.5%**	**42.45**	**66.4**	**38.71**

Table 1: GPT4 evaluation of experiments of Zephyr-7B-β and Zephyr-7B-Gemma on Alpaca benchmark. Reading is short for Reading Comprehension, which is evaluated by F1. CommonSence is evaluated by Accuracy (%). Trans is short for Translation Fr-En, evaluated by BLEU.

Other models results. To further validate our method on larger LLMs, e.g., Mistral-7B (Jiang et al., 2023a) based models, we apply model averaging (MA) and Heterogeneous Model Averaging (HMA) on Zephyr-7B-β[5] (Tunstall et al., 2023) which is trained with DPO on the SFT version, Mistral-7B-SFT-β[6]. We also apply HMA on Zephyr-7B-Gemma [7] which is aligned based on Gemma-7B[8] model. Here we use the the publicly available preference model PairRM (Jiang et al., 2023b) to judge the helpfulness and evaluate models on AlpacaEval 2.0 (Li et al., 2023). We reports the win rates of each model. Figure 6 (Top) shows that the trends of averaging different layers evaluated by PairRM are similar with the results evaluated by our own reward model. The results range across $\alpha = 0, 0.2, \ldots, 1.0$ depicted in Figure 6 (Bottom) demonstrate that MA effectively achieves a strong Pareto front to mitigate forgetting in the Mistral-7B models. Additionally, our HMA algorithm shows further improvement compared to the MA method.

GPT4 Evaluation. We also use GPT4 to evaluate HMA on AlpacaEval 2.0 (Li et al., 2023). Due to the limited quota, we only compare HMA with $\alpha = 0.2$ with vanilla Zephyr-7B-β ($\alpha = 0.2$ is recommended by the previous discussion). In Table 1, we summarize their Win-Rate against GPT4 as well as their performance on NLP tasks. We show that HMA consistently outperforms Zephyr-7B-β on all the metrics.

7 Conclusion

In this paper, we highlight the surprisingly effectiveness of model averaging and propose the Heterogeneous Model Averaging (HMA) framework to further enhance the performance.

---

[5] https://huggingface.co/HuggingFaceH4/zephyr-7b-beta
[6] https://huggingface.co/HuggingFaceH4/mistral-7b-sft-beta
[7] https://huggingface.co/HuggingFaceH4/zephyr-7b-gemma-v0.1
[8] https://huggingface.co/google/gemma-7b

Limitations

Though our HMA significantly alleviates the alignment tax, it has not been fully eliminated. Future work could explore the theoretical lower bound of the alignment tax and determine which method could achieve the optimal trade-off.

References

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2024. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154.

Zeyuan Allen-Zhu and Yuanzhi Li. 2020. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*.

Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. 2021. The evolution of out-of-distribution robustness throughout fine-tuning. *arXiv preprint arXiv:2106.15831*.

Anthropic. 2023. Introducing claude.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. 2021. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2104.05025*.

Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. 2020. Online learned continual compression with adaptive quantization modules. In *International Conference on Machine Learning*, pages 1240–1250. PMLR.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*.

Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. 2021a. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 9516–9525.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. 2021b. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. 2023. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pages 5383–5395. PMLR.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2019. On the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1907.01752*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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

Xu Chu, Yujie Jin, Wenwu Zhu, Yasha Wang, Xin Wang, Shanghang Zhang, and Hong Mei. 2022. Dna: Domain generalization with diversified neural averaging. In *International Conference on Machine Learning*, pages 4010–4034. PMLR.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. *arXiv preprint arXiv:2306.12420*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2020. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Zheng Gong, Kun Zhou, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2022. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5923–5933.

Google. 2023. Bard.

Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. 2022. Finetune like you pretrain: Improved finetuning of zero-shot vision models. *arXiv preprint arXiv:2212.00638*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. *arXiv preprint arXiv:2104.05489*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. 2022a. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. 2022b. Surgical fine-tuning improves adaptation to distribution shifts. *ArXiv*, abs/2210.11466.

Shengzhi Li, Rongyu Lin, and Shichao Pei. 2024. Multimodal preference alignment remedies regression of visual instruction tuning on language model. *arXiv preprint arXiv:2402.10884*.

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

Yong Lin, Hanze Dong, Hao Wang, and Tong Zhang. 2022a. Bayesian invariant risk minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16021–16030.

Yong Lin, Lu Tan, Yifan Hao, Honam Wong, Hanze Dong, Weizhong Zhang, Yujiu Yang, and Tong Zhang. 2023. Spurious feature diversification improves out-of-distribution generalization. *arXiv preprint arXiv:2309.17230*.

Yong Lin, Shengyu Zhu, Lu Tan, and Peng Cui. 2022b. Zin: When and how to learn invariance without environment partition? *Advances in Neural Information Processing Systems*, 35:24529–24542.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

Zihan Liu, Genta Indra Winata, and Pascale Fung. 2021. Continual mixed-language pre-training for extremely low-resource neural machine translation. *arXiv preprint arXiv:2105.03953*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Michael Noukhovitch, Samuel Lavoie, Florian Strub, and Aaron Courville. 2023. Language model alignment with elastic reset. *arXiv preprint arXiv:2312.07551*.

Michael Noukhovitch, Samuel Lavoie, Florian Strub, and Aaron C Courville. 2024. Language model alignment with elastic reset. *Advances in Neural Information Processing Systems*, 36.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

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

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. *arXiv preprint arXiv:2203.06311*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. 2024. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. 2024. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, Christoph H Lampert, and icarl. Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2018. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Sunny Sanyal, Atula Tejaswi Neerkaje, Jean Kaddour, Abhishek Kumar, et al. 2023. Early weight averaging meets high learning rates for llm pre-training. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Ziang Song, Tianle Cai, Jason D Lee, and Weijie J Su. 2023. Reward collapse in aligning large language models. *arXiv preprint arXiv:2305.17608*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

Xiaoyu Tan, LIN Yong, Shengyu Zhu, Chao Qu, Xihe Qiu, Xu Yinghui, Peng Cui, and Yuan Qi. 2023. Provably invariant learning without domain information.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Haoqin Tu, Bingchen Zhao, Chen Wei, and Cihang Xie. 2023. Sight beyond text: Multi-modal training enhances llms in truthfulness and ethics. *arXiv preprint arXiv:2309.07120*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Jeffrey S Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen. 2023a. Beyond reverse kl: Generalizing direct preference optimization with diverse divergence constraints. *arXiv preprint arXiv:2309.16240*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023b. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022a. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. 2022b. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. 2021. Robust fine-tuning of zero-shot models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7949–7961.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021a. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2021b. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2022. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Wei Xiong, Hanze Dong, Chen Ye, Han Zhong, Nan Jiang, and Tong Zhang. 2023. Gibbs sampling from human feedback: A provable kl- constrained framework for rlhf.

LI Xuhong, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pages 2825–2834. PMLR.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.

Pengfei Yu and Heng Ji. 2023. Self information update for large language models through mitigating exposure bias. In *arxiv*.

Pengfei Yu, Heng Ji, and Premkumar Natarajan. 2021. Lifelong event detection with knowledge transfer. In *Proc. The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP2021)*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

Michael Zhang and Christopher Ré. 2022. Contrastive adapters for foundation model group robustness. *arXiv preprint arXiv:2207.07180*.

Tong Zhang. 2023. *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022. Continual sequence generation with adaptive compositional modules. *arXiv preprint arXiv:2203.10652*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*.

Xiao Zhou, Yong Lin, Renjie Pi, Weizhong Zhang, Renzhe Xu, Peng Cui, and Tong Zhang. 2022a. Model agnostic sample reweighting for out-of-distribution learning. In *International Conference on Machine Learning*, pages 27203–27221. PMLR.

Xiao Zhou, Yong Lin, Weizhong Zhang, and Tong Zhang. 2022b. Sparse invariant risk minimization. In *International Conference on Machine Learning*, pages 27222–27244. PMLR.

Banghua Zhu, Jiantao Jiao, and Michael I Jordan. 2023. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Related Work

Large Language Models. Large Language Models (LLMs) are pre-trained using vast amounts of data and has the ability to handle a diverse set of tasks. An excellent line of LLMs includes GPT (Brown et al., 2020; OpenAI, 2023), Bard (Google, 2023), Claude (Anthropic, 2023), LLaMA (Touvron et al., 2023), Galactica (Taylor et al., 2022), Bloom (Scao et al., 2022). It is a common practice to fine-tune the LLMs to obtain better performance on a specific task (Diao et al., 2023), follow the instruction of humans (Ouyang et al., 2022; Sanh et al., 2021; Wang et al., 2022) and align with humans' preferences (Christiano et al., 2017; Askell et al., 2021; Bai et al., 2022; Ouyang et al., 2022; Dong et al., 2023).

Reinforcement Learning with Human Preference (RLHF). RLHF (Christiano et al., 2017) has attracted considerable attention in the past few years, particularly after the tremendous success of the ChatGPT (Ouyang et al., 2022; OpenAI, 2023). There is a rich literature on RLHF and the related discussions which cannot be comprehensively reviewed here due to the space constraint. We thus refer the interested readers to the survey paper like (Casper et al., 2023) but focus on the algorithmic designs here. Proximal Policy Optimization (PPO) (Schulman et al., 2017) is the predominant approach in RLHF whose effectiveness has been showcased by ChatGPT (OpenAI, 2023), Claude (Anthropic, 2023), and Bard (Google, 2023). However, it is known that the PPO is unstable and sample-inefficient in aligning LLMs (Choshen et al., 2019) and imposes a heavy burden on the GPU resources as it requires loading multiple (typically four) models at the same time (Yuan et al., 2023; Dong et al., 2023). In view of this, attempts have been made to propose alternative approaches to the PPO algorithm. There is a line of work using the rejection sampling (also referred to as the best-of-n sampling in the literature) (Nakano et al., 2021), to reinforce the dataset used to finetune the LLM, including (Dong et al., 2023; Yuan et al., 2023; Touvron et al., 2023; Gulcehre et al., 2023). Among them, (Dong et al., 2023; Touvron et al., 2023; Gulcehre et al., 2023) adopt an iterative framework, which is more sample-efficient and effective, while (Yuan et al., 2023) highlights the importance of sampling strategy. In comparison to the original rejection sampling algorithm, which generates n responses but only output the one with the highest reward, the LLMs aligned by iterative rejection sampling balance the goal of alignment and the inference cost. Meanwhile, there is also another line of work aiming to derive algorithms from the reverse KL-constrained contextual bandit (Rafailov et al., 2023; Zhao et al., 2023; Wang et al., 2023a; Azar et al., 2023; Xiong et al., 2023), whose theoretical property is studied in (Xiong et al., 2023). Among them, Direct Preference Optimization (DPO) (Rafailov et al., 2023) has appeared to be one of the most attractive algorithms, which optimizes the LLMs without the reward modeling and directly by preference learning from an offline dataset. In view of the success of DPO, there has also been a debate on whether reward modeling is necessary, where (Rafailov et al., 2023; Zhao et al., 2023; Azar et al., 2023) support bypassing reward modeling. Although there are many works on reward optimization, the forgetting issue (also referred to as the alignment tax (Casper et al., 2023) in the literature) of RLHF algorithms has not been comprehensively studied. Therefore, we choose three representative algorithms, including the PPO (Schulman et al., 2017), RSF (Dong et al., 2023), and DPO (Rafailov et al., 2023) in this work, to study the catastrophic forgetting issue of LLMs after RLHF.

Pretraining, fine-tuning, and distributional shift. Before the emergence of foundation models, the pre-training and fine-tuning paradigm had already achieved remarkable accomplishments across numerous applications (He et al., 2016; Radford et al., 2021; Devlin et al., 2018). However, when deploying pre-trained models into real-world applications and fine-tuning them, a common challenge arises: encountering novel samples from a target distribution that differs from the fine-tuning distribution (Andreassen et al., 2021; Goyal et al., 2022; Zhang and Ré, 2022; Lin et al., 2022a; Zhou et al., 2022a,b; Lin et al., 2022b; Tan et al., 2023). To address this issue, several approaches have been proposed. For instance, (Wortsman et al., 2021; Cha et al., 2021b; Chu et al., 2022) suggest leveraging the weight ensemble of the pre-trained model and the fine-tuned model to enhance out-of-distribution (OOD) performance. Another strategy, as proposed in (Kumar et al., 2022), is the LP-FT technique, which involves initializing the pre-trained feature extractor with a reasonably good classifier. This initialization is particularly important when the classifier is randomly initialized, as the pre-trained features can easily be distorted to accommodate the

random classifier during fine-tuning, exacerbating the issue of catastrophic forgetting.

Catastrophic forgetting and continual learning. DNN tends to lose the knowledge of previously learned task (e.g., pretraining task) when it begins to learn a new task (e.g., the fine-tuning task) (McClelland et al., 1995). Various attempts have been made to alleviate catastrophic forgetting. (Xuhong et al., 2018; Ritter et al., 2018; Aljundi et al., 2018; Schwarz et al., 2018) impose a penalty on the change of the parameter on the new task. (Yu et al., 2021) transfers knowledge from related new knowledge types back to the old types by continually training the representations of old knowledge with the data for new knowledge using a self-training loss. (Yu and Ji, 2023) observes that LLMs tend to rely on pre-existing knowledge, neglecting recent facts and leading to incorrect reasoning chains that ultimately diminish the efficacy of information updates, and proposes to mitigate exposure bias by incorporating the selection of relevant facts into training losses. (Kirkpatrick et al., 2017) gain intuition from Taylor expansion of the losses of the old task at the point of fine-tuned parameter, and further proposes EWC by incorporating the Hassien matrix into parameter regularization. The reply-based method tries to approximate and recover the old data distribution. Popular methods in this direction include sampling methods which store a few old training samples with a small memory buffer (Vitter, 1985; Riemer et al., 2018; Chaudhry et al., 2018; Cha et al., 2021a; Caccia et al., 2021), and generative methods which generate samples from the old distributions with a generative model (Caccia et al., 2020). Knowledge distillation (KD) methods try to keep the prediction of the fine-tuned model close to that of the old model. KD can be naturally combined with experience reply. For example, (Rebuffi et al., 2017) proposes to perform KD on the samples of new tasks as well as the old samples stored in the buffer.

Notably, previous continual learning focuses on sequentially learning tasks which learns a sequence of tasks in order and measures the forgetting of older tasks when learning new tasks (Wang et al., 2023b). Whereas, we focus on the generality forgetting of the pre-trained foundation model during fine-tuning a specific task.

Alignment tax. (Ouyang et al., 2022) reports that they observe significant alignment tax when developing InstructGPT. They have also tried to adopt Experience Replay to alleviate this issue, which is followed by (Zheng et al., 2023). However, we show in Appendix C.1 that Experience Relay is less favorable when compared with model averaging. (Noukhovitch et al., 2024) tried to use stochastic weight averaging, which still under-performs our method as shown in Figure 3. (Li et al., 2024) finds that DPO induces less alignment tax compared with other RLHF algorithms, which is consistent with our findings (e.g., Figure 5). (Askell et al., 2021) reports that they didn't observe significant alignment tax when prompting LLM to align with humans. However, we focus on a more standard setting that the LLM is fully fine-tuned for RLHF.

B RLHF Basics

Following (Ouyang et al., 2022; Bai et al., 2022; Dong et al., 2023; Touvron et al., 2023; Rafailov et al., 2023), we assume that there exists a ground-truth reward function $r^*(x, a) : \mathcal{X} \times \mathcal{A} \to [0, 1]$ where $\mathcal{X}$ and $\mathcal{A}$ are the spaces of prompt and action. The preference ranking satisfies the Bradley-Terry model (Bradley and Terry, 1952): the probability of $a^1 \in \mathcal{A}$ being preferred is

$$\mathbb{P}(a^1 \succ a^2 | x, a^1, a^2) = \frac{\exp(r^*(x, a^1))}{\exp(r^*(x, a^1)) + \exp(r^*(x, a^2))}. \tag{4}$$

We denote an LLM by a policy π that maps x to a distribution over the response space $\mathcal{A}$. The main goal of RLHF is to align the staring checkpoint π_{θ_0} with the human preference so that it achieves high reward measured by r^*, but we may also impose additional constraints to avoid overfitting like requiring the models to stay close to the π_{θ_0}. In practice, we learn from a preference dataset of the form $\mathcal{D} = \{(x, a_w, a_l)\}$, where a_w is the preferred response. Typically, we will first train a reward model r as the Maximum Likelihood Estimation (Ouyang et al., 2022; Bai et al., 2022; Touvron et al., 2023) on the preference dataset $\mathcal{D}$ and then perform reward optimization by different algorithms.

Rejection Sampling Finetuning (RSF) is proposed in (Dong et al., 2023; Touvron et al., 2023; Yuan et al., 2023; Gulcehre et al., 2023) with several variants. Essentially, the RSF learns from the best-of-n

policy (Nakano et al., 2021), which samples n responses for each prompt query and returns the one with the highest reward. As suggested by (Dong et al., 2023; Touvron et al., 2023; Gulcehre et al., 2023), we adopt an iterative training set-up for the implementation instead of always sampling the samples from the starting checkpoint because we find that the iterative training is far more sample-efficient. Specifically, for each iteration, we first sample a batch of prompts and generate n responses for each prompt from current model. Then, we use the reward model to compute the rewards for each prompt-response pair and for each prompt, we select the one with the highest reward into a small subset. By this process, we collect a batch of samples from the best-of-n policy that are with high reward. We simply fine-tune the current model on this subset to get the next model and the next iteration begins.

PPO is the the classical method for RLHF and has gained its success in aligning Chat-GPT (OpenAI, 2023). In contrast to the implementation in traditional DRL scenario, for alignment of LLMs, following (Ziegler et al., 2019; Wu et al., 2021a; Ouyang et al., 2022; Rafailov et al., 2023; Liu et al., 2023), we modify the reward optimization as the following KL-regularized form:

$$\tilde{r}(x,a) = r(x,a) - \eta \log \frac{\pi(a|x)}{\pi_{\theta_0}(a|x)},$$

where $\eta > 0$ is a hyper-parameter to control the level of KL penalty.

Direct Preference Optimization (DPO) is proposed by (Rafailov et al., 2023) from the following KL-constraint optimization problem:

$$\max_{\pi} \mathbb{E}_x \mathbb{E}_{a \sim \pi(\cdot|x)} \left[r^*(x,a) + \eta \log \frac{\pi_{\theta_0}(a|x)}{\pi(a|x)} \right]. \tag{5}$$

It is known that (5) admits the following closed-form solution $\pi^*(\cdot|x) = \frac{1}{Z(x)} \pi_0(\cdot|x) \cdot \exp\left(\frac{1}{\eta} r^*(x,\cdot)\right)$ (see e.g. Proposition 7.16 of (Zhang, 2023)), where $Z(x)$ is the normalization constant. We can now represent r^* by π^* as follows:

$$r^*(x,a) = \eta \log \frac{\pi^*(a|x)}{\pi_0(a|x)} + \eta \log Z(x).$$

Plugging the reparameterization of r^* into the preference model in (4), we get

$$\mathbb{P}(a^1 \succ a^2 | x, a^1, a^2) = \frac{1}{1 + \exp\left(\eta \log \frac{\pi^*(a^2|x)}{\pi_0(a^2|x)} - \eta \log \frac{\pi^*(a^1|x)}{\pi_0(a^1|x)}\right)}. \tag{6}$$

The idea of DPO is to find a model π so that it maximizes the likelihood given in (6) on the offline preference dataset. Therefore, it chooses to minimize the following loss function:

$$\mathcal{L}(\theta, \pi_{\theta_0}, \mathcal{D}) = - \sum_{(x,a_w,a_l) \in \mathcal{D}} \left[\log \sigma \left(\eta \log \frac{\pi_\theta(a_w|x)}{\pi_{\theta_0}(a_w|x)} - \eta \log \frac{\pi_\theta(a_l|x)}{\pi_{\theta_0}(a_l|x)} \right) \right], \tag{7}$$

where the reward modeling step is bypassed.

B.1 Algorithm of Heterogeneous Model Averaging

Reward Preserving Updating. It is noteworthy that Eqn. (3) represents a RL problem. To implement Eqn. (3), RL algorithms such as RSF, PPO, or DPO need to be implemented, involving extra implementation details that depend on the algorithm. To address this issue, we propose a proxy distillation method. Specifically, given a policy π_θ after RLHF, we generate a proxy dataset by

$$\mathcal{D}_\theta = \{(x,a) : a \sim \pi_\theta(\cdot|x), \text{ for } x \in \mathcal{X}\}. \tag{8}$$

Since the data in $\mathcal{D}_\theta$ is generated by π_θ, this data should have a high reward. Therefore, maximizing the likelihood on $\mathcal{D}_\theta$ could result in a model with a high reward. Specifically, we optimize the following

$$\max_{\alpha_1,\dots,\alpha_K \in \Omega} \frac{1}{|\mathcal{D}_\theta|} \sum_{(x,a) \in \mathcal{D}_\theta} \log[\pi_{\theta(K)}(a|x)]. \tag{9}$$
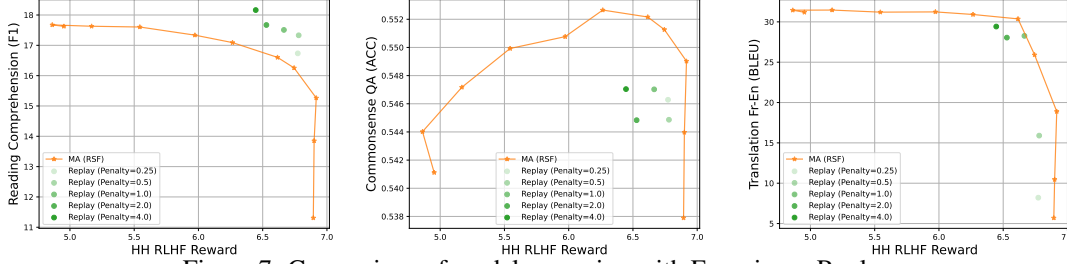
Figure 7: Comparison of model averaging with Experience Replay.

The algorithm of Heterogeneous Model Averaging is summarized as follows:

Algorithm 1 HMA: Heterogeneous Model Averaging

Input: The reward model $r(\cdot, \cdot)$, initial policy π_{θ_0}, prompt set $\mathcal{D}_x$, hyper-parameter K, merge ratio α.
Output: The output policy $\pi_{\theta(K)}$.
1: Perform vanilla RLHF by Eqn (1) and obtain π_θ.
2: Distill $\mathcal{D}_\theta$ from π_θ according to Eqn. (8).
3: Initialize $\alpha_1, ..., \alpha_K \in [0, 1]$ for the K parts of the Transformer, respectively.
4: Obtain the averaged model $\theta(K)$ with $\alpha_1, ..., \alpha_K$.
5: Solve Heterogeneous ratios $\alpha_1, ..., \alpha_K$ according to Eqn. (9).
6: Return the $\theta(K)$ with the optimized $\alpha_1, ..., \alpha_K$.

C More Results

C.1 Experience Replay

In our alignment tax situation, we aim to preserve a wide range of abilities gained during pre-training. It is possible to replay a small subset of pretraining data, which also known as Experience Replay (ER) (Rebuffi et al.; Shin et al., 2017). However, this method is less practical since pre-training datasets of most models are often not publicly available. Further more, even if we can access the pre-training data, retaining a subset of the pre-training data entails extra computational costs and implementation intricacies, making it less preferable (Noukhovitch et al., 2023). In this part, we compare ER with MA. Specifically, we include a small proportion of randomly subsampled pre-training data during the RLHF stage. Here, we denote $\mathcal{D}_{pre}$ as the pre-training data distribution, and our objective is to solve the following:

$$\max_\theta \mathbb{E}_x \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}[r^*(x, a)] + \lambda \mathbb{E}_{(x,a) \sim \mathcal{D}_{pre}} \log \pi_\theta(a|x)$$

We experiment with different penalty weights λ such as 0.25, 0.5, 1, 2, and 4. Importantly, we utilize the data proportion as a proxy for setting the penalty weight. For instance, we do not explicitly apply a penalty of 4 when $\lambda = 4$; instead, we include 4 times the replay data over the RLHF data in a batch. Refer to the Appendix D for more details.

Results. The results of ER are displayed in Figure 7. Additionally, we include the performance of model averaging for comparison. It is evident that while ER has access to pre-training data, it only demonstrates superior performance over model averaging in the Reading Comprehension dataset (Figure 7 - Left), and falls short of model averaging in the Commonsense QA (Figure 7 - Middle) and Translation (Figure 7 - Right) benchmarks.

Discussion of ER results. The differing performance of ER compared to model averaging is somewhat surprising. Despite maintaining extra pre-training data, which is four times larger than the RLHF data (400M token), ER under-performs model averaging in two out of three benchmarks. This may be attributed to the vast size of the pre-training data (1.2T token), such that even when replaying a subset four times larger than the RLHF data, it only covers about 0.03% of the pre-training data. Consequently, the data corresponding to certain abilities may be underrepresented in the replay dataset. With a substantial pre-training dataset and a wide range of abilities to preserve, it becomes challenging to maintain all abilities through replay.
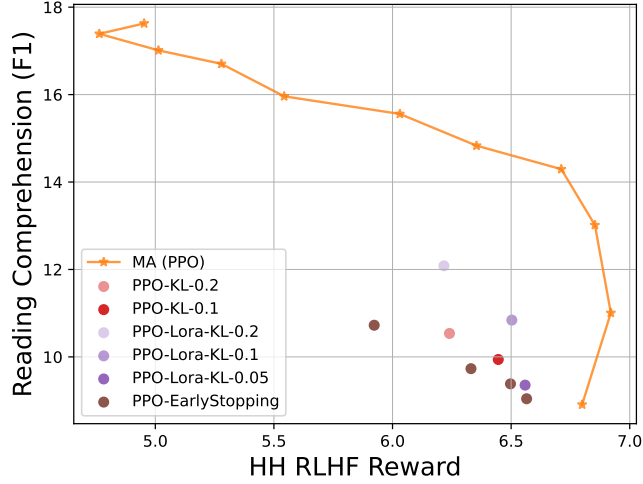
Figure 8: Comparison of model averaging with reward penalty for PPO.

C.2 Reward Penalty

It is a common practice to impose Kullback–Leibler (KL) penalty on the RL reward in the PPO. Such a penalty can also regularize the policy to stay closer to the initial policy, which in return can reduce the alignment tax. Following (Ziegler et al., 2019; Wu et al., 2021a; Ouyang et al., 2022; Yuan et al., 2023), we modify the raw reward function with an additional KL penalty (Ziegler et al., 2019).

$$\max_\pi \mathbb{E}_x \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}[r^*(x, a)] - \mathrm{KL}(\pi_\theta || \pi_{\theta_0}), \tag{10}$$

where we use $\mathrm{KL}(\pi_\theta || \pi_{\theta_0})$ to denote $\mathbb{E}_x[\mathrm{KL}(\pi_\theta(\cdot|x) || \pi_{\theta_0}(\cdot|x))]$ for short. We compare vanilla model averaging methods with the reward penalty by considering different KL penalties in $\{0.05, 0.1, 0.2\}$. The results are shown in Figure 8. We can see that while a larger KL penalty can partially mitigate the forgetting issue, the model averaging is much more effective than the reward penalty in terms of the alignment-forgetting trade-off.

C.3 Consistency of different combination ratios among various tasks

We try three patterns experiment given a base $\alpha \in \{0.2, 0.3, 0.4\}$: (a) $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$; (b) $\alpha_1 = \alpha_2 = \alpha, \alpha_3 = \alpha - 0.1$, and (c) $\alpha_1 = \alpha, \alpha_2 = \alpha_3 = \alpha - 0.1$. We use $(\alpha|\alpha|\alpha)$, $(\alpha|\alpha|\alpha - 0.1)$ and $(\alpha|\alpha - 0.1|\alpha - 0.1)$ to denote these three patterns, respectively. These results confirm that certain ratio combinations exceed the trade-off curve of vanilla model averaging, as displayed in Figure 9. Notably, some combination ratios consistently outperform the equal ratio across various benchmarks. This affirms the potential to identify consistent combination ratios that demonstrate superior performance across a broad spectrum of benchmarks in terms of alignment-forgetting trade-off.
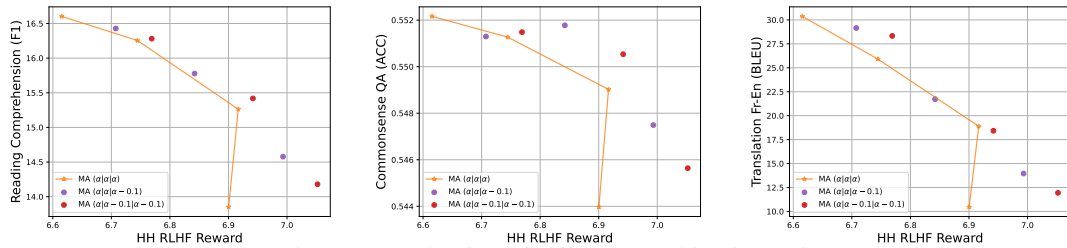

Figure 9: Evaluation of different combination ratios.

C.4 Results of $\alpha = 0.2$

The following results show that when we chose $\alpha = 0.2$, MA and HMA consistently alleviate the alignment tax without sacrificing any alignment performance.
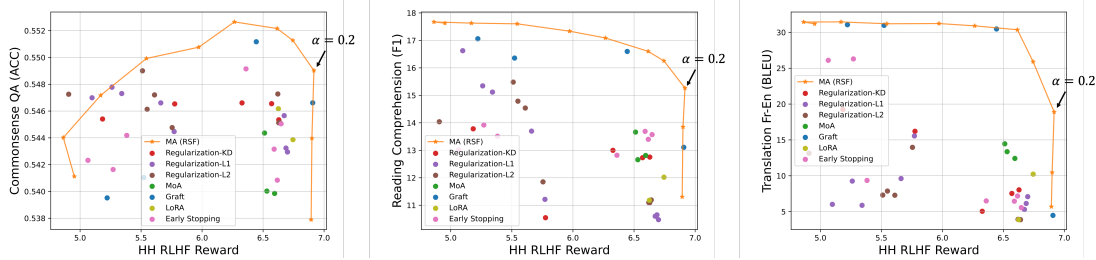
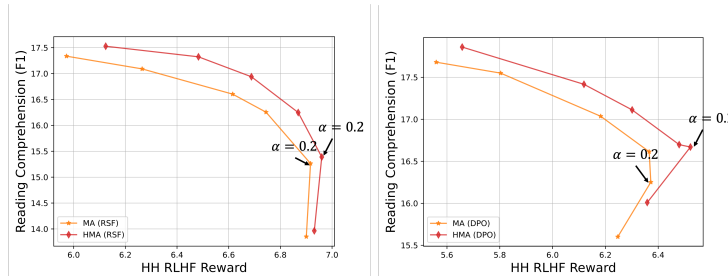Figure 10: Illustration of $\alpha = 0.2$ on vanilla model averaging

Figure 11: Illustration of $\alpha = 0.2$ on HMA

D Implementation Details

In this section, we introduce the implementation details for the methods mentioned in Section 3.

D.1 Rejection Sampling Fine-tuning Implementation

The rejection sampling fine-tuning (RSF) is proposed in (Dong et al., 2023; Touvron et al., 2023; Yuan et al., 2023; Gulcehre et al., 2023) with several variants. Essentially, RSF earns from the best-of-n policy (Nakano et al., 2021), which samples n responses for each prompt query and returns the one with the highest reward. In this work, we implement the algorithm with the official code provided in LMFlow[9]. We adopt most of the hyper-parameters as suggested by (Dong et al., 2023) and focusing on tuning the learning rate by searching over $\{1 \times 10^{-6}, 2 \times 10^{-6}, 1 \times 10^{-5}\}$ and 1×10^{-5} is taken for our main experiments.

As suggested by (Dong et al., 2023; Touvron et al., 2023; Gulcehre et al., 2023), we adopt an iterative training set-up for the implementation instead of always sampling the samples from the starting checkpoint because we find that the iterative training is far more sample-efficient. Specifically, for each iteration, we first sample a batch (2048) of prompts and generate $n = 32$ responses for each prompt from current model. Then, we use the reward model to compute the rewards for each prompt-response pair, and for each prompt, we select the one with the highest reward into a small subset. Through this process, we collect 2048 samples from the best-of-32 policy that are with high reward. We simply fine-tune the current model on this subset to get the next model and the next iteration begins.

When RSF is combined with other methods for preventing the model from forgetting, we follow (Touvron et al., 2023; Dong et al., 2023) to align the models in a distillation style. Specifically, we run RSF algorithm as described above until the model converges to a rather stable level of reward. Then, we collect the best-of-32 samples along the way of training and fine-tune the model from the starting checkpoint with the additional methods for mitigating the forgetting issue. In comparison, we note that (Touvron et al., 2023) only uses the largest 70B Llama 2-Chat models to collect best-of-n samples and other smaller models are then fine-tuned on these collected data and (Dong et al., 2023) uses LLaMA-7B to run RSF and uses the collected data to fine-tune other LLMs.

---

[9] https://github.com/OptimalScale/LMFlow

D.2 Implementation of PPO

The experiments with PPO in this work are conducted using the open-source package Transformer Reinforcement Learning (TRL)[10]. It is known that the PPO is significantly less stable as compared to supervised learning (Choshen et al., 2019) and sensitive to the hyper-parameter and code-level optimization (Engstrom et al., 2020). To tune PPO to its best performance, we include several empirical enhancements and we record our tuning process, as well as the successful/unsuccessful attempts in this subsection for interested readers.

First, we follow (Ramamurthy et al., 2022) to warm up by finetuning the model on the preferred samples of the preference dataset for 1 epoch for a more stable training process. Moreover, in contrast to the implementation in traditional DRL scenario, for alignment of LLMs, following (Ziegler et al., 2019; Wu et al., 2021a; Ouyang et al., 2022; Rafailov et al., 2023; Liu et al., 2023), we will also modify the reward optimization as the following KL-regularized form:

$$\tilde{r}(x, a) = r(x, a) - \eta \log \frac{\pi(a|x)}{\pi_0(a|x)},$$

where $\eta > 0$ is a hyper-parameter to control the level of KL penalty.

However, even though we first finetune the models with the preferred samples and train with an additional KL penalty, the PPO training can still lead to an unstable reward level and failure. For the first issue, with the ultimate hyper-parameter, we will run PPO with three independent seeds and take the best models. We now focus on the second issue. One notable failure signal of PPO training is that the models suddenly refuse to answer the question (prompt), or reply with incomplete sentences, which may be detected by (1) a shorter average response length; (2) the incomplete sentences in randomly displayed sample responses within one iteration; (3) sudden drop in reward value. Once such a drop happens, the models just collapse and the training fails.

Hyper-parameter tuning. To mitigate this issue, we carefully tune the learning rate, KL coefficient, update epoch, batchsize by grid search. We observe that for full training (without LoRA), a learning rate with 1×10^{-6} is most suitable in terms of the trade-off between reward learning and training stability. Update epoch $= 2$ performs best in our preliminary experiments for parameter tuning. A batchsize that is too large (2048) or too small (128) leads to unstable training. Therefore, we fix the batchsize as 512 and the update epoch as 2 to further tune the KL coefficient and learning rate. Ideally, in the mathematical formulation of KL-constrained RLHF, a smaller KL coefficient should lead to a higher reward value. In practice, we observe that for KL coefficient $\beta \in [0.05, 0.3]$, a smaller KL coefficient leads to a higher ultimate reward value of the obtained policy. However, for $\beta < 0.05$, the model collapses before it achieves the highest reward possible, leading to a even worse model compared to $\beta = 0.05$. The results are observed across more than 20 independent runs. Therefore, in the ablation study of the impact of KL coefficient for PPO, we choose $\beta = 0.05$ as the smallest KL coefficient. We mention in passing that due to the same instability issue, the LoRA training may also achieve better reward because we can optimize the model well with LoRA, while the full-trained models collapse before it achieve its best performance.

Restart trick in critic training. To further understand the reason why the PPO fails, we examine several training records provided by wandb. We found that before (or simultaneously) the models collapse, the critic loss increases significantly. After looking at the source code of TRL, we notice that there is a scaling factor of the critic loss of 0.1, which may also suggest that the training processes of the critic and actor are different. Motivated by these observations, we try out different learning rates for the critic: (1) a larger learning rate for the critic; (2) a smaller learning rate for the critic; (3) decay/increase the learning rate of the critic every 10 batch of the training. Unfortunately, we do not see significant improvement in either the training stability or the ultimate reward value. We noticed that the instability from value estimation (critic training) seems to be a well-studied problem in the DRL literature. For instance, (Lee et al., 2022a) proposes to use a pessimistic (conservative) reward signal, which is obtained by reward model ensemble, which is also recommended in theoretical RLHF studies (Zhu et al., 2023; Xiong et al., 2023). However, this requires to load multiple reward models at the same time, which is infeasible for us

---

[10]https://github.com/huggingface/trl

due to computational constraint. Motivated by the trick of PaLM (in the pre-trained stage) (Chowdhery et al., 2023), which call back whenever the spikes happen in the loss curve, we simply train the model twice. Specifically, we run PPO training first and save the intermediate models for every iteration. Once the model collapses, we simply restart from a model 3 iterations before the training fails and re-initiate the critic model. Then, we skip the actor training for 1 iteration as a warm-up stage of the restarted critic. We observe that though the training still collapses easily after 10-20 iterations of training, we do achieve a much higher reward value.

It is also interesting to design new algorithms to mitigate the value estimation error for a more stable PPO-based training, and we leave it for future study since it is beyond the scope of this work.

D.3 Implementation of DPO

We implement DPO by the open-source package Transformer Reinforcement Learning (TRL). We mainly use 0.1 in our experiments but also try out 0.3 and 0.5 since the authors of original paper recommend to set it from 0.1 to 0.5. Then, we mainly tune the learning rate. We use the evaluation loss (which generally aligns with the evaluation accuracy) on the validation set of reward modeling for the model selection. We observe that for learning rate in $\{1 \times 10^{-6}, 2 \times 10^{-6}, 1 \times 10^{-5}\}$, 1×10^{-6} achieves the lowest evaluation loss so it is adopted in our experiments. We train DPO for up to 3 epochs and evaluate the model every 0.5 epoch by the evaluation loss on the validation set. The lowest evaluation loss and highest evaluation accuracy are achieved at the end of the first epoch so we use the model as the representative model of DPO though we do observe the validation reward of the model at 0.5 epoch of the training is slightly higher. We suspect that this is because the equivalence of reward modeling and policy training are equivalent for DPO only when the optimization error is zero (see (Rafailov et al., 2023; Azar et al., 2023) for a detailed proof). In practice, since the samples are finite and we may not solve the non-convex optimization by finding its exact minimizer, the reward of the generator may not align with the accuracy of the discriminator (reward model).

D.4 Implementations of Existing Methods to Alleviate Alignment Tax

We test existing methods mainly on the RSF method which is implemented as discussed in Appendix D.1. Details about how we implement existing methods to mitigate forgetting are described as follows.

(a) Early Stopping: The whole RSF is conducted for 10 iterations and we choose the model of RSF at numbers of iterations of 2,4,6,8 as the early stopping checkpoints.

(b) Regularization towards θ_0 in the weight space: For these kinds of methods. We alternative the training loss at the SFT stage in RSF by adding the regularization terms with different penalties. Specifically, we test $\{0.04, 0.1, 0.4, 0.6, 1\}$ for the L1 penalty and $\{0.01, 0.04, 0.06, 0.08, 0.1\}$ for L2 penalty.

(c) Low-Rank Adaptation (LoRA): We implement two levels of LoRA. The typical version only considers the low-rank adaptation of MLP blocks and we have tested several ranks for 16-512, while only rank 512 gives a reasonable performance on the final alignment result. The other is the low-rank adaptation of both MLP and attention blocks, in this case, rank 16 makes a good performance on alignment.

(d) Knowledge distillation: The implementation of this approach is similar to the Regularization method. We add the knowledge distillation term as a regularization term in the SFT stage. The penalty used here are $\{10^{-5}, 10^{-3}, 10^{-1}\}$.

(e) Model Averaging: We simply interpolate the modules of linear layers in the whole model, e.g., Q, K, V projection layers in attention and MLP layers. We will vary the α from 0 to 1. The start point of the model averaging is the model after instruction following and the end point of that is the model after RLHF.

For the experience replay (ER) method, we uniformly sample the pre-trained data of Open-LLaMA-3B according to the penalty. Specifically, given the alignment data of 400M tokens and a penalty of 2, we will sample 800M token data from the pre-trained data. And then add data to conduct the SFT loss as a penalty.

D.5 Implementations of Heterogeneous Model Averaging

Notice that it is difficult to directly solve the Eqn. (9) on the support set Ω. So instead of directly optimizing the $\alpha_1, \ldots, \alpha_K$, we reparameterize the $\alpha_1, \ldots, \alpha_K$ as follows,

$$\hat{\alpha}_i = \sigma(s_i) + \epsilon; \quad \alpha_i = \frac{\hat{\alpha}_i}{\sum_{i=1,\ldots,K} \hat{\alpha}_i} \alpha \tag{11}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function s_i can take any real number. For each $s_1, \ldots, s_K$, we can easily find the corresponding $\alpha_1, \ldots, \alpha_K$ of Eqn. (11) belongs to the Ω. In this way we can optimize on $s_1, \ldots, s_K$ rather than $\alpha_1, \ldots, \alpha_K$. Moreover, the ϵ in Eqn. (11) can serve as a boundary control parameter, that is, if we set $K = 3, \epsilon = 1$, then each α_i can just take values over $[0.2\alpha, 0.5\alpha]$. In practice, we will search the $\epsilon \in \{0, 0.1, \ldots, 0.9\}$ to get the best model.

To get D_θ, we will use the prompts from the training RLHF dataset to generate the full response with different policy π_θ. Then we sample about 2000 pieces generated responses from the set consisting of the 5000 samples with the highest rewards. Then we can just take the $s_1, \ldots, s_K$ as the optimization parameters and just finetuning them on the D_θ.

Besides directly optimizing the Eqn. (9), we also test adding regularization terms of $\alpha_1, \ldots, \alpha_K$. Generally we just add weighted L1 loss $\sum_i w_i |\alpha_i - \alpha|$ as the regularization terms. w_i is chosen to make the middle part of the module change not too much.

Typically, we only average the weights in the linear layers and the $\alpha_1, \ldots, \alpha_K$ works on transformer layers which contain self-attention and MLP. For the head layer, we just set the average weight as α.

We give the hyper-parameters for the optimization in Table 4

E More Results

E.1 The Alignment Tax during Training (Results of Early Stopping)

The following figure shows the RLHF reward and alignment tax during different training steps.
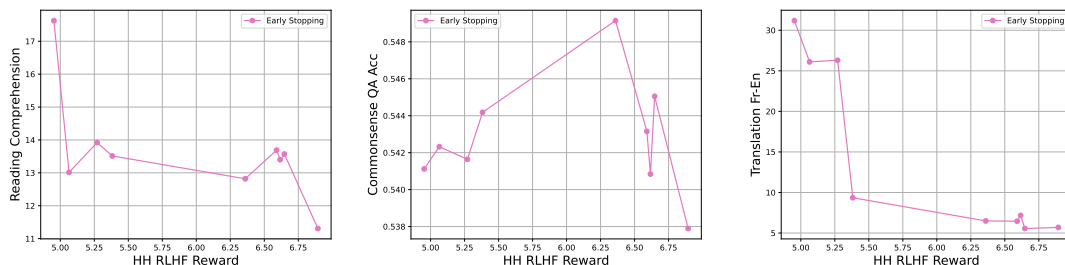

Figure 12: The alignment-forgetting trade-off during training

E.2 More Results of Averaging Different Parts

In this part, we include the full results (e.g., RSF, DPO, PPO) of averaging different parts.
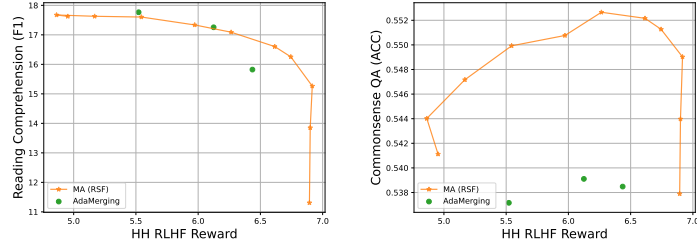
Figure 15: Results of AdaMerging. We optimize AdaMerging on Reading Comprehension and found it can hardly do well on Common Sense.
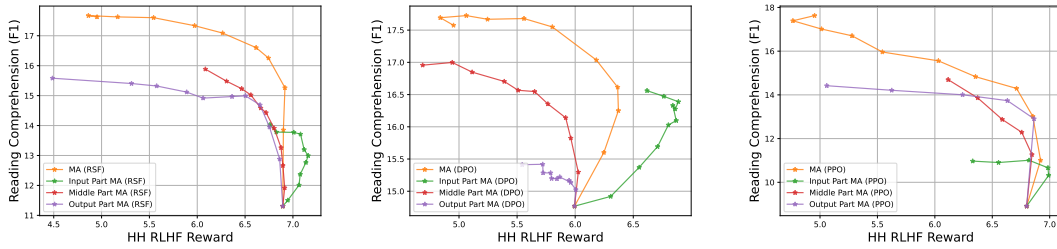

Figure 13: The performance of averaging different parts. (Left) RSF; (Middle) DPO; (Right) PPO

E.3 Comparison of RLHF Algorithms

We compare the alignment-forgetting trade-off of RSF, DPO and PPO in Figure 14. We observe that RSF is consistently better than DPO. However, we also note that this is not a fair comparison since DPO does not directly optimize for the reward.
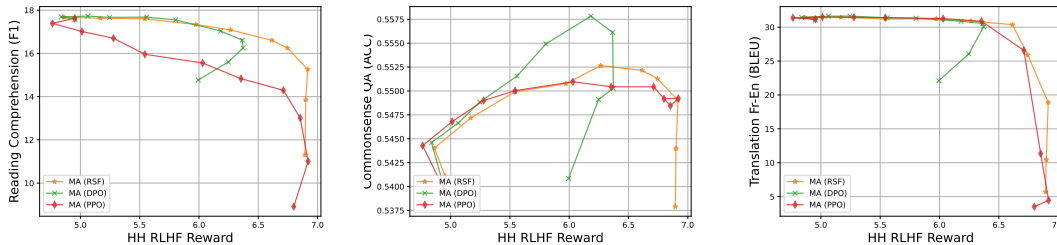

Figure 14: Comparison of RLHF algorithms in terms of alignment-forgetting trade-off.

E.4 Results of AdaMerging (Yang et al., 2023)

Previous studies (Yang et al., 2023) have also discussed the idea of dynamically assigning different weights to different layers when merging models, aiming to maximize performance on a specific task (e.g., $\mathcal{T}_i$). These approaches assume access to the task-specific data $\mathcal{T}_i$. However, considering the nature of alleviating alignment tax, which aims to mitigate forgetting across a extremely wide range of tasks ($\mathcal{T}_{j_1}...\mathcal{T}_{j_K}$), these methods fail to effectively optimize performance for multiple tasks simultaneously. Specifically, we want to preserve the abilities on a wide range of tasks and it is hard to get the data for all these tasks. Further more, some ability such as in-context learning does not have a clear corresponding training set. So it is less practical to find training set for AdaMerging.

Here we demonstrate when we use AdaMerging to optimizes for task A and the training set does not cover task B, AdaMerging can not preserve the ability on task B. Specifically, we provide AdaMerging with labeled data for Reading Comprehension (i.e., task A) and optimize the 26 layer-wise merging ratios as (Yang et al., 2023). To have a clear comparison with vanilla model averaging, we try different mean averaging ratio for AdaMerging among 0.2, 0.4 and 0.6. We also show both the results on task A and B.

In contrast, our HMA only require the RLHF data and does not need any data from the tasks which we want to preserve ability. Figure 16 shows that HMA can alleviate the alignment tax evaluated on a wide range of tasks.

E.5 Detailed Results of Heterogeneous Model Averaging

We provide the detailed results of Heterogeneous model averaging on various benchmarks, e.g., Reading Comprehension, Commonsense QA and translation, and different RLHF methods, e.g., RSF, PPO, and DPO.
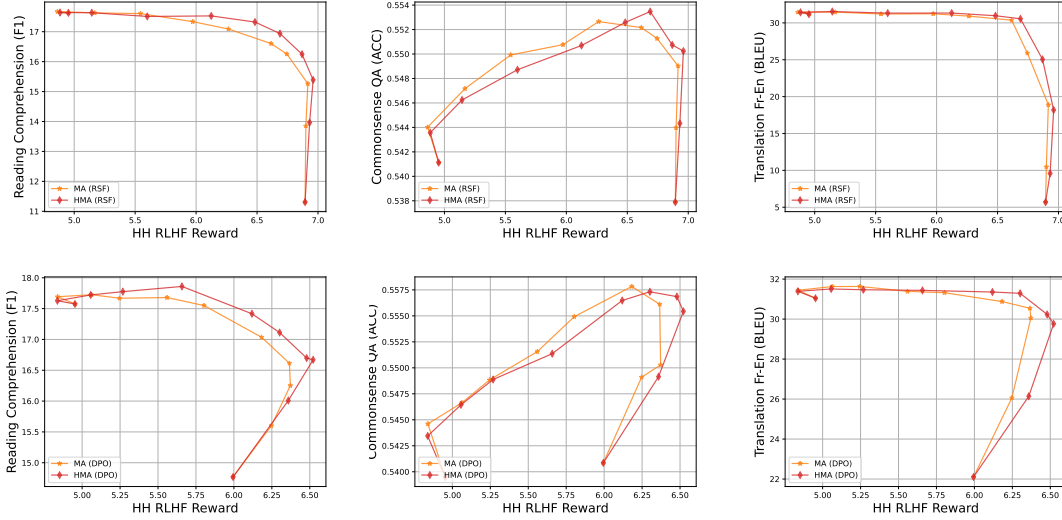

Figure 16: Detailed results of Heterogeneous model averaging on various benchmarks and RLHF methods.

F Theoretical Settings, Proofs and Discussions

F.1 Re-statement of Formal Settings

Notation. Consider that the full class space $\mathcal{M}$ contains M classless, i.e. $\boldsymbol{y} \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_M\}^\top$, where $\boldsymbol{e}_i$ denotes the M-dimensional unit vector with ith element equaling 1, e.g., $\boldsymbol{e}_2 = [0, 1, 0, ..., 0]^\top$. $\boldsymbol{a}(k)$ means the kth element of vector $\boldsymbol{a}$, $\boldsymbol{A}(k)$ means the kth column of matrix $\boldsymbol{A}$. We use $\mathbf{I}_M$ to represent a $M \times M$ identity matrix, e.g., $\mathbf{I}_M = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_M]$. We omit the subscript of $\mathbf{I}$ when no confusion arises.

Following (Lin et al., 2023), suppose we have N weak features $\{\boldsymbol{x}_i\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and the whole feature $\boldsymbol{x} \in \mathbb{R}^{d \times N}$ is the concatenation of them, i.e., $\boldsymbol{x} = \mathrm{Concat}\left(\{\boldsymbol{x}_i\}_{i=1}^N\right) = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$. Consider that each model f is composed of a featurizer $\Phi \in \{0, 1\}^N$ and a classifier $\boldsymbol{w} \in \mathbb{R}^{d \times K}$. Φ first selects feature by $\boldsymbol{x}\Phi$. For example, suppose $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3]$ and $\Phi = [1, 1, 0]^\top$, then $\boldsymbol{x}\Phi = \boldsymbol{x}_1 + \boldsymbol{x}_2$. Then the classifier $\boldsymbol{w} \in \mathbb{R}^{d \times K}$ is fit based on the features selected by Φ as $\boldsymbol{w} = \arg\min_{\boldsymbol{v}} \mathbb{E}[\ell(\boldsymbol{v}^\top(\boldsymbol{x}\Phi), \boldsymbol{y})] + \|v\|_2^2$, where ℓ is the cross-entropy loss function.

We simplified (Lin et al., 2023)'s Definition 1 and only consider weak features as following:

Definition F.1 (Data Generation Process). The whole data generation process is as follows:

$$\boldsymbol{y} \sim \mathrm{Unif}\{\boldsymbol{e}_1, \boldsymbol{e}_2, ...\boldsymbol{e}_M\}, \boldsymbol{x} = \mathrm{Concat}\left(\{\boldsymbol{x}_i\}_{i=1}^M\right),$$
$$\mathbb{P}_\theta(\boldsymbol{x}_i \mid \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{\mu}_i \mathbf{Q}_i \boldsymbol{y}, \sigma^2 \boldsymbol{I}_d\right), \forall i. \tag{12}$$

where $\mathbf{Q}_i \in \{0, 1\}^{M \times M}$. the mth column of $\mathbf{Q}$, i.e., $\mathbf{Q}_j(m)$, is as follows for $m = 1, 2, \cdots, M$:

$$\mathbf{Q}_j(m) = \begin{cases} \boldsymbol{e}_m, & \text{with probability } 1 - p \\ \mathrm{Unif}\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_M\}, & \text{with probability } p. \end{cases}$$

Definition F.2 (Model Averaging, Definition 4 of (Lin et al., 2023)). Given the two individual models $(\bar{w}, \bar{\Phi})$ and $(\tilde{w}, \tilde{\Phi})$, the prediction of the model averaging is $f_{\text{avg}}(\boldsymbol{x}) = \frac{1}{4}(\bar{\boldsymbol{w}} + \tilde{\boldsymbol{w}})^\top \left(\boldsymbol{x}(\bar{\Phi} + \tilde{\Phi}) \right)$

We impose the following mild assumptions as (Lin et al., 2023).

Assumption F.3 (Small Noise). Denote N_s as the the maximum number of invariant features and spurious features that a model can learn, respectively. We need the overall noise to be small to satisfy $\boldsymbol{F}^K(\frac{1}{\sigma(N_s)}) \geq 1 - \epsilon$, in which $\boldsymbol{F}$ is the cumulative distribution function of standard Gaussian random variable, and K refers to the class number.

Assumption F.4 (Orthogonal features (Lin et al., 2023; Allen-Zhu and Li, 2020)). (1) $\|\boldsymbol{\mu}_i(k)\|_2 = 1$ for $i = 1, \cdots, n$, (2) $\mu_i(k) \perp \mu_{i'}(k')$ for any $(i, k) \neq (i', k')$, $k, k' = 1, \cdots, K$, $i, i' \in 1, \cdots, n$.

F.2 Proof of Proposition 5.1

Estimating $\xi^{(1)}$ corresponding to Case (1). The estimation of $\xi^{(1)}$ is a direct application of Proposition 7 of (Lin et al., 2023). Specifically, according to Proposition 7 of (Lin et al., 2023), we have

$$\mathcal{A}_a(f_a) = \mathcal{A}_b(f_b) = F_b((1-p)\sqrt{n}), \mathcal{A}_a(f_{\text{avg}}) = \mathcal{A}_b(f_{\text{avg}}) = F_b((1-p)\frac{\sqrt{2}n}{\sqrt{n+n_o}}) \quad (13)$$

Estimating $\xi^{(2)}$ corresponding to Case (2). Without loss of generality, we assume the $\mathcal{Y}_a$ is $\{1, ..., K\}$ and $\mathcal{Y}_b$ is $\{K+1, ..., 2K\}$. Denote the feature learnt by $(\boldsymbol{w}_a, \Phi_a)$ and $(\boldsymbol{w}_b, \Phi_b)$ as $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ and $\boldsymbol{x}_{n-n_o+1}, ..., \boldsymbol{x}_n, ... \boldsymbol{x}_{2n-n_o}$. Since $\mathcal{A}_a(f_{\text{avg}}), \mathcal{A}_b(f_{\text{avg}}) \geq 0$, we trivially have $\xi^{(1)} \geq -F_p((1-p))\sqrt{n}$ by combing Proposition 7 of (Lin et al., 2023).

According to the Lemma 5 of (Lin et al., 2023), we have

$$\bar{\boldsymbol{w}}_a(k) = \sum_{i=1}^{n} \mu_i(k), \forall k = 1, \cdots, K, \quad \bar{\boldsymbol{w}}_b(k') = \sum_{i=n-n_o+1}^{2n-n_o} \mu_i(k'), \forall k' = K+1, \cdots, 2K, .$$

We first estimate the accuracy of f_{avg} on task (a), i.e., $\mathcal{A}_a(f_{\text{avg}})$, for a sample from class $k \in 1, \cdots, K$ and $k' \neq k, k' \in 1, \cdots, K$. Then by $|\mathcal{Y}_a \cap \mathcal{Y}_b| = 0$ and Assumption F.4, we have

$$(\boldsymbol{w}_a(k) + \boldsymbol{w}_b(k))^\top \boldsymbol{x}(\bar{\Phi}_a + \bar{\Phi}_b)|_{y=\boldsymbol{e}_k} = \boldsymbol{w}_a(k)^\top \boldsymbol{x}\bar{\Phi}_a + \boldsymbol{w}_b(k)\boldsymbol{x}\bar{\Phi}_b|_{y=\boldsymbol{e}_k} = \boldsymbol{w}_a(k)^\top \boldsymbol{x}\bar{\Phi}_a|_{y=\boldsymbol{e}_k}$$
$$(\boldsymbol{w}_a(k') + \boldsymbol{w}_b(k'))^\top \boldsymbol{x}(\bar{\Phi}_a + \bar{\Phi}_b)|_{y=\boldsymbol{e}_k} = \boldsymbol{w}_a(k')^\top \boldsymbol{x}\bar{\Phi}_a + \boldsymbol{w}_b(k')\boldsymbol{x}\bar{\Phi}_b|_{y=\boldsymbol{e}_k} = \boldsymbol{w}_a(k')^\top \boldsymbol{x}\bar{\Phi}_a|_{y=\boldsymbol{e}_k}$$

The last equality is due to $w_b(k) = 0$ and $w_b(k') = 0$ for $k, k' \in 1, ..., K$. Then it is straightforward to see that $\mathcal{A}_a(f_{\text{avg}}) = \mathcal{A}_a(f_a)$. We similarly have $\mathcal{A}_b(f_{\text{avg}}) = \mathcal{A}_b(f_b)$. Then we have $\xi^{(2)} = 0$.

We finish the proof by collecting the results.

F.3 Discussion on the Effect of Task Similarity on Model Averaging

We illustrate why model averaging would not lead to much improvement if two tasks are dissimilar, i.e., $|\mathcal{Y}_a \cap \mathcal{Y}_b| = 0$. Without loss of generality, we assume the $\mathcal{Y}_a$ is $\{1, ..., K\}$ and $\mathcal{Y}_b$ is $\{K+1, ..., 2K\}$. Since $\boldsymbol{w}$ is the minimum norm solution based on Φ, we know that $\boldsymbol{w}_b(k) = 0$ for $k = 1, ..., K$. From the previous proof, we know that

$$(\boldsymbol{w}_a(k) + \boldsymbol{w}_b(k))^\top \boldsymbol{x}(\bar{\Phi}_a + \bar{\Phi}_b)|_{y=\boldsymbol{e}_k} = \boldsymbol{w}_a(k)^\top \boldsymbol{x}\bar{\Phi}_a + \boldsymbol{w}_b(k)\boldsymbol{x}\bar{\Phi}_b|_{y=\boldsymbol{e}_k}$$

Since $\boldsymbol{w}_b(k) = 0$, the above equation equals $\boldsymbol{w}_a(k)^\top \boldsymbol{x}\bar{\Phi}_a$, which is simply the performance of f_a. Intuitively, $\boldsymbol{w}_b(k)\boldsymbol{x}\bar{\Phi}_b$ maps the feature $\boldsymbol{x}\bar{\Phi}_b$ into the space spanned by $\boldsymbol{w}_b$. However, since $\boldsymbol{w}_b$ is all zero in the dimension $1, ..., K$, so $\boldsymbol{w}_b(k)\boldsymbol{x}\bar{\Phi}_b$ has no impact on the prediction of task a (i.e., among class $1, ..., K$).

F.4   Close Form of $F_p(x)$

Here we provide the explicit expression of function $F_p(x)$ in K class situation, which is monotonically increasing with x.

We denote a $K - 1$-dim random variable $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{x}, \boldsymbol{M})$, in which

$$\boldsymbol{M}_{i,i} = \frac{p(K + 2 - pK)}{K}, \boldsymbol{M}_{i,j} = \frac{p(K + 1 - pK)}{K},$$

then $F_p(x)$ is defined as

$$F_p(x) = \mathbb{P}(\boldsymbol{\eta}_1 > 0, \ldots, \boldsymbol{\eta}_{K-1} > 0).$$

G   Hyper-Parameters

Table 2: Hyper-parameters for RLHF experiments with Open-LLaMA-3B. Δ means that the parameter will be specified in each individual experiment. For LoRA training, the omitted hyper-parameters are set as the full training.

MODELS AND METHODS	HYPER-PARAMETER	VALUE
PPO TRAINING	TEMPERATURE	1.0
	DATA COLLECTION BATCH SIZE	512
	LEARNING RATE	1×10^{-6}
	UPDATE EPOCH	2
	UPDATE BATCH SIZE	32
	KL COEFFICIENT	Δ
	REWARD BASELINE	5.5625
PPO LoRA TRAINING	LEARNING RATE	1×10^{-5}
	UPDATE EPOCH	4
	UPDATE BATCH SIZE	32
	KL COEFFICIENT	Δ
	REWARD BASELINE	5.5625
	LoRA RANK	16
	LoRA α	32
	LoRA DROPOUT	0.05
RSF TRAINING	TEMPERATURE	1.0
	BATCH SIZE	2048
	LEARNING RATE	1×10^{-5}
	EPOCH	2
	UPDATE BATCH SIZE	32
RSF LoRA TRAINING	LEARNING RATE	1×10^{-5}
	EPOCH	2
	UPDATE BATCH SIZE	32
	LoRA RANK	16-512
	LoRA α	32
DPO	LEARNING RATE	1×10^{-6}
	BATCH SIZE	32
	KL COEFFICIENT	0.1

Table 3: Hyper-parameters for auxiliary experiments.

Models and Methods	Hyper-parameter	Value
ShareGPT SFT	Learning rate	1×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	128
	Block size	2048
HH-RLHF SFT	Learning rate	1×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	12
	Block size	2048
RM SFT	Learning rate	2×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	2
	Batch size	12
RM Training	Learning rate	5×10^{-6}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	16
Test Settings	Temperature λ	1.0
	Max new token	196
	Do sample	True

Table 4: Hyper-parameters for HMA experiments.

Models and Methods	Hyper-parameter	Value
RSF HMA	Learning rate	2×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	1
	Block size	512
PPO HMA	Learning rate	4×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	1
	Block size	512
DPO HMA	Learning rate	4×10^{-5}
	Scheduler	Cosine decay with 0.03 warm-up
	Epoch	1
	Batch size	1
	Block size	512