Google DeepMind

WARP: On the Benefits of Weight Averaged Rewarded Policies

Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, Olivier Bachem
Google DeepMind

Reinforcement learning from human feedback (RLHF) aligns large language models (LLMs) by encouraging their generations to have high rewards, using a reward model trained on human preferences. To prevent the forgetting of pre-trained knowledge, RLHF usually incorporates a KL regularization; this forces the policy to remain close to its supervised fine-tuned initialization, though it hinders the reward optimization. To tackle the trade-off between KL and reward, in this paper we introduce a novel alignment strategy named Weight Averaged Rewarded Policies (*WARP*). *WARP* merges policies in the weight space at three distinct stages. First, it uses the exponential moving average of the policy as a dynamic anchor in the KL regularization. Second, it applies spherical interpolation to merge independently fine-tuned policies into a new enhanced one. Third, it linearly interpolates between this merged model and the initialization, to recover features from pre-training. This procedure is then applied iteratively, with each iteration's final model used as an advanced initialization for the next, progressively refining the KL-reward Pareto front, achieving superior rewards at fixed KL. Experiments with Gemma policies validate that *WARP* improves their quality and alignment, outperforming other open-source LLMs.

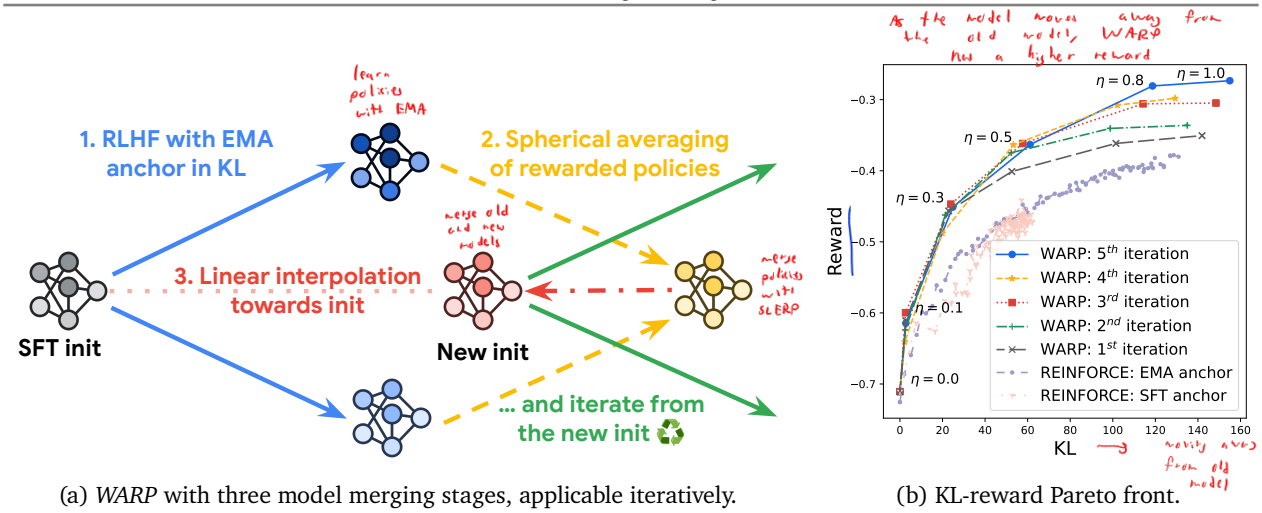
Keywords: Alignment, RLHF, LLM, Model Merging

1. Introduction

LLM alignment. Conversational agents like Gemini [36, 110] and GPT-4 [93], along with their open-weight counterparts like Gemma [129], have demonstrated remarkable abilities in complex tasks including mathematics, coding, and tool use [13]. These capabilities largely emerge from pre-training on next-token prediction [101, 102], subsequently refined through supervised fine-tuning (SFT) [105, 135]. As these LLMs become more powerful, aligning them with human values becomes increasingly crucial to ensure safe deployment [5, 46]. To this end, reinforcement learning from human feedback (RLHF) has become the prominent strategy [20, 122, 145], first learning a reward model (RM) on human preferences, before optimizing the LLM to maximize predicted rewards.

Challenges in RLHF. However, RLHF introduces several unresolved challenges [16]. First, the limited scope of fine-tuning, often restricted to relatively small datasets, can lead to excessive specialization and catastrophic forgetting [31] of the broad and diverse knowledge acquired during pre-training [38, 66, 67, 79]. Such *alignment tax* [97] can degrade the LLM's reasoning capabilities and performance on NLP benchmarks [25, 81]. Second, maximizing an imperfect RM presents several issues on its own, as the LLM can learn to exploit loopholes in the RM [21, 98] when it deviates significantly from its initialization [33]. Such *reward hacking* [7, 120] can produce outputs that are linguistically flawed [77], excessively verbose [119], or sycophantic [99, 116], thereby raising misalignment [90, 128] and safety [5, 46] concerns. Finally, RLHF can reduce the diversity of generations [65], potentially leading to policy collapse [42, 86]. Such *loss of diversity* limits use in creative or exploratory tasks and can result in the LLM systematically refusing to answer. Overall, achieving high rewards based on an imperfect RM on a selected distribution of prompts is insufficient due to potential reward misspecification and distribution shifts upon deployment.

Corresponding author: alexandrerame@google.com



(a) *WARP* with three model merging stages, applicable iteratively. (b) KL-reward Pareto front.

Figure 1 | Figure 1(a) illustrates the RLHF alignment process with *WARP* from a supervised fine-tuned (SFT) LLM. *WARP* uses model merging by weight averaging at three different stages. First, the exponential moving average (*EMA*) [55] of the policy serves as the anchor for KL regularization [59]. Second, the independently fine-tuned policies are merged by spherical linear interpolation (*SLERP*) [118] of their task vectors [53]. Third, we interpolate towards the initialization (*LITI*) [138], revealing a Pareto front of solutions as we slide the interpolating coefficient η from 1 to 0. This results in the "*WARP*: 1st iteration" curve from Figure 1(b) which improves over the REINFORCE [136] fine-tuning trajectories. Critically, iteratively using a point from this Pareto front as an advanced initialization for the next episode *WARP* improves performance. Details in Figure 4(c).

RL with KL regularization. To address these issues, previous works constrained the reward optimization by integrating a Kullback-Leibler (KL) regularization [35, 59], using the SFT initialization as the anchor. As clarified in Section 2, this KL regularization forces the policy to remain close to its initialization [74, 84], mitigating forgetting and reward hacking [33]. However, employing the SFT model as the anchor may lead to reward underfitting: indeed, there is a fundamental tension between reducing KL and maximizing reward. Thus, different policies should be compared in terms of KL-reward Pareto optimality as in Figure 1(b), where the x-axis is the KL and the y-axis is the reward as estimated by the RM, with the optimal policies located in the top-left of the plot.

On model merging by weight averaging. To improve the trade-off between KL and reward during RLHF, we leverage the ability to merge LLMs by weight averaging (WA) [131]. WA relies on the linear mode connectivity [30, 89], an empirical observation revealing linear paths of high performance between models fine-tuned from a shared pre-trained initialization. Model merging was shown to improve robustness under distribution shifts [55, 106, 137] by promoting generalization and reducing memorization [108], to combine models' abilities [52, 53, 109], to reduce forgetting in continual learning [123], to enable collaborative [104] and distributed [27] learning at scale, without computational overheads at inference time. Model merging is increasingly adopted within the open-source community [37, 72], leading to state-of-the-art models in specialized domains [70] but also significant advancements on general-purpose benchmarks [68, 69]. In particular, while WA was initially mostly used for discriminative tasks [137] such as reward modeling [108], it is now becoming popular for generative tasks [4, 111]; its use in KL-constrained RLHF has already shown preliminary successes in a few recent works [39, 81, 83, 88, 92, 109], further elaborated in Section 5.

WARP. In this paper, we propose Weight Averaged Rewarded Policies (*WARP*), a simple strategy for aligning LLMs, illustrated in Figure 1(a) and detailed in Section 3. *WARP* is designed to optimize the KL-reward Pareto front of solutions, as demonstrated in Figure 1(b). *WARP* uses three variants of WA at three different stages of the alignment procedure, for three distinct reasons.

Stage 1: *Exponential Moving Average* (*EMA*). During RL fine-tuning, instead of regularizing the policy towards the SFT initialization, *WARP* uses the policy's own exponential moving average [100] as a dynamic updatable anchor in the KL. This stage enables stable exploration with distillation from a mean teacher [127] and annealed constraint.

Stage 2: *Spherical Linear intERPolation of task vectors* (*SLERP*). Considering M policies RL fine-tuned independently with their own *EMA* anchor, we merge them by spherical linear interpolation [118] of their task vectors [53]. This stage creates a merged model with higher reward by combining the strengths of the M individual policies.

Stage 3: *Linear Interpolation Towards Initialization* (*LITI*). Considering the merged policy from *SLERP*, *WARP* linearly interpolates towards the initialization, akin to WiSE-FT [138]. This stage allows to run through an improved Pareto-front simply by adjusting the interpolating coefficient η between 1 (high reward but high KL) and 0 (small KL but small reward). Critically, selecting an intermediate value for $0 < \eta < 1$ offers a balanced model that can serve as a new, improved initialization for subsequent iterations of *WARP*.

Experiments and discussion. In Section 4, we validate the efficacy of *WARP* for the fine-tuning of Gemma "7B" [129]. Finally, in Section 6, we discuss the connections between *WARP*, the distributed learning literature [27, 104] and iterated amplification [19], illustrating how *WARP* embodies their principles to enable scaling post-training, for continuous alignment and improvement of LLMs.

2. Context and notations

RL for LLMs. We consider a transformer [132] LLM $f(\cdot, \theta)$ parameterized by θ. Following the foundation model paradigm [12] and the principles of transfer learning [94], those weights are trained via a three-stage procedure: pre-training through next token prediction, supervised fine-tuning resulting in θ_{sft}, and ultimately, RLHF [20, 97] to optimize a reward r as determined by a RM trained to reflect human preferences. In this RL stage, θ defines a policy $\pi_\theta(\cdot \mid \boldsymbol{x})$ by auto-regressively generating token sequences $\boldsymbol{y}$ from the prompt $\boldsymbol{x}$. The primary objective is to find weights maximizing the average reward over a dataset of prompts $\mathcal{X}$: $\operatorname{argmax}_\theta \mathbb{E}_{x\in\mathcal{X}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\Big[r(\boldsymbol{x},\boldsymbol{y})\Big]$.

KL **vs. reward.** Optimizing solely for r can (i) forget general abilities from pre-training [31] as an alignment tax [81, 97], (ii) hack the reward [7, 120] leading to potential misalignment, or (iii) reduce the diversity of possible generations [65] (confirmed in Appendix F). To mitigate these risks, a KL regularization is usually integrated to balance fidelity to the initialization and high rewards:

$$\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{x\in\mathcal{X}}\Big[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)} r(\boldsymbol{x},\boldsymbol{y}) - \beta \mathrm{KL}\big(\pi_\theta(\cdot \mid \boldsymbol{x}) \| \pi_{\theta_{\text{anchor}}}(\cdot \mid \boldsymbol{x})\big)\Big], \tag{1}$$

where $\theta_{\text{anchor}} \leftarrow \theta_{\text{sft}}$ and β is the regularization strength, with high values leading to low KL though also lower reward. The reward function adjusted with this KL is $r(\boldsymbol{x},\boldsymbol{y}) - \beta \log\Big(\frac{\pi_\theta(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\theta_{\text{anchor}}}(\boldsymbol{y}|\boldsymbol{x})}\Big)$. Our base RL algorithm is a variant of REINFORCE [136]. This choices follows recent RLHF works [75, 108, 112] and the findings from [2, 80, 126] that, in terms of KL-reward Pareto optimality, REINFORCE performs better than the more complex PPO [114] and also better than various offline algorithms such as DPO [103], IPO [11] or RAFT [26]. Practitioners then typically employ early stopping to select an optimal point on the training trajectory based on their specific use cases.

3. *WARP*

We introduce a novel alignment strategy named Weight Averaged Rewarded Policies (*WARP*), illustrated in Figure 1(a) and described in Algorithm 1 below. *WARP* merges LLMs in the weight space to enhance the KL-reward Pareto front of policies. The following Sections 3.1 to 3.3 describe the motivations behind applying three distinct variants of WA at the three different stages of *WARP*. In particular, we summarize the key insights as observations, that will be experimentally validated in Section 4 (and in Appendices C and D), and theoretically motivated in Appendix B when possible. Overall, *WARP* outperforms other RL alignment strategies, without any memory or inference overhead at test time. However, training *WARP* is costly, requiring multiple RL runs at each iteration: see Section 6 for a detailed discussion on the required compute scaling.

Algorithm 1 *WARP* for KL-reward Pareto optimal alignment

Input: Weights θ_{sft} pre-trained and supervised fine-tuned
Reward model r, prompt dataset $\mathcal{X}$, optimizer Opt
I iterations with M RL runs each for T training steps
μ *EMA* update rate, η *LITI* update rate

Define $\theta_{\text{init}} \leftarrow \theta_{\text{sft}}$
for iteration i from 1 to I **do**
  for run m from 1 to M **do** ▷ Run in parallel
    Define $\theta^m, \theta^m_{\text{ema}} \leftarrow \theta_{\text{init}}$
    for step t from 1 to T **do**
      Generate completion $\boldsymbol{y} \sim \pi_{\theta^m}(\cdot \mid \boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$
      Compute $r_\beta(\boldsymbol{y}) \leftarrow r(\boldsymbol{x}, \boldsymbol{y}) - \beta \log \frac{\pi_{\theta^m}(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\theta^m_{\text{ema}}}(\boldsymbol{y}|\boldsymbol{x})}$ ▷ KL regularized reward
      Update $\theta^m \leftarrow \text{Opt}\big(\theta^m, r_\beta(\boldsymbol{y}) \nabla_\theta [\log \pi_{\theta^m}(\boldsymbol{y} \mid \boldsymbol{x})]\big)$ ▷ Policy gradient
      Update $\theta^m_{\text{ema}} \leftarrow (1-\mu) \cdot \theta^m_{\text{ema}} + \mu \cdot \theta^m$ ▷ Equation (*EMA*): update anchor
    end for
  end for
  Define $\theta^i_{\text{slerp}} \leftarrow \text{slerp}\Big(\theta_{\text{init}}, \{\theta^m\}_{m=1}^M, \lambda = \frac{1}{M}\Big)$ ▷ Equation (*SLERP*): merge M weights
  Update $\theta_{\text{init}} \leftarrow (1-\eta) \cdot \theta_{\text{init}} + \eta \cdot \theta^i_{\text{slerp}}$ ▷ Equation (*LITI*): interpolate towards init
end for

Output: KL-reward Pareto front of weights $\{(1-\eta) \cdot \theta_{\text{sft}} + \eta \cdot \theta^I_{\text{slerp}} \mid 0 \leq \eta \leq 1\}$

3.1. Stage 1: exponential moving average as a dynamic anchor in KL regularization

***EMA* anchor.** KL-regularized methods typically use the SFT initialization as a static anchor [59, 112], but in RL for control tasks, it is common to regularly update the anchor [1, 113]. In this spirit, *WARP* uses the policy's own exponential moving average (*EMA*) [100], updated throughout the RL fine-tuning process such as, at each training step with $\mu = 0.01$:

$$\theta_{\text{ema}} \leftarrow (1-\mu) \cdot \theta_{\text{ema}} + \mu \cdot \theta_{\text{policy}}. \qquad (\textit{EMA})$$

Using θ_{ema} as the anchor θ_{anchor} in Equation (1) provides several benefits, outlined below.

Observation 1 (*EMA*). *Policies trained with an exponential moving average anchor benefit from automatic annealing of the KL regularization and from distillation from a dynamic mean teacher [127]. Empirical evidence in Section 4.1.*

Benefits from *EMA*. Unlike a static SFT anchor, the dynamic nature of an *EMA* anchor induces a gradual automatic annealing and relaxation of the KL regularization. Specifically, the policy is initially strongly tied to the SFT initialization, and then progressively unleashed, allowing for more aggressive gradient updates later in training, leading to higher rewards. Moreover, by progressively incorporating knowledge from the training, *EMA* acts as slow weight [76, 123], and thus performing better than the initialization. But, by also maintaining essential information from the initialization, *EMA* can even perform better than the final policy's weights; studies [6, 55, 125] (see [87] for a review), and specifically [62] within the context of LLMs, indicate that averaging checkpoints over steps improves internal representations and thus predictions. Then, *EMA* guides the policy by KL distillation [49] of high-quality target predictions, akin to a mean teacher [127] for self-supervised [15, 40, 44, 95, 121] learning. This also relates to deep RL techniques where *EMA* stabilizes exploration toward a Nash equilibrium [9, 10, 39, 88], and approximates mirror descent [14, 35, 130].

3.2. Stage 2: spherical linear interpolation of independently rewarded policies

***SLERP*.** While *EMA* helps for a single RL and a fixed compute budget, it faces limitations due to the similarity of the weights collected along a single fine-tuning [106]. In this second stage, we merge M weights RL fine-tuned independently (each with their own *EMA* anchor). This follows model soups from Wortsman et al. [137] and its variants [106, 107] showing that WA improves generalization, and that task vectors [53] (the difference between fine-tuned weights and their initialization) can be arithmetically manipulated by linear interpolation (*LERP*) [131]. Yet, this time, we use spherical linear interpolation (*SLERP*) [118], illustrated in Figure 2 and defined below for $M = 2$:

$$\text{slerp}\Big(\theta_{\text{init}}, \theta^1, \theta^2, \lambda\Big) = \theta_{\text{init}} + \frac{\sin[(1-\lambda)\Omega]}{\sin\Omega}\cdot\underbrace{\delta^1}_{\theta^1-\theta_{\text{init}}} + \frac{\sin[\lambda\Omega]}{\sin\Omega}\cdot\underbrace{\delta^2}_{\theta^2-\theta_{\text{init}}}, \qquad \text{(SLERP)}$$

where Ω is the angle between the two task vectors $\delta^1 = \theta^1 - \theta_{\text{init}}$ and $\delta^2 = \theta^2 - \theta_{\text{init}}$, and λ the interpolation coefficient. Critically *SLERP* is applied layer by layer, each having a different angle. In Appendix B.3 we clarify how *SLERP* can be used iteratively to merge $M > 2$ models. To enforce diversity across weights, we simply vary the order in which text prompts $\boldsymbol{x}$ are given in each run: this was empirically sufficient, though other diversity strategies could help, e.g., varying the hyperparameters or the reward objectives (as explored in Figure 18(c)).

Benefits from *SLERP* vs. *LERP*. Merging task vectors, either with *SLERP* or *LERP*, combines their abilities [53]. The difference is that *SLERP* preserves their norms, reaching higher rewards than the base models; this is summarized in Observation 2. In contrast, and as summarized in Observation 3, the more standard *LERP* has less impact on reward, but has the advantage of reducing KL; indeed, as shown in Appendix B, *LERP* tends to pull the merged model towards the initialization, especially as the angle Ω between task vectors is near-orthogonal (see Observation 3).

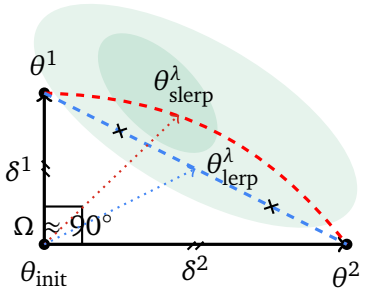


Figure 2 | *SLERP* vs. *LERP*.

Observation 2 (*SLERP*). *Spherical linear interpolation boosts rewards, yet slightly increases* KL. *Empirical evidence in Section 4.2 and theoretical insights in Lemma 1.*

Observation 3 (*LERP*). *Linear interpolation reduces* KL, *yet has reduced impact on reward. Empirical evidence in Appendix C.1 and theoretical insights in Lemmas 2 and 3.*

Observation 4 (Task vectors). *Task vectors δ are close to orthogonal with $\Omega \approx 90°$, while the full weights θ are collinear. Empirical evidence in Appendix C.2.*

3.3. Stage 3: linear interpolation towards initialization

LITI. In the previous stage, *SLERP* combines multiple policies into one with higher rewards and slightly higher KL. This third stage, inspired by WiSE-FT from Wortsman et al. [138], interpolates from the merged model towards the initialization:

$$\theta^{\eta} \leftarrow (1-\eta) \cdot \theta_{\text{init}} + \eta \cdot \theta_{\text{slerp}}. \qquad (\textit{LITI})$$

Adjusting the interpolating coefficient $\eta \in [0, 1]$ trades off between some newly acquired behaviors leading to high rewards vs. general knowledge from the SFT initialization. Specifically, large values $\eta \approx 1$ provide high rewards but also high KL, while smaller values $\eta \approx 0$ lean towards smaller rewards and minimal KL. Fortunately, we observe that the reduction in KL is proportionally greater than the reduction in reward when decreasing η. Then, *LITI* empirically yields Pareto fronts that are noticeably above the "diagonal", but also above those revealed during the base RLs.

Observation 5 (*LITI*). *Interpolating weights towards the initialization reveals a better Pareto front than the one revealed during RL fine-tuning. Empirical evidence in Figure 1(b) and Section 4.3, and theoretical insights in Lemmas 4 and 5.*

Benefits from *LITI*. Previous works tried to understand how weight interpolation can mitigate forgetting while increasing robustness and generalization. [138] argues that WiSE-FT, akin to *LITI* in supervised learning contexts, recovers generalizable features from pre-training that might be lost during fine-tuning [67], consistently with WA reducing catastrophic forgetting [28, 123] in continual learning. Then in the context of RL, [81] argues that *LITI* increases feature diversity, efficiently balancing between generality and task specificity. Finally, [58] argues that the geometric projection of the ideal weights is located between the merged model and the initialization.

3.4. Iterative *WARP*

Iterative training. The model merging strategies previously described not only establish an improved Pareto front of solutions, but also set the stage for iterative improvements. Indeed, if the computational budget is sufficient, we can apply those three stages iteratively, using θ^{η} from previous Pareto front (usually with $\eta = 0.3$, choice ablated in Appendix D.3) as the initialization θ_{init} for the next iteration, following the model recycling [24, 107] strategies. Then, the entire training procedure is made of multiple iterations, each consisting of those three stages, where the final weight from a given iteration serves as an improved initialization for the next one.

Observation 6 (Iterative *WARP*). *Applying WARP iteratively improves results, converging to an optimal Pareto front. Empirical evidence in Sections 4.4 and 4.5.*

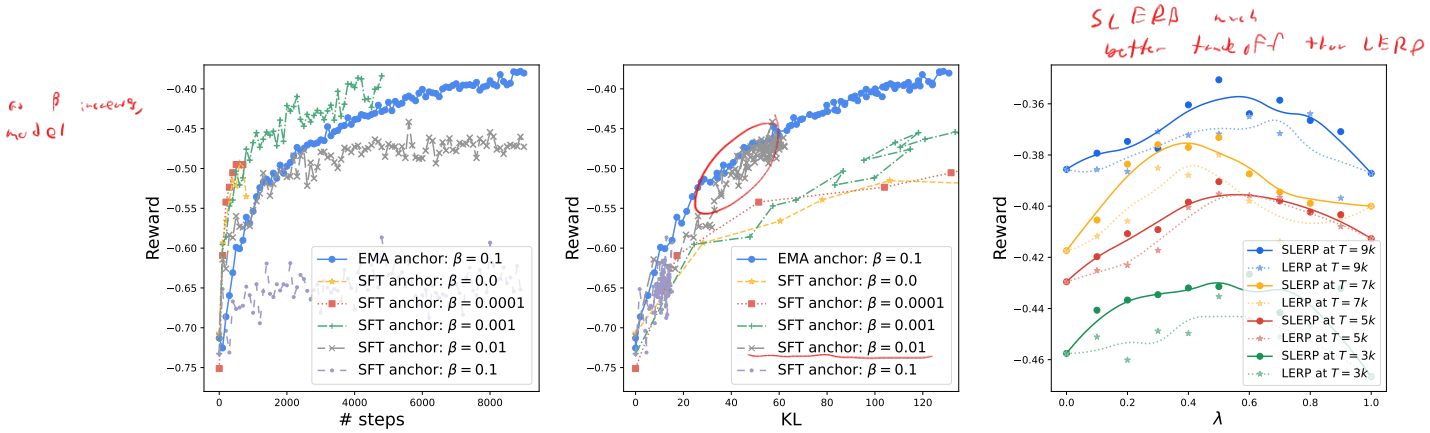
4. Experiments: on the benefits of *WARP*

Setup. We consider the Gemma "7B" [129] LLM, which we seek to fine-tune with RLHF into a better conversational agent. We use REINFORCE [136] policy gradient to optimize the KL-regularized reward. The dataset $\mathcal{X}$ contains conversation prompts. We generate on-policy samples with temperature 0.9, batch size of 128, Adam [64] optimizer with learning rate 10^{-6} and warmup of 100 steps. *SLERP* is applied independently to the 28 layers. Except when stated otherwise, we train for $T = 9k$ steps, with KL strength $\beta = 0.1$, *EMA* update rate $\mu = 0.01$, merging $M = 2$ policies uniformly $\lambda = 0.5$, and *LITI* update rate $\eta = 0.3$; we analyze those values in Appendix D. We rely on a high capacity reward model, the largest available, which prevents the use of an oracle control RM as done in [33, 108].

Summary. In our experiments, we analyze the KL to the SFT policy (reflecting the forgetting of pre-trained knowledge) and the reward (evaluating alignment to the RM). In Section 4.1, we first show the benefits of using an *EMA* anchor; then in Section 4.2, we show that merging policies trained independently helps. Moreover, in Section 4.3, we show that *LITI* improves the KL-reward Pareto front; critically, repeating those three *WARP* stages can iteratively improve performances in Section 4.4. A limitation is that our RM accurately approximates true human preferences only in low KL region, though can be hacked away from the SFT [33]. Therefore, we finally report other metrics in Section 4.5, specifically comparing against open-source baselines such as Mixtral [61], and reporting performances on standard benchmarks such as MMLU [47].

4.1. Stage 1: exponential moving average as a dynamic anchor in KL regularization

In Figures 3(a) and 3(b), we compare the training trajectories of different REINFORCE variants, where the changes lie in the choice of the anchor in the KL regularization and of the hyperparameter β controlling its strength. Results are computed every 100 training steps. In our proposed version, the anchor is the *EMA* of the trained policy with $\beta = 0.1$ and an *EMA* update rate $\mu = 0.1$ (other values are ablated in Figure 15). As the Pareto front for our strategy is above and to the left in Figure 3(b), this confirms the superiority of using such an adaptive anchor. The baseline variants all use the SFT as the anchor, with different values of β. The lack of regularization ($\beta = 0.0$) leads to very fast optimization of the reward in Figure 3(a), but largely through hacking, as visible by the KL exploding in just a few training steps in Figure 3(b). In contrast, higher values such as $\beta = 0.1$ fail to optimize the reward as regularization is too strong, causing a quick reward saturation around -0.62 in Figure 3(a). Higher values such as $\beta = 0.01$ can match our *EMA* anchor in low KL regime, but saturates around a reward of -0.46. In contrast, as argued in Observation 1, the dynamic *EMA* anchor progressively moves away from the SFT initialization, causing implicit annealing of the regularization. In conclusion, relaxing the anchor with *EMA* updates allows the efficient learning of KL-reward Pareto-optimal policies, at any given KL level, for a fixed compute budget. We refer the interested reader to additional experiments in Figure 14 from Appendix D.2 where we compare the trained policies with their online *EMA* version.



(a) Reward vs. steps. (b) Reward vs. KL. (c) *SLERP* vs. *LERP*.

Figure 3 | ***EMA* and *SLERP* experiments.** We first compare RL runs with different anchors and strengths β in the KL regularization. We show their results along training in Figure 3(a), and their KL-reward Pareto fronts in Figure 3(b). We perform evaluation every 100 steps, and train them for $T = 9k$ steps, though we stopped the trainings if they ever reach a KL of 200 (e.g., after $T = 1k$ training steps when $\beta = 0.0$). Figure 3(c) plots the reward obtained when merging two policies (trained independently after T RL steps with their own *EMA* anchor) with interpolating coefficient λ; highest rewards are with *SLERP* for $\lambda = 0.5$ and $T = 9k$ steps.

4.2. Stage 2: spherical linear interpolation of independently rewarded policies

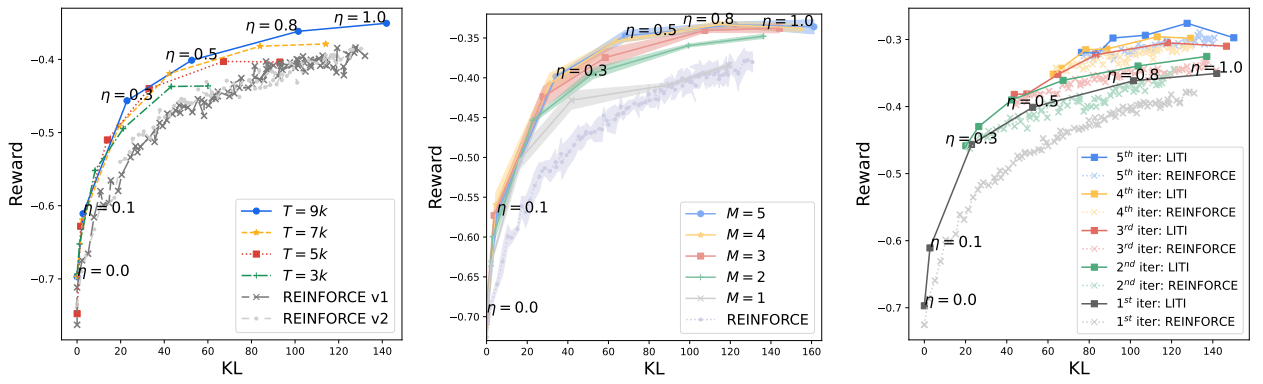
In Figure 3(c), we plot $\lambda \to r(\text{slerp}(\theta_{\text{init}}, \theta^1, \theta^2, \lambda))$ showing reward convexity when interpolating policies via *SLERP*, validating Observation 2. This mirrors the linear mode connectivity [30] property across weights fine-tuned from a shared initialization, i.e., the fact that interpolated weights perform better than the initial models (recovered for $\lambda = 0$ or $\lambda = 1$). Moreover, *SLERP* consistently obtains higher rewards than *LERP*; yet, this is at slightly higher KL, as further detailed in Appendices B and C.1, where we analyze respectively their theoretical and empirical differences.

4.3. Stage 3: linear interpolation towards initialization

In Figure 4(a), we merge policies trained for T steps, and then apply the *LITI* procedure. Critically, sliding the interpolating coefficient $\eta \in \{0, 0.1, 0.3, 0.5, 0.8, 1.0\}$ reveals various Pareto fronts, consistently above the training trajectories obtained during the two independent RL fine-tunings. Interestingly, longer fine-tunings improve performances, at high KL, but also at lower KL, simply by using a smaller η afterwards. Then in Figure 4(b), we report the Pareto fronts when merging up to $M = 5$ weights. We note that all Pareto fronts revealed when applying *LITI* are consistently above the ones from RL fine-tunings, validating Observation 5. More precisely, best results are achieved by merging an higher number of policies M, suggesting a promising scaling direction.

4.4. Iterative *WARP*

In Figure 4(c), we apply the iterative procedure described in Section 3.4. At each of the $I = 5$ iterations we train $M = 2$ policies for T steps, with $T = 9k$ for the first iteration, and $T = 7k$ for iterations 2 and 3, and then $T = 5k$ for computational reasons. The *LITI* curves interpolate towards their own initialization (while Figure 1(b) interpolated towards the SFT initialization, see Appendix D.4 for a comparison). We systematically observe that *LITI* curves are above the RL training trajectories used to obtain the inits. Results get better at every iteration, validating Observation 6, although with reduced returns after a few iterations.



(a) *LITI* of *SLERP* after T steps. (b) *LITI* of *SLERP* of M weights. (c) Iterative *WARP*.

Figure 4 | ***LITI* and iterative experiments.** Figure 4(a) considers the *LITI* of the *SLERP* of $M = 2$ policies after T steps with $\lambda = 0.5$, interpolating towards their SFT init as we slide η, revealing Pareto fronts above the $M = 2$ REINFORCE training trajectories. Then Figure 4(b) plots the *LITI* of the *SLERP* of M weights with $\lambda = \frac{1}{M}$ after $T = 9k$ steps: light-colored areas show standard deviations across 5 experiments. The iterative *WARP* procedure is illustrated in Figure 4(c); we fine-tune $M = 2$ policies with their own *EMA* as the anchor, merge them with *SLERP*, interpolate towards their init with *LITI*, and iteratively leverage the weights obtained with $\eta = 0.3$ as the new initialization for the next iteration.

Table 1 | **Side by side comparisons**.

Methods	Mistral 7B v1	Mistral 7B v2	Mixtral 8x7B
Gemma "7B" 1.0	0.24	-0.01	-0.08
Gemma "7B" 1.1	0.37	0.16	0.08
REINFORCE *EMA* anchor	0.37	0.16	0.07
WARP: 1st iter	0.42	0.23	0.13
WARP: 2nd iter	**0.45**	0.25	0.16
***WARP*: 3rd iter**	**0.45**	**0.26**	**0.18**
WARP: 4th iter	**0.45**	0.25	0.16
WARP: 5th iter	**0.45**	0.24	0.17

4.5. Comparisons and benchmarks

Side by side comparisons. To conclude our experiments, we compare our trained policies against Mistral [60] and Mixtral [61] LLMs. Each policy generates a candidate answer on an held-out collection of prompts, as in the Gemma tech report [129]. Then similarly to Gemini 1.5 [110], we compute side by side preference rates [144] with "much better", "better" and "slightly better" receiving scores of ±1.5, ±1, and ±0.5 respectively (and ties receiving a score of 0). A positive score represents better policies. The results in Table 1 validate the efficiency of *WARP*, as our policies are preferred over Mistral variants, and also outperform the two previous Gemma "7B" releases. However, we note that the results stagnate after the 3rd iteration.

Benchmarks. Table 2 compares *WARP* (3rd iter) and the latest Gemma "7B" 1.1 release [129] on popular benchmarks in the zero-shot setup: MBPP [8] and HumanEval [18] benchmarking coding capabilities, MMLU [47] assessing STEM knowledge, the GSM8K [22] and MATH [48] benchmarks targeting reasoning abilities, and the Big Bench Hard (BBH) [124] benchmark evaluating general capabilities through questions that were deemed difficult for frontier LLMs. *WARP* has particularly strong results on mathematics benchmarks, suggesting higher analytical capabilities.

Table 2 | **Benchmark results**.

Methods	MBPP	MMLU	GSM8K	MATH	HumanEval	BBH
Gemma "7B" 1.1	39.0	56.4	55.6	25.6	46.9	53.1
WARP	**45.4**	**57.6**	**66.8**	**31.0**	**50.0**	**58.8**

5. Related work

How to merge models. The question of how best to merge models has recently garnered significant attention, driven by the discoveries that deep models can be merged in the weight space [131] instead of in the prediction space, as traditionally done in ensembling [43, 71]. For clarity, we collectively refer to these methods as weight averaging (WA). The most common is *LERP*, initially used to average checkpoints collected along a single run, uniformly [55, 125] or with an exponential moving average (*EMA*) [100], notably as a mean teacher [127] for self-supervision [15, 40, 44, 95, 121]. Following the linear mode connectivity [30] observation, the model soups variants [53, 107, 137] linearly interpolate from different fine-tunings; this relies on the shared pre-training, limiting divergence [89] such as models remain in constrained weight regions [41], which also suggests that pre-training mitigates the need to explicitly enforce trust regions in gradient updates [113, 114]. Subsequent works such as TIES merging [140] and DARE [141] reduce interferences in multi-task setups with sparse task

vectors [53]. In contrast, we use *SLERP*, introduced in [118], increasingly popular in the open-source community [37] but relatively underexplored in the academic literature, with limited studies such as [63]. Some tried to align weights trained from scratch [3, 29] or with different architectures [133]; yet, the methods are complex, less robust, and usually require additional training.

Benefits of model merging. WA boosts generalization by reducing variance [106, 137], decreasing memorization [82, 108, 142] and flattening the loss landscape [17]. Additionally, merging weights combines their strengths [53], which helps in multi-task setups [52, 109], to tackle catastrophic forgetting [28, 123] or to provide better initializations [24], as explored in [51, 57, 58] for iterative procedures in classification tasks. In particular, we considered using the geometric insights from Eq. 2 in [58]; yet, as our task vectors are nearly orthogonal $\Omega \approx 90°$ (see Appendix C.2), using the update rule $\eta \rightarrow \frac{2\cos\Omega}{1+\cos\Omega}$ failed. WA is now also used in RL setups [34, 73, 91]; for example, *WARM* [108] merges reward models to boost their efficiency, robustness and reliability. Actually, *WARP* is conceived as a response to *WARM*, demonstrating that model merging can tackle two key RLHF challenges; policy learning in *WARP* and reward design in *WARM*. The most similar works are the following, which also explore how WA can improve policy learning. [92] proposes an iterative approach with the *EMA* as a new initialization for subsequent iterations. [39] and [88] uses *EMA* as the reference, but only for direct preference optimization. [109] employs *LERP* to improve alignment in multi-objective RLHF when dealing with different objectives; similarly, [139] targets multi-task setups with *LERP*. Finally, [81] and [32] use model merging to reduce the alignment tax, although without incorporating *EMA* during training, without merging multiple rewarded policies and not iteratively. Critically, none of these works focus on KL as a measure of forgetting, use *EMA* as the anchor in KL, apply *SLERP* or use *LITI* as the initialization for subsequent RL iterations. In contrast, *WARP* integrates all those elements, collectively leading to an LLM outperforming Mixtral [61].

6. Discussion

Distributed learning for parallelization and open-source. *WARP* addresses a crucial challenge: aligning LLMs with human values and societal norms, while preserving the capabilities that emerged from pre-training. To this end, we leverage a (perhaps surprising) ability: policies trained in parallel can combine their strengths within a single policy by weight averaging. Then, the distributed nature of *WARP* makes it flexible and scalable, as it is easily parallelizable by enabling intermittent weight sharing across workers. Actually, iterative *WARP* shares similarities with DiLoCo [27]: by analogy, the first stage performs inner optimization on multiple workers independently; the second stage merges gradients from different workers; the third stage performs SGD outer optimization with a learning rate equal to η. More generally, *WARP* could facilitate open-source [37] collaborative training of policies [104], optimizing resource and supporting privacy in federated learning [85] scenarios; collaborators could train and share their LLMs, while keeping their data and RMs private. In particular, we show in Appendix E that *WARP* can handle diverse objectives, similarly to [109].

Iterated amplification. *WARP* improves LLM alignment by leveraging the principles of iterated amplification [19] and progressive collaboration of multiple agents. By analogy, model merging via WA acts as an effective alternative to debate [54], with agents communicating within the weight space instead of the token space, ensuring that only essential information is retained [108]. Then, *WARP* refines the training signal by combining insights and exploration from diverse models, iteratively achieving higher rewards through self-distillation [127], surpassing the capabilities of any single agent. If this is the way forward, then an iterative safety assessment would be required to detect and mitigate potential risks early, ensuring that the development remains aligned with safety standards.

Scaling alignment. The *WARP* procedure increases the compute training cost by performing multiple fine-tunings at each iteration. Yet, this should be viewed as "a feature rather than a bug". Specifically, by preventing memorization and forgetting, we see *WARP* as a fine-tuning method that can transform additional compute allocated to alignment into enhanced capabilities and safety. This would allow scaling (the traditionally cheap) post-training alignment, in the same way pre-training has been scaled [50]. Critically for large-scale deployment, the acquired knowledge is within a single (merged) model, thus without inference or memory overhead, in contrast to "more agents" approaches [78, 134]. Finally, although *WARP* improves policy optimization, it is important to recognize that *WARP* does not address other critical challenges in RLHF [16]: to mitigate the safety risks [5, 45, 46] from misalignment [90, 128], *WARP* should be part of a broader responsible AI framework.

7. Conclusion

We introduce Weight Averaged Rewarded Policies (*WARP*), a novel RLHF strategy to align LLMs with three distinct stages of model merging: exponential moving average as a dynamic anchor during RL, spherical interpolation to combine multiple policies rewarded independantly, and interpolation towards the shared initialization. This iterative application of *WARP* improves the KL-reward Pareto front, aligning the LLMs while protecting the knowledge from pre-training, and compares favorably against state-of-the-art baselines. We hope *WARP* could contribute to safe and powerful AI systems by scaling alignment, and spur further exploration of the magic behind model merging.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *ICLR*, 2018. (p. 4)

[2] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint*, 2024. (p. 3)

[3] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *ICLR*, 2022. (p. 10)

[4] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *arXiv preprint*, 2024. (p. 2)

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint*, 2016. (pp. 1 and 11)

[6] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. In *NeurIPS*, 2021. (pp. 5 and 30)

[7] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment. *arXiv preprint*, 2021. (pp. 1 and 3)

[8] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint*, 2021. (p. 9)

[9] Mostafa D Awheda and Howard M Schwartz. Exponential moving average Q-learning algorithm. In *ADPRL*, 2013. (p. 5)

[10] Mostafa D Awheda and Howard M Schwartz. Exponential moving average based multiagent reinforcement learning algorithms. *Artificial Intelligence Review*, 2016. (p. 5)

[11] Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint*, 2023. (p. 3)

[12] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint*, 2021. (p. 3)

[13] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint*, 2023. (p. 1)

[14] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in ML*, 2015. (p. 5)

[15] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. (pp. 5 and 9)

[16] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *TMLR*, 2023. (pp. 1 and 11)

[17] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain generalization by seeking flat minima. In *NeurIPS*, 2021. (p. 10)

[18] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint*, 2021. (p. 9)

[19] Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint*, 2018. (pp. 3 and 10)

[20] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017. (pp. 1 and 3)

[21] Jack Clark and Dario Amodei. Faulty Reward Functions in the Wild. `https://openai.com/research/faulty-reward-functions`, 2016. (p. 1)

[22] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. (p. 9)

[23] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975. (p. 25)

[24] Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. ColD fusion: Collaborative descent for distributed multitask finetuning. In *ACL*, 2023. (pp. 6 and 10)

[25] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint*, 2023. (p. 1)

[26] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *TMLR*, 2023. (p. 3)

[27] Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint*, 2023. (pp. 2, 3, and 10)

[28] Steven Vander Eeckt et al. Weight averaging: A simple yet effective method to overcome catastrophic forgetting in automatic speech recognition. *arXiv preprint*, 2022. (pp. 6 and 10)

[29] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *ICLR*, 2022. (p. 10)

[30] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *ICML*, 2020. (pp. 2, 8, 9, and 26)

[31] Robert M French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 1992. (pp. 1 and 3)

[32] Tingchen Fu, Deng Cai, Lemao Liu, Shuming Shi, and Rui Yan. Disperse-then-merge: Pushing the limits of instruction tuning via alignment tax reduction. *arXiv preprint*, 2024. (p. 10)

[33] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *ICML*, 2023. (pp. 1, 2, 6, and 7)

[34] Jean-Baptiste Gaya, Laure Soulier, and Ludovic Denoyer. Learning a subspace of policies for online adaptation in reinforcement learning. In *ICLR*, 2022. (p. 10)

[35] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *ICML*, 2019. (pp. 2 and 5)

[36] Google Gemini Team. Gemini: A family of highly capable multimodal models. 2023. (p. 1)

[37] Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint*, 2024. (pp. 2 and 10)

[38] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint*, 2013. (p. 1)

[39] Alexey Gorbatovski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov. Learn your reference model for real good alignment. *arXiv preprint*, 2024. (pp. 2, 5, and 10)

[40] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 2020. (pp. 5 and 9)

[41] Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a region in weight space for fine-tuned language models. In *EMNLP*, 2023. (p. 9)

[42] Sil Hamilton. Detecting mode collapse in language models via narration. *arXiv preprint*, 2024. (pp. 1 and 34)

[43] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *TPAMI*, 1990. (p. 9)

[44] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. (pp. 5 and 9)

[45] Dan Hendrycks. Natural selection favors AIs over humans. *arXiv preprint*, 2023. (p. 11)

[46] Dan Hendrycks and Mantas Mazeika. X-risk analysis for AI research. *arXiv preprint*, 2022. (pp. 1 and 11)

[47] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint*, 2020. (pp. 7 and 9)

[48] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*, 2021. (p. 9)

[49] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPS*, 2015. (p. 5)

[50] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *NeurIPS*, 2022. (p. 11)

[51] Zitong Huang, Ze Chen, Bowen Dong, Chaoqi Liang, Erjin Zhou, and Wangmeng Zuo. Imwa: Iterative model weight averaging benefits class-imbalanced learning tasks. *arXiv preprint*, 2024. (p. 10)

[52] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *NeurIPS*, 2022. (pp. 2 and 10)

[53] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023. (pp. 2, 3, 5, 9, 10, 22, and 23)

[54] Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint*, 2018. (p. 10)

[55] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018. (pp. 2, 5, 9, 22, and 30)

[56] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. (p. 24)

[57] Samyak Jain, Sravanti Addepalli, Pawan Kumar Sahu, Priyam Dey, and R Venkatesh Babu. Dart: Diversify-aggregate-repeat training improves generalization of neural networks. In *CVPR*, 2023. (p. 10)

[58] Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. *arXiv preprint*, 2024. (pp. 6, 10, 23, 24, and 29)

[59] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *ICML*, 2017. (pp. 2, 4, and 22)

[60] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint*, 2023. (p. 9)

[61] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. (pp. 7, 9, and 10)

[62] Jean Kaddour. Stop wasting my time! saving days of ImageNet and BERT training with latest weight averaging. In *NeurIPS Workshop*, 2022. (p. 5)

[63] Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token fusion: Bridging the gap between token pruning and token merging. In *WACV*, 2024. (p. 10)

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. (p. 6)

[65] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *ICLR*, 2024. (pp. 1, 3, and 34)

[66] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. In *PNAS*, 2017. (p. 1)

[67] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*, 2022. (pp. 1 and 6)

[68] Maxime Labonne. Merge Large Language Models with mergekit, 2024. URL https://huggingface.co/blog/mlabonne/merge-models. (p. 2)

[69] Maxime Labonne. NeuralBeagle14-7B. https://huggingface.co/mlabonne/NeuralBeagle14-7B-GGUF, 2024. (p. 2)

[70] Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains. *arXiv preprint*, 2024. (p. 2)

[71] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. (p. 9)

[72] Nathan Lambert and Jacob Morrison. Model merging lessons in The Waifu Research Department, 2024. URL https://www.interconnects.ai/p/model-merging. (p. 2)

[73] Daniel Lawson and Ahmed H Qureshi. Merging decision transformers: Weight averaging for forming multi-task policies. In *ICLR RRL Workshop*, 2023. (p. 10)

[74] Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *ACL*, 2020. (p. 2)

[75] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. In *ICML*, 2024. (p. 3)

[76] Jaerin Lee, Bong Gyun Kang, Kihoon Kim, and Kyoung Mu Lee. Grokfast: Accelerated grokking by amplifying slow gradients. *arXiv preprint*, 2024. (p. 5)

[77] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint*, 2017. (p. 1)

[78] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint*, 2024. (p. 11)

[79] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017. (p. 1)

[80] Ziniu Li, Tian Xu, Yushun Zhang, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint*, 2023. (p. 3)

[81] 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. Mitigating the alignment tax of rlhf. *arXiv preprint*, 2024. (pp. 1, 2, 3, 6, and 10)

[82] Yong Lin, Lu Tan, Yifan Hao, Honam Wong, Hanze Dong, Weizhong Zhang, Yujiu Yang, and Tong Zhang. Spurious feature diversification improves out-of-distribution generalization. In *ICLR*, 2024. (p. 10)

[83] Tianlin Liu, Shangmin Guo, Leonardo Bianco, Daniele Calandriello, Quentin Berthet, Felipe Llinares, Jessica Hoffmann, Lucas Dixon, Michal Valko, and Mathieu Blondel. Decoding-time realignment of language models. In *ICML*, 2024. (p. 2)

[84] Yuchen Lu, Soumye Singhal, Florian Strub, Aaron Courville, and Olivier Pietquin. Countering language drift with seeded iterated learning. In *ICML*, 2020. (p. 2)

[85] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. (p. 10)

[86] Skander Moalla, Andrea Miele, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in ppo. *arXiv preprint*, 2024. (pp. 1 and 34)

[87] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *TMLR*, 2024. (p. 5)

[88] Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, et al. Nash learning from human feedback. *arXiv preprint*, 2023. (pp. 2, 5, and 10)

[89] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *NeurIPS*, 2020. (pp. 2, 9, 24, and 29)

[90] Richard Ngo, Lawrence Chan, and Soren Mindermann. The alignment problem from a deep learning perspective. *arXiv preprint*, 2022. (pp. 1 and 11)

[91] Evgenii Nikishin, Pavel Izmailov, Ben Athiwaratkun, Dmitrii Podoprikhin, Timur Garipov, Pavel Shvechikov, Dmitry Vetrov, and Andrew Gordon Wilson. Improving stability in deep reinforcement learning with weight averaging. In *UDL*, 2018. (p. 10)

[92] Michael Noukhovitch, Samuel Lavoie, Florian Strub, and Aaron Courville. Language model alignment with elastic reset. In *NeurIPS*, 2023. (pp. 2 and 10)

[93] OpenAI. Gpt-4 technical report. 2023. (p. 1)

[94] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. (p. 3)

[95] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. (pp. 5 and 9)

[96] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *NeurIPS*, 2023. (p. 24)

[97] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022. (pp. 1 and 3)

[98] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *ICLR*, 2022. (p. 1)

[99] Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint*, 2022. (p. 1)

[100] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM*, 1992. (pp. 3, 4, and 9)

[101] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. (p. 1)

[102] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. (p. 1)

[103] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint*, 2023. (p. 3)

[104] Colin Raffel. Building Machine Learning Models Like Open Source Software. *ACM*, 2023. (pp. 2, 3, and 10)

[105] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020. (p. 1)

[106] Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In *NeurIPS*, 2022. (pp. 2, 5, 10, 24, 30, and 33)

[107] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *ICML*, 2023. (pp. 5, 6, and 9)

[108] Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. WARM: On the benefits of weight averaged reward models. In *ICML*, 2024. (pp. 2, 3, 6, and 10)

[109] Alexandre Ramé, Guillaume Couairon, Mustafa Shukor, Corentin Dancette, Jean-Baptiste Gaya, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In *NeurIPS*, 2023. (pp. 2, 10, and 33)

[110] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint*, 2024. (pp. 1 and 9)

[111] Mark Rofin, Nikita Balagansky, and Daniil Gavrilov. Linear interpolation in parameter space is good enough for fine-tuned language models. *arXiv preprint*, 2022. (p. 2)

[112] Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Léonard Hussenot, Orgad Keller, et al. Factually consistent summarization via reinforcement learning with textual entailment feedback. In *ACL*, 2023. (pp. 3 and 4)

[113] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015. (pp. 4 and 9)

[114] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. (pp. 3 and 9)

[115] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *ACL*, 2020. (p. 34)

[116] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint*, 2023. (p. 1)

[117] Wei Shen, Rui Zheng, Wenyu Zhan, Jun Zhao, Shihan Dou, Tao Gui, Qi Zhang, and Xuanjing Huang. Loose lips sink ships: Mitigating length bias in reinforcement learning from human feedback. In *ACL*, 2023. (p. 33)

[118] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH*, 1985. (pp. 2, 3, 5, 10, and 22)

[119] Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in rlhf. *arXiv preprint*, 2023. (pp. 1 and 33)

[120] Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *NeurIPS*, 2022. (pp. 1 and 3)

[121] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. (pp. 5 and 9)

[122] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *NeurIPS*, 2020. (p. 1)

[123] Zafir Stojanovski, Karsten Roth, and Zeynep Akata. Momentum-based weight interpolation of strong zero-shot models for continual learning. In *NeurIPS Workshop*, 2022. (pp. 2, 5, 6, and 10)

[124] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint*, 2022. (p. 9)

[125] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. (pp. 5 and 9)

[126] Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *arXiv preprint*, 2024. (p. 3)

[127] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *NeurIPS*, 2017. (pp. 3, 4, 5, 9, 10, and 31)

[128] Jessica Taylor, Eliezer Yudkowsky, Patrick LaVictoire, and Andrew Critch. Alignment for advanced machine learning systems. *Ethics of AI*, 2016. (pp. 1 and 11)

[129] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint*, 2024. (pp. 1, 3, 6, and 9)

[130] Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint*, 2020. (p. 5)

[131] Joachim Utans. Weight averaging for neural networks and local resampling schemes. In *AAAI*, 1996. (pp. 2, 5, 9, and 23)

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. (p. 3)

[133] Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. *arXiv preprint*, 2024. (p. 10)

[134] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint*, 2024. (p. 11)

[135] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022. (p. 1)

[136] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, 1992. (pp. 2, 3, 6, and 22)

[137] 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. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022. (pp. 2, 5, 9, 10, 23, and 24)

[138] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Hanna Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *CVPR*, 2022. (pp. 2, 3, 6, 22, and 24)

[139] Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. LM-cocktail: Resilient tuning of language models via model merging. *arXiv preprint*, 2023. (p. 10)

[140] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *NeurIPS*, 2023. (p. 9)

[141] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint*, 2023. (p. 9)

[142] Kerem Zaman, Leshem Choshen, and Shashank Srivastava. Fuse to forget: Bias reduction and selective memorization through model fusion. *arXiv preprint*, 2023. (p. 10)

[143] Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Weak-to-strong extrapolation expedites alignment. *arXiv preprint*, 2024. (pp. 27 and 28)

[144] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS*, 2023. (p. 9)

[145] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint*, 2019. (p. 1)

WARP: On the Benefits of Weight Averaged Rewarded Policies

Supplementary material

This supplementary material is organized as follows:

- Appendix A provides additional illustration of the *WARP* procedure.
- Appendix B details theoretical insights on task vectors, *SLERP*, *LERP* and *LITI*.
- Appendix C details empirical insights on task vectors, *SLERP*, *LERP* and *LITI*.
- Appendix D shows the impact of different design choices in *WARP*.
- Appendix E investigates a potential length bias in *WARP*, and how to fix it.
- Appendix F explores the relationship between KL and diversity in generations.

A. Strategy illustration

In Figure 5, we propose an alternative illustration of *WARP*, where the different stages are more detailed than in Figure 1(a). Then in Figure 6, we also refine our illustration showcasing the similarity and difference between *SLERP* and *LERP*.

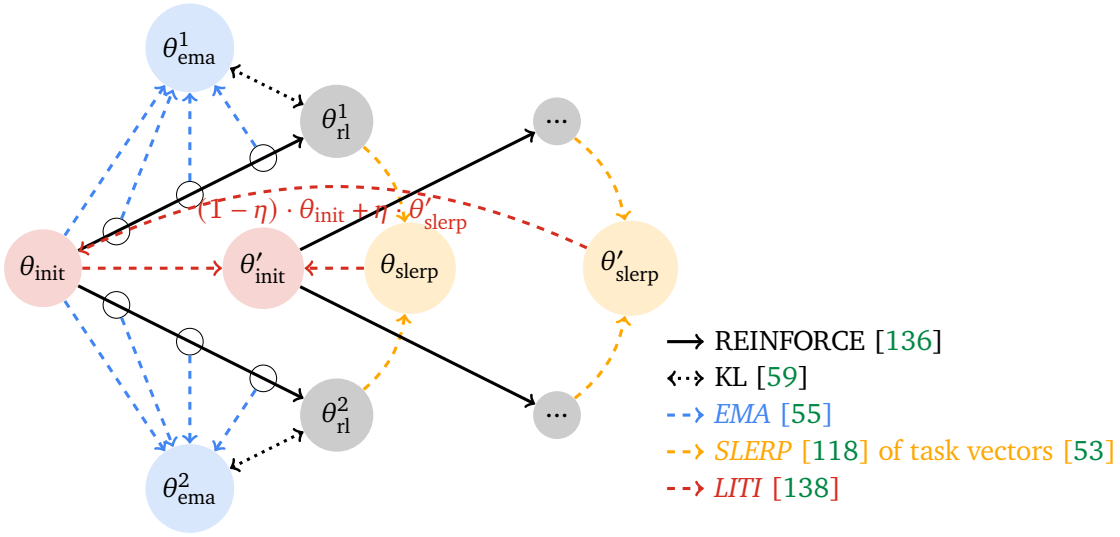


Figure 5 | **Detailed illustration of the *WARP* strategy**. From a (pre-trained and supervised fine-tuned) LLM θ_{init}, we launch $M = 2$ fine-tunings (black arrows ⟶). The innovation of *WARP* lies in the use of model merging by weight averaging at three different stages. First, the exponential moving averages (*EMA*, blue dashed arrows - - ›) of the policy (collected at different training steps) serves as the anchor for the KL regularization (black double-headed dotted arrows ‹··›). The fine-tuned networks are weight averaged using spherical linear interpolation of task vectors (*SLERP*, yellow dashed arrows - - ›). Third, we interpolate towards the initialization (*LITI*, red dashed arrows - - ›). This obtained model θ'_{init} serves as an updated initialization for the next iteration, progressively refining the model's capabilities and alignment. Overall, the final model θ'_{slerp} has high reward but also high KL. Then, by interpolation towards the SFT init, we reveal a KL-reward Pareto front of solutions: $\{(1-\eta)\cdot\theta_{\text{sft}}+\eta\cdot\theta^I_{\text{slerp}} \mid 0\leq\eta\leq 1\}$.

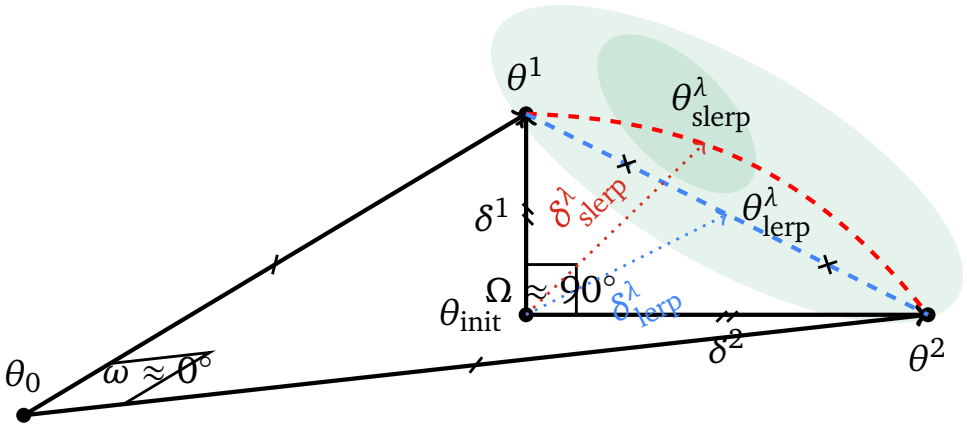


Figure 6 | Illustration of the difference between the full weights θ^m and their task vectors $\delta^m = \theta^m - \theta_{\text{init}}$, where darker areas are of better performance. We found in Appendix C.2 that $\Omega \approx 90°$ where Ω is the angle between task vectors such as $\cos\Omega = \frac{\delta^1 \cdot \delta_2}{\|\delta^1\|\|\delta^2\|}$, while ω the angle between the full weights such as $\cos\omega = \frac{\theta^1 \cdot \theta_2}{\|\theta^1\|\|\theta^2\|}$ satisfies $\omega \approx 0°$.

B. Theoretical insights on task vectors, *SLERP*, *LERP* and *LITI*

Based on the insights from [53] that task vectors (the differences between a fine-tuned model and its initialization) are semantically manipulable and interpretable units in the weight space, we compare *SLERP* and *LERP* merging operations by analyzing their task vectors.

Background. Linear interpolation (*LERP*) [131] is the simplest merging strategy, notably used in the model soups variants [137], and defined as:

$$\text{lerp}\left(\theta^1, \theta^2, \lambda\right) = (1-\lambda)\cdot\theta^1 + \lambda\cdot\theta^2. \tag{*LERP*}$$

Then, as illustrated in Figure 6, the task vector for *LERP* with interpolating coefficient λ is given by: $\delta^\lambda_{\text{lerp}} = \text{lerp}(\theta^1, \theta^2, \lambda) - \theta_{\text{init}} = (1-\lambda)\cdot\delta^1 + \lambda\cdot\delta^2$. Similarly, we define $\delta^\lambda_{\text{slerp}} = \text{slerp}(\theta_{\text{init}}, \theta^1, \theta^2, \lambda) - \theta_{\text{init}}$ where slerp is defined in Equation (*SLERP*).

B.1. Theoretical insights on the *SLERP* and *LERP* task vectors

We denote Ω the angle between the the task vectors δ^1 and δ^2:

$$\cos\Omega = \frac{\delta^1 \cdot \delta_2}{\|\delta^1\|\|\delta^2\|}. \tag{2}$$

Based on the empirical observations from [58], confirmed in our Figure 11(c), we introduce the following Assumption 1 for simplicity.

Assumption 1 (Task vectors of equal norm)**.** *Independently fine-tuned task vectors have a same norm l:*

$$\|\delta^1\| = \|\delta^2\| = l. \tag{3}$$

Lemma 1 (*SLERP* task vector)**.** *Under Assumption 1,* SLERP *preserves the norm of the task vector:*

$$\|\delta^\lambda_{\text{slerp}}\| = l. \tag{4}$$

Proof. By definition,

$$\delta^{\lambda}_{\text{slerp}} = \frac{\sin[(1-\lambda)\Omega]}{\sin\Omega} \cdot \delta^1 + \frac{\sin[\lambda\Omega]}{\sin\Omega} \cdot \delta^2 \tag{5}$$

Then, as $\delta^1 \cdot \delta^2 = l^2 \cos\Omega$,

$$\frac{\|\delta^{\lambda}_{\text{slerp}}\|^2}{l^2} = \left(\frac{\sin[(1-\lambda)\Omega]}{\sin\Omega}\right)^2 + 2\frac{\sin[(1-\lambda)\Omega]}{\sin\Omega}\frac{\sin[\lambda\Omega]}{\sin\Omega}\cos(\Omega) + \left(\frac{\sin[\lambda\Omega]}{\sin\Omega}\right)^2 \tag{6}$$

$$= \frac{\sin^2[(1-\lambda)\Omega] + 2\sin[(1-\lambda)\Omega]\sin[\lambda\Omega]\cos(\Omega) + \sin^2[\lambda\Omega]}{\sin^2\Omega} \tag{7}$$

$$= \frac{\sin^2\Omega}{\sin^2\Omega} \tag{8}$$

$$= 1 \tag{9}$$

using trigonometric identities, proving Lemma 1. □

Lemma 2 (*LERP* task vector)**.** *Under Assumption 1,* LERP *reduces the norm of the task vector:*

$$\|\delta^{\lambda}_{\text{lerp}}\| = l\sqrt{1 - 2(1-\cos\Omega)(\lambda - \lambda^2)}. \tag{10}$$

We recover that averaging weights with $\lambda = 0.5$ tends to reduce the norm of the task vectors, as previously highlighted in [58].

Proof. By definition:

$$\delta^{\lambda}_{\text{lerp}} = (1-\lambda)\cdot\delta^1 + \lambda\cdot\delta^2. \tag{11}$$

Then, as $\delta^1 \cdot \delta^2 = l^2\cos\Omega$,

$$\frac{\|\delta^{\lambda}_{\text{slerp}}\|^2}{l^2} = (1-\lambda)^2 + 2\lambda(1-\lambda)\cos\Omega + \lambda^2 \tag{12}$$

$$= 1 - 2\lambda(1-\cos\Omega) + 2\lambda^2(1-\cos\Omega) \tag{13}$$

$$= 1 - 2(1-\cos\Omega)(\lambda - \lambda^2), \tag{14}$$

proving Lemma 2 when $0 < \lambda < 1$. □

B.2. Theoretical insights on the KL

B.2.1. Linear regime

Assumption 2 (Linear regime [138])**.** *We assume that the predictions of a model f, with weights initialized from θ_0 and fine-tuned into θ, can be approximated by first-order Taylor expansion:* $\forall \boldsymbol{x}$,

$$f(\boldsymbol{x},\theta) \approx f(\boldsymbol{x},\theta_0) + (\theta-\theta_0)\cdot\nabla_\theta f(\boldsymbol{x},\theta_0). \tag{15}$$

Assumption 2 defines a neural tangent [56] space in which the relationship between weights and functions is linear. As previously argued in [106, 137], this Taylor expansion is reasonable partly because weights remain close during fine-tunings [89], as confirmed in Figure 11 where they have equal norms and a cosine of one. Yet, please note that [96] highlighted some limitations.

B.2.2. KL *variations for* LERP

We consider θ^1 and θ^2 weights fine-tuned from a shared SFT initialization θ_{sft}. Then in the linear regime from Assumption 2, weight and prediction ensembling behaves similarly:

$$f\big(\boldsymbol{x}, (1-\lambda)\cdot\theta^1 + \lambda\cdot\theta^2\big) \approx (1-\lambda)\cdot f(\boldsymbol{x},\theta^1) + \lambda\cdot f(\boldsymbol{x},\theta^2). \tag{16}$$

This similarity enables to prove the following Lemma 3.

Lemma 3 (LERP reduces KL)**.** *For an interpolating coefficient $0 \le \lambda \le 1$, denoting π_λ the* LERP *policy from weight interpolation $(1-\lambda)\cdot\theta^1 + \lambda\cdot\theta^2$, and $\hat{\pi}_\lambda$ the ensembling policy from prediction interpolation $(1-\lambda)\cdot\pi_{\theta^1} + \lambda\cdot\pi_{\theta^2}$, then under Assumption 2,*

$$\text{KL}(\pi_\lambda||\pi_{\theta_{\text{sft}}}) \approx \text{KL}(\hat{\pi}_\lambda||\pi_{\theta_{\text{sft}}}) \le (1-\lambda)\text{KL}(\pi_{\theta^1}||\pi_{\theta_{\text{sft}}}) + \lambda\text{KL}(\pi_{\theta^2}||\pi_{\theta_{\text{sft}}}), \tag{17}$$

i.e., the KL *for* LERP *is lower than the interpolated* KL*.*

Proof. The following proof applies the linear assumption and properties of the KL divergence.

Approximation of KL. The first approximate equality is a direct application of Assumption 2 to π_λ. Precisely, applying Equation (16) to the definition of $\pi_\lambda = \pi_{(1-\lambda)\theta_1+\lambda\theta_2}$ yields that $\pi_\lambda \approx \hat{\pi}_\lambda$.

Upper bound of the KL. The KL divergence is convex in both its arguments [23], thus we directly have that

$$\text{KL}((1-\lambda)\cdot\pi_{\theta^1} + \lambda\cdot\pi_{\theta^2}||\pi_{\theta_{\text{sft}}}) \le (1-\lambda)\text{KL}(\pi_{\theta^1}||\pi_{\theta_{\text{sft}}}) + \lambda\text{KL}(\pi_{\theta^2}||\pi_{\theta_{\text{sft}}}), \tag{18}$$

which completes the proof.

□

Remark 1. *Lemma 3 shows that the LERP π_λ is closer in* KL *to the original SFT initialization. This relates to Lemma 2, where we show that the linear interpolation reduces the norm to the initialization. As the interpolation brings the weights of the models closer, it is natural that it would also bring the resulting policies closer.*

B.2.3. KL *and reward variation for* LITI

We now consider a given weight θ (in practice either obtained from *LERP* or *SLERP* of multiple fine-tuned weights) and its associated task vector $\delta = \theta - \theta_{\text{sft}}$. In the linear regime from Assumption 2, for each $\eta \in [0,1]$, we have the following:

$$f(\boldsymbol{x}, \theta_{\text{sft}} + \eta\cdot\delta) - f(\boldsymbol{x},\theta_{\text{sft}}) \approx \eta\cdot(f(\boldsymbol{x},\theta_{\text{sft}}+\delta) - f(\boldsymbol{x},\theta_{\text{sft}})). \tag{19}$$

We try to show that:

$$\text{KL}\big(\pi_{\theta_{\text{sft}}+\eta\cdot\delta}\|\pi_{\theta_{\text{sft}}}\big) \le \eta\cdot\text{KL}\big(\pi_{\theta_{\text{sft}}+\delta}\|\pi_{\theta_{\text{sft}}}\big). \tag{20}$$

Lemma 4 (KL upper bound for interpolated distributions)**.** *For an interpolating coefficient $0 \le \eta \le 1$, denoting π_η the* LITI *policy from weight interpolation $\theta_{\text{sft}} + \eta\cdot\delta$, and $\hat{\pi}_\eta$ the ensembling policy from prediction interpolation $(1-\eta)\cdot\pi_{\theta_{\text{sft}}} + \eta\cdot\pi_{\theta_{\text{sft}}+\delta}$, then under Assumption 2,*

For each $\eta \in [0,1]$, we have that

$$\text{KL}\big(\pi_\eta\|\pi_{\theta_{\text{sft}}}\big) \approx \text{KL}(\hat{\pi}_\eta\|\pi_{\theta_{\text{sft}}}) \le \eta\text{KL}\big(\pi_{\theta_{\text{sft}}+\delta}\|\pi_{\theta_{\text{sft}}}\big). \tag{21}$$

Proof. The following proof uses the same method as the one of Lemma 3. We use Assumption 2 to link the policy with the interpolation of polices, and the inequality is a result of the KL convexity.

Approximation of KL. The first approximate equality is a direct application of Assumption 2 to π_η. Precisely, applying Equation (19) to the definition of $\pi_\eta = \pi_{\theta_{\text{sft}}+\eta\cdot\delta}$ yields that $\pi_\eta \approx \hat{\pi}_\eta$.

Upper bound of the KL. Using the fact that the KL is convex, we have

$$\text{KL}(\eta \cdot \pi_{\theta_{\text{sft}}+\delta} + (1-\eta)\cdot \pi_{\theta_{\text{sft}}} || \pi_{\theta_{\text{sft}}}) \leq \eta \text{KL}\big(\pi_{\theta_{\text{sft}}+\delta} \| \pi_{\theta_{\text{sft}}}\big). \tag{22}$$

□

Assumption 3 (LITI reward is above the expected reward)**.** *The rewards for the* LITI *interpolated weights are above the interpolated rewards:*

$$r(\pi_0 + \eta \cdot (\pi - \pi_{\theta_{\text{sft}}})) \geq \eta r(\pi) + (1-\eta) r(\pi_{\theta_{\text{sft}}}), \tag{23}$$

This Assumption 3 is based on observations from Figure 9(b), and extends to a reward maximization setup the notion of linear mode connectivity [30], usually defined w.r.t. the accuracy in supervised learning.

Lemma 5 (LITI Pareto optimality)**.** *Be given the convexity of the* KL *from Lemma 4 and the concavity of the reward r in Assumption 3, then the reward vs.* KL *front of* LITI *is above the diagonal. Illustration in Figure 7.*

Proof. We obtain a policy π_θ fine-tuned from $\pi_{\theta_{\text{sft}}}$. The *LITI* policy for $\theta_\eta = (1-\eta)\cdot\theta_{\text{sft}} + \eta\cdot\theta$ is noted π_η. Combining the approximation from Lemma 4 and Assumption 3, we have that

$$r(\pi_\eta) \geq (1-\eta) r(\pi_{\theta_{\text{sft}}}) + \eta r(\pi_\theta). \tag{24}$$

And, from Lemma 4, we also have that

$$\text{KL}\big(\pi_\eta \| \pi_{\theta_{\text{sft}}}\big) \leq \eta \text{KL}\big(\pi_\theta \| \pi_{\theta_{\text{sft}}}\big). \tag{25}$$

This means that for every *LITI* coefficient η, the *LITI* policy has a higher reward than the interpolated reward at a lower KL. Geometrically, this means that each point on the Reward-KL front from LITI is on the top left quadrant of the plane according to the corresponding point on the diagonal. □

B.3. Uniformly averaging $M > 2$ weights with *SLERP*

The *SLERP* merging formula from Equation (*SLERP*) is only defined for $M = 2$ weights. We trivially (and certainly suboptimally) generalize this to $M > 2$ weights in the uniform averaging setup, thus giving an equal coefficient to each of them, i.e., $\lambda = \frac{1}{M}$. In that setup, removing the dependency of θ_{init} that is assumed shared, we generalize *SLERP* to merge M weights uniformly through the iterative procedure defined below:

$$\text{slerpm}\big(\{\theta^m\}_{m=1}^M\big) = \text{slerp}\left(\text{slerpm}\left(\{\theta^m\}_{m=1}^{M-1}\right), \theta^M, \lambda = \frac{1}{M}\right). \tag{26}$$

Though these operations are not associative, the standard deviations in performances are small, as indicated by the shaded areas in Figure 4(b).

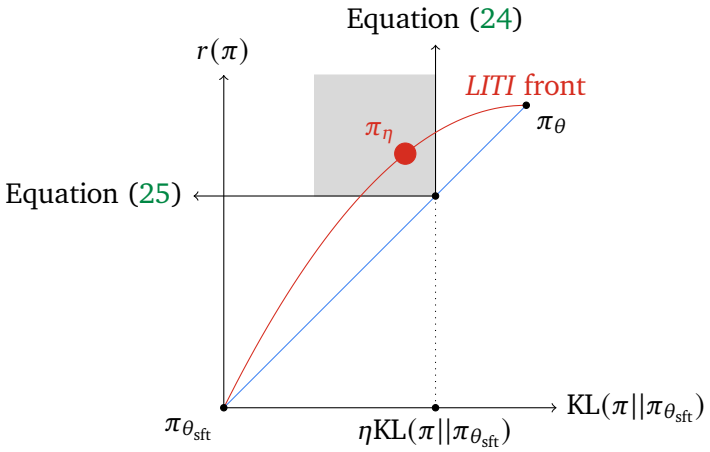


Figure 7 | Illustration of Lemma 5. Based on experimental observation and theoretical insights, we see that the Pareto front of the *LITI* policy is better than the identity. It highlights how Equations (24) and (25) place *LITI* policies on the KL-reward plane.

C. Empirical insights on task vectors, *SLERP*, *LERP* and *LITI*

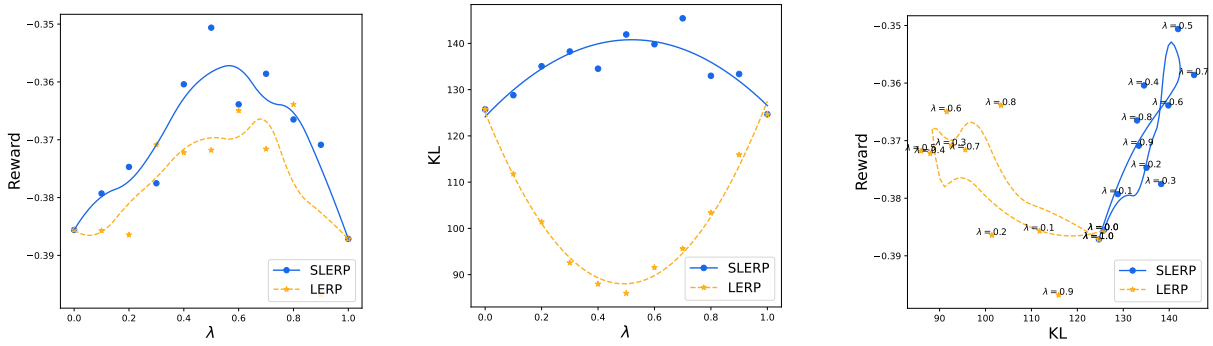
C.1. Empirical insights on the difference between *SLERP* and *LERP*

We now empirically investigate how those theoretical differences between *SLERP* and *LERP* affect the performance of the merged policies.

***SLERP* vs. *LERP*.** In Figure 8 we adjust the interpolating coefficient λ, highlighting distinct behaviors for *SLERP* and *LERP*. *SLERP* consistently enhances rewards more than *LERP*, as depicted in Figures 3(c) and 8(a). However, a comprehensive evaluation must consider both KL and reward. As shown in Figure 8(b), *LERP* consistently reduces KL, corroborating with Lemma 2 that *LERP* reduces the norm of updates (while *SLERP* preserves it). When plotting these metrics together in Figure 8(c), we observe that *SLERP* and *LERP* target different regions on the Pareto front: *SLERP* achieves higher rewards at the expense of increased KL, while the main impact of *LERP* is to lower KL. This is consistent with Lemmas 2 and 3, be given the orthogonal angles between task vectors $\Omega \approx 90°$ (as shown in Figure 11(a)).

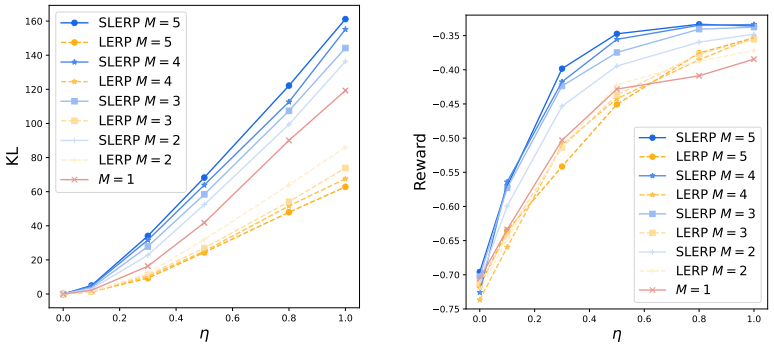
Combining *SLERP* and *LERP* with *LITI*. We also compare the behaviours of *SLERP* and *LERP* when we apply *LITI*, as we adjust the interpolating coefficient η. Figure 9(a) and Figure 9(b) validate that KL is convexe with regard to η while the reward is concave with regard to η, for different values of M. This is also highlighted in Figure 10(a), which reproduces the results from Figure 4(b) (and maintaining the same axis limits), replacing *SLERP* by *LERP*: this leads to critical changes in the Pareto fronts. Inded, increasing M now tends to decrease KL for *LERP*, while it used to increase reward with *SLERP*. In Figure 10(b), we explore the extrapolation strategies from [143], using $0 \leq \eta \leq 2$ to compare the full extrapolated fronts from *LERP* and *SLERP*. While both perform similarly on low KL, our results suggest that *SLERP* perform better in high KL regions.

Conclusion. *SLERP* demonstrates some key advantages. In particular, it reveals the full Pareto front of solutions, while *LERP* only exposes a portion; extrapolation Figure 10(b) with $\eta > 1$ can partially mitigate this but as our experiments suggest, *LERP* curves consistently lag behind *SLERP* curves in high-reward regions. Moreover, from a practical perspective, *SLERP* scales the choice of η effectively, where 1 represents full updates and a fixed value of 0.3 always corresponds to the same operational region, optimizing for high reward and KL.



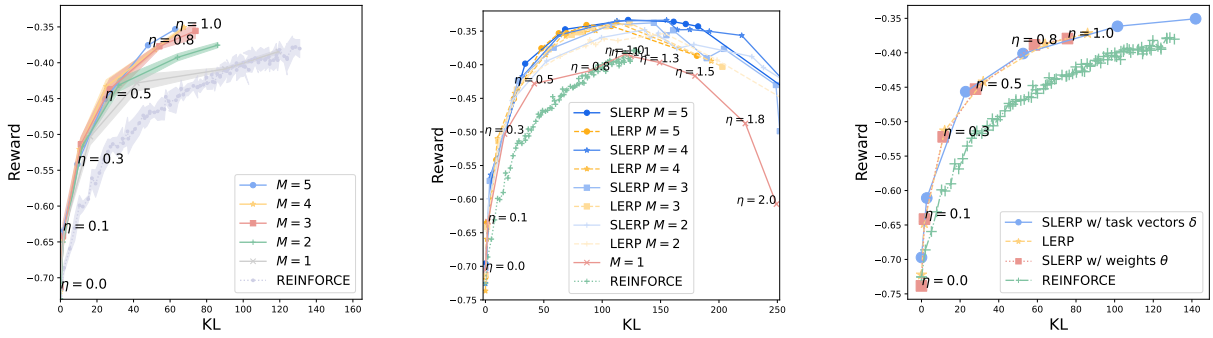
(a) Reward vs. λ. (b) KL vs. λ. (c) Reward vs. KL.

Figure 8 | ***SLERP* vs. *LERP* when sliding the interpolating coefficient λ.** Considering $M = 2$ weights after $T = 9k$ RL steps, we merge them using either *SLERP* or *LERP*, while sliding the interpolating coefficient λ between 0 and 1. We then evaluate the merged checkpoints. Figure 8(a) shows that *SLERP* leads to higher reward than *LERP*, as previously in Figure 3(c). Figure 8(b) shows that *LERP* signicantly reduces the KL (consistently with Lemma 3) while *SLERP* slightly increases it. Figure 8(c) shows how this impact the KL-reward Pareto front, where larger markers/darker colors indicate higher values of λ; while *SLERP* covers high KL-high reward regions, *LERP* tends to cover regions of lower KL and thus also lower rewards.



(a) KL for η. (b) Reward for η.

Figure 9 | ***SLERP* vs. *LERP* when sliding the interpolating coefficient η of *LITI*.** In Figure 9(a) we show that the KL is convex (and almost linear) with regard to η, consistently with Lemma 4. In contrast, Figure 9(b) shows that the reward is concave, validating Assumption 3.



(a) *LITI* of *LERP* of M weights. (b) Extrapolation with $0 \leq \eta \leq 2$. (c) *SLERP* w/o task vectors.

Figure 10 | ***SLERP* vs. *LERP* when sliding the interpolating coefficient η of *LITI*.** Figure 10(a) merges M policies with *LERP* and $\lambda = \frac{1}{M}$ (the endpoints on the top right of the solid lines), and then interpolates towards their SFT init, where light-colored areas show standard deviations across 5 experiments, and with $0 \leq \eta \leq 1$. In contrast, in Figure 10(b) we investigate extrapolation [143], using $0 \leq \eta \leq 2$ enabling to compare the full fronts of solutions with *LERP* and *SLERP*. Finally, Figure 10(c) confirms that applying *SLERP* on the full weights θ rather than on the task vectors δ perform very similarly to *LERP*.

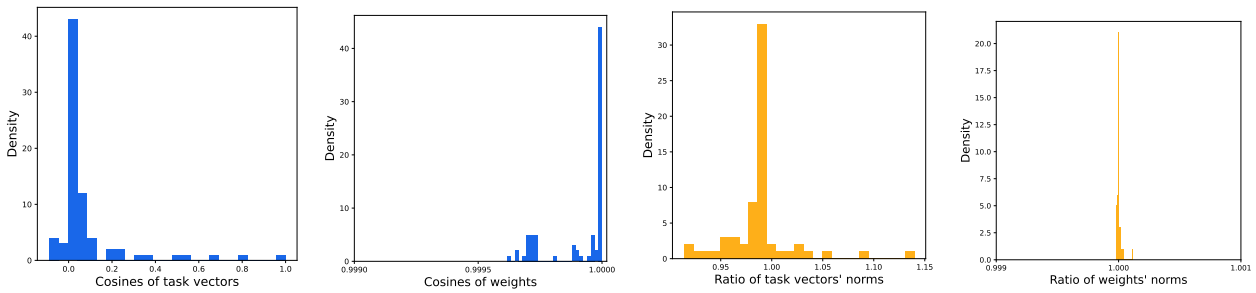
C.2. Empirical insights on the role of task vectors

We now explore the effectiveness of applying *SLERP* on task vectors δ vs. full weights θ, as illustrated in Figure 6. To this end, in Figure 11 we draw inspiration from [58] and plot the angles Ω and ω and norms of δ and θ.

Angles of task vectors $\Omega \approx 90°$. Figure 11(a) shows that the task vectors are typically orthogonal ($\Omega \approx 90°$), highlighting the diverse trajectories of the different RL fine-tunings. This is in contrast with [58] for supervised fine-tunings, where Ω typically range between 40° and 80°. We suspect that this is related to the underlying differences between reinforcement and supervised learning; in RL the policies are trained on their own generations, creating more orthogonal task vectors, whereas in supervised learning the LLM try to imitate the groundtruth labels, leading to more similar task vectors. The orthogonality of our task vectors prevents the use of the update rule $\eta \to \frac{2\cos\Omega}{1+\cos\Omega}$ suggested from Eq. 2 in [58], as it would lead to $\eta \approx 0$, deleting any potential update.

Angles of full weights $\omega \approx 0°$. In contrast, Figure 11(b) show that full weights remain collinear ($\omega \approx 0°$). This explains the empirical results from Figure 10(c), where applying *SLERP* directly to full weights results in behaviors similar to *LERP*. Indeed, as the angles $\omega \approx 0°$, spherical interpolation effect is minimal because $\sin(x) \approx x + O(x^3)$, and thus $\frac{\sin[\lambda\omega]}{\sin\omega} \approx \frac{\lambda\omega}{\omega} \approx \lambda$.

Norms consistency. Figure 11(c) confirms the consistency in the norms of different task vectors, supporting our Assumption 1. This uniformity is aligned with previous research [58]. As a side note, this consistency extends to full weights θ, confirming that fine-tuning typically results in minimal changes to the overall weight [89].



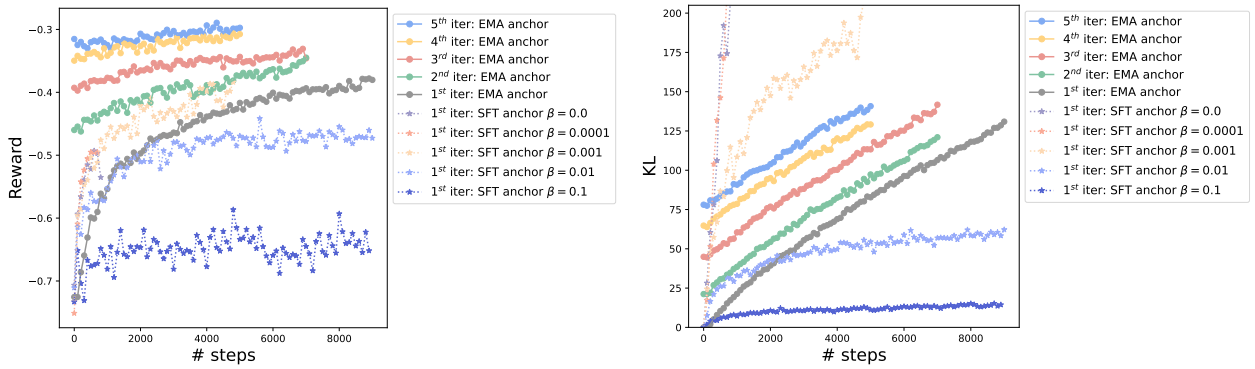
(a) $\frac{\delta^1 \cdot \delta^2}{\|\delta^1\|\|\delta^2\|}$. (b) $\frac{\theta^1 \cdot \theta^2}{\|\theta^1\|\|\theta^2\|}$. (c) $\frac{\|\delta^1\|}{\|\delta^2\|}$. (d) $\frac{\|\theta^1\|}{\|\theta^2\|}$.

Figure 11 | **Angles and norms of (full) weights θ^m and their task vectors $\delta^m = \theta^m - \theta_{\text{init}}$.** The histograms represent data across the 28 layers of the Gemma "7B" architecture. Figure 11(a) plots the histograms of task vector cosines. Figure 11(b) plots the histograms of weights cosines. Figure 11(c) plots the histograms of task vector norms ratio. Figure 11(d) plots the histograms of weights norms ratio.

D. Empirical investigation of several design choices

We include several experiments showcasing the robustness of *WARP* to different design choices, while further demonstrating its superiority in terms of KL-reward Pareto optimality. Specifically, Appendix D.1 analyzes the performances along training at different steps T; Appendix D.2 provides results with different values for the hyperparameters μ and β; Appendix D.3 shows the impact of the update rate η to provide an improved initialization for the 2$^{\text{nd}}$ iteration of *WARP*; finally, Appendix D.4 shows that in iterative *WARP*, interpolating towards the episode initialization or the SFT initialization both perform similarly.

D.1. Analyzing the number of training steps



(a) Reward along training.

(b) KL along training.

Figure 12 | **Rewards and** KL **at different number of training steps** T. Figures 12(a) and 12(b) complement Figure 3(b) and Figure 4(c), this time plotting rewards and KL separately as a function of the number of training steps T. Regarding iterative *WARP*, we observe that each iteration has higher rewards but also higher KL (by starting at training step 0 from a new initialization). Regarding the baseline (REINFORCE with SFT anchor), we observe that low values of β lead to very fast hacking of the reward, as visible by the KL exploding, while high values of β slow down the learning procedure.

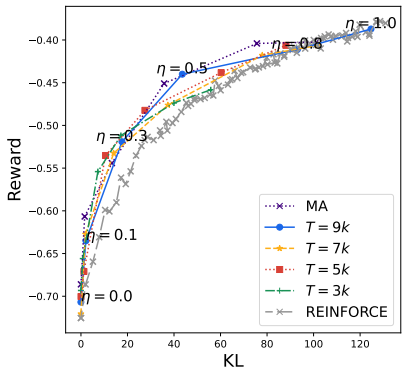
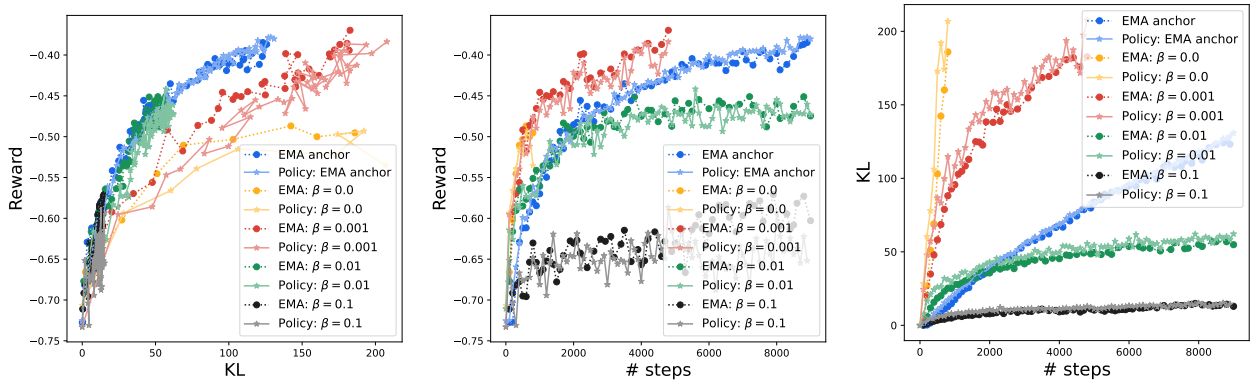


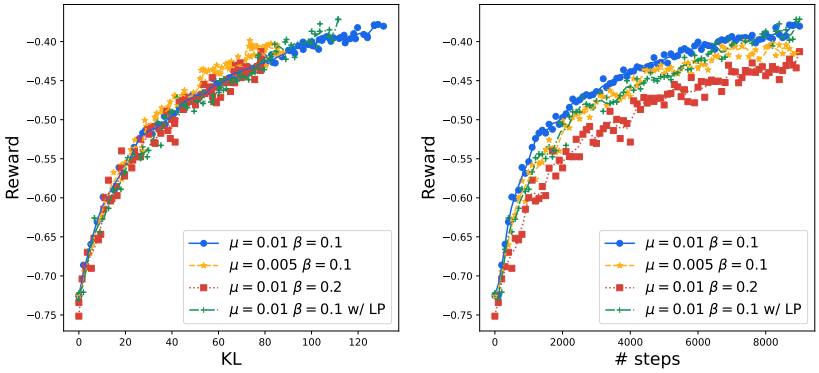
Figure 13 | ***LITI* with** $M = 1$ **at different number of training steps** T. The reward gain is significantly reduced compared to Figure 4(a) where we first merged $M = 2$ policies before applying *LITI*. We also try to perform moving average (MA) [6, 55] before applying *LITI*, averaging multiple checkpoints collected along a single RL fine-tuning at steps $\{6k, 7k, 8k, 9k\}$; this does not improve performances, suggesting the need to merge weights from independent fine-tunings to have enough diversity [106].

D.2. Analyzing the values of μ and β



(a) Reward vs. KL. (b) Reward vs. steps. (c) KL vs. steps.

Figure 14 | ***EMA* vs. their base policies**, complementing the results from Figures 3(a) and 3(b). We confirm in Figure 14(a) that the *EMA* of all variants (with SFT anchor) perform similarly or better than their base policies in terms of KL-reward Pareto optimality. As a reminder, we perform evaluation every 100 steps, and train them for $T = 9k$ steps, though we stopped the trainings if the base policy ever reaches a KL of 200. This confirms Observation 1; the benefits of our variant with *EMA* anchor is partly explained by distillation from an improved mean teacher [127].



(a) Reward vs. KL. (b) Reward vs. steps.

Figure 15 | **Experiments ablating the values for the *EMA* update rate μ and the** KL **regularization strength β.** So far we have systematically used $\mu = 0.01$ and $\beta = 0.1$ for all *EMA*-based runs, including in the iterative *WARP*. These hyperparameters were chosen at the project's onset and have remained unchanged. In Figures 15(a) and 15(b) we increase regularization with $\mu = 0.005$ and $\beta = 0.2$. Our results indicate that reducing μ or increasing β behaves similarly, marginally improving the KL-reward Pareto front but slowing down training. Additionally, we include the training trajectory when using a length penalty (LP), as detailed in Appendix E.

D.3. Analyzing the values of η

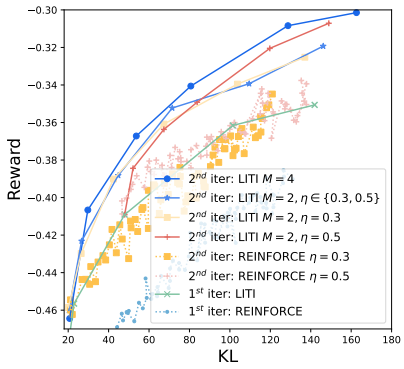


Figure 16 | **Experiments ablating the *LITI* update rate** η. As we initiate the 2$^{\text{nd}}$ iteration of *WARP*, selecting an appropriate value for η is key, as it determines the starting point θ^{η} and functions similarly to an outer learning rate (see Section 6). We usually set $\eta = 0.3$, but now provide results with an increased $\eta = 0.5$, starting the 2$^{\text{nd}}$ iteration from a more "advanced" position on the previous Pareto front. We run and average $M = 2$ fine-tunings from each of those two initializations for $T = 7k$ steps, before applying *LITI*. Our results indicate that a higher η (0.5) performs better in regions of high KL, whereas a lower η (0.3) helps in regions with KL below 65. This suggests that the optimal choice for η is compute-dependent; a lower η is appropriate if further iterations can explore high KL regions, whereas a limited compute budget might benefit from a higher η. This resembles the learning rate trade-off in optimization, where lower rates improve results but require more training steps. As a final note, we can also use different η for the different fine-tunings; notably, we observe that merging all those $M = 4$ RLs perform better (though it doubles the compute).

D.4. Interpolate towards the initialization? or towards the SFT?

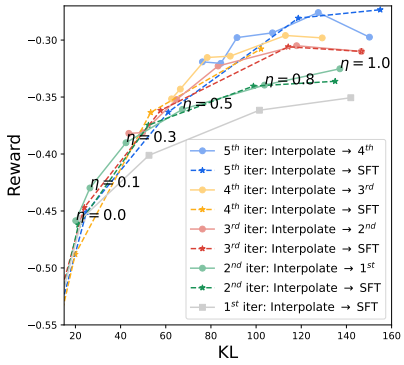
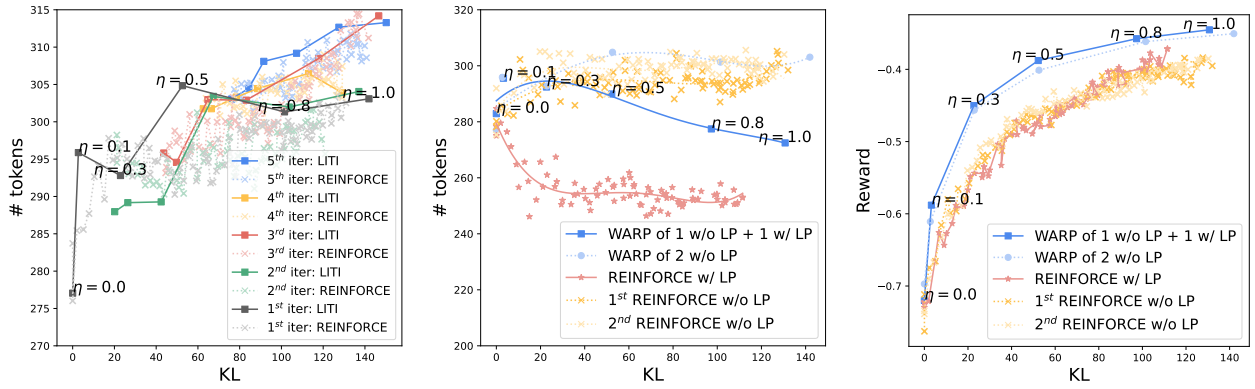


Figure 17 | **Experiments ablating the initialization in *LITI*.** We compare linear interpolating either towards the episode-specific initialization (i.e., the θ^{η} selected from previous iteration) or towards the SFT, which was the initialization of the 1$^{\text{st}}$ episode. The two resulting Pareto fronts are similar. However, in our iterative experiments we interpolate towards the episode-specific initialization as it allows maintaining a constant η at each *WARP* iteration, enabling a smooth progression towards the high KL regions.

E. Addressing length bias in *WARP*



(a) Length bias in iterative *WARP*. (b) Adding length penalty (LP). (c) Benefits of diversity.

Figure 18 | **Addressing length bias in *WARP*.** Figure 18(a) explores how length and KL change in successive *WARP* iterations. Figure 18(b) demonstrates the effectiveness of length penalty (LP) in reducing output length, and how such policies can be merged with others trained without LP. Finally, Figure 18(c) suggest an additional benefit of merging policies with different objectives, as it improves the KL-reward Pareto front.

Problem: length bias. We investigate a potential length bias in *WARP* [117]. LLMs after RLHF tend to be unnecessarily verbose because RMs often prefer longer generations to shorter ones, leading to this form of reward hacking. We confirm such a phenomenon in Figure 18(a), where the length of the generated text increases with higher KL values. This trend is even more pronounced in iterative *WARP*, where the 3^{rd} iteration generates longer sentences than the 1^{st} iteration at the same KL.

Mitigation strategy: length penalty. To mitigate this length bias, we integrate a length penalty (LP) into the reward: $-0.0005 \times \text{len}(y)$, following [119]. From the SFT init, we launch one RL fine-tuning run with this LP, highlighted with red stars in Figure 18(b). This LP leads to significantly shorter outputs as training occurs and KL increases, in contrast to policies trained without LP.

***SLERP* with different configurations.** Figure 18(b) also displays the lengths of generations from a *SLERP* merging of two policies, one trained with the LP and the other without. Critically, merging policies from diverse training configurations not only mitigates the length bias but also improves the Pareto front, as illustrated in Figure 18(c). This improvement is likely due to the increased diversity in weights and predictive mechanisms across policies, which seems beneficial for generalization, as shown in supervised learning [106].

Conclusion. Those experiments highlight the possibility to fix the length bias, and also the benefits of merging policies trained with diverse rewards, supporting previous suggestions from [109].

F. Diversity in predictions

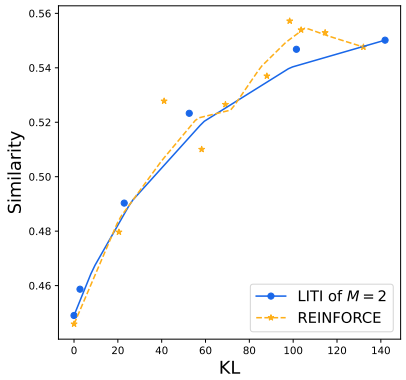


Figure 19 | **Confirming diversity loss in RLHF**. The x-axis is the KL compared to the SFT initialization; the y-axis is the similarity across two generations from a given policy when decoding with temperature 0.9.

Finally, we investigate the loss in diversity across generations when aligning LLMs, as reported in [65]. This could have negative consequences for creative or exploratory tasks, or even lead to policy collapse [42, 86]. In Figure 19 we plot the BLEURT similarity [115] across generations, during REINFORCE, or in *LITI* (as we interpolate back towards the SFT initialization). We observe that KL is strongly positively correlated with similarity across generations, confirming that RLHF induces a loss of diversity across generations. This experiment confirms that optimizing the Pareto optimality between reward and KL enables to trade-off between alignment and other benefits from pre-training, such as diversity in generations.