AutoMixAlign: Adaptive Data Mixing for Multi-Task Preference Optimization in LLMs

Nicholas E. Corrado[1,2], Julian Katz-Samuels[2], Adithya Devraj[2], Hyokun Yun[2], Chao Zhang[2,3], Yi Xu[2], Yi Pan[2], Bing Yin[2], Trishul Chilimbi[2],

[1]University of Wisconsin–Madison, [2]Amazon, [3]Georgia Institute of Technology

Correspondence: ncorrado@wisc.edu, jkatzsamuels@gmail.com

Abstract

When aligning large language models (LLMs), their performance on various tasks (such as being helpful, harmless, and honest) depends heavily on the composition of their training data. However, selecting a data mixture that achieves strong performance across all tasks is challenging. Existing approaches rely on large ablation studies, heuristics, or human intuition, but these can be prohibitively expensive and suboptimal. We study this problem in the setting of preference optimization via DPO and introduce AutoMixAlign (AMA), a theoretically-grounded algorithm that adaptively mixes datasets during training to balance performance across tasks. AMA first trains *specialist models* for each task to determine losses that correspond to strong task performance. Then, it trains a generalist model using a novel minimax optimization that prioritizes tasks for which generalist model losses deviate most from specialist model losses. To optimize this problem, we propose two algorithms: (1) AMA-R, which adaptively reweights the objective to prioritize tasks, and (2) AMA-S, which adaptively adjusts how much data is sampled from each task to prioritize tasks. Both algorithms achieve a convergence rate of $O(1/\sqrt{T})$ in the convex case. AMA-R's convergence result follows from Sagawa et al. (2019), and we provide a convergence proof for AMA-S using online learning techniques such as EXP3 (Auer, 2002). We evaluate AMA on several multitask alignment setups and find that AMA outperforms the standard alignment approach—which simply optimizes the total loss across all tasks—and also outperforms model merging methods.

1 Introduction

As LLMs are increasingly deployed in real-world applications, it has become essential to develop generalist models with strong capabilities across a broad range of tasks (*e.g.* math, coding, safety). A key step toward this objective is preference optimization, learning from human or synthetic preference data with notable methods like RLHF (Ouyang et al., 2022) or DPO (Rafailov et al., 2023). To produce generalist models, LLMs are typically trained on a diverse mix of datasets, each targeting different tasks. However, the training data composition can significantly affect performance in each task (Brown et al., 2020; Du et al., 2022; Chowdhery et al., 2023; Ivison et al., 2024). This observation motivates the central question in this work: *How should we mix each task dataset to produce a model that performs well across a variety of tasks?*

Currently, practitioners tackle this data mixing problem using heuristics or large-scale ablation studies on dataset mixtures (Touvron et al., 2023; Dubey et al., 2024; Ivison et al., 2023; Lambert et al., 2024). Both approaches have significant drawbacks: heuristics tend to produce suboptimal models while ablation studies become prohibitively expensive due to the combinatorial nature of dataset mixing. This problem is particularly acute in production settings, where regular model releases must accommodate evolving datasets and product requirements, creating significant ongoing expenses in computation and human effort.

In this paper, we introduce a novel and theoretically justified approach to the data mixing problem in DPO-based preference optimization. Our algorithm, AutoMixAlign (AMA), automatically mixes datasets during LLM training to balance performance across multiple tasks. AMA has two phases. First, AMA trains *specialist models* on each task dataset individually, yielding a strong specialized model for each task. Second, AMA trains a generalist model to match the specialist model performance by optimizing a novel minimax optimization problem that prioritizes more challenging tasks—tasks with large *excess loss* (He et al., 2024; Xie et al., 2024), defined as the difference between the generalist model's losses and the specialist models'

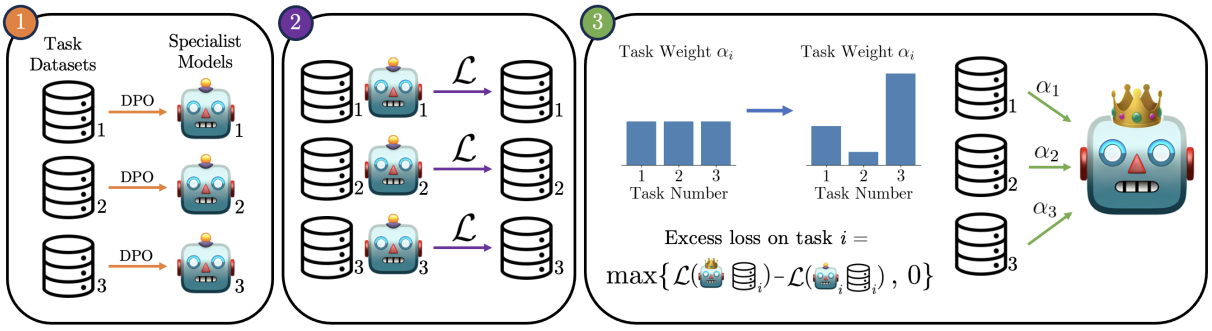

Figure 1: **Overview of AMA**. First, we train *specialist models* on each task dataset individually to obtain a model that performs well on that specific task. Next, we pre-compute the losses achieved by each specialist model and add these losses to their respective datasets. Lastly, we use a minimax optimization algorithm to optimize a generalist model towards the losses achieved by the specialist models.

losses. We present two algorithms for this optimization: a reweighting algorithm that adaptively reweights tasks in the objective, and a resampling algorithm that adaptively adjusts how much data is sampled from each task.

Both algorithms enjoy theoretical guarantees. The reweighting algorithm has an $O(1/\sqrt{T})$ convergence rate in the convex setting, following immediately from Sagawa et al. (2019). For the resampling algorithm, we prove convergence by modeling it as a two-player zero-sum game in which the learner aims to minimize the excess losses, while the adversary shifts the mixture distribution to sample more from tasks with larger excess losses. Using online learning techniques such as EXP3 (Auer, 2002), we prove that the resampling algorithm has an $O(1/\sqrt{T})$ convergence rate in the convex case.

We conduct extensive experiments demonstrating that AMA effectively balances helpfulness, harmlessness, and reasoning tasks. In all experiments, AMA outperforms standard baselines such as uniform data mixing and model merging, which often perform poorly on one or more tasks. AMA improves average performance over standard DPO with uniform data mixing by up to 9.42% while maintaining robust performance across each task. In summary, we make the following contributions:

1. We introduce AMA, a multi-task learning algorithm that adaptively prioritizes more challenging tasks via reweighting or resampling.

2. We prove that the resampling-based AMA algorithm has a $O(1/\sqrt{T})$ convergence rate in the convex case.

3. We show that AMA has strong empirical performance, improving over standard data mixing approaches by up to 9.42%.

2 The Setup

We assume we have k preference datasets, one for each task i. Each dataset is a collection of triples: $\mathcal{D}_i = \{(x_1^{(i)}, y_{1,+}^{(i)}, y_{1,-}^{(i)}), \ldots, (x_n^{(i)}, y_{n,+}^{(i)}, y_{n,-}^{(i)})\}$ with $x_j^{(i)}$ denoting the jth query, $y_{j,+}^{(i)}$ denoting the *preferred* response on the query, and $y_{j,-}^{(i)}$ denoting the *not* preferred response on the query. Each dataset may correspond to different tasks such as coding, math, general helpfulness, or safety.

We focus on preference learning using DPO (Rafailov et al., 2023). Let $\pi_{\boldsymbol{\theta}}$ denote a language model parameterized by $\boldsymbol{\theta}$ with $\pi_{\boldsymbol{\theta}}(y|x)$ denoting the probability that the model outputs response y given prompt x. In DPO, we are given a *reference* model $\pi_{\texttt{ref}}$ and use it to initialize $\pi_{\boldsymbol{\theta}}(y|x)$. The loss for a particular sample (x, y_+, y_-) is as follows: $\mathcal{L}(\boldsymbol{\theta}; x, y_+, y_-) :=$

$$-\log\left(\sigma\left(\beta\frac{\pi_{\boldsymbol{\theta}}(y_+|x)}{\pi_{\texttt{ref}}(y_+|x)} - \beta\frac{\pi_{\boldsymbol{\theta}}(y_-|x)}{\pi_{\texttt{ref}}(y_-|x)}\right)\right)$$

where σ is the sigmoid function and $\beta > 0$ is a hyperparameter.

To train a model via DPO across k task datasets, the standard practice optimizes a reweighted loss over all datasets (Ivison et al., 2024):

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{k} \alpha_i \sum_{(x,y_+,y_-)\in\mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}; x, y_+, y_-) \quad (1)$$

where $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$ are positive weights. In practice, two choices of dataset weightings are typically used: (1) uniform weighting, $\alpha_i = 1$ for all $i \in [k]$, and (2) task-normalized weighting, $\alpha_i = \frac{1}{|\mathcal{D}_i|}$ for all $i \in [k]$.

Both weighting choices have several limitations. Uniform weighting is sensitive to the size of task datasets; if $|\mathcal{D}_i| > |\mathcal{D}_j|$, then task i receives more weight since it contributes more to the total loss than task j. As a result, practitioners often need to run many experiments to determine a dataset mixture that yields strong performance across all tasks (Ivison et al., 2024). Task-normalized weighting ensures all tasks are weighted equally by normalizing by the dataset size. However, this rigidity can be suboptimal; if some tasks are easier to learn than others, giving all tasks equal weight may cause the model to focus too much on easy tasks at the expense of harder ones.

To address these challenges with standard data mixing approaches in post-training, in the next section, we introduce AutoMixAlign (AMA), our new framework for balancing performance across tasks. To simplify our notation, we henceforth let $z = (x, y_+, y_-)$ denote an example and write $\mathcal{L}(\boldsymbol{\theta}, z) := \mathcal{L}(\boldsymbol{\theta}; x, y_+, y_-)$.

3 AutoMixAlign (AMA)

In this section, we present AMA. We derive the AMA optimization problem, describe two algorithms to optimize it, and then provide theoretical analysis.

3.1 Deriving the AMA Optimization Problem

Suppose we have model parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$ where each $\boldsymbol{\theta}_i$ was obtained via DPO training on $\mathcal{D}_i$. We refer to $\boldsymbol{\theta}_i$ as the *specialist* model parameters for task i since $\boldsymbol{\theta}_i$ should perform well on task i but may perform poorly on the remaining tasks.[1] Since task losses tend to be predictive of performance (Chen et al., 2024), a natural approach would be to find model parameters $\boldsymbol{\theta}$ that match the losses achieved by $\boldsymbol{\theta}_i$ in each task i by solving the following feasibility problem:

$$\text{Find } \boldsymbol{\theta} \text{ s.t.} \quad \forall i \in [k], \quad \frac{1}{|\mathcal{D}_i|}\left[\sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, z) - \sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}_i, z)\right] \leq 0. \tag{2}$$

[1]We train specialist models on individual datasets to ensure they achieve strong performance, though alternative approaches may exist. For instance, it may be possible to train a strong specialist for two tasks i and j by training on $\mathcal{D}_i \cup \mathcal{D}_j$.

LLMs are typically sufficiently expressive so that there exists a solution to feasibility problem 2. However, since this problem is not computationally tractable, we relax it into a minimization problem over constraint violations:

$$\min_{\boldsymbol{\theta}} \max_{i \in [k]} \max \left\{ \frac{1}{|\mathcal{D}_i|}\left[\sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, z) - \sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}_i, z)\right], 0 \right\}. \tag{3}$$

Unfortunately, we cannot apply stochastic gradient descent directly to optimization problem 3 because the outer $\max\{\cdot, 0\}$ function makes the gradient computed over a random mini-batch a biased estimator of the true gradient with respect to the full data distribution. To make this optimization problem 3 tractable, we can upper bound it with the following optimization problem:

$$\min_{\boldsymbol{\theta}} \max_{i \in [k]} \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \max\left\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\right\}. \tag{4}$$

Optimization problem 4 is the core optimization problem that AMA aims to solve. In this formulation, $\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z)$ denotes the excess loss for sample z on task i, and $\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z) := \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$ is the *clipped* excess loss, quantifying how much our generalist model's loss deviates from the loss achieved by our specialist model $\boldsymbol{\theta}_i$. The excess loss has been used in prior works to prevent optimization from focusing on tasks that have already been learned (*i.e.*, tasks with zero excess loss) at the expense of more challenging tasks (He et al., 2024; Xie et al., 2024). For completeness, we further discuss the excess loss in Appendix E.[2]

How do we interpret optimization problem 4? By minimizing the maximum clipped excess loss, this optimization aims to train a generalist model that matches the performance of all specialist models $\boldsymbol{\theta}_i$. Toward this aim, it focuses on the task with the largest average clipped excess losses. In particular, once it has learned a task so that all the clipped excess losses are zero, it stops optimizing that task; if $\boldsymbol{\theta}_i$ maximizes performance on task i, then optimizing beyond $\mathcal{L}(\boldsymbol{\theta}_i, z)$ has no benefit and may

[2]Since specialist model losses $\mathcal{L}(\boldsymbol{\theta}_i, z)$ are fixed throughout training, we reduce AMA's memory footprint by precomputing $\mathcal{L}(\boldsymbol{\theta}_i, z)$ and adding them to a new column in each $\mathcal{D}_i$ before training our generalist model.

eventually lead to overfitting. In the following sections, we discuss two different algorithms to solve this optimization problem: objective reweighting and task resampling.

3.2 Reweighting Algorithm (AMA-R)

The optimization problem 4 is equivalent to

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha} \in \Delta^k} \sum_{i=1}^{k} \alpha_i \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z) \quad (5)$$

where $\Delta^k = \{\boldsymbol{\alpha} \in \mathbb{R}^k : \sum_i \alpha_i = 1, \alpha_i \geq 0\}$ is a probability simplex. To optimize this new objective, we follow Sagawa et al. (2019) and interleave gradient updates on task weights $\boldsymbol{\alpha}$ and model parameters $\boldsymbol{\theta}$, increasing the weight on tasks with larger excess losses. Specifically, we sample a batch of data from each task uniformly at random, update $\boldsymbol{\alpha}$ using exponentiated gradient ascent on the clipped excess loss of each task, and then update $\boldsymbol{\theta}$ using the loss $\sum_{i=1}^{k} \alpha_i \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z)$ and a standard first-order optimization algorithm. Since this algorithm reweights the objective using α, we call it AMA Reweighting (AMA-R). Due to space constraints, we present this algorithm in Algorithm 2 of Appendix B. For this interleaved optimization approach, Sagawa et al. (2019) previously established a convergence rate of $O(1/\sqrt{T})$ in the convex setting.[3]

3.3 Resampling Algorithm (AMA-S)

The reweighting algorithm described in the previous section prioritizes tasks by adjusting the objective weights $\boldsymbol{\alpha}$. However, when some weights α_i are very small, this method is inefficient; we incur the same cost to compute each task's contribution to the gradient, but tasks with small weights have little impact on the model update. To illustrate, suppose the extreme: all weight is concentrated on task i ($\alpha_i = 1, \alpha_j = 0 \; \forall j \neq i$). Letting b denote the batch size, AMA-R samples b/k examples from every task, but examples from all tasks other than task i are wasteful as they don't contribute to the gradient calculation.

We can alternatively use $\boldsymbol{\alpha}$ to control the probability of sampling from each task's dataset when optimizing problem 4. We refer to this resampling algorithm as AMA Resampling (AMA-S) and provide its implementation in Algorithm 1. Like AMA-R, AMA-S alternates between updating $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$. For a given $\boldsymbol{\alpha}$, we sample batches according to Multinomial$(\alpha_1, \ldots, \alpha_k)$ and update $\boldsymbol{\theta}$ using standard first-order optimization. The $\boldsymbol{\alpha}$ updates use the EXP3 online learning algorithm (Auer, 2002) to adaptively adjust sampling probabilities to focus computation on tasks with large excess loss. For EXP3 updates, we maintain an internal weight vector $\boldsymbol{q} \in \Delta^k$ and compute the sampling distribution as $\boldsymbol{\alpha} = (1-c)\boldsymbol{q} + c\frac{1}{k}\mathbf{1}$, where $c \in (0, 1)$ smooths the sampling distribution towards a uniform distribution. With AMA-S, all examples from every task have equal weight in the gradient calculation, improving the computational efficiency when task weights are unequal. In Section 4.4, we empirically validate this argument.

Since the convergence proof given by Sagawa et al. (2019) applies only to the reweighting algorithm, we provide a new convergence result for this resampling based algorithm. In the next section, we will show that AMA-S can be viewed as a two-player zero-sum game between an $\boldsymbol{\alpha}$-player and a $\boldsymbol{\theta}$-player and prove a standard $O(1/\sqrt{T})$-style convergence rate in the convex case.

3.4 Theoretical Analysis

In this section, we show that AMA-S is a theoretically principled multi-task learning algorithm. Again, we consider the setting where we have k datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, each with $\mathcal{D}_i = \{x_1^{(i)}, \ldots, x_n^{(i)}\}$. We consider the function class $\{\boldsymbol{\theta} \in \Theta\}$ and suppose $\mathcal{D}_i \subset \mathcal{Z}$ for all $i \in [k]$. For simplicity, we abstract away from the excesses losses and consider the following:

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\alpha} \in \Delta^k} \sum_{i} \alpha_i \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, z).$$

We can model this problem as a zero-sum game between two players, the $\boldsymbol{\theta}$-player and the $\boldsymbol{\alpha}$-player. The protocol is as follows. For round $t = 1, 2, \ldots$

1. The $\boldsymbol{\alpha}$-player chooses $\boldsymbol{\alpha}_t \in \Delta^k$.
2. The $\boldsymbol{\theta}$-player chooses $\boldsymbol{\theta}_t \in \Theta$.
3. The $\boldsymbol{\theta}$-player samples $i_t \sim \boldsymbol{\alpha}_t$.
4. The $\boldsymbol{\theta}$-player suffers loss $\frac{1}{|\mathcal{D}_{i_t}|} \sum_{(x,y) \in \mathcal{D}_{i_t}} \mathcal{L}(\boldsymbol{\theta}_t, x)$.

[3] While the convergence rate of AMA-R follows from Sagawa et al. (2019), we note that AMA-R optimizes a different objective; Sagawa et al. (2019) optimizes the loss $\mathcal{L}(\boldsymbol{\theta}, z)$ whereas AMA-R optimizes the clipped excess loss $\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z)$.

Algorithm 1 AutoMixAlign Resampling (AMA-S)

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, specialist model parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$, batch size b, smoothing parameter $c \in (0, 1)$, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
For all tasks, compute $\mathcal{L}(\boldsymbol{\theta}_i, z)$ for each $z \in \mathcal{D}_i$, and add them to a new column in $\mathcal{D}_i$
Initialize model parameters $\boldsymbol{\theta}$
$\boldsymbol{q}^{(1)} \leftarrow (\frac{1}{k}, \ldots, \frac{1}{k})$
for $t = 1, \ldots, T$ **do**
  $\alpha_i^{(t)} \leftarrow (1 - c)q_i^{(t)} + c\frac{1}{k}, \forall i$
  $\mathcal{T} \sim \text{Multinomial}(\alpha_1^{(t)}, \ldots, \alpha_k^{(t)}, b)$
  $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
  $\mathcal{B}_i \leftarrow \mathcal{B} \cap \mathcal{D}_i$ for each task i
  Update $\boldsymbol{q}^{(t)}$ using EXP3 (Auer, 2002):

$$\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z) := \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$$

$$q_i^{(t+1)} \leftarrow q_i^{(t)} \exp\left\{\frac{1}{q_i^{(t)}} \frac{1}{|\mathcal{B}_i|} \sum_{z \in \mathcal{B}_i} \mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\theta}_i, z)\right\}$$

$$q_i^{(t+1)} \leftarrow \frac{q_i^{(t+1)}}{\sum_{j=1}^k q_j^{(t+1)}}$$

  Update $\boldsymbol{\theta}$ using one step of gradient descent on the loss

$$\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$$

end for

For our convergence result, we require access to a $\boldsymbol{\theta}$-player that satisfies the following property.

Definition 3.1. We say that the $\boldsymbol{\theta}$-player is C-low regret if for any sequence of indices $\{i_1, \ldots i_T\}$, we have that

$$\sum_{t=1}^{T} \frac{1}{|\mathcal{D}_{i_t}|} \sum_{z \in \mathcal{D}_{i_t}} \mathcal{L}(\boldsymbol{\theta}_t, z) \leq \min_{\boldsymbol{\theta} \in \Theta} \frac{1}{|\mathcal{D}_{i_t}|} \sum_{z \in \mathcal{D}_{i_t}} \mathcal{L}(\boldsymbol{\theta}, z) + C\sqrt{T}.$$

Now, we show a case where such a C-low-regret $\boldsymbol{\theta}$-player exists. Suppose that $z = (x, y)$, $f_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta} \cdot x$, $\mathcal{L}(\boldsymbol{\theta}, x) = l(f_{\boldsymbol{\theta}}(x), y)$. Note that we have $f_{\boldsymbol{\theta}}(x) = w \cdot \phi(x)$ where ϕ is a feature extractor. Then, online gradient descent is an example of such a low-regret $\boldsymbol{\theta}$-player.

Example 1. *Let* $l(f_{\boldsymbol{\theta}}(x), y) = (\boldsymbol{\theta} \cdot x - y)^2$, $\Theta = \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\|_2 \leq 1\}$, $y \in [-1, 1]$, *and* $x \in \{x : \|x\|_2 \leq 1\}$. *Then, online gradient descent is* C*-low-regret algorithm with* $C = 4\sqrt{2}$.

See the Appendix for a proof based on online convex optimization (Shalev-Shwartz et al., 2012).

Now, we can prove our main theoretical result under mild simplifying assumptions. It shows that by framing AMA-S as a zero-sum two-player game, the $\boldsymbol{\theta}$-player minimizes task losses while the $\boldsymbol{\alpha}$-player adversarially upweights challenging tasks, leading to balanced convergence.

Theorem 3.2. *Fix* $\epsilon, \delta \in (0, 1)$. *Suppose that the* $\boldsymbol{\theta}$*-player is a* C*-low-regret algorithm. Suppose* $\max_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta}, z) \leq 1$ *for all* $z \in \mathcal{Z}$. *Then, if* $T \geq \max(\frac{64C^2}{\epsilon^2}, \frac{32k \log(k)}{\epsilon})$ *and* $\eta = \frac{1}{4k}$, *we have that*

$$\max_{i \in [k]} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}_t, z) \leq \min_{\boldsymbol{\theta} \in \Theta} \max_{i \in [k]} \frac{1}{|\mathcal{D}_i|} \sum_{z \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, z) + \epsilon.$$

This result establishes a standard $O(1/\sqrt{T})$ convergence rate for the AMA-S algorithm on multi-task learning under mild assumptions. The argument

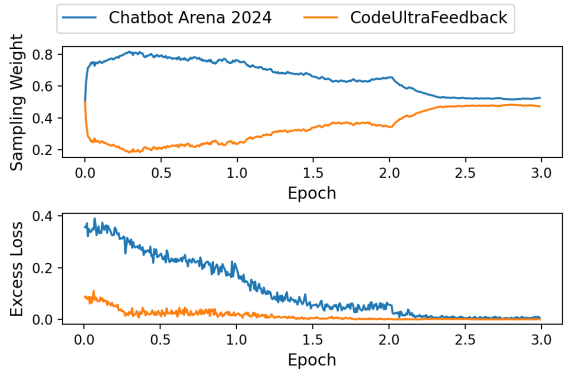

Figure 2: Excess losses and task weights for AMA-S in Setup 1, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback).

extends classical arguments from (Shalev-Shwartz and Wexler, 2016) to the multi-task setting and beyond the realizable setting to the agnostic setting where there may not be a single predictor that perfectly fits the data.

4 Experiments

We consider the following baselines:

1. **Standard optimization (Standard)**: Combine all task datasets and minimize the total loss—the standard approach in DPO training.

2. **Standard optimization with uniform task sampling (Standard Uniform)**: Minimize the total loss across tasks but sample data from each task with probability $1/k$.

3. **Model Averaging** (Yang et al., 2024): A heuristic model merging method that averages the parameters of the specialist models, giving equal weight to each model.

We provide pseudocode for Standard and Standard Uniform in Algorithm 3 and 4 of Appendix B.

Our experiments focus on Helpfulness (Alpaca Eval 2.0 (Dubois et al., 2024) and IFEval (Zhou et al., 2023)), Harmlessness (Toxigen (Hartvigsen et al., 2022)), and Coding (MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021)). We consider the following datasets: UltraFeedback (Cui et al., 2023), Chatbot Arena 2024 (Zheng et al., 2023), CodeUltraFeedback (Weyssow et al., 2024), and PKU-SafeRLHF with harmlessness preference labels (SafeRLHF) (Ji et al., 2023). Each dataset primarily improves performance on only one of three tasks, so we combine datasets to investigate how to produce a model that performs well across all tasks. We consider the following dataset combinations: Chatbot Arena 2024 + CodeUltraFeedback (Helpfulness + Coding), UltraFeedback + SafeRLHF (Helpfulness + Harmlessness), and Chatbot Arena 2024 + Coding + SafeRLHF (Helpfulness + Coding + Harmlessness).

In all experiments, we use the open source Zephyr 7b SFT Full (Tunstall et al., 2023) as the base model, and train all models using DPO (Rafailov et al., 2023). To obtain specialist models, we train separate models via DPO on each task dataset for 3 epochs. We provide full training details for both specialist and generalist models Appendix G as well as evaluation details in Appendix H. Due to space constraints, we provide ablations on different features of AMA and different model merging approaches in Appendix C.

4.1 Setup 1: Helpfulness + Coding

Our first experiment focuses on balancing helpfulness and harmlessness using Chatbot Arena 2024 and SafeRLHF. As shown in Table 1, our Chatbot Arena 2024 specialist improves helpfulness but not coding. On the other hand, our CodeUltraFeedback specialist marginally improves helpfulness but greatly improves coding.

From Table 1, we see that both AMA algorithms achieve the highest average scores and balance performance across both tasks. In fact, they are the only methods that do not degrade performance on at least one benchmark. On the other hand, Standard and Standard Uniform degrade coding performance while Model Averaging scores lower than both AMA algorithms on AlpacaEval and MBPP.

To understand how AMA balances tasks, Figure 2 shows task-specific excess losses and task weights during AMA-S training. As expected, AMA-S initially shifts weights to the task with higher excess loss, Chatbot Arena 2024, and then increases the weight on CodeUltraFeedback as the gap between the two tasks decreases. Since the excess loss on Chatbot Arena 2024 remains slightly larger than that of CodeUltraFeedback at convergence, the weight on Chatbot Arena 2024 also remains slightly larger. We plot task weights and excess losses for AMA-R in Figure 5 of Appendix G and observe qualitatively similar behavior.

4.2 Setup 2: Helpfulness + Harmlessness

Our second experiment focuses on balancing helpfulness and harmlessness using UltraFeedback and SafeRLHF. As shown in Table 2, our UltraFeed-

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%)↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	36.08
Chatbot Arena 2024 Specialist	43.44	16.05	44.60	50.61	38.68
CodeUltraFeedback Specialist	35.86	9.93	52.16	57.32	38.82
Standard	39.74	15.23	37.77	47.82	35.14
Standard Uniform	42.51	11.20	36.33	51.83	35.47
Model Averaging	42.14	11.51	48.56	**55.49**	39.43
AMA Reweighting	42.51	**18.15**	**51.44**	54.88	**41.75**
AMA Resampling	**43.44**	15.61	51.08	50.61	40.19

Table 1: Setup 1, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback).

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	51.60	31.85
UltraFeedback Specialist	42.51	17.59	36.10	32.07
SafeRLHF Specialist	29.02	6.66	100.00	45.23
Standard	41.22	6.90	**99.99**	49.37
Standard Uniform	44.18	15.51	99.93	53.21
Model Averaging	39.37	12.21	99.96	50.51
AMA Reweighting	**44.55**	16.21	99.74	53.50
AMA Resampling	**44.55**	**16.88**	**99.99**	**53.81**

Table 2: Setup 2, Helpfulness (UltraFeedback) + Harmlessness (SafeRLHF).

back specialist improves helpfulness, but leaves substantial room for improvement on Toxigen. However, our SafeRLHF specialist achieves a perfect score on Toxigen but degrades helpfulness.

In Table 2, both AMA algorithms achieve the highest average scores and achieve the highest scores on each benchmark individually. Here, AMA-S and AMA-R perform similarly. We observe that Standard optimization can be sensitive to the task weighting: Standard Uniform performs similarly to both AMA algorithms, but Standard *degrades* performance on AlpacaEval significantly. Model averaging offers only modest improvements on helpfulness benchmarks.

4.3 Setup 3: Helpfulness + Coding + Harmlessness

Our third experiment builds off of Setup 1 and focuses on balancing helpfulness, coding, and harmlessness using Chatbot Arena 2024, CodeUltraFeedback, and SafeRLHF. As shown in Table 3, the Chatbot Arena 2024 and CodeUltraFeedback specialists have significant room for improvement on Toxigen, while our SafeRLHF specialist achieves a perfect Toxigen score but degrades performance on helpfulness and coding.

As shown in Table 3, both AMA algorithms achieve the highest average score across benchmarks while also improving on each benchmark individually. AMA-R outperforms AMA-S on IFEval, AlpacaEval, and HumanEval. We again observe that Standard optimization can be sensitive to task weighting: Standard scores 6 points higher on average than Standard Uniform. Model Averaging balances task scores but nevertheless achieves relatively low scores.

4.4 Resampling vs. Reweighting

Although we can optimize the AMA objective using reweighting or resampling, we now show that AMA converges more slowly with certain dataset mixtures when using reweighting. In particular, we show how *uniformly* sampling across datasets (as done by AMA-R) can result in much slower convergence compared to adaptive sampling (as done by AMA-S). We train AMA-S and AMA-R for one epoch on a Coding + Harmlessness dataset mixture consisting of 1 copy of a 10^4-sample subset of SafeRLHF and 10 copies of a 250-sample subset of CodeUltraFeedback *each representing separate tasks* so that we have $k = 11$ tasks.

How might AMA-R struggle in this setting? Let b denote the batch size so that one epoch corresponds to $\frac{1}{b}(250 \cdot 10 + 10^4)$ updates. Since AMA-R samples tasks uniformly, it will in expectation observe $\frac{b}{11} \cdot \frac{1}{b}(250 \cdot 10 + 10^4) \approx 1136$ SafeRLHF samples in one epoch, which may not be sufficient to produce a harmless model. In contrast, AMA-S

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	51.60	39.18
Chatbot Arena 2024 Specialist	43.44	16.05	44.60	50.61	69.77	44.89
CodeUltraFeedback Specialist	35.86	9.93	52.16	57.32	76.31	46.31
SafeRLHF Specialist	29.02	6.66	48.56	50.61	100.00	46.97
Standard	38.63	11.32	50.00	54.88	99.91	50.95
Standard Uniform	47.50	11.78	27.70	37.80	**99.99**	44.96
Model Averaging	42.70	12.28	52.16	53.66	96.70	51.50
AMA Reweighting	**48.43**	**17.77**	54.32	**55.49**	95.91	**54.38**
AMA Resampling	43.62	13.04	**55.76**	53.66	99.80	53.18

Table 3: Setup 3, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback) + Harmlessness (SafeRLHF).

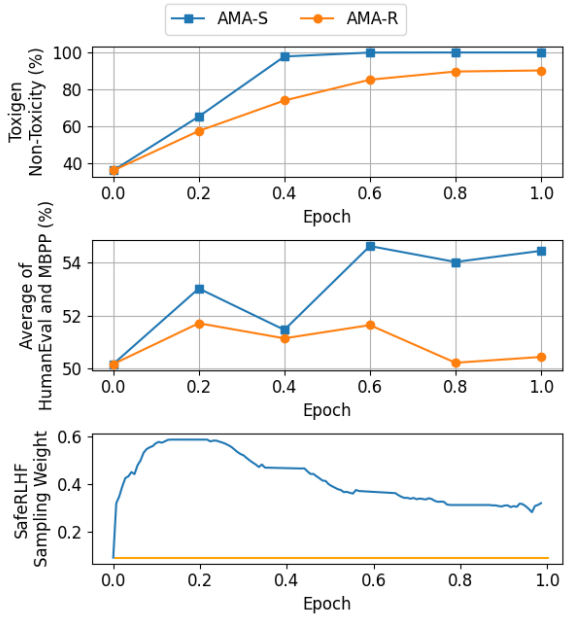

Figure 3: AMA-R and AMA-S in Coding (CodeUltraFeedback) + Harmlessness (SafeRLHF).

can adaptively increase the probability of sampling from SafeRLHF to improve harmlessness.

In Figure 3, we observe that AMA-S quickly converges to almost 100% non-toxicity, whereas AMA-R reaches a maximum of 90%. To understand why, we also plot the probability of sampling from SafeRLHF with AMA-R (which is always $1/11$) and AMA-S. We see that AMA-S quickly places significantly more weight on SafeRLHF and thus observes more SafeRLHF data than AMA-R.

5 Related Work

Data Mixing. Prior works often use heuristic filters (Brown et al., 2020; Du et al., 2022; Chowdhery et al., 2023) or ablations studies to produce a data mixture that yields strong performance (Ivison et al., 2024). Recently, many works have leveraged group distributionally robust optimization (GroupDRO) to automatically determine an appropriate data mixture (Oren et al., 2019; Sagawa et al., 2019; Michel et al., 2021; Albalak et al., 2023; Xia et al., 2023; Xie et al., 2024; He et al., 2024). Similar to AMA, DRO methods use a minimax optimization to optimize the worst-case loss or excess loss. However, AMA has two core differences: (1) AMA focuses on alignment rather than pretraining, and (2) AMA optimizes the clipped excess loss during model updates rather than the loss or excess loss. We highlight other algorithmic differences between AMA and closely-related DRO-style algorithms in Appendix D. Pan et al. (2024) use bilevel minimax optimization to reweight tasks for LLM fine-tuning, though this approach also minimizes the loss rather than the clipped excess loss. De Langis et al. (2024) consider dynamically weighting multiple dimensions of alignment, but they focus on controllable generation and two-stage RLHF which assume reward models are separately trained before LLM alignment.

Gradient Conflict Resolution. Multi-task learning methods based on multiple-gradient descent (MGDA) (Desideri, 2009) compute gradients that reduces the loss of all tasks at every step in a balanced fashion (Sener and Koltun, 2018; Navon et al., 2022; Yu et al., 2020; Liu et al., 2021b,a; Fernando et al., 2022). Other methods seek to eliminate conflicting components of task-specific gradients (Chen et al., 2020; Yu et al., 2020) or simply rescale task-specific gradients to have similar norms (Chen et al., 2018). These gradient-based approaches differ fundamentally from data reweighting and resampling methods, as they address multi-task conflicts through geometric operations in gradient space rather than through adjustments to the loss or training distribution.

Model Merging. Model merging methods combine the parameters of separate models to produce a new model that shares the capabilities of each

individual model. The most basic approach is to uniformly average the parameters across models (McMahan et al., 2017; Stich, 2018; Wortsman et al., 2022), though non-uniform merging methods also exist (Matena and Raffel, 2022; Jin et al., 2022; Tam et al., 2024). While model merging is cheaper than fine-tuning—the approach taken in this paper—model merging is heuristic in nature and generally leads to worse performance than fine-tuning (Yang et al., 2024).

6 Conclusion

We introduced AutoMixAlign (AMA), an adaptive data mixing strategy that uses minimax optimization to prioritize tasks with larger excess loss. We provide two algorithms for AMA: one adaptively reweights how much each task contributes to the objective function (AMA-R), while the other adaptively updates how much data is sampled from each task during model updates (AMA-S). Both algorithms are theoretically justified and have $O(1/\sqrt{T})$ convergence rates in the convex case. AMA-R's convergence result follows from Sagawa et al. (2019), and we provide a proof of AMA-S's convergence rate. We evaluate both AMA algorithms on several multitask alignment setups, and observe that they outperform the standard alignment approach which simply optimizes the total loss across all tasks and also outperform model merging methods. We outline directions for future exploration in Appendix F.

7 Limitations

Specialist model training overhead. AMA computes excess losses using separate specialist models trained on each task dataset individually. Training these specialist models can introduce significant training overhead compared to other multi-task learning methods like MGDA (Desideri, 2009), model merging (Wortsman et al., 2022), or simply optimizing the total loss, especially when the number of tasks is large or when the size of each task dataset is large. However, we emphasize that AMA requires the compute equivalent to *just two optimization runs of DPO*. More concretely, training separate specialist models for m epochs requires $\sum_{i=1} m|\mathcal{D}_i| = m|\mathcal{D}|$ model updates, and training the generalist model requires an additional $m|\mathcal{D}|$ model updates—a total of $2m|\mathcal{D}|$ model updates. In other words, the ratio of the number of training steps needed for AMA training vs. standard DPO does not depend on the number of tasks. Since the common practice in the literature is to run many data mixture ablations to optimize performance across many tasks (*e.g.*, Ivison et al. (2024)), AMA in its current form can already improve the efficiency of the practice. Existing techniques in the literature can reduce the expense of computing specialist losses. For instance, ExcessMTL (He et al., 2024) uses a diagonal estimation of the generalist model's Hessian matrix to approximate excess losses, while DoReMi and RegMix (Liu et al., 2024) estimate specialist losses using models that are much smaller than the generalist model.[4] While DoReMi (Xie et al., 2024) trains a single multitask model to compute excess losses, we note that this approach requires just as many model updates as training k specialist models. AMA has the added advantage that specialist models can be trained in parallel, reducing the total time required to train them by a factor of $1/k$.

AMA is specific to DPO. Our experiments and theoretical contributions focus on preference alignment with DPO, though empirical findings may not transfer to other alignment techniques like on-policy learning or supervised fine-tuning. However, one could in principle borrow the algorithmic ideas discussed in this work and use reweighting and resampling to prioritize tasks in other parts of the learning stack, such as on-policy learning or supervised fine-tuning.

Impact Statement

This paper focuses on aligning LLMs to multiple tasks while achieving strong performance across each task individually and represents a significant step towards the development of generalist LLMs. We showcase how our method encourages LLMs to balance its capabilities (*e.g*, to be helpful while also being harmless), and we expect that our findings will have a positive impact on the research community as well as industry practitioners. While it is possible that researchers or practitioners might use our proposed method for negative purposes (*e.g.* training an LLM to balance multiple toxic capabilities), we maintain that this work is largely positive, and that potential negative societal impacts of this work are minimal.

[4] In these works, the specialist models are called *reference models*. We use the term "specialist" throughout this work to avoid confusion with the reference model π_{ref} in the DPO loss.

References

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. 2023. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*.

P Auer. 2002. Finite-time analysis of the multiarmed bandit problem.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. 2024. Scaling laws for predicting downstream performance in llms. *arXiv preprint arXiv:2410.08527*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050.

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.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback.

Karin De Langis, Ryan Koo, and Dongyeop Kang. 2024. Dynamic multi-reward weighting for multi-style controllable generation. *arXiv preprint arXiv:2402.14146*.

Jacopo Desideri. 2009. Multiple-gradient descent algorithm for multiobjective optimization. *Journal of Optimization Theory and Applications*, 142(3):639–656.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

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

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Xiequan Fan, Ion Grama, and Quansheng Liu. 2012. Hoeffding's inequality for supermartingales. *Stochastic Processes and their Applications*, 122(10):3545–3559.

Heshan Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. 2022. Mitigating gradient bias in multi-objective learning: A provably convergent stochastic approach. *arXiv preprint arXiv:2210.12624*.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*.

Elad Hazan, Tomer Koren, and Nati Srebro. 2011. Beating sgd: Learning svms in sublinear time. *Advances in Neural Information Processing Systems*, 24.

Yifei He, Shiji Zhou, Guojun Zhang, Hyokun Yun, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Han Zhao. 2024. Robust multi-task learning with excess risks. *arXiv preprint arXiv:2402.02009*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hanna Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *ArXiv*, abs/2406.09279.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou

Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of LLM via a human-preference dataset. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

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.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021a. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021b. Towards impartial multi-task learning. iclr.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2024. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*.

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Paul Michel, Sebastian Ruder, and Dani Yogatama. 2021. Balancing average and worst-case accuracy in multitask learning. *arXiv preprint arXiv:2110.05838*.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. 2023. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*.

Shay Oren et al. 2019. Distributionally robust language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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.

Rui Pan, Jipeng Zhang, Xingyuan Pan, Renjie Pi, Xiaoyu Wang, and Tong Zhang. 2024. Scalebio: Scalable bilevel optimization for llm data reweighting. *arXiv preprint arXiv:2406.19976*.

Rafael Rafailov et al. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2309.09341*.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. 2019. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Shai Shalev-Shwartz and Yonatan Wexler. 2016. Minimizing the maximal loss: How and why. In *International Conference on Machine Learning*, pages 793–801. PMLR.

Shai Shalev-Shwartz et al. 2012. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194.

Sebastian U Stich. 2018. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*.

Derek Tam, Mohit Bansal, and Colin Raffel. 2024. Merging by matching models in task parameter subspaces. *Transactions on Machine Learning Research*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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*.

Martin Weyssow, Aton Kamanda, and Houari Sahraoui. 2024. Codeultrafeedback: An llm-as-a-judge dataset for aligning large language models to coding preferences. *arXiv preprint arXiv:2403.09032*.

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. 2022. 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.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2024. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Appendix

Table of Contents

A Theory **14**

A.1 Lemmas . . . 14

A.2 Proof of Theorem 3.2 . . . 14

A.3 Additional Results . . . 16

B Algorithms **16**

C Additional Experiments **18**

C.1 AMA Ablations . . . 18

C.2 Model Merging Ablations . . . 21

D Comparing AMA with Existing DRO Algorithms **22**

E Excess Loss **22**

F Future Directions **23**

G Training Details **23**

H Evaluation **24**

H.1 Model Selection . . . 24

H.2 Benchmarks . . . 25

I Licenses **25**

A Theory

In this section, we prove the theoretical results from Section 3.4. We begin with some definitions. A sequence $B_1, \ldots, B_T$ of random variables is said to be Markovian if, for any $t \in [T]$, the variable B_t is independent of $B_1, \ldots, B_{t-2}$ given B_{t-1}. Additionally, a sequence $A_1, \ldots, A_T$ of random variables forms a martingale difference sequence relative to $B_1, \ldots, B_T$ if the conditional expectation $E[A_t|B_1, \ldots, B_t] = 0$ for every $t \in [T]$.

A.1 Lemmas

The Exponential Gradient (EG) algorithm generates a series of vectors, denoted as $z_1, \ldots, z_T$, each belonging to $\mathbb{R}^k$. Let $\eta > 0$. Initialize $\tilde{q}_1 = (1, \ldots, 1) \in \mathbb{R}^k$. For $t \geq 1$,

- Set $\tilde{q}_{t+1,i} = \tilde{q}_{t,i} \exp(-\eta z_{t,i})$ for each $i \in [k]$
- Normalize $\tilde{q}_t$ to obtain q_t such that $q_t = \frac{\tilde{q}_t}{\sum_{i=1}^k \tilde{q}_{t,i}}$

The EG algorithm satisfies the following Lemma.

Lemma A.1 (Shalev-Shwartz et al. (2012), Theorem 2.22)**.** *Assume that $\eta z_{t,i} \geq -1$ for every t and i. Then, for every $u \in \Delta^k$, we have that*

$$\sum_{t=1}^{T} (q_t - u)^\top z_t \leq \frac{\log(k)}{\eta} + \eta \sum_{t=1}^{T} \sum_{i=1}^{k} q_{t,i} z_{t,i}^2.$$

Lemma A.2 (Hazan et al. (2011); Fan et al. (2012))**.** *Let $B_1, \ldots, B_T$ be a Markovian sequence and let $A_1, \ldots, A_T$ be a martingale difference sequence wrt $B_1, \ldots, B_T$. Assume that for every $t \in [T]$, we have $|A_t| \leq V$ and $\mathbb{E}[A_t^2|B_1, \ldots, B_T] \leq s$. Then, for every $\alpha > 0$, we have that*

$$\mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} A_t \geq \alpha\right) \leq \exp\left(-T\frac{\alpha^2}{2s + 2\alpha V/3}\right).$$

In particular, for every $\delta \in (0, 1)$, if

$$T \geq \frac{2(s + \alpha V/3)\log(1/\delta)}{\alpha^2}$$

then with probability of at least $1 - \delta$ we have that $\frac{1}{T}\sum_{t=1}^{T} A_t \leq \alpha$.

A.2 Proof of Theorem 3.2

Proof of Theorem 3.2. To simplify notation, moving forward, we define:

$$L_i(\boldsymbol{\theta}) := \frac{1}{|\mathcal{D}_i|} \sum_{x \in \mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, z).$$

Step 1: Concentration of Probability. To begin, we note that by definition $\alpha_{t,i} = \frac{1}{2k} + \frac{q_{t,i}}{2}$, implying that for all $i \in [k]$ and $t \in [T]$,

$$\frac{q_{t,i}}{\alpha_{t,i}} = \frac{q_{t,i}}{\frac{1}{2k} + \frac{q_{t,i}}{2}} \leq 2$$

and

$$\frac{1}{\alpha_{t,i}} \leq 2k.$$

Fix $i \in [k]$ and let $A_1, \ldots, A_T$ be a martingale difference sequence where $A_t = L_i(\boldsymbol{\theta}_t) - \langle e_i, \frac{L_{i_t}(\boldsymbol{\theta}_t)}{\alpha_{t,i_t}} e_{i_t} \rangle$. We will apply Lemma A.2. Note that $|A_t| \leq (2k+1)$. Furthermore, $\mathbb{E}[A_t^2 | q_t, \boldsymbol{\theta}_t] \leq 2k$ since

$$\mathbb{E}\left[\langle e_{i_t} \frac{L_{i_t}(\boldsymbol{\theta}_t)}{\alpha_{t,i_t}}, e_i \rangle^2 | q_t, \boldsymbol{\theta}_t \right] = \sum_{j=1}^{k} \frac{\alpha_{t,j}}{\alpha_{t,j}^2} \langle e_j L_j(\boldsymbol{\theta}_t), e_i \rangle^2 \leq \frac{1}{\alpha_{t,i}} \leq 2k.$$

where in the first inequality we used $L_j(\theta_t) \leq 1$ by assumption and in the second inequality we used $\frac{1}{\alpha_{t,i}} \leq 2k$. Then by Lemma A.2 if $T \geq 6k \log(k/\delta)/(\varepsilon/8)^2$ we have that with probability at least $1 - \delta/k$, $\frac{1}{T}\sum_{t=1}^{T} A_t \leq \varepsilon/8$. Then, by the union bound, we have that with probability at least $1 - \delta$

$$\forall i \in [m], \quad \frac{1}{T}\sum_{t=1}^{T} L_i(\boldsymbol{\theta}_t) \leq \frac{1}{T}\sum_{t=1}^{T} \langle e_i, \frac{L_{i_t}(\boldsymbol{\theta}_t)}{\alpha_{t,i}} e_{i_t} \rangle + \varepsilon/8. \tag{6}$$

For the rest of the proof, we condition on the above event.

Step 2: the $\boldsymbol{\alpha}$-player. Setting $z_t = -\frac{L_{i_t}(\boldsymbol{\theta}_t)}{\alpha_{t,i_t}} e_{i_t}$, we see that the $\boldsymbol{\alpha}$-player applies the EG algorithm wrt $z_1, \ldots, z_T$. Since $z_{t,i} \geq -2k$ and $\eta \leq \frac{1}{2k}$, Lemma A.1 implies that for every $u \in \Delta^k$,

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \langle q_t - u, z_t \rangle &\leq \frac{\log(k)}{\eta T} + \frac{\eta}{T}\sum_{t=1}^{T}\sum_{i=1}^{k} q_{t,i} z_{t,i}^2 \\
&\leq \frac{\log(k)}{\eta T} + \frac{4k\eta}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t)^2 \\
&\leq \frac{\log(k)}{\eta T} + \frac{4k\eta}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t) \\
&\leq \varepsilon/8 + \frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t)
\end{aligned} \tag{7}$$

where in the last inequality we used $\eta = \frac{1}{4k}$ and $T \geq \frac{32k\log(k)}{\epsilon}$.

Step 3: the $\boldsymbol{\theta}$-player. By definition of our low regret $\boldsymbol{\theta}$-player, we have that for every sequence $i_1, \ldots, i_T$

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t) - \min_{\boldsymbol{\theta}\in\Theta} \frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}) &\leq \frac{C}{\sqrt{T}} \\
&\leq \frac{\epsilon}{8}
\end{aligned}$$

where we used $T \geq \frac{64C^2}{\epsilon^2}$. Then, rearranging and using the fact that the average is always less than or equal to the max, we have:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t) &\leq \min_{\boldsymbol{\theta}\in\Theta} \frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}) + \frac{\epsilon}{8} \\
&\leq \min_{\boldsymbol{\theta}\in\Theta} \max_{i\in[k]} L_i(\boldsymbol{\theta}) + \frac{\epsilon}{8}.
\end{aligned} \tag{8}$$

Step 4: Wrapping it up. Putting it all together, we have that for all $i \in [k]$,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} L_i(\boldsymbol{\theta}_t) &\leq \frac{1}{T}\sum_{t=1}^{T}\langle e_i, \frac{1}{\alpha_{t,i}} L_i(\boldsymbol{\theta}_t) e_i\rangle + \varepsilon/8 \\ &\leq \frac{1}{T}\sum_{t=1}^{T} L_{i_t}(\boldsymbol{\theta}_t) + \varepsilon/4 \\ &\leq \min_{\boldsymbol{\theta}\in\Theta} \max_{i\in[k]} L_i(\boldsymbol{\theta}) + \varepsilon\end{aligned}$$

where the first inequality follows from equation 6, the second inequality follows from equation 7 with $u = e_i$, and the last inequality follows from equation 8.

□

A.3 Additional Results

Proof of Example 1. (Shalev-Shwartz et al., 2012) implies that online gradient descent with learning rate $\eta = \frac{1}{L\sqrt{2T}}$ is $BL\sqrt{2}$-low-regret where L is the Lipschitz constant of $l(f_w(x), y)$ and $\|w\|_2 \leq B$. By assumption, $B = 1$. To bound on the Lipschitz constant of $l(f_w(x), y)$ wrt w over $\mathcal{W}$, it suffices to bound the euclidean norm of the gradient wrt w. Therefore,

$$\begin{aligned}\|\nabla_w l(f_w(x), y)\|_2 &= \|2|w \cdot x - y|x\|_2 \\ &\leq 2|w \cdot x - y|\|x\|_2 \\ &\leq 2|w \cdot x - y| \\ &\leq 2(\|w\|_2\|x\|_2 + |y|) \\ &\leq 4,\end{aligned}$$

which yields a bound of 4, completing the proof. □

B Algorithms

Algorithm 2 AutoMixAlign Reweighting (AMA-R)

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, specialist parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$, batch size b, smoothing parameter $c \in (0, 1)$, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
For all tasks, compute $\mathcal{L}(\boldsymbol{\theta}_i, z)$ for each $z \in \mathcal{D}_i$, and add them to a new column in $\mathcal{D}_i$
Initialize model parameters $\boldsymbol{\theta}$
$\boldsymbol{q}^{(1)} \leftarrow (\frac{1}{k}, \ldots, \frac{1}{k})$
for $t = 1, \ldots, T$ **do**
  $\alpha_i^{(t)} \leftarrow (1 - c)q_i^{(t)} + c\frac{1}{k}, \forall i$
  $\mathcal{T} \sim \text{Uniform}(1, \ldots, k, b)$
  $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
  $\mathcal{B}_i \leftarrow \mathcal{B} \cap \mathcal{D}_i$ for each task i
  Update $\boldsymbol{q}^{(t)}$ using exponentiated gradient ascent:

$$q_i^{(t+1)} \leftarrow q_i^{(t)} \cdot \exp\left\{\frac{1}{|\mathcal{B}_i|} \sum_{z \in \mathcal{B}_i} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}\right\}$$

$$q_i^{(t+1)} \leftarrow \frac{q_i^{(t+1)}}{\sum_{j=1}^k q_j^{(t+1)}}$$

  Perform one step of gradient descent on the loss

$$\sum_{i=1}^k \alpha_i \frac{1}{|\mathcal{B}_i|} \sum_{z \in \mathcal{B}_i} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$$

end for

Algorithm 3 Standard Optimization (Standard)

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, batch size b, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
Initialize model parameters $\boldsymbol{\theta}$
for $t = 1, \ldots, T$ **do**
  Sample minibatch $\mathcal{B}$ containing b samples drawn uniformly at random from $\bigcup_{i=1}^k \mathcal{D}_i$
  Perform one step of gradient descent on the loss $\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \mathcal{L}(\boldsymbol{\theta}, z)$
end for

Algorithm 4 Standard Optimization with Uniform Task Sampling (Standard Uniform)

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, batch size b, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
Initialize model parameters $\boldsymbol{\theta}$
for $t = 1, \ldots, T$ **do**
  $\mathcal{T} \sim \text{Uniform}(1, \ldots, k, b)$
  $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
  Perform one step of gradient descent on the loss $\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \mathcal{L}(\boldsymbol{\theta}, z)$
end for

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	51.60	39.18
Standard	38.63	11.32	50.00	**54.88**	99.91	50.95
AMA-S	43.62	13.04	**55.76**	53.66	99.80	**53.18**
AMA-S Uniform	42.51	**13.90**	52.16	51.83	99.80	52.05
AMA-S DoReMi	**44.81**	10.97	45.32	53.05	**99.99**	50.70
AMA-S Unclipped	42.51	12.48	48.56	53.66	99.84	51.42

Table 4: AMA-S ablations for Setup 3, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback) + Harmlessness (SafeRLHF).

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	51.60	39.18
Standard	38.63	11.32	50.00	54.88	**99.91**	50.95
AMA-R	48.43	17.77	54.32	**55.49**	95.91	**54.38**
AMA-R Uniform	45.47	9.69	**54.68**	53.66	99.54	52.61
AMA-R DoReMi	**49.35**	14.02	53.95	53.66	99.47	54.09
AMA-R Unclipped	47.50	**18.56**	48.56	51.83	96.59	52.61

Table 5: AMA-R ablations for Setup 3, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback) + Harmlessness (SafeRLHF).

C Additional Experiments

In this appendix, we first present AMA ablations to understand how adaptive task weighting and excess loss clipping affect performance. Next, since our model merging baseline in our main experiments computes a uniform average of specialist parameters, we present ablations on model merging with non-uniform parameter averaging. We focus on Setup 3 (Helpfulness + Coding + Harmlessness).

C.1 AMA Ablations

In this appendix, we run additional ablation experiments to better understand how adaptive task weighting and excess loss clipping affect AMA-R and AMA-S performance. We consider the following ablations

1. **AMA-R/S Uniform** (Algorithm 5). Fix task weights at $1/k$ throughout training.
2. **AMA-R/S DoReMi** (Algorithm 6). Fix task weights at the average weights obtained throughout AMA-S training, as is done in DoReMi. (Xie et al., 2024).[5]
3. **AMA-R/S Unclipped** (Algorithm 7). Remove the inner $\max$ in Eq. 4 so that the excess loss is no longer clipped at 0 for model updates.

Note that Algorithms 5, 6, and 7 follow AMA-S. We omit pseudocode for AMA-R ablations because they are analogous to the pseudocode for AMA-S ablations.

We present AMA-S ablation results in Table 4 and AMA-R ablation results in Table 5 All AMA-S ablations yield lower average performance than AMA-S, and all AMA-R ablations yield lower average performance than AMA-R, indicating that fixed weights lead to suboptimal task balancing and that excess loss clipping is critical for good performance.

AMA-S Uniform scores higher on average than Standard whereas AMA-S DoReMi and Standard have similar average scores. This result emphasizes that while there exist fixed task weights that can improve over Standard (uniform weights in this setup), other fixed task weights may not yield improvements. Since it is not obvious how to determine the ideal fixed weighting, adaptive weighting is critical.

AMA-R Unclipped and AMA-S Unclipped degrade performance on MBPP w.r.t. the base model. Without clipping, the generalist model optimizes beyond the losses achieved by specialist models. Since

[5]We use this baselines to determine if a fixed, non-uniform task weighting used in DoReMi is superior to a fixed, uniform weighting. We note that DoReMi determines task weights by optimizing the *unclipped* excess loss, whereas we are optimizing the clipped excess loss. Despite this difference, this baseline captures the spirit of DoReMi.

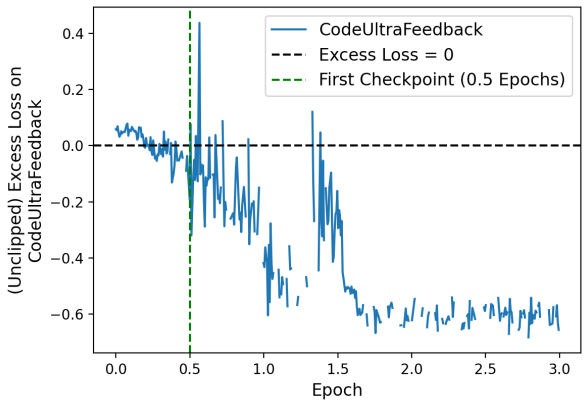

Figure 4: Excess loss on CodeUltraFeedback during AMA-S Unclipped training in Setup 3. We save model checkpoints every 0.5 epochs, and the green dashed line indicates where we save our first model checkpoint. The horizontal dashed line indicates zero excess loss. After 0.5 epochs of training, AMA-S Unclipped achieves negative excess loss, indicating that it has optimized beyond the specialist model losses. The gaps in the excess loss curve arise because in some updates, no CodeUltraFeedback samples were sampled.

specialist model losses correspond to strong performance, further optimization causes our generalist model to overfit to CodeUltraFeedback.

We now show this phenomenon empirically. In Figure 4, we show the (unclipped) excess loss on CodeUltraFeedback during AMA-S Unclipped training. After 0.5 epochs of training, AMA-S Unclipped achieves negative excess loss, indicating that it has optimized beyond the specialist model losses. Since we save model checkpoints every 0.5 epochs (as described in Appendix H), *all* model checkpoints were optimized beyond the specialist model.

Algorithm 5 AutoMixAlign Resampling Uniform (AMA-S Uniform)

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, specialist parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$, batch size b, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
For all tasks, compute $\mathcal{L}(\boldsymbol{\theta}_i, z)$ for each $z \in \mathcal{D}_i$, and add them to a new column in $\mathcal{D}_i$
Initialize model parameters $\boldsymbol{\theta}$
for $t = 1, \ldots, T$ **do**
    $\mathcal{T} \sim \text{Uniform}(1, \ldots, k, b)$
    $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
    Update $\boldsymbol{\theta}$ using one step of gradient descent on the loss

$$\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$$

end for

Algorithm 6 DoReMi

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, specialist parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$, batch size b, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
For all tasks, compute $\mathcal{L}(\boldsymbol{\theta}_i, z)$ for each $z \in \mathcal{D}_i$, and add them to a new column in $\mathcal{D}_i$
Initialize model parameters $\boldsymbol{\theta}$
Initialize $\boldsymbol{\alpha}$ to be the average task weighting achieved during a training run of AMA (Algorithm 1).
for $t = 1, \ldots, T$ **do**
    $\mathcal{T} \sim \text{Multinomial}(\alpha_1, \ldots, \alpha_k, b)$
    $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
    Update $\boldsymbol{\theta}$ using one step of gradient descent on the loss

$$\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\}$$

end for

Algorithm 7 AMA-S Unclipped

Inputs: Task datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, specialist parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$, batch size b, smoothing parameter $c \in (0, 1)$, training steps T
Outputs: Trained model parameters $\boldsymbol{\theta}$
For all tasks, compute $\mathcal{L}(\boldsymbol{\theta}_i, z)$ for each $z \in \mathcal{D}_i$, and add them to a new column in $\mathcal{D}_i$
Initialize model parameters $\boldsymbol{\theta}$
$\boldsymbol{q}^{(1)} \leftarrow (\frac{1}{k}, \ldots, \frac{1}{k})$
for $t = 1, \ldots, T$ **do**
    $\alpha_i^{(t)} \leftarrow (1 - c) q_i^{(t)} + c\frac{1}{k}, \forall i$
    $\mathcal{T} \sim \text{Multinomial}(\alpha_1^{(t)}, \ldots, \alpha_k^{(t)}, b)$
    $\mathcal{B} \leftarrow \{z \sim \mathcal{D}_j : \forall j \in \mathcal{T}\}$
    $\mathcal{B}_i \leftarrow \mathcal{B} \cap \mathcal{D}_i$ for each task i
    Update $\boldsymbol{\alpha}$ using one step of EXP3 (Auer, 2002):

$$q_i^{(t+1)} \leftarrow q_i^{(t)} \cdot \exp\left\{ \frac{1}{q_i^{(t)}} \frac{1}{|\mathcal{B}_i|} \sum_{z \in \mathcal{B}_i} \max\{\mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z), 0\} \right\}$$

$$q_i^{(t+1)} \leftarrow \frac{q_i^{(t+1)}}{\sum_{j=1}^{k} q_j^{(t+1)}}$$

    Update $\boldsymbol{\theta}$ using one step of gradient descent on the loss

$$\frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \mathcal{L}(\boldsymbol{\theta}, z) - \mathcal{L}(\boldsymbol{\theta}_i, z)$$

end for

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	36.08
Chatbot Arena 2024 Specialist	43.44	16.05	44.60	50.61	38.68
CodeUltraFeedback Specialist	35.86	9.93	52.16	57.32	38.82
$0.25\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + 0.75\boldsymbol{\theta}_{\text{CodeUltraFeedback}}$	37.15	8.48	50.36	54.88	37.72
$0.50\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + 0.50\boldsymbol{\theta}_{\text{CodeUltraFeedback}}$	42.14	11.51	48.56	**55.49**	39.43
$0.75\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + 0.25\boldsymbol{\theta}_{\text{CodeUltraFeedback}}$	44.73	17.56	48.20	51.22	40.43
AMA-R	42.51	**18.15**	**51.44**	54.88	**41.75**
AMA-S	**43.44**	15.61	51.08	50.61	40.19

Table 6: Weighted Model Averaging results for Setup 2, Chatbot Arena 2024 (Helpfulness) + CodeUltraFeedback (Coding)

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	51.60	31.85
UltraFeedback Specialist	42.51	17.59	36.10	32.07
SafeRLHF Specialist	29.02	6.66	100.00	45.23
$0.25\boldsymbol{\theta}_{\text{UltraFeedback}} + 0.75\boldsymbol{\theta}_{\text{SafeRLHF}}$	36.78	8.48	**99.99**	48.42
$0.50\boldsymbol{\theta}_{\text{UltraFeedback}} + 0.50\boldsymbol{\theta}_{\text{SafeRLHF}}$	39.37	12.21	99.96	50.51
$0.75\boldsymbol{\theta}_{\text{UltraFeedback}} + 0.25\boldsymbol{\theta}_{\text{SafeRLHF}}$	43.99	15.69	78.17	45.95
AMA-R	**44.55**	16.21	99.74	53.50
AMA-S	**44.55**	**16.88**	**99.99**	**53.81**

Table 7: Weighted Model Averaging results for Setup 2, UltraFeedback (Helpfulness) + SafeRLHF (Harmlessness)

Model	IFEval (%) ↑	Alpaca Eval (%) ↑	MBPP (%) ↑	HumanEval (%) ↑	Toxigen (%) ↑	Average (%) ↑
Zephyr-7b-sft-full	35.30	8.64	49.90	50.46	51.60	39.18
Chatbot Arena 2024 Specialist	43.44	16.05	44.60	50.61	69.77	44.89
CodeUltraFeedback Specialist	35.86	9.93	52.16	57.32	76.31	46.31
SafeRLHF Specialist	29.02	6.66	48.56	50.61	100.00	46.97
$\frac{1}{6}\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + \frac{5}{12}(\boldsymbol{\theta}_{\text{CodeUltraFeedback}} + \boldsymbol{\theta}_{\text{SafeRLHF}})$	35.86	7.58	50.00	54.27	96.50	48.84
$\frac{1}{3}\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + \frac{1}{3}(\boldsymbol{\theta}_{\text{CodeUltraFeedback}} + \boldsymbol{\theta}_{\text{SafeRLHF}})$	42.70	12.28	52.16	53.66	96.70	51.50
$\frac{5}{12}\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + \frac{1}{6}(\boldsymbol{\theta}_{\text{CodeUltraFeedback}} + \boldsymbol{\theta}_{\text{SafeRLHF}})$	43.81	17.56	48.20	51.22	70.51	46.26
AMA-R	**48.43**	**17.77**	54.32	**55.49**	95.91	**54.38**
AMA-S	43.62	13.04	**55.76**	53.66	**99.80**	53.18

Table 8: Weighted Model Averaging results for Setup 3, Chatbot Arena 2024 (Helpfulness) + CodeUltraFeedback (Coding) + SafeRLHF (Harmlessness)

C.2 Model Merging Ablations

The model merging baseline included in our main experiments uniformly averages the parameters of specialist models. To ablate the effects of parameter weighting with model merging, we investigate non-uniform merging methods. For the setups 1 and 2, we merge specialist model parameters as

$$\boldsymbol{\theta}_{\text{merged}} = \lambda\boldsymbol{\theta}_{\text{specialist 1}} + (1-\lambda)\boldsymbol{\theta}_{\text{specialist 2}}$$

for $\lambda \in \{0.25, 0.5, 0.75\}$. For setup 3, we use

$$\boldsymbol{\theta}_{\text{merged}} = \lambda\boldsymbol{\theta}_{\text{base}} + \frac{(1-\lambda)}{2}(\boldsymbol{\theta}_{\text{specialist 1}} + \boldsymbol{\theta}_{\text{specialist 2}})$$

for $\lambda \in \{1/6, 1/3, 5/12\}$ with Chatbot Arena specialist model as the base model $\boldsymbol{\theta}_{\text{base}}$.

We present results in Tables 6, 7, and 8. In setups 2 and 3, both AMA algorithms score higher on average than all model merging methods. In setup 1, AMA-R has the highest average score, though $0.75\boldsymbol{\theta}_{\text{Chatbot Arena 2024}} + 0.25\boldsymbol{\theta}_{\text{CodeUltraFeedback}}$ has a higher average score than AMA-S. However, AMA-S improves performance on MBPP whereas this merging method degrades performance on MBPP.

D Comparing AMA with Existing DRO Algorithms

In this section, we discuss differences between AMA and several closely-related DRO-style algorithms: GroupDRO (Sagawa et al., 2019), ODM (Albalak et al., 2023), Sheared LLama (Xia et al., 2023), ExcessMTL (He et al., 2024), and DoReMi (Xie et al., 2024). GroupDRO minimizes the worst-case loss across multiple tasks (*i.e.*, groups) by reweighting the loss function to put more weight on tasks with larger loss. DoReMi and ExcessMTL are DRO-based pretraining algorithms that reweight tasks by prioritizing those with the largest clipped excess loss (while optimizing the usual loss). ODM and Sheared Llama are online data mixing algorithms that increases the probability of sampling data from tasks with large loss or excess loss. AMA is similar to DoReMi and GroupDRO because all three introduce a minimax optimization problem used to optimize the worst-case loss or excess loss. Sheared LLama and ODM are similar to AMA in that they are adapt the probability of sampling data from each task during training. However, AMA differs in several respects:

1. AMA focuses on preference alignment, whereas all of these prior works focus on pretraining. GroupDRO additionally considers NLP and vision tasks.

2. AMA optimizes the clipped excess loss for both task weight and model updates. DoReMi and Sheared LLama optimize the usual loss for model updates. GroupDRO and ODM optimize the usual loss for both task weight and model updates.

3. AMA *adaptively* updates task weights *during* model optimization, whereas DoReMi learns a *fixed* task weighting prior to model optimization.

4. AMA uses single-task specialist models to compute excess losses. DoReMi uses a single multitask model (called the *proxy model*) to compute excess losses. ExcessMTL uses a diagonal approximation of the generalist's Hessian. Sheared LLaMA approximates these losses with single aggregate reference loss computed as the average loss computed over a validation dataset: $\mathcal{L}(\mathcal{D}_i^{\text{val}}, \boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_i^{\text{val}}|} \sum_{x \in \mathcal{D}_i^{\text{val}}} \mathcal{L}(x, \boldsymbol{\theta})$. Thus, Sheared LLaMA will underestimate the excess loss of easy samples and overestimate the excess loss of hard samples. AMA computes references loss for each sample individually and thus more accurately estimates excess losses.

5. We introduce two AMA algorithms—one that uses reweighting (AMA-R) and another that uses resampling (AMA-S)—whereas GroupDRO and DoReMi only introduce a reweighting algorithm. We stress that while Sagawa et al. (2019) provide a convergence result for GroupDRO, their proof only applies to the reweighting-based AMA-R algorithm and does not extend to our resampling-based AMA-S algorithm. A primary contribution of this work is that we provide a new theoretical argument to prove convergence for AMA-S.

E Excess Loss

In this section, we provide background motivation for the using the excess loss in AMA. The following discussion is largely summarized from He et al. (2024).

Consider a supervised learning setting where we want to train a model $\boldsymbol{\theta} \in \Theta$ to predict the label $y \in \mathcal{Y}$ given input data $x \in \mathcal{X}$ where samples (x, y) are drawn from some distribution $\mathcal{P}$. Given loss function $\mathcal{L}$, the risk of $\boldsymbol{\theta}$ is

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(\boldsymbol{\theta}, x, y)]. \tag{9}$$

The risk can be decomposed as

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\underbrace{\mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}^*_\Theta)}_{\text{Estimation error}} + \underbrace{\mathcal{L}(\boldsymbol{\theta}^*_\Theta) - \mathcal{L}(\boldsymbol{\theta}^*)}_{\text{Approximation error}}}_{\text{Excess risk}} + \underbrace{\mathcal{L}(\boldsymbol{\theta}^*)}_{\text{Bayes error}}, \tag{10}$$

where $\boldsymbol{\theta}^*_\Theta = \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{(x,y)\sim\mathcal{P}}[\mathcal{L}(\boldsymbol{\theta}, x, y)]$ is the optimal model in model family Θ and $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(x,y)\sim\mathcal{P}}[\mathcal{L}(\boldsymbol{\theta}, x, y)]$ is the optimal model in any model family. Approximation error arises from

Hyperparameter	Value
Learning rate	$5 \cdot 10^{-7}$
Batch size	256
Per-device batch size	8
Gradient accumulation steps	4
Optimizer	Adam with $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$
Learning rate scheduler	cosine
Learning rate scheduler warmup ratio	0.1
Number of training epochs	3
AMA task weight learning rate η	1
AMA smoothing parameter c	0.1

Table 9: Hyperparameters fixed across our main experiments.

our choice of model family, but when working with expressive function approximators like LLMs, the approximation error is generally very small. The Bayes error arises from label noise and is irreducible. The *excess risk* $\mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}^*)$ denotes the gap between our model's loss and the loss of the Bayes optimal model (*i.e.* the minimum loss achievable). Since excess loss removes the loss contribution of label noise and approximation error is effectively 0, it can be interpreted as our model's "room for improvement."

The *excess loss* $\mathcal{L}(\boldsymbol{\theta}, x, y) - \mathcal{L}(\boldsymbol{\theta}^*, x, y)$ quantifies the room for improvement on a particular sample. Samples with high excess loss are learnable ($\mathcal{L}(\boldsymbol{\theta}^*, x, y)$ is small) but our model has not yet learned them. Samples with low excess loss are high entropy samples that are less learnable ($\mathcal{L}(\boldsymbol{\theta}^*, x, y)$ is large) or samples that our model has already learned ($\mathcal{L}(\boldsymbol{\theta}, x, y) \approx \mathcal{L}(\boldsymbol{\theta}^*, x, y)$).

The excess loss is particularly useful in minimax optimization problems where we want to minimize the worst-case loss over different tasks, as is the case with AMA. Let $\mathcal{L}_i(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_i|} \sum_{(x,y)\in\mathcal{D}_i} \mathcal{L}(\boldsymbol{\theta}, x, y)$ denote the loss for task i, and consider the following optimization problem:

$$\min_{\boldsymbol{\theta}} \max_{i\in[k]} \mathcal{L}_i(\boldsymbol{\theta}) \tag{11}$$

Suppose the task i has zero excess loss but has a larger loss than task j. Such a scenario can arise if task i is difficult to learn. In this case, optimization problem 11 will focus on optimizing task i even though it cannot minimize its loss any further and make no progress on task j. Minimizing worst-case excess loss prevents optimization from focusing on difficult tasks at the expense of easier tasks.

F Future Directions

Partitioning data into tasks. We relied on intuitive notions of what a "task" represents, *e.g.* datasets designed to improve helpfulness fall under the helpfulness task. However, this intuitive approach is not necessarily optimal, and future work should study how to partition data into tasks to maximize performance in DRO-based algorithms. For instance, one could automatically partition data into tasks based on classification from another LLM.

Applying the methodology to other layers of the post-training stack. We mainly focused on the DPO layer of the post-training learning layer. In principle, our AMA methodology can be applied to others, including reward modeling, supervised-fine-tuning and on-policy learning algorithms such as PPO. We are excited to see what benefit this methodology can bring to these other layers.

G Training Details

We provide DPO and AMA hyperparameters for our main experiments in Table 9 and links to datasets in Table 10. We built our codebase on top of the alignment-handbook repository[6] and run all training jobs on 8 NVIDIA A100 80GB GPUs. AMA trains k specialist models and one generalist model and

[6] https://github.com/huggingface/alignment-handbook

Dataset Name	Link
UltraFeedback	https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized
Chatbot Arena 2024	https://huggingface.co/datasets/lmsys/chatbot_arena_conversations
CodeUltraFeedback	https://huggingface.co/datasets/coseal/CodeUltraFeedback
PKU-SafeRLHF	https://huggingface.co/datasets/PKU-Alignment/PKU-SafeRLHF

Table 10: Datasets used in our experiments.

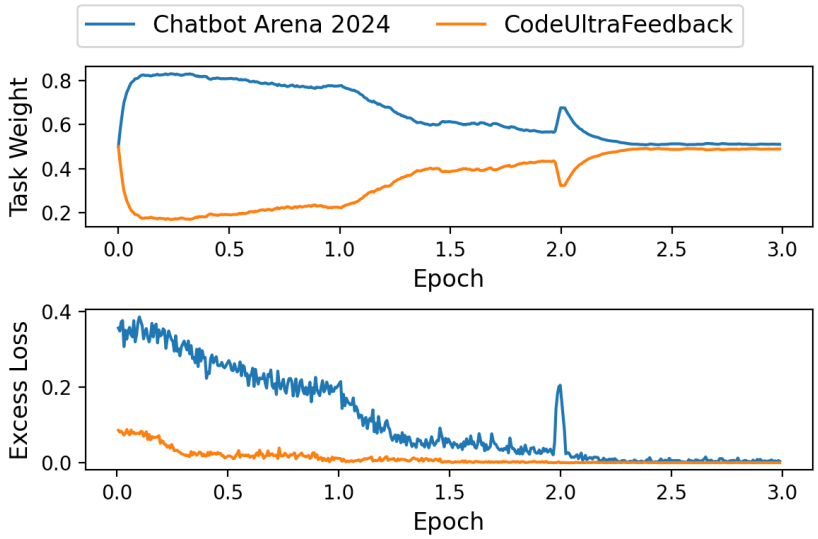

Figure 5: Excess losses and task weights for AMA-R in Setup 1, Helpfulness (Chatbot Arena 2024) + Coding (CodeUltraFeedback).

thus requires $2\sum_{i=1}^{k}|D_i|$ model updates in total, twice as many required to train a multitask model using standard optimization approaches.

H Evaluation

In this appendix, we describe our model checkpoint selection method and how we evaluate the selected checkpoint.

H.1 Model Selection

We save model checkpoints every 0.5 epochs. For each checkpoint, compute a confidence interval for the evaluation accuracy across all tasks (+/- 1 standard error):

$$CI = [\hat{p} - \sigma, \hat{p} + \sigma], \quad \sigma = \sqrt{\sum_{i=1}^{k} \sigma_i^2}, \qquad \sigma_i = \sqrt{\frac{\hat{p}_i(1-\hat{p}_i)}{n_i}} \tag{12}$$

Here, $\hat{p}$ is the evaluation accuracy, σ_i is the standard error for task i, n_i is the number of samples in the eval dataset for task i, and σ is the standard error for all tasks combined. We then select the earliest checkpoint whose CI overlaps with the CI of the checkpoint with the maximum evaluation accuracy.

Researchers often look at the evaluation loss for model selection (*e.g.*, selecting the checkpoint saved just before the model begins overfitting), but this metric can be misleading in multi-task settings. The losses for different tasks are often on different scales (*i.e.*, the excess loss of task 1 may always be larger than that of task 2), and it's unclear how we should adjust for these differences in loss scale. Since task accuracy, the fraction of samples in the evaluation dataset for which $\pi(y_w|x) > \pi(y_l|x)$, is always on a scale from 0 to 1, we instead look at task accuracies to select model checkpoints.

H.2 Benchmarks

We evaluate a single model for each algorithm and experimental setup and base our evaluation on a subset of the benchmarks used by (Ivison et al., 2024) described below. To run AlpacaEval, we use the original `alpaca_eval` codebase (Li et al., 2023; Dubois et al., 2024).[7] For the remaining benchmarks, we use the `open-instruct` codebase.[8]. Since we use the Zephyr 7b SFT Full model (Tunstall et al., 2023) as the base policy in all experiments, we use the Zephyr chat template in all evaluations.

- **IFEval (Helpfulness)** (Zhou et al., 2023): We use the default setup and report the "Strict Accuracy" metric at the prompt level, *i.e.* the percentage of prompts for which a model's output satisfies all verifiable instructions.
- **Alpaca Eval 2.0 (Helpfulness)** (Dubois et al., 2024): We use the AlpacaEval 2.0 configuration provided for zephyr-7b-alpha-ExPO, reporting the length-controlled winrate. We use a temperature of 0.7, nucleus sampling with $p = 0.9$ (top-p), and a top-k value of 50. To control repetition, we applied a presence penalty of 0.1 and a frequency penalty of 0.1. report the length-controlled win rate. We allow the evaluated model to generate up to 2048 tokens.
- **MBPP (Coding)** (Austin et al., 2021): We report pass@10 accuracy and use a sampling temperature of 0.8.
- **HumanEval (Coding)** (Chen et al., 2021)): In addition to the original HumanEval prompts, we use the prompts provided by HumanEvalPack (Muennighoff et al., 2023). Just as with MBPP, we report pass@10 accuracy and use a sampling temperature of 0.8.
- **Toxigen (Harmlessness)** (Hartvigsen et al., 2022): We use the default setup and report the percentage of *non-toxic* model responses. In particular, Toxigen outputs the fraction of toxic responses f, and we report $1 - f$ expressed as a percentage.

I Licenses

Below, we give all the artifacts that we used and their respective licenses:

- Alignment Handbook: Apache-2.0 License
- open-instruct: Apaches-2.0 License
- Zephyr 7B SFT Full: Apache-2.0 License
- IFEval: Eclipse Public License - v 2.0 License
- AlpacaEval: Apache 2.0 License
- MBPP: MIT License
- HumanEval: MIT License
- Toxigen: MIT License
- GPT-4 outputs: AlpacaEval uses GPT-4 Turbo outputs. Since we use output for research, we are in compliance with the GPT-4 terms of service.

[7] https://github.com/tatsu-lab/alpaca_eval
[8] https://github.com/allenai/open-instruct