Online Merging Optimizers for Boosting Rewards and Mitigating Tax in Alignment

Keming Lu, Bowen Yu, Fei Huang, Yang Fan, Runji Lin, Chang Zhou
Qwen Team, Alibaba Inc.
{lukeming.lkm,yubowen.ybw,feihu.hf}@alibaba-inc.com
{suyang.fy,linrunji.lrj,ericzhou.zc}@alibaba-inc.com

Abstract

Effectively aligning Large Language Models (LLMs) with human-centric values while preventing the degradation of abilities acquired through Pre-training and Supervised Fine-tuning (SFT) poses a central challenge in Reinforcement Learning from Human Feedback (RLHF). In this paper, we first discover that interpolating RLHF and SFT model parameters can adjust the trade-off between human preference and basic capabilities, thereby reducing the alignment tax at the cost of alignment reward. Inspired by this, we propose integrating the RL policy and SFT models at each optimization step in RLHF to continuously regulate the training direction, introducing the Online Merging Optimizer. Specifically, we merge gradients with the parameter differences between SFT and pretrained models, effectively steering the gradient towards maximizing rewards in the direction of SFT optimization. We demonstrate that our optimizer works well with different LLM families, such as Qwen and LLaMA, across various model sizes ranging from 1.8B to 8B, various RLHF algorithms like DPO and KTO, and existing model merging methods. It significantly enhances alignment reward while mitigating alignment tax, achieving higher overall performance across 14 benchmarks.[1]

1 Introduction

Reinforcement Learning From Human Feedback (RLHF) has propelled the success of the most recent wave of generative AI models (Ouyang et al., 2022; Bai et al., 2022b). While RLHF garners the bonus of steering large language models (LLMs) to meet human expectations, past studies have highlighted that this approach can inadvertently lead to forgetting the diverse abilities that the LLMs have already mastered through pre-training and supervised finetuning (SFT) (Bai et al., 2022a; Zheng et al., 2023; Lin et al., 2024; Dong et al., 2024). This is evident in their declining performance on certain public language benchmarks and a loss of fundamental language skills. Additionally, responses tend to exhibit unexpected code-switching and get significantly longer (Park et al., 2024). These undesirable side effects are commonly referred to as the "alignment tax" (Dong et al., 2023; Sun et al., 2024).

Ideally, an optimal RLHF strategy should maintain the bonuses of alignment while avoiding the associated tax, striving to maximize rewards while minimizing forgetting. Relying on the linear mode connectivity of neural networks (Garipov et al., 2018; Frankle et al., 2020; Entezari et al., 2021), the trade-off in model capabilities can be succinctly described as the interpolation of model parameters (Garipov et al., 2018; Ilharco et al., 2022; Zheng et al., 2024). Studies have shown that combining different models, fine-tuned from the same pre-trained model through weight interpolation, often leads to a more balanced performance among the original models (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024). Motivated by this insight, we conducted initial investigations into

[1] Codes can be found at https://github.com/QwenLM/online_merging_optimizers

Preprint. Under review.

merging an RLHF model with the reference SFT model it trained from. Our observations indicate that this offline model merging effectively mitigates the alignment cost. As depicted in Tab. 4, the offline merged model restores performance comparable to the SFT model across language benchmarks and language proficiency. However, this improvement comes at the expense of a reduction in preference rewards compared to the RLHF model.

The modest performance gains from offline merging are unsurprising, given that single-time parameter interpolation only allows for trade-offs between models with fixed capabilities. During the RLHF training process, each optimization step improves the model's capabilities. Thus, we have the chance to ensure that the direction of these changes aligns with the reference SFT model. In this paper, we integrate model merging into each RLHF optimization step and introduce the Online Merging Optimizer. This innovative optimizer enhances rewards more effectively than traditional optimizers like AdamW (Loshchilov and Hutter, 2018) while simultaneously reducing the alignment tax, akin to offline merging. Specifically, we consider the alterations in model parameters before and after the SFT phase, termed delta parameters, as the final update direction for the SFT model. Throughout each RLHF step, we blend gradients with the SFT delta parameters, efficiently guiding the gradient to maximize rewards in the direction of SFT optimization. This results in well-calibrated gradients that balance reward maximization and alignment tax.

We conduct extensive experiments to demonstrate that our optimizer is effective across various dimensions: (1) for different backbones, including Qwen1.5 (Bai et al., 2023) and LLaMA3 [2]; (2) for models of different sizes, ranging from 1.5B to 8B parameters; (3) for various RLHF algorithms, such as DPO (Rafailov et al., 2024), IPO (Amini et al., 2024), and KTO (Ethayarajh et al., 2024); and (4) compatible with different offline merging algorithms, such as DARE (Yu et al., 2024) and TIES (Yadav et al., 2023). Notably, the OnDARE optimizer consistently achieves better performance across 12 benchmarks in 7 categories along with MT-Bench and AlpacaEval 2.0, surpassing all regularization and offline merging baselines, demonstrating its superior capability as an optimizer designed for RLHF.

2 Related Works

Reinforcement Learning with Human Feedback (RLHF). The objective of RLHF is to enhance the alignment of LLMs with human preferences and values, aiming to generate responses that are more helpful, accurate, and safer (Ouyang et al., 2022; Bai et al., 2022b; Ji et al., 2023). Essentially, RLHF techniques treat human preferences as rewards and employ online or offline RL algorithms (Zhao et al., 2023; Rafailov et al., 2024; Amini et al., 2024; Ethayarajh et al., 2024; Song et al., 2024; Yuan et al., 2023c), to achieve this alignment. Typically, RLHF is built upon models trained via SFT (Wang et al., 2023; Xu et al., 2024; Lu et al., 2024; Wu et al., 2023). One of the biggest challenges in RLHF is the bonus-tax trade-off (Ouyang et al., 2022; Lin et al., 2024), which shows RLHF models forget basic language modeling while pursuing high rewards. In this work, we mitigate this issue by using online merging optimizers, which have proven effective across various sizes and backbone LLM families.

Continual Learning (CL). The Alignment Tax can be viewed as a form of catastrophic forgetting in CL. Catastrophic forgetting refers to a dramatic degradation of performance on old tasks when learning new tasks(Goodfellow et al., 2013; Kirkpatrick et al., 2017; Wang et al., 2024). For example, constraining the shift in output space between reference and policy models (Ouyang et al., 2022; Rafailov et al., 2023)) is a common strategy employed in continual learning (Li and Hoiem, 2017; Rebuffi et al., 2017; Wang et al., 2024). However, traditional methods mitigating forgetting issues in CL usually require data for previous training tasks for experience replay, which may be impractical for current open- or closed-source LLMs, whose alignment data is in-house and already multi-task. Although we mainly experiment with our online merging optimizers on LLM alignment, they can be applied to a wider range of CL scenarios in the future.

Robust Fine-tuning. Overcoming alignment tax can be seen from another perspective, similar to overcoming generalization degradation, involving the research area of robust fine-tuning. Nonetheless, robust fine-tuning methods often necessitate supplementary forward and backward computations, rendering them inefficient for large-scale models (Liang et al., 2021; Aghajanyan et al., 2021; Jiang

[2] https://ai.meta.com/blog/meta-llama-3/

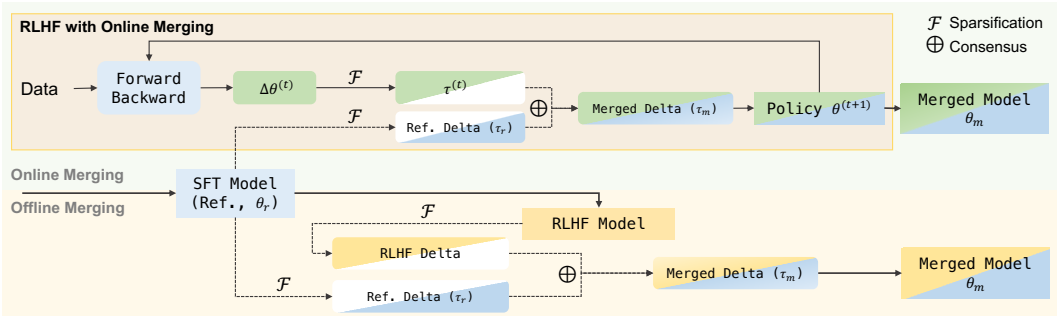

Figure 1: An illustrastion of RLHF with online merging optimizers described in §4. In each RLHF iteration, we first obtain the update weight $\Delta\theta^{(t)}$, and then sparcify it and make a consensus with the delta parameters of the reference model. We use this merged delta as the update of the policy model in this iteration. We also compare online merging with offline merging, shown in the lower part of the figure and further introduced in §3.

et al., 2020; Yuan et al., 2023a). Some works aim to enhance robustness by eliminating invaluable parameter updates during training to achieve robust generalization (Lee et al., 2020; Jiang et al., 2022; Xu et al., 2021). Our proposed optimizer can be viewed as an extension of these gradient dropout methods. However, we further derive that merging the gradients with the delta parameters of the SFT model can better balance alignment rewards and tax. Comprehensive experiments show our methods surpass established robust fine-tuning baselines in alignment scenarios.

Offline Model Merging. Model merging is widely studied as an effective training-free method to combine the capabilities of fine-tuned large language models (Wortsman et al., 2022; Lin et al., 2023; Matena and Raffel, 2022). Ilharco et al. (2023) proposed the concept of a "task vector" (also referred to as "delta parameters") and demonstrates that the arithmetic combination of them can adjust model behaviors. Yadav et al. (2023) further addressed conflicts between different task vectors through norm-based sparsification and consensus on the signs of weights. Tam et al. (2024) employed the conjugate gradient method to find the optimal merging combination. Yu et al. (2024) shown LLMs are highly robust to sparsify task vectors and proposes DARE for merging LLMs with random sparsification. Recently, Lin et al. (2024) demonstrates that simple offline model averaging can help mitigate alignment tax. Unlike these offline model merging methods that combine the parameters of different models in one shot, our work introduces an approach where, at each step of RLHF optimization, we online merge the gradients of the policy model with the delta parameters of the SFT model. Our results show that this optimizer is more effective than offline merging in boosting alignment rewards while eliminating alignment tax.

3 Preliminaries and Motivations

In this section, we begin by briefly defining alignment tax and introducing offline model merging. We then present preliminary results indicating that while offline merging can reduce alignment tax, it also diminishes rewards. These findings motivate our proposal for online merging optimizers.

Alignment Tax. Typically, aligning LLMs with human preferences involves two phases: SFT to establish an instruction-following model, followed by RLHF for enhanced human preference (Ouyang et al., 2022). Current mainstream RLHF methods, such as PPO (Ouyang et al., 2022) and DPO (Rafailov et al., 2023), guide the model to optimize the reward while incorporating a Kullback-Leibler (KL) divergence penalty between the output of learned RL policy and the reference SFT model. This penalty prevents the policy from deviating too far from its original goal of maintaining acquired language proficiency while pursuing preference rewards. However, despite these efforts, RLHF models still exhibit fluctuations in performance across various NLP tasks and language capabilities (Park et al., 2024). We experimented with different KL divergence weights β in the DPO setting, as shown in Fig. 4. An increase in β correlates with an elevation in the mean benchmark performance, albeit at the expense of diminished performance on MT-Bench and AlpacaEval. Conversely, reducing β leads the model to lose its fundamental abilities. Struggling to balance reward optimization and maintaining linguistic taxonomies has become the foremost challenge in RLHF training.

Model Merging. Instead of relying solely on the KL penalty to constrain the variation of the model's output space, we aim to find a trade-off path between the parameters of SFT and RLHF. This enables us to freely adjust whether the final model leans towards maximizing reward or minimizing tax. Our objective relies on the discovery of Mode Connectivity (Garipov et al., 2018), which suggests that the local optima of modern deep neural networks are connected by simple curves, like a polygonal chain with only one bend. These connections are often categorized under model merging methods. In this work, we specifically focus on general task arithmetic merging methods, which have proven effective in combining the abilities of fine-tuned models and have good characteristics such as linearity. Here, we present a straightforward merging case where parameters θ_s and θ_r, trained from the same pre-trained model θ_b, are merged. The new capabilities learned by θ_s and θ_r relative to θ_b can be represented by their delta parameters: $\tau_s = \theta_s - \theta_b$ and $\tau_r = \theta_r - \theta_b$. Merging methods typically begin by applying a sparsification operation to each delta parameter and then linearly combining them using consensus methods to derive the merged model $\theta_m = \theta_b + \tau_m = \theta_b + \mathcal{F}(\tau_s) \oplus \mathcal{F}(\tau_r)$, where $\mathcal{F}(\cdot)$ denotes a sparsification operator with corresponding hyper-parameters p, and $\oplus$ represents the consensus method used to handle parameter interference. For instance, Yadav et al. (2023) employed top-k sparsification and sign-based consensus, while Yu et al. (2024) utilized random sparsification and linear combination to achieve this merging process.

Offline Merging Mitigates Alignment Tax at Cost. To investigate whether Model Merging can restore the ability of the aligned model to recover the SFT model's capability, we conducted preliminary experiments. As shown in Tab. 4, we observe that the current mainstream model merging methods, when merging the SFT model and RLHF model, indeed improve performance across various linguistic abilities compared to the RLHF model, and outperform the SFT model on average. However, this improvement comes at the cost of decreased human preference, evidenced by a significant drop in MT-Bench and AlpcaEval 2.0 scores. In other words, while merging the SFT and RLHF models once can alleviate alignment tax, it also substantially diminishes the alignment bonus.

From Offline Merging to Online. It is not surprising that simply merging the SFT and RLHF models in an offline, one-time manner did not yield satisfactory results, as our reliance on the Mode Connectivity assumption has indicated that interpolating parameters results in interpolated model performance. Thus, finding a single point along this continuum that outperforms both ends of the spectrum simultaneously is indeed challenging. However, through iterative model merging at each step of RLHF training, integrating the strengths of SFT with each update to maximize reward, we can progressively steer our training trajectory. This online merging could lead us to discover local minima that effectively balance foundational capabilities and human preferences.

4 Online Merging Optimizers

Motivated by the offline merging, we explore incorporating model merging into the RLHF optimization steps in this section. We begin by examining the commonly used gradient-based optimizers. At each optimization step t, LLM parameters are adjusted according to $\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}$, where $\Delta\theta^{(t)}$ is a gradient-based weight modification. For instance, in the Adam optimizer (Kingma and Ba, 2017), $\Delta\theta$ is computed using the normalized gradient by exponential moving average, as shown in Alg. 1. From the perspective of model merging, in RLHF training, $\Delta\theta^{(t)}$ can be seen as a representation of improved human preference—a set of parameters ready to be merged into the policy model at step t. By post-processing $\Delta\theta^{(t)}$ to encapsulate not only human preference but also the basic capabilities of the SFT model, we can achieve an online merging of the RLHF policy model and the SFT model. To implement this concept, we propose an online merging optimizer.

4.1 From Gradient-based Optimizer to Online Merging Optimizer

As introduced in §3, offline task arithmetic techniques merge LLMs by aggregating their delta parameters as $\theta_m = \theta_b + \tau_m = \theta_b + \mathcal{F}(\tau_s) \oplus \mathcal{F}(\tau_r)$. In alignment, we aim to merge the reference SFT model θ_r and the policy model $\theta^{(t+1)}$ at the t-th training step. Here, we define the merged parameter at step t as:

$$\theta^{(t+1)} = \theta_b + \mathcal{F}(\theta^{(t)} - \theta_b + \Delta\theta^{(t)}) \oplus \mathcal{F}(\theta_r - \theta_b) = \theta_b + \mathcal{F}(\tau^{(t)} + \Delta\theta^{(t)}) \oplus \mathcal{F}(\tau_r), \quad (1)$$

where θ_b is the the parameters of pre-trained model that the reference and policy models are finetuned on, and $\tau_r, \tau^{(t)}$ are delta parameters $\tau_r = \theta_r - \theta_b$, $\tau^{(t)} = \theta^{(t)} - \theta_b$.

However, we empirically find directly optimizing Eq. 1 is unstable and hard to converge, and Eq. 1 requires an additional cache for pre-trained model parameters. Therefore, we apply a relaxation on Eq. 1 by moving the $\tau^{(t)}$ out of the merging and only focus on $\Delta\theta^{(t)}$, meaning we relax the merging for the past but only keep it toward the future update:

$$\theta^{(t+1)} \sim \theta_b + \tau^{(t)} + \mathcal{F}(\Delta\theta^{(t)}) \oplus \mathcal{F}(\tau_r) = \theta^{(t)} + \mathcal{F}(\Delta\theta^{(t)}) \oplus \mathcal{F}(\tau_r). \tag{2}$$

The delta parameter $\tau^{(t)}$ is already sparsified and consolidated with the reference in the previous update, so this relaxation still aligns well with our motivation for applying offline merging in optimization steps. Another essential benefit of this relaxation is avoiding caching additional parameters of θ_b, enhancing memory efficiency. With this relaxation, we show that the online merging at each optimization step can be approximated by consolidation between gradient-based delta weights and the delta parameters of the reference model.

4.2 Implementations

Note that our optimizer framework is highly flexible and compatible with exiting model merging methods. In this work, we develop two online merging optimizers, OnDARE and OnTIES, based on widely-used model merging methods, DARE (Yu et al., 2024) and TIES (Yadav et al., 2023).

OnDARE Optimizer. DARE employs a random sparsification method and a linear combination as the consensus method. So, the OnDARE optimizer following Eq. 2 is displayed in Eq. 3:

$$\theta_m^{(t)} = \theta_m^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) + \alpha\mathcal{F}_R(\tau_r), \tag{3}$$

where α is a hyper-parameter controlling the weight of delta parameters of the reference model. Larger α introduces stronger regularization on the training. $F_R(\cdot)$ shown in Eq. 4 is a random sparsification operator based on the Bernoulli distribution with a fixed reserving probability p:

$$\mathcal{F}_R(x)_i = \begin{cases} x_i, & \text{Bernoulli}(p) = 1 \\ 0, & \text{Bernoulli}(p) = 0 \end{cases} \tag{4}$$

OnTIES Optimizer. TIES uses a top-k percentage sparsification and a sign-based consensus method. Specifically, it reserves the top-p percentage of parameters regarding absolute values from each merging candidate. It calculates the element-wise majority signs based on the sign and norms, drops parameters with different signs with the majority, and weighted sum up the rest. Similarly, the formulation of the OnTIES optimizer is shown in Eq. 5:

$$\theta_m^{(t)} = \theta_m^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) \oplus_{sign} \alpha\mathcal{F}_R(\tau_r), \tag{5}$$

where $\mathcal{F}_R(\cdot)$ is a top-p percentage sparsification detailed in Eq. 6. The consensus method $\oplus_{sign}$ used in the OnTIES optimizer is a sign-based consensus among sparsified delta parameters as shown in Eq. 7, where $\mathbb{I}$ is the indicator function:

$$\mathcal{F}_R(x)_i = \begin{cases} x_i, & |x_i| \in \text{top-p}(|x|) \\ 0, & o.w. \end{cases} \tag{6}$$

$$a \oplus_{sign} b = \begin{cases} a + b, & \text{sign}(a) = \text{sign}(b) \\ a\mathbb{I}(|a| \geq |b|) + b\mathbb{I}(|b| \geq |a|), & \text{sign}(a) \neq \text{sign}(b) \end{cases} \tag{7}$$

The pseudo-code examples for these variants of online merging optimizers are shown in Alg. 1. It is worth noticing that we neglect the rescaling of sparsified delta parameters, which are proven to be essential in the offline merging (Yu et al., 2024; Yadav et al., 2023), as we find rescaling harms the numeric stabilities in multi-step optimization.

5 Experiments

5.1 Experimental Setup

Dataset. We conduct our experiment on the widely-used preference dataset ULTRAFEEDBACK. To be more specific, we use the binarized version of ULTRAFEEDBACK from Tunstall et al. (2023)[3], which is a reannotated and clean dataset for preference learning, compared with the original release (Cui et al., 2023). The training and evaluation splits of ULTRAFEEDBACK contain about 61K and 2K preference pairs, respectively, ranked by GPT-4, cleaned with manual efforts, and decontaminated with popular benchmarks such as TruthfulQA. The prompts in ULTRAFEEDBACK are large-scale, fine-grained, and diverse from multiple sources.

Training. We mainly explore our online merging optimizers in direct preference optimization (DPO, Rafailov et al. (2023)) on the ULTRAFEEDBACK dataset, as it is widely applied in nowadays large-scale LLM alignment due to its lower cost of training compared with Proximal Policy Optimization (PPO, (Schulman et al., 2017)). The general DPO includes sampling and annotating the responses from the policy model. In this work, we use an off-policy setting of DPO that directly trained our policy models on ULTRAFEEDBACK dataset, which has also proved effective in enhancing the helpfulness and harmlessness. The training loss of DPO is shown in Eq. 8, where β is the hyper-parameter controlling the strength of KL penalty between policy and reference models, π_θ and π_{ref} are policy and reference models, y_w and y_l are chosen and rejected samples in the preference pairs, respectively:

$$\mathcal{L}(\pi_\theta, \pi_{\text{ref}}) = -\mathbb{E}_{x,y_w,y_l\sim D}[\log \sigma(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)})]. \tag{8}$$

Note that online merging optimizers are agnostic with the training loss and optimizers, so they can be easily applied to PPO, other variants of DPO (Azar et al., 2023; Hong et al., 2024; Ethayarajh et al., 2024), and other online preference learning methods (Wu et al., 2024; Guo et al., 2024).

Evaluation. Assessing aligned large language models is a challenging task. We follow the straightforward principle that comprehensive evaluation yields trustworthy assessment. Therefore, our evaluation includes 12 public benchmarks in 7 categories evaluating the comprehensive abilities of aligned LLMs: **Mathematics**: (1) GSM8K (Cobbe et al., 2021) (2) Math401 (Yuan et al., 2023b) (3) Math23K (Wang et al., 2017); **Coding**: (1) HumanEval (Chen et al., 2021) (2) MBPP (Austin et al., 2021) (3) DS1000 (Lai et al., 2022) (4) CodeApex (Fu et al., 2024); **Instruction-following (IF)**: (1) IFEval (Zhou et al., 2023); **Reading Comprehension (RC)**: (1) COQA (Reddy et al., 2019) (2) DROP (Dua et al., 2019); **Knowledge** (1) MMLU (Hendrycks et al., 2021); **Agent** (1) NousResearch; **Code-switching**; We report the details of benchmarks in Appx. §B and inference configurations in Appx. §C. We use average scores within a category as the final category score and an average score among all benchmarks in all categories as the overall assessment. We also involve MT-Bench [4] and AlpacaEval 2.0 (Dubois et al., 2023, 2024) with length-controlled scoring, two leading and popular benchmarks that assess LLMs' alignment with human preferences using GPT-4-based evaluators.

Baselines. A naive baseline of our methods is the vanilla AdamW. We further consider offline merging methods, such as linear merging, DARE Yu et al. (2024), and TIES (Yadav et al., 2023) as our strong baselines as Lin et al. (2024) shows simple merging can mitigate alignment tax. As alignment tax is related to forgetting, we also involve traditional regularization methods, such as the KL penalty (Ouyang et al., 2022), EMA (Hunter, 1986; Noukhovitch et al., 2023), and ChildTuning (Xu et al., 2021) as our baselines. Specifically, the DPO algorithm adjusts the KL penalty with the hyper-parameter β. We also include LoRA (Hu et al., 2021) as one of our baselines, as parameter-efficient methods apply regularization regarding the weight space in the training (Biderman et al., 2024). Implementations of baselines are detailed in Appx. §C.

Configurations. We experiment with three LLM sizes, Qwen1.5-1.8B, Qwen1.5-7B, and LLaMa-3-8B series. Specifically, we use Qwen-1.8B-Base, Qwen-7B-Base, and LLaMa-3-8B as the base models for online optimizers. And we conduct direct preference optimization on ULTRAFEEDBACK on Qwen1.5-1.8B-SFT, Qwen1.5-7B-SFT, and LLaMa-3-8B-it as reference models. Two Qwen1.5 supervised finetuned models were trained on multi-lingual instruction data but no overlap with

[3] https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized

[4] https://huggingface.co/spaces/lmsys/mt-bench

Table 1: Main results of baselines and online merging optimizers on ULTRAFEEDBACK trained from various backbone LLMs. In merging and dropping settings, we experiment with two variants, OnDARE and OnTIES. I.F., R.C., and Know. are short for instruction-following, reading comprehension, and knowledge. The Avg. columns are average scores except for MT-Bench and AlpacaEval 2.0. Benchmarks in each category are detailed in §5.1. We report the improvement against the vanilla AdamW (marked in gray). Higher scores are marked in darker green, while lower in darker red. The best and second-best results are marked in **bold** and underline, respectively.

	Method	Math	Code	I.F.	R.C.	Know.	Agent	CodeSwitch	Benchmark Avg.	MT-Bench	AlpacaEval 2.0 (LC)
Qwen1.5-1.8B-Chat	Reference	37.3	7.8	23.1	52.2	43.1	70.8	58.6	41.8	4.37	3.99
	AdamW	**37.8**	8.4	23.1	50.8	43.2	67.5	61.9	41.8	4.59	4.79
	Regularization										
	KL Penalty	-0.3	0.0	0.0	+0.5	+0.3	+1.2	+0.8	+0.4	0.00	+0.02
	EMA	-2.0	-0.5	-0.8	-1.4	+0.1	+3.0	+1.6	0.0	+0.09	-0.10
	ChildTuning	-0.9	+0.1	-0.2	+1.9	+0.7	0.0	+0.4	+0.3	+0.03	-0.04
	LoRA	-1.3	**+0.2**	+0.8	+1.6	+0.3	+0.3	-2.5	-0.1	-0.07	-0.23
	Offline Merging										
	Linear	-1.7	0.0	+0.8	-0.3	-0.2	+3.0	0.0	+0.2	-0.03	-0.96
	DARE	-2.4	-0.3	+0.2	-1.2	-0.1	**+5.6**	-0.8	+0.2	+0.09	-0.72
	TIES	-1.6	-0.2	+0.8	+0.2	-0.6	+1.5	-0.8	-0.1	-0.10	-0.41
	Online Merging (Ours)										
	OnTIES	-2.0	-0.6	+1.0	-0.5	+0.2	+1.5	**+2.0**	+0.3	+0.15	**+0.12**
	OnDARE	-1.3	-0.6	**+1.1**	**+4.9**	**+0.8**	-1.7	+0.4	**+0.5**	**+0.24**	+0.05
Qwen1.5-7B-Chat	Reference	65.5	23.0	34.3	69.2	59.1	74.6	75.0	57.3	6.97	10.71
	AdamW	69.3	22.1	38.0	71.9	59.4	76.3	71.7	58.4	7.18	11.66
	Regularization										
	KL Penalty	-1.9	+0.9	-2.2	-2.0	+0.1	+3.2	+2.9	+0.1	-0.23	+0.08
	EMA	-2.0	+0.5	-1.9	-3.0	-0.4	+1.5	+3.7	-0.2	-0.32	**+1.24**
	ChildTuning	-2.8	-0.47	+1.0	-2.1	-1.1	-0.6	+2.6	+2.5	-0.1	-0.11
	LoRA	-1.9	+0.4	-2.4	-2.1	-0.4	+2.4	**+5.3**	+0.2	-0.14	-0.33
	Offline Merging										
	Linear	-1.2	**+1.9**	-2.3	-2.1	-0.2	+1.5	+0.8	-0.2	-0.28	-1.65
	DARE	-1.8	+1.2	-2.1	-1.7	-0.5	+0.9	**+5.3**	+0.2	-0.13	-1.59
	TIES	-2.2	+1.7	-0.2	-1.4	-0.1	**+3.8**	+2.5	+0.6	-0.21	-1.35
	Online Merging (Ours)										
	OnTIES	-2.2	+0.1	+0.5	+0.3	**+1.0**	+2.4	+1.3	+0.5	-0.08	+0.91
	OnDARE	+0.1	+0.8	**+2.3**	+0.7	+0.3	-0.9	+4.1	**+1.1**	**+0.12**	+0.28
LLaMa-3-8B-Instruct	Reference	67.8	39.5	68.3	81.2	65.7	39.5	41.8	57.7	7.97	22.92
	AdamW	68.3	39.3	68.4	81.7	65.7	40.9	41.4	58.0	8.04	24.10
	Regularization										
	KL Penalty	**+1.9**	-0.8	-0.6	-0.4	**+0.1**	+1.2	+5.7	+1.0	+0.08	-1.02
	EMA	-2.9	-0.1	**+1.0**	-0.1	-0.1	+1.8	+2.5	+0.3	+0.18	+0.34
	ChildTuning	-1.5	-1.3	+0.2	-0.7	-0.7	-2.0	**+7.8**	+0.2	+0.14	+1.04
	LoRA	+0.1	-0.6	-0.2	-0.4	**+0.1**	**+4.1**	+1.6	+0.6	+0.07	-0.16
	Offline Merging										
	Linear	-1.1	-0.2	+0.3	-1.2	-0.5	-0.8	-4.5	-1.2	+0.11	+0.02
	DARE	-1.0	-0.3	-0.1	-0.2	-0.4	+1.5	+4.5	+0.5	+0.14	+0.18
	TIES	-1.2	**+0.3**	-0.3	-1.5	-0.3	+1.8	+1.2	0.0	+0.01	+0.21
	Online Merging (Ours)										
	OnTIES	-0.1	-0.7	0.0	-0.8	-0.1	+2.7	-0.4	0.0	+0.16	+0.84
	OnDARE	+0.1	-1.3	+0.8	**+0.5**	-0.2	+3.0	+6.6	**+1.3**	**+0.19**	**+1.57**

the ULTRAFEEDBACK dataset. More configuration about hyper-parameter searching, statistical significance, and experiment resources are detailed in Appx. §C

5.2 Main Results

We present our main results in Tab. 4, which showcases the performance of baseline methods and our proposed online merging optimizers on ULTRAFEEDBACK, trained from Qwen1.5-1.8B-Chat, Qwen1.5-7B-Chat, and LLaMa-3-8B-Chat. Overall, regularization and offline model merging methods do not significantly improve the average performance of RLHF models on the benchmarks compared with vanilla AdamW in most settings, and lead to a decrease in preference scores on MT-Bench and AlpacaEval 2.0. This indicates that simply relying on techniques such as dropout for gradients (ChildTuning), suppressing changes in model gradient updates (EMA), or making one-time adjustments to RLHF model parameters based on SFT models (Merging) are not effective solutions for the alignment bonus-tax trade-off. The regularization baselines particularly work well for LLama-3-8B-It, as all regularization methods achieves consistent improvements on the average benchmark scores, as well as MT-Bench and AlpacaEval scores. In contrast, our proposed Online Merging Optimizer, particularly the OnDARE variant, achieved the most significant improvements on all the test sets. OnDARE achieves the highest improvements on the benchmark average score and consistently enhances MT-Bench and AlpacaEval 2.0 on all three backbone LLMs, significantly surpasses other baselines, especially on the LLaMa-3-8B-Instruct experiments, where it achieves 1.3, 0.19, and 1.57 improvements on the benchmark, MT-Bench, and AlpacaEval, respectively. Though OnTIES and OnDARE show effectiveness in boosting rewards and mitigating tax, OnDARE is

slightly better than OnTIES in terms of the average benchmark scores in most cases, while OnTIES sometimes has higher LC win rates on AlpacaEval 2.0. Detailed scores of each benchmark are reported in Tab. 5, Tab. 6, Tab. 7.

5.3 Hyper-parameter Effects

This section analyzes how the two main hyper-parameters, parameter reserve rate p and merging weight α, affect the overall performance of online merging optimizers.

Parameter Reserve Rate is the reserve rate of parameters during online merging, which is p in Eq. 4 and Eq. 6. We explore reserve rates ranging from 1 to $1e^{-5}$ on Qwen1.5-1.8B-Chat to maximize the search space within our limited computational resources. As shown in Fig. 2, online merging optimizers remain robust even at low parameter reserve rates down to $5e^{-4}$. This indicates that discarding 99.95% of gradient-based parameter modifications at each RLHF step still results in stable training. OnTIES is more sensitive to extremely low parameter reserve rates compared to OnDARE. This sensitivity arises because OnDARE employs an unbiased random sparsification method, whereas the top-k sparsification used by OnTIES introduces significant bias during training.

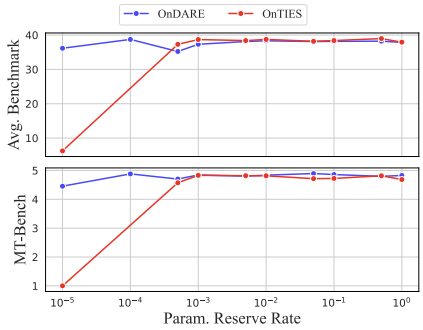

Figure 2: Analysis of Parameter Reserve Rates

Table 2: Analysis of the merging weight α with the OnDARE optimizer

α	Benchmarks(avg.)	MT-Bench
1e-4	7.4	1.00
5e-5	37.2	3.13
1e-5	40.1	4.37
5e-6	**41.8**	4.66
1e-6	41.6	4.76
5e-7	40.1	4.79
1e-7	39.2	**4.85**

Merging Weight is the aggregated weight for gradients and delta parameters of the reference (SFT) model. Larger merging weights introduce stronger regularization in the online merging optimizer. We experiment various merging weights α from $1e^{-4}$ to $1e^{-7}$, and report the results at Tab. 2. As the merging weight increases, the MT-Bench scores rise due to fewer regularizations being added during training, while the average benchmark scores initially increase but then decrease, peaking at $\alpha = 5e^{-7}$. Similar to the exponential coefficient in the EMA training (Hunter, 1986), large α leads to unstable training. So, we suggest beginning the hyper-parameter searching of α from a minor number such as $1e^{-7}$. It is worth noticing that a special case $\alpha = 0$ makes OnDARE equivalent to gradient dropout regularization methods, such as ChildTuning (Xu et al., 2021) in our baselines.

5.4 The Impact of RLHF Algorithms

In §5.1, we have validated the effectiveness of our Online Merging Optimizers in compatibility with DPO. In this section, we further study its application on other RLHF algorithms. In this section, we further explore their application to other RLHF algorithms. Specifically, we implement OnDARE and OnTIES with IPO (Azar et al., 2023) and KTO (Ethayarajh et al., 2024). We train Qwen1.5-7B-Chat under these settings on the ULTRAFEEDBACK datasets and present the results in Tab. 3. The Online Merging Optimizers, OnDARE, and OnTIES outperform AdamW across all algorithms on MT-Bench, except for the average benchmark score with the IPO algorithm. This demonstrates that their effectiveness extends to multiple RLHF algorithm variants.

Table 3: Analysis on different RLHF algorithms with online merging optimizers. We report the average scores of benchmark and MT-Bench scores as Tab. 4. B.M. is short for Benchmark. Best scores are marked in **bold**.

Method	Optimizer	B.M.(avg.)	MT-Bench
IPO	AdamW	**58.1**	7.06
	OnDARE	57.0	7.09
	OnTIES	57.3	**7.34**
KTO	AdamW	58.3	7.18
	OnDARE	58.6	**7.46**
	OnTIES	**59.3**	7.42

6 Discussion

6.1 Bridging Online and Offline Merging via Step-K Online Merging Optimizers

Offline Model Merging occurs after RLHF training concludes, combining the trained model with the SFT model. In contrast, Online Model Merging merges models at each training step. To bridge the gap, we propose step-K online merging optimizers, as detailed in Alg. 2. In step-K online merging, the optimizer caches accumulated update Δ_c, which is zero at the beginning of the training. Then, it applies the standard Adam update to the parameters for K optimization steps. At the K-th step, it first reverts the parameters to their cached state from the previous K-th step and calculates the delta parameters as the difference between current and cached parameters. These delta parameters are then merged with those of the reference model. The optimizer then updates the current parameters with the merged ones and refreshes the cached parameters. The online gap step K controls the **onlineness** for the step-K online merging optimizers. The pseudo-codes are detailed in Alg. 2. When K equals 1, the step-K online merging optimizer functions as the online merging optimizer introduced in §4; When K equals the total number of training steps, implying a single merge at the end of the training, the step-K online merging optimizer operates as the offline merging optimizer described in §3.

We then examine how the onlineness may affect the overall performance of step-K online merging optimizers. We run step-K optimizers on UltraFeedback trained from Qwen1.5-1.8B-Chat with online gap steps in $\{5, 10, 50, 100, 200\}$, whose results are shown in Fig. 3. As the online merging gap step increases, we witness a significant drop in MT-Bench scores for both OnDARE and OnTIES, with scores gradually approaching the performance of AdamW. The decline in average benchmark scores is less pronounced in MT-Bench, indicating that the "onlineness" of model merging is more critical for achieving higher rewards than for mitigating forgetting.

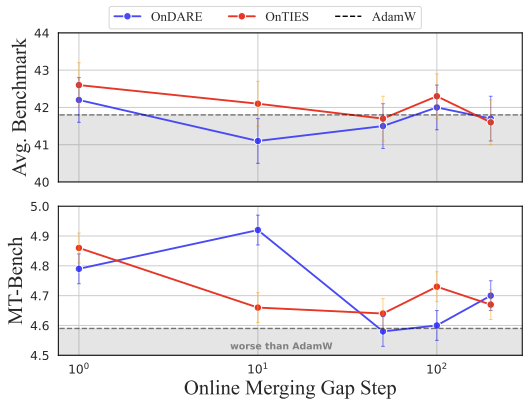

Figure 3: Analysis of Online Merging Gap Step

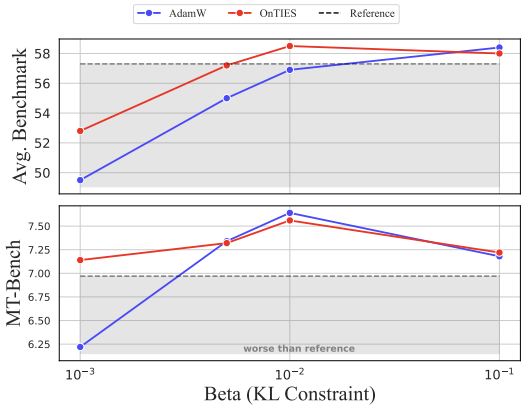

Figure 4: Analysis of KL constraints

6.2 Complementary Effect of KL Constraints and Online Merging

We further analyze the complementary effect of widely used KL constraint and online merging. We conduct experiments on Qwen1.5-7B-Chat with AdamW and OnTIES under extremely small β of the DPO algorithm, indicating very few KL constraints on the training. The results are shown in Fig. 4. At $\beta = 1e^{-1}$, AdamW and OnTIES show similar performance on both the MT-Bench and the average benchmark scores. However, as the β decreases under $1e^{-1}$, OnTIES is significantly better than AdamW on the average benchmark score, showing regularization from the online merging optimizer mitigates forgetting under lower KL constraints. Furthermore, OnTIES still achieves MT-Bench over 6.9 (MT-Bench of the reference SFT model) under an extremely small β ($1e^{-3}$), while AdamW falls to 6.25, demonstrating the complementary effect of KL constraints and online merging.

7 Conclusion

In this work, we investigate the issue of alignment tax in the RLHF training of LLMs. Drawing inspiration from offline merging methods, we develop online merging optimizers that integrate general task arithmetic merging into each RLHF optimization step. Extensive experiments conducted on

various backbone LLMs demonstrate that online merging optimizers better mitigate alignment tax and achieve superior alignment performance compared to regularization and offline merging baselines. Additionally, we propose step-K online merging optimizers to bridge the gap between online and offline merging, providing in-depth analyses on the effects of hyper-parameters and ablations. While our primary focus is on using online merging optimizers for LLM alignment, their applications can naturally extend to other domains facing catastrophic forgetting, such as the continual learning of LLMs. We hope that online merging optimizers will become a representative optimizer in future RLHF training, enabling researchers to more boldly pursue reward maximization without concerns about model degradation, ultimately producing more helpful, honest, and harmless LLMs.

Limitations

Major limitations of online merging optimizers are related to the parameter efficiency. Online merging optimizers enhance the memory requirements as they require a cache of additional delta parameters of the reference model as the counterpart of the delta updated weights in training. At the same time, they can not be applied to the LoRA training, except for the cases where the reference models are also trained with LoRA adapters. However, this limitation can be further eliminated by applying GaLore (Zhao et al., 2024) with the online merging optimizers.

Acknowledgements

We sincerely appreciate the support from Kai Dang, An Yang, Junyang Lin, and others on Qwen's Alignment team. We thank Hongyi Yuan for providing insights on robust alignment and helping with engineering implementation, Le Yu for valuable feedback about model merging that improves our manuscript, and Tingyu Xia for visualization enhancement.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Direct preference optimization with an offset. *arXiv preprint arXiv:2402.10571*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models.

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.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report.

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

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, E Perez, Jamie Kerr, Jared Mueller, Jeff Ladish, J Landau, Kamal Ndousse, Kamilė Luko

vsiūtė, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noem'i Mercado, Nova Dassarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Sam Bowman, Zac Hatfield-Dodds, Benjamin Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom B. Brown, and Jared Kaplan. 2022b. Constitutional ai: Harmlessness from ai feedback. *ArXiv*, abs/2212.08073.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

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. *arXiv preprint arXiv:2310.01377*.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, 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.

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*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2021. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR.

Lingyue Fu, Huacan Chai, Shuang Luo, Kounianhua Du, Weiming Zhang, Longteng Fan, Jiayi Lei, Renting Rui, Jianghao Lin, Yuchen Fang, Yifan Liu, Jingkuan Wang, Siyuan Qi, Kangning Zhang, Weinan Zhang, and Yong Yu. 2024. Codeapex: A bilingual programming evaluation benchmark for large language models.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. 2024. Direct language model alignment from online ai feedback.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model.

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

J Stuart Hunter. 1986. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. 2023. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Lan Jiang, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, and Rui Jiang. 2022. Rose: Robust selective fine-tuning for pre-trained language models.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

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.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. 2024. Mitigating the alignment tax of rlhf.

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.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

Michael Matena and Colin Raffel. 2022. Merging models with fisher-weighted averaging.

Michael Noukhovitch, Samuel Lavoie, Florian Strub, and Aaron Courville. 2023. Language model alignment with elastic reset.

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.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

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

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-play preference optimization for language model alignment.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning.

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

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2023a. Hype: Better pre-trained language model fine-tuning with hidden representation perturbation.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023b. How well do large language models perform in arithmetic tasks?

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. 2024. Weak-to-strong extrapolation expedites alignment. *arXiv preprint arXiv:2404.16792*.

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*.

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.

Appendix

A Detailed Algorithms

Algorithm 1: Online Merging Optimizers (OnDARE, OnTIES) based on Adam

Inputs : Reference and base models θ_r, θ_b, total step T, coefficients β_1, β_2, learning rate η, parameter reserve rate p, merging weight α

States : First momentum $m^{(t)}$, second momentum $v^{(t)}$ at step t, delta of the reference model τ_r

Initialze $\tau_r = \theta_r - \theta_b$;
for *step* $t \leq T$ **do**
  for *parameter* $\theta^{(t-1)}$ *in* $\Theta^{(t-1)}$ *and its gradient* g^t **do**
    if $t == 0$ **then**
      $m^{(0)} = \mathbf{0}$; $v^{(0)} = \mathbf{0}$;
    end
    $m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$;
    $v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(g^{(t)})^2$;
    $\Delta\theta = -\eta m^{(t)} / \sqrt{v^{(t)} + \epsilon}$;
    `// Online merging begin`
    if *Using OnDARE* **then**
      $\theta^{(t)} = \theta^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) + \alpha\mathcal{F}_R(\tau_r)$
      `// Detailed in Eq. 3,4`
    end
    else if *Using OnTIES* **then**
      $\theta^{(t)} = \theta^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) \oplus_{sign} \alpha\mathcal{F}_R(\tau_r)$
      `// Detailed in Eq. 5,6,7`
    end
    `// Online merging end`
  end
end

B Benchmark Details

We all report the overall accuracy of the following benchmarks unless explaining with special annotations.

- **Mathematics**:
  - (1) GSM8K (Cobbe et al., 2021): we use the full test set and test in the zero-shot setting with greedy decoding;
  - (2) Math401 (Yuan et al., 2023b): we randomly downsample to obtain 101 samples and test in the zero-shot setting with default sampling decoding hyper-parameters;
  - (3) Math23K (Wang et al., 2017): we randomly downsample to obtain 250 samples and we test in the zero-shot setting with default sampling decoding hyper-parameters;
- **Coding**:
  - (1) HumanEval (Chen et al., 2021): we use the full test set and test in the zero-shot setting with greedy decoding and report the PASS@1;
  - (2) MBPP (Austin et al., 2021): we use the full test set, test in the zero-shot setting with default sampling decoding hyper-parameters, and report the PASS@1;
  - (3) DS1000 (Lai et al., 2022): we randomly downsample to obtain 600 samples, test in the zero-shot setting with default sampling decoding hyper-parameters, and report the PASS@1;
  - (4) CodeApex (Fu et al., 2024): we use the full test set, test in the zero-shot setting with default sampling decoding hyper-parameters, and report the PASS@1;
- **Instruction-following (IF)**: (1) IFEval (Zhou et al., 2023): we use the full test set, strictly follow the evaluation of IFEval, using greedy decoding and report strict instruction-level accuracy.

Algorithm 2: Step-K Online Merging Optimizers (OnDARE, OnTIES) based on Adam

Inputs : Reference and base models θ_r, θ_b, total step T, coefficients β_1, β_2, learning rate η, parameter reserve rate p, merging weight α, **online merging gap step** t_m

States : First momentum $m^{(t)}$, second momentum $v^{(t)}$ at step t, delta of the reference model τ_r, **cached delta** Δ_c

Initialze $\tau_r = \theta_r - \theta_b$;
for *step* $t \leq T$ **do**
  for *parameter* $\theta^{(t-1)}$ *in* $\Theta^{(t-1)}$ *and its gradient* g^t **do**
    if $t == 0$ **then**
      $m^{(0)} = \mathbf{0}$; $v^{(0)} = \mathbf{0}$; $\Delta_c = \mathbf{0}$;
    end
    $m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$;
    $v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(g^{(t)})^2$;
    $\Delta\theta = -\eta m^{(t)} / \sqrt{v^{(t)} + \epsilon}$;
    if *current step* % t_m == *0* **then**
      `// Rollback current parameters`
      $p = p - \Delta_c$;
      $\Delta\theta = \Delta\theta + \Delta_c$;
      $\Delta_c = \mathbf{0}$;
      if *Using OnDARE* **then**
        $\theta^{(t)} = \theta^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) + \alpha\mathcal{F}_R(\tau_r)$
        `// Detailed in Eq. 3,4`
      end
      else if *Using OnTIES* **then**
        $\theta^{(t)} = \theta^{(t-1)} + (1-\alpha)\mathcal{F}_R(\Delta\theta) \oplus_{sign} \alpha\mathcal{F}_R(\tau_r)$
        `// Detailed in Eq. 5,6,7`
      end
    end
    else
      `// Accumulate delta without merging`
      $\Delta_c = \Delta_c + \Delta\theta$;
      $\theta^{(t)} = \theta^{(t-1)} + \Delta\theta$;
    end
  end
end

- **Reading Comprehension (RC)**:
  - (1) COQA (Reddy et al., 2019): we use the full validation set, test in the zero-shot setting with default sampling decoding hyper-parameters, and report the recall of the keywords in the golden answer;
  - (2) DROP (Dua et al., 2019): we use the full validation set (only the last turn in the session) and test in the zero-shot setting with default sampling decoding hyper-parameters, and report the recall of the keywords in the golden answer;
- **Knowledge** (1) MMLU (Hendrycks et al., 2021): we use the full test set and test in the zero-shot setting with greedy decoding;
- **Agent** (1) NousResearch: we report the right rates of function calling from four agent evaluation sets delivered by NousResearch[5] with default decoding parameters.
- **Code-switching** We collect 113 queries for translation and 131 open-ended queries (most of which are in Chinese). For translation, we check if all tokens in the generated response are in the target

[5] https://huggingface.co/datasets/NousResearch/json-mode-eval
https://huggingface.co/datasets/NousResearch/func-calling-eval-singleturn
https://huggingface.co/datasets/NousResearch/func-calling-eval-glaive
https://huggingface.co/datasets/NousResearch/func-calling-eval

language; For open-ended queries, we check if all tokens are in the same language of the query. The final score is the ratio of responses that do not have code-switching problem.

MT-Bench and AlpacaEval(LC) are judged by GPT-4 and GPT-4-1106-preview, respectively. MT-Bench has a standard deviation about 0.05, and AlpacaEval(LC) has a standard deviation about 0.5. Both are caused by the scoring instability of LLM-as-Judge.

C Configurations

Hyper-parameter Searching. We use grid search for all optimal hyper-parameters in this work. To be more specific, we search 6 learning rates $\{1e^{-5}$,$5e^{-5}$,$1e^{-6}$,$5e^{-6}$,$1e^{-7}$,$5e^{-7}\}$ for all AdamW experiments to build strong baselines. For hyper-parameters in baselines and proposed methods, we search at least three different values for each parameter to achieve optimal methods. For every ablation study of online merging optimizers, we search 4 learning rates $\{5e^{-6}$,$1e^{-6}$,$1e^{-7}$,$5e^{-7}\}$. We search batch-size in {64, 128, 512}, and we find batch-size 128 consistently works better for all settings with different backbone LLMs. So, we run all main and ablation experiments with batch-size 128 without further annotations.

Generation Configuration. We run models with their suggested generation hyper-parameters. We run the generation of LLaMa-3-8B-It with top-p 0.8 and temperature 0.7. We run the generation of the Qwen1.5 series with top-p 0.8, temperature 0.7, top-k 20, and repetition penalty 1.05.

Baseline Implementations.

- **AdamW**: We run AdamW with $\beta = 0.1$ in DPO, which is also the same setting as our proposed online merging parameters.
- **KL Penalty**: We construct the KL Penalty baseline by searching the optimal β from $1e^{-5}$ to 1.
- **EMA**: We implement the exponential moving average in each optimization step and search the optimal exponential coefficient from $1e^{-2}$, $1e^{-3}$, and $1e^{-4}$. We find $1e^{-3}$ consistently achieves the best performance across all settings.
- **ChildTuning**: We implement the ChildTuning$_{\text{F}}$ following official implementation [6]. We compare ChildTuning with our online merging optimizers under the same drop rates.
- **Offline Merging**: We use the implementation in *mergekit*[7]. For each merging, we search three densities {0.3, 0.5, 0.7} and five weights {0.1, 0.3, 0.5, 0.7, 0.9} to make sure achieving the best baseline performance.

Experiment Resources. We run all training on 64 NVIDIA H800 GPUs. Qwen1.5-1.8B-Chat, Qwen1.5-7B-Chat, and LLaMa-3-8B-it need training for about 3 hours. We run all evaluations on 32 NVIDIA H800 GPUs for half an hour.

Asset Licences. All Qwen models are distributed with the Tongyi Qwen Licence. All LLaMa-3 models used in this work are distributed with the LLaMa-3 Licence [8]. The training dataset ULTRAFEEDBACK and other open-source evaluation datasets are distributed with non-commercial licenses, such as the MIT Licence and Apache 2.0 Licence.

D Detailed Results

[6] https://github.com/RunxinXu/ChildTuning

[7] https://github.com/arcee-ai/mergekit

[8] https://llama.meta.com/llama3/license/

Table 4: Main results of baselines and online merging optimizers on ULTRAFEEDBACK trained from various backbone LLMs. In merging and dropping settings, we experiment with two variants, OnDARE and OnTIES. I.F., R.C., and Know. are short for instruction-following, reading comprehension, and knowledge. The Avg. columns are average scores except for MT-Bench and AlpacaEval. Benchmarks in each category are detailed in §5.1.

	Method	Math	Code	I.F.	R.C.	Know.	Agent	CodeSwitch	Benchmark Avg.	MT-Bench	AlpacaEval(LC)
Qwen1.5-1.8B-Chat	Reference	37.3	7.8	23.1	52.2	43.1	70.8	58.6	41.8	4.37	3.99
	AdamW	**37.8**	8.4	23.1	50.8	43.2	67.5	61.9	41.8	4.59	4.79
	Regularization										
	KL	37.5	8.4	23.1	51.3	43.5	68.7	62.7	42.2	4.59	4.81
	EMA	35.8	7.9	22.3	49.4	43.3	70.5	63.5	41.8	4.68	4.69
	ChildTuning	36.9	8.5	22.9	52.7	43.9	67.5	62.3	42.1	4.62	4.75
	LoRA	36.5	**8.6**	23.9	52.4	43.5	67.8	59.4	41.7	4.52	4.56
	Offline Merging										
	linear	36.1	8.4	23.9	50.5	43.0	70.5	61.9	42.0	4.56	3.83
	DARE	35.4	8.1	23.3	49.6	43.1	**73.1**	61.1	42.0	4.68	4.07
	TIES	36.2	8.2	23.9	51.0	42.6	69.0	61.1	41.7	4.49	4.38
	Online Merging (Ours)										
	OnTIES	35.8	7.8	24.1	50.3	43.4	69.0	**63.9**	42.1	4.74	4.91
	OnDARE	36.5	7.8	**24.2**	**55.7**	**44.0**	65.8	62.3	**42.3**	**4.83**	4.84
Qwen1.5-7B-Chat	Reference	65.5	23.0	34.3	69.2	59.1	74.6	75.0	57.3	6.97	10.71
	AdamW	69.3	22.1	38.0	71.9	59.4	76.3	71.7	58.4	7.18	11.66
	Regularization										
	KL	67.4	23.0	35.8	69.9	59.5	79.5	74.6	58.5	6.95	11.74
	EMA	67.3	22.6	36.1	68.9	59.0	77.8	75.4	58.2	6.86	12.92
	ChildTuning	66.5	23.1	35.9	70.8	58.8	78.9	74.2	58.3	7.07	11.55
	LoRA	67.4	22.5	35.6	69.8	59.0	78.7	**77.0**	58.6	7.04	11.33
	Offline Merging										
	linear	68.1	**24.0**	35.7	69.8	59.2	77.8	72.5	58.2	6.9	10.01
	DARE	67.5	23.3	35.9	70.2	58.9	77.2	**77.0**	58.6	7.05	10.07
	TIES	67.1	23.8	37.8	70.5	59.3	**80.1**	74.2	59.0	6.97	10.31
	Online Merging (Ours)										
	OnTIES	67.1	22.2	38.5	72.2	**60.4**	78.7	73.0	58.9	7.1	12.57
	OnDARE	69.4	22.9	**40.3**	72.6	59.7	75.4	75.8	**59.5**	7.3	11.94
LLaMa-3-8B-Instruct	Reference	67.8	39.5	68.3	81.2	65.7	39.5	41.8	57.7	7.97	22.92
	AdamW	68.3	39.3	68.4	81.7	65.7	40.9	41.4	58.0	8.04	24.10
	Regularization										
	KL	**70.2**	38.5	67.8	81.3	**65.8**	42.1	47.1	59.0	8.12	23.08
	EMA	65.4	39.2	**69.4**	81.6	65.6	42.7	43.9	58.3	8.22	24.44
	ChildTuning	66.8	38.0	68.6	81.0	65.0	38.9	**49.2**	58.2	8.18	25.05
	LoRA	68.4	38.7	68.2	81.3	**65.8**	**45.0**	43.0	58.6	8.11	23.94
	Offline Merging										
	linear	67.2	39.1	68.7	80.5	65.2	40.1	36.9	56.8	8.15	24.12
	DARE	67.3	39.0	68.3	81.5	65.3	42.4	45.9	58.5	8.18	24.28
	TIES	67.1	**39.6**	68.1	80.2	65.4	42.7	42.6	58.0	8.05	24.31
	Online Merging (Ours)										
	OnDARE	68.4	38.0	69.2	**82.2**	65.5	43.9	48.0	**59.3**	**8.23**	24.94
	OnTIES	68.2	38.6	68.4	80.9	65.6	43.6	41.0	58.0	8.2	25.67

Table 5: Detailed results on each benchmark of experiments trained from LLaMa-3-8B-It

method	gsm8k/acc	math401/acc	math23k/acc	ds1000/pass-1	mbpp/pass-1	codeapex/pass-1	humaneval/pass-1	ifeval/strict_prompt_acc	ifeval/strict_instruction_acc	coqa/acc	drop/acc	mmlu/acc	agent_nous/function_call/acc	code_switch_detect/acc
Reference	79.15	60.40	64.0	29.91	47.8	40.42	40.05	63.49	73.20	83.26	79.05	65.70	39.47	41.80
AdamW	79.15	61.39	64.4	26.50	49.6	40.63	40.62	63.49	73.38	84.05	79.25	65.72	40.94	41.39
KL	80.89	62.38	67.2	25.64	50.8	37.26	40.16	62.85	72.84	83.93	78.64	65.76	42.11	47.13
EMA	78.85	55.45	62.0	25.93	50.2	40.63	40.21	64.23	74.58	83.83	79.36	65.55	42.69	43.85
ChildTuning	77.86	57.43	65.2	26.78	49.0	35.37	40.83	64.23	72.96	85.23	76.83	65.01	38.89	49.18
LoRA	79.08	60.40	65.6	26.50	49.2	38.32	40.62	63.12	73.20	83.77	78.89	65.78	45.03	43.03
linear	79.45	58.42	63.6	26.21	49.6	40.42	40.36	63.86	73.44	82.03	79.02	65.20	40.06	36.89
DARE	78.24	56.44	67.2	25.36	50.8	38.95	40.83	63.59	73.02	83.75	79.27	65.33	42.40	45.90
TIES	78.01	56.44	66.8	28.49	51.4	39.79	38.70	63.03	73.26	82.26	78.11	65.35	42.69	42.62
OnDARE	78.85	60.40	66.0	25.07	49.4	37.89	39.69	64.33	73.98	84.79	79.52	65.47	43.86	47.95
OnTIES	79.61	61.39	63.6	27.07	47.8	39.37	40.36	63.22	73.50	83.06	78.77	65.57	43.57	40.98

Table 6: Detailed results on each benchmark of experiments trained from Qwen1.5-1.8B-Chat

method	gsm8k/acc	math401/acc	math23k/acc	ds1000/pass-1	mbpp/pass-1	codeapex/pass-1	humaneval/pass-1	ifeval/strict_prompt_acc	ifeval/strict_instruction_acc	coqa/acc	drop/acc	mmlu/acc	agent_nous/function_call/acc	code_switch_detect/acc
ChildTuning	24.56	44.55	41.6	3.42	13.6	11.79	5.16	17.38	28.48	64.37	41.01	43.89	67.54	62.30
KL	24.11	47.52	40.8	3.70	12.2	13.47	4.17	17.84	28.36	62.89	39.68	43.48	68.71	62.70
Reference	25.55	41.58	44.8	3.13	11.0	11.79	5.36	18.11	28.00	63.28	41.15	43.11	70.76	58.61
AdamW	25.32	45.54	42.4	3.13	13.4	11.58	5.31	17.93	28.18	62.25	39.26	43.18	67.54	61.89
DARE	23.73	40.59	42.0	4.27	12.4	10.95	4.84	17.84	28.72	62.26	36.97	43.13	73.10	61.07
TIES	22.74	42.57	43.2	2.85	12.8	11.58	5.42	18.67	29.14	62.61	39.37	42.64	69.01	61.07
EMA	22.59	41.58	43.2	3.42	11.2	11.79	5.36	17.38	27.22	59.75	39.10	43.26	70.47	63.52
linear	23.73	44.55	40.0	2.28	13.2	13.68	4.58	18.48	29.26	62.17	38.88	43.01	70.47	61.89
LoRA	23.65	44.55	41.2	2.28	14.2	12.84	4.95	18.85	29.02	63.38	41.32	43.51	67.84	59.43
OnDARE	25.25	41.58	42.8	3.13	11.6	11.16	5.31	19.04	29.44	67.89	43.54	43.96	65.79	62.30
OnTIES	24.34	44.55	38.4	3.99	11.4	11.37	4.64	19.04	29.14	61.50	39.11	43.44	69.01	63.93

Table 7: Detailed results on each benchmark of experiments trained from Qwen1.5-7B-Chat

method	gsm8k/acc	math401/acc	math23k/acc	ds1000/pass-1	mbpp/pass-1	codeapex/pass-1	humaneval/pass-1	ifeval/strict_prompt_acc	ifeval/strict_instruction_acc	coqa/acc	drop/acc	mmlu/acc	agent_nous/function_call/acc	code_switch_detect/acc
KL	56.18	69.31	76.8	16.81	28.8	24.63	21.61	30.31	41.25	79.38	60.41	59.50	79.53	74.59
\rowcolor[gray]{0.95}Reference	56.41	69.31	70.8	16.24	32.2	24.42	19.32	28.93	39.75	78.72	59.73	59.09	74.56	75.00
AdamW	58.83	67.33	81.6	16.24	28.6	22.32	21.30	32.81	43.11	81.68	62.16	59.38	76.32	71.72
DARE	58.68	68.32	75.6	15.95	30.6	26.95	19.64	30.13	41.73	80.65	59.83	58.94	77.19	77.05
TIES	57.62	65.35	78.4	16.24	33.0	25.26	20.62	32.35	43.35	79.15	61.95	59.25	80.12	74.18
EMA	57.24	67.33	77.2	14.25	31.2	25.68	19.22	31.05	41.19	80.39	57.48	58.99	77.78	75.41
linear	56.33	71.29	76.8	15.38	32.2	27.37	21.09	30.13	41.31	80.11	59.57	59.24	77.78	72.54
LoRA	58.53	66.34	77.2	16.24	29.8	25.47	18.39	30.50	40.65	79.73	59.80	59.01	78.65	77.05
OnDARE	59.82	69.31	79.2	15.67	30.6	25.26	20.26	34.75	45.92	82.30	62.96	59.69	75.44	75.82
OnTIES	58.38	69.31	73.6	14.53	28.6	25.05	20.68	32.81	44.12	81.89	62.50	60.41	78.65	72.95