Rewarded soups: towards Pareto-optimal alignment
by interpolating weights fine-tuned on diverse rewards

Abstract

Foundation models are first pre-trained on vast unsupervised datasets and then fine-tuned on labeled data. Reinforcement learning, notably from human feedback (RLHF), can further align the network with the intended usage. Yet the imperfections in the proxy reward may hinder the training and lead to suboptimal results; the diversity of objectives in real-world tasks and human opinions exacerbate the issue. This paper proposes embracing the heterogeneity of diverse rewards by following a multi-policy strategy. Rather than focusing on a single a priori reward, we aim for Pareto-optimal generalization across the entire space of preferences. To this end, we propose rewarded soup, first specializing multiple networks independently (one for each proxy reward) and then interpolating their weights linearly. This succeeds empirically because we show that the weights remain linearly connected when fine-tuned on diverse rewards from a shared pre-trained initialization. We demonstrate the effectiveness of our approach for text-to-text (summarization, helpful assistant), text-image (image captioning, text-to-image generation, visual grounding), and control (locomotion) tasks. We hope to enhance the alignment of deep models, and how they interact with the world in all its diversity.

1. Introduction

Foundation models (Bommasani et al., 2021) have emerged as the standard paradigm to learn neural networks’ weights. They are typically first pre-trained through self-supervision (Devlin et al., 2019; Brown et al., 2020; Caron et al., 2021; Radford et al., 2021) and then fine-tuned (Oquab et al., 2014; Yosinski et al., 2014) via supervised learning (Vapnik, 1999). Yet, collecting labels is expensive, and thus supervision may not cover all possibilities and fail to perfectly align (Amodei et al., 2016; Taylor et al., 2016; Ngo, 2022) the trained network with the intended applications. Recent works (Stiennon et al., 2020; Ouyang et al., 2022; Pinto et al., 2023) showed that deep reinforcement learning (DRL) helps by learning from various types of rewards. A prominent example is reinforcement learning from human feedback (RLHF) (Stiennon et al., 2020; Christiano et al., 2017; Ziegler et al., 2019; Wu et al., 2021), which appears as the current go-to strategy to refine large language models (LLMs) into conversational agents such as ChatGPT (Ouyang et al., 2022; OpenAI, 2023). After pre-training on next token prediction (Radford et al., 2018), the LLMs are fine-tuned to follow instructions (Wei et al., 2022; Wang et al., 2022c; Taori et al., 2023) before reward maximization. This RL strategy enhances alignment by evaluating the entire generated sentence instead of each token independently, handling the diversity of correct answers and allowing for negative feedback (Goldberg, 2023). Similar strategies have been useful in computer vision (CV) (Pinto et al., 2023; Rennie et al., 2017), for example to integrate human aesthetics into image generation (Lee et al., 2023; Wu et al., 2023b).

Diversity of proxy rewards. RL is usually seen as more challenging than supervised training (Dulac-Arnold et al., 2021), notably because the real reward—ideally reflecting the users’ preferences—is often not specified at training time. Proxy rewards are therefore developed to guide the learning, either as hand-engineered metrics (Papineni et al., 2002; Lin & Hovy, 2003; Vedantam et al., 2015) or more recently in RLHF as models trained to reflect human preferences (Christiano et al., 2017; Kwon et al., 2023; Xu et al., 2023). Nonetheless, designing reliable proxy rewards for evaluation is difficult. This reward misspecification (Amodei et al., 2016; Pan et al., 2022) between the proxy reward and the users’ actual rewards can lead to unforeseen consequences (Michaud et al., 2020). Moreover, the diversity of objectives in real-world applications complicates the challenge. In particular, human opinions can vary significantly (Wildavsky, 1987; Coello, 2000; Schwartz et al., 2012) on subjects such as aesthetics (Nadal & Chat-
In this paper, we propose **rewarded soup** (RS), an efficient and flexible multi-policy strategy to fine-tune any foundation model. As shown in Section 1, we first use RL to learn one network for each proxy reward; then, we combine these expert networks according to user preferences. This a posteriori selection allows for better-informed trade-offs, improved transparency and increased fairness (Hayes et al., 2022; Mannion et al., 2021). The method to combine those networks is our main contribution: we do this through **linear interpolation in the weight space**, despite the non-linearities in the network. This is in line with recent findings on linear mode connectivity (LMC) (Frankle et al., 2020; Neyshabur et al., 2020): weights fine-tuned from a shared pre-trained initialization remain linearly connected and thus can be interpolated. This LMC inspired a plethora of weight interpolation (WI) strategies (Wortsman et al., 2022a; Ramé et al., 2022; Matena & Raffel, 2022; Ilharco et al., 2022; Don-Yehiya et al., 2022; Ramé et al., 2023), discussed in Section 4. Actually, the name rewarded soups follows the terminology of model soups (Wortsman et al., 2022a), as we combine various ingredients each rewarded differently. Unlike previous works, which focused on supervised learning, we explore LMC in RL, in a challenging setup where each training run uses a different reward. Perhaps surprisingly, we show that we can trade off the capabilities of multiple weights in a single final model, thus without any computational overhead. This enables the creation of custom weights for any preference over the diverse rewards.

- We propose a new practical strategy named rewarded soup for fine-tuning foundation models with diverse rewards. It defines a continuous set of (close to) Pareto-optimal solutions by weight interpolation, approximating more costly multi-policy strategies.
- We analyze the linear mode connectivity between weights fine-tuned on diverse rewards.
- We validate that RS mitigates reward misspecification.

In Section 3, we demonstrate the consistent effectiveness of rewarded soup across a variety of tasks: RLHF fine-tuning of LLaMA, multimodal tasks such as image captioning, text-to-image generation with diffusion models or visual grounding, as well as locomotion tasks. More results are on our anonymized website.

## 2. Rewarded soups

### 2.1. RL fine-tuning with diverse rewards

We consider a deep neural network $f$ of a fixed non-linear architecture (e.g., with batch normalization (Ioffe & Szegedy, 2015), ReLU layers (Agarap, 2018) or self-attention (Vaswani et al., 2017)). It defines a policy by

---

**Figure 1:** We detail the steps in rewarded soup (RS). After unsupervised pre-training and supervised fine-tuning, we launch $N$ independent RL fine-tunings on the proxy rewards $\{R_i\}_{i=1}^N$. Then we combine the trained networks by interpolation in the weight space: The final weights are adapted at test time by selecting $\lambda$. The resulting weights can be interpolated. This LMC inspired a plethora of weight interpolation (WI) strategies (Wortsman et al., 2022a), as we can trade off the capabilities of multiple weights in a single final model, thus without any computational overhead. This enables the creation of custom weights for any preference over the diverse rewards.

---

[1] Akshay, M., & Nishio, T. (2022). Towards multi-policy strategies. Considering these challenges, it may not be feasible to develop a single model simultaneously aligned with everyone’s preferences (Ouyang et al., 2022). Current strategies tend to align towards a consensus-based user (Bukker et al., 2022; Ovadya, 2023), inherently prioritizing certain values over others, potentially resulting in unfair representations of marginalized groups (Kirk et al., 2023). Moreover, these trade-offs (Pan et al., 2023) are decided a priori before training, shifting the responsibility to the engineers and reducing transparency and explainability (Hayes et al., 2022). These limitations, further discussed in Appendix A.2.1, highlight a key limitation of single-policy alignment strategies; their inability to handle the diversity of human preferences. Yet, “human-aligned artificial intelligence is a multi-objective problem” (Vamplew et al., 2018). Thus, we draw inspiration from the multi-objective reinforcement learning (MORL) literature (Barrett & Narayanan, 2008; Li et al., 2020; Tanaka & Yamamura, 2003; Van Moffaert & Nowé, 2014; Roijers et al., 2013; Rădulescu et al., 2020) and notably (Hayes et al., 2022) arguing that tackling diverse rewards requires shifting from single-policy to multi-policy approaches. As optimality depends on the relative preferences across those rewards, the goal is not to learn a single network but rather a set of Pareto-optimal networks (Pareto, 1964).

---
mapping inputs \(x\) to \(f(x, \theta)\) when parametrized by \(\theta\). For a reward \(R\) and a test distribution \(T\) of deployment, our goal is to maximize \(\int_{x \in T} R(f(x, \theta))\). Learning \(\theta\) is now commonly a three-step process: unsupervised pre-training, supervised fine-tuning, and reward optimization. Yet \(\hat{R}\) is usually not specified, meaning we can only optimize a proxy reward \(\hat{R}\) during training. This reward misspecification between \(R\) and \(\hat{R}\) may hinder the alignment of the network with \(\hat{R}\). Moreover, the diversity of human preferences complicates the design of \(R\).

Rather than optimizing one single proxy reward, our paper’s first key idea is to consider a family of \(N\) diverse proxy rewards \(\{R_i\}_{i=1}^N\). Each of these rewards evaluates the prediction according to different (potentially conflicting) criteria. The goal then becomes obtaining a coverage set of policies that trade-off between these rewards. To this end, we first introduce the costly MORL baseline. Its inefficiency motivates our rewarded soups, which leverages our second key idea: weight interpolation.

**MORL baseline.** The standard MORL scalarization strategy (Barrett & Narayanan, 2008; Li et al., 2020) linearizes the problem by interpolating the proxy rewards using \(M\) different weightings. Specifically, during the training phase, \(M\) trainings are launched, with the \(j\)-th optimizing the reward \(\sum_{i=1}^N \mu_i^j R_i\), where \(\forall j \in \{1, \ldots, M\}, \{\mu_i^j\}_{i=1}^N \in \Delta_N\) the \(N\)-simplex s.t. \(\sum_{i=1}^N \mu_i^j = 1\) and \(0 \leq \mu_i^j \leq 1\). Then, during the selection phase, the user’s reward \(\hat{R}\) becomes known and the \(j\)-th policy that maximizes \(\hat{R}\) on some validation dataset is selected. We typically expect to select \(j\) such that \(\sum_{i=1}^N \mu_i^j R_i \approx \hat{R}\) linearly approximates the user’s reward. Finally, this \(j\)-th weight is used during the inference phase on test samples. Yet, a critical issue is that “minor [preference] variations may result in significant changes in the solution” (Vamplew et al., 2008). Thus, a high level of granularity in the mesh of \(\Delta_N\) is necessary. This requires explicitly maintaining a large set of \(M \gg N\) networks, practically one for each possible preference. Ultimately, this MORL strategy is unscalable in deep learning due to the computational, memory, and engineering costs involved (see further discussion in Appendix A.2.2).

**Rewarded soup (RS).** In this paper, we draw inspiration from the weight interpolation literature. The idea is to learn expert weights and interpolate them linearly to combine their abilities. Specifically, we propose RS, illustrated in Section 1 and whose recipe is described below. RS alleviates MORL’s scaling issue as it requires only \(M = N\) trainings while being flexible and transparent.

1. During the training phase, we optimize a set of \(N\) expert weights \(\{\theta_i\}_{i=1}^N\), each corresponding to one of the \(N\) proxy rewards \(\{R_i\}_{i=1}^N\), and all from a shared pre-trained initialization.
2. For the selection phase, we linearly interpolate those weights to define a continuous set of rewarded soups policies: \(\{\sum_{i=1}^N \lambda_i \cdot \theta_i\}_{\lambda \in \Delta_N}\). Practically, we uniformly sample \(M\) interpolating coefficients \(\{\lambda^j_i\}_{i=1}^N\) from the \(N\)-simplex \(\Delta_N\) and select the \(j\)-th that maximizes the user’s reward \(\hat{R}\) on validation samples, i.e., \(\arg \max_{j=1}^M \hat{R}\left(\sum_{i=1}^N \lambda^j_i \theta_i\right)\).
3. For the inference phase, we predict using the network \(f\) parameterized by \(\sum_{i=1}^N \lambda^j_i \theta_i\).

While MORL interpolates the rewards, RS interpolates the weights. This is a considerable advantage as the appropriate weighting \(\lambda\), which depends on the desired trade-off, can be selected a posteriori: the selection is achieved without additional training, only via inference on some samples. In the next Section 2.2 we explicitly state the Hypotheses 1 and 2 underlying in RS. These are considered Working Hypotheses as they enabled the development of our RS strategy. Their empirical verification will be the main motivation for our experiments on various tasks in Section 3.

### 2.2. Exploring the properties of the rewarded soups set of solutions

#### 2.2.1. Linear mode connectivity of weights

**Fine-tuned on diverse rewards**

We consider \(\{\theta_i\}_{i=1}^N\) fine-tuned on \(\{R_i\}_{i=1}^N\) from a shared pre-trained initialization. Previous works (Frankle et al., 2020; Neyshabur et al., 2020; Wortsman et al., 2022a; Ramé et al., 2023) defined linear mode connectivity (LMC) w.r.t. a single performance measure (e.g., accuracy or loss) in supervised learning. We extend this notion in RL with \(N\) rewards, and define that the LMC holds if all rewards for the interpolated weights exceed the interpolated rewards. It follows that the LMC condition which underpins RS’s viability is the Hypothesis 1 below.

**Working Hypothesis 1 (LMC).** For all \(\{\lambda_i\}_{i=1}^N \in \Delta_N\) and \(k \in \{1, \ldots, N\}\), \(R_k(\sum \lambda_i \cdot \theta_i) \geq \sum \lambda_i R_k(\theta_i)\).

#### 2.2.2. Pareto optimality of rewarded soups

The Pareto front (PF) is the set of undominated weights, for which no other weights can improve a reward without sacrificing another, i.e., \(\{\theta : \exists \theta' \in \Theta s.t. \{R_i(\theta')\}_{i=1}^N >_N \{R_i(\theta)\}_{i=1}^N\}\) where \(>_N\) is the dominance relation in \(\mathbb{R}^N\).

In practice, we only need to retain one policy for each possible value vector, i.e., a Pareto coverage set (PCS). We now introduce the key Hypothesis 2.

**Working Hypothesis 2 (Pareto optimality).** The set \(\{\sum \lambda_i \cdot \theta_i \in \Delta_N\}\) is a PCS of \(\{R_i\}_{i=1}^N\).

Hypothesis 2 holds if the rewarded soups solutions, uncovered by interpolation, are Pareto-optimal. Overall, we
empathically validate Hypotheses 1 and 2 in Section 3, yet
also report a few limitations in Appendix and research di-
tections to fix them. Moreover, we theoretically prove in
Appendix B.2 they approximately hold when rewards are
replaced by their second-order Taylor expansion with co-
diagonalizable Hessians, a simplified setup justifiable when
weights remain close.

Remark 1. Hypotheses 1 and 2 rely on a good pre-trained
initialization, making RS particularly well-suited to fine-
tune foundation models. This is because pre-training pre-
vents the weights from diverging during training (Neyshabur
et al., 2020). When the weights remain close, we can theo-
retically justify Hypotheses 1 and 2 (see Appendix B.2) and,
more broadly, demonstrate that WI approximates ensem-
bling (Hansen & Salamon, 1990; Lakshminarayanan et al.,
2017) (see Lemma 4). In contrast, the LMC does not hold
when training from scratch (Neyshabur et al., 2020).

Remark 2. Pareto-optimality in Hypothesis 2 is defined
w.r.t. a set of weights $\Theta$. Yet, in full generality, improve-
ments in initialization, RL algorithms, data, or specific hyperpa-
rameters could enhance performances. In other words, for
real-world applications, the true PF is unknown and needs
to be defined w.r.t. a training procedure. In this case, $\Theta$
represents the set of weights attainable by fine-tuning within a
shared procedure. As such, in Section 3 we analyze Hypothe-
sis 2 by comparing the fronts obtained by RS and scalarized
MORL while keeping everything else constant.

2.2.3. CONSEQUENCES OF PARETO OPTIMALITY IF THE
USER’S REWARD IS LINEAR IN THE PROXY
REWARDS

Lemma 1 (Reduced reward misspecification in the linear
case). If Hypothesis 2 holds, and for linear reward $\hat{R} = \sum_i \hat{\mu}_i \hat{R}_i$ with $\{\hat{\mu}_i\} \in \Delta_N$, then $\exists\{\lambda_i\} \in \Delta_N$ such that
$\sum_i \lambda_i \cdot \hat{\theta}_i$ is optimal for $\hat{R}$.

The proof outlined in Appendix B.1 directly follows the
definition of Pareto optimality. In simpler terms, Lemma 1
implies that if Hypothesis 2 is true, then RS can mitigate
reward misspecification. For any preference $\hat{\mu}$, there exists
a $\lambda$ such that the $\lambda$-interpolation over weights maximizes
the $\hat{\mu}$-interpolation over rewards. In practice, as we will see in
Figure 4(a), we can set $\lambda = \hat{\mu}$, or cross-validate $\lambda$ on other
samples. Yet, this theoretically holds only for $\hat{R}$ linear over
the proxy rewards. This follows the linear utility functions
setup from the MORL literature (Rădulescu et al., 2020),
whose limitations (Vamplew et al., 2008) are discussed in
Appendix A.1. This motivates having sufficiently rich and
diverse proxy rewards to capture the essential aspects of
all possible users’ rewards. Despite the lack of theoretical
guarantees, we will show in Figures 4(b) and 9 that weight
interpolation improves results even for non-linear $\hat{R}$.

3. Experiments

In this section we implement RS across a variety of standard
learning tasks: text-to-text generation, image captioning,
image generation, visual grounding, and locomotion. We
use either model or statistical rewards. We follow a system-
atic procedure. First, we independently optimize diverse
rewards on training samples. For all tasks, we employ the
default architecture, hyperparameters and RL algorithm; the
only variation being the reward used across runs. Second,
we evaluate the rewards on the test samples: the results
are visually represented in series of plots. Third, we verify
Hypothesis 1 by examining whether RS’s rewards exceed
the interpolated rewards. Lastly, as the true Pareto front is
unknown in real-world applications, we present empirical
support for Hypothesis 2 by comparing the front defined by
RS (sliding $\lambda$ between 0 and 1) to the MORL’s solutions opti-
mitizing the $\mu$-weighted rewards for $0 \leq \mu \leq 1$ (sometimes
only $\mu = 0.5$ for computational reasons).

3.1. Text-to-text: LLaMA with diverse RLHF

Given the significance of RLHF to train LLMs, we begin
our experiments with text-to-text generation tasks. Our
pre-trained network is LLaMA-7b (Touvron et al., 2023),
instruction fine-tuned (Wei et al., 2022; Wang et al., 2022b)
on Alpaca (Taori et al., 2023). For RL training with PPO
(Schulman et al., 2017), we employ the trl package (von
Werra et al., 2020) and the setup from (Beeching et al.,
2023) with low-rank adapters (LoRA) (Hu et al., 2022a) for
efficiency. We consider the following tasks: summariza-
tion (Stiennon et al., 2020; Wu et al., 2021) on two datasets
(Reuter news (Ahmed, 2017) in Figure 2(a) and Reddit posts
(Völske et al., 2017) in Figure 2(b)), and helpfulness as a
conversational assistant (Bai et al., 2023a) in Figures 2(c)
and 2(d). To evaluate the generation in the absence of super-
vision, we utilized $N = 2$ different reward models (RM)
for each task, except in Figure 2(d) where $N = 4$. These
RMs were trained on human preferences datasets (Chris-
tiano et al., 2017) and all open-sourced on HuggingFace
(Wolf et al., 2020). For example in summarization, $R_1$
follows the “Summarize from Human Feedback” paper (Sti-
ennon et al., 2020), while $R_2$ leverages “contrast candidate
generation” (Chen et al., 2021). For the assistant task, we
rely on diverse RMs from OpenAssistant (Köpf et al., 2023);
they differ by their training procedures.

The results are reported in Figure 2. The green front, de-
defined by RS between the two weights specialized on $R_1$ and
$R_2$, is above the straight line connecting those two points,
validating Hypothesis 1. Second, the front passes through
the point obtained by MORL fine-tuning on the average
of the two rewards, supporting Hypothesis 2. Moreover,
when comparing both full fronts, they have qualitatively
the same shape; quantitatively in hypervolume (Yen & He,
3.2. Image-to-text: captioning with statistical rewards

RL training is also effective for multimodal tasks (Pinto et al., 2023), for example in image captioning (Rennie et al., 2017) where the task is to generate textual descriptions of images. Precisely evaluating the quality of a prediction w.r.t. a set of human-written captions is a challenging task, thus the literature relies on various hand-engineered, non-differentiable metrics: e.g., the precision-focused BLEU (Papineni et al., 2002), the recall-focused ROUGE (Lin & Hovy, 2003), METEOR (Banerjee & Lavie, 2005) handling synonyms and CIDEr (Vedantam et al., 2015) using TF-IDF. Thus the literature relies on various hand-engineered, non-differentiable metrics: e.g., the precision-focused BLEU (Papineni et al., 2002), the recall-focused ROUGE (Lin & Hovy, 2003), METEOR (Banerjee & Lavie, 2005) handling synonyms and CIDEr (Vedantam et al., 2015) using TF-IDF. As these metrics are proxies for human preferences, good trade-offs are desirable. We conduct our experiments on COCO (Lin et al., 2014), with an ExpansionNet2 (Hu et al., 2022b) network and a Swin Transformer (liu et al., 2022) visual encoder, initialized from the state-of-the-art weights of (Hu et al., 2022b) optimized on CIDEr. We then utilize the code of (Hu et al., 2022b) and their self-critical (Rennie et al., 2017) procedure (a variant of REINFORCE (Williams, 1992)) to reward the network on BLEU1, BLEU4, ROUGE or METEOR. More details are in Appendix D.

We observe in Figure 3 that tuning solely BLEU1 sacrifices some points on ROUGE or BLEU4. Yet interpolating between $\theta_1$ and $\theta_2$ uncovers a convex set of solutions approximating the ones obtained through scalarization of the rewards in MORL. When comparing both full fronts in Figure 3(a), they qualitatively have the same shape, and quantitatively the same hypervolume (Yen & He, 2013) of 0.140. One of the strengths of RS is its ability to scale to any number of rewards. In Figure 3(c), we uniformly $(\lambda_i = \frac{1}{N})$ average $N = 5$ weights fine-tuned independently. It improves upon the initialization (Hu et al., 2022b) and current state-of-the-art on all metrics, except for CIDEr, on which (Hu et al., 2022b) was explicitly optimized.

Figure 4 refines our analysis of RS. In Figures 4(a) and 4(b), rewards are normalized to 1 for the initialization and 0 for the worst model. Figure 4(a) validates Lemma 1: for any linear preference $\mu$ over the proxy rewards, there exists an optimal solution in the set described by RS. Two empirical strategies to set the value of $\lambda$ are close to optimal: selecting $\lambda = \hat{\mu}$ if $\hat{\mu}$ is known, or cross-validating (CV) $\lambda$ if a different data split (Karpathy & Fei-Fei, 2015) is available. Moreover, Figure 4(b) (and Figure 9 in Appendix D) investigate all metrics as evaluation. Excluding results’ variance, we observe monotonicity in both training rewards, linear in BLEU1 and quadratic in ROUGE. For other evaluation rewards that cannot be linearly expressed over the training rewards, the curves’ concavity shows that RS consistently improves the endpoints, thereby mitigating reward misspecification. The optimal $\lambda$ depends on the similarity between the evaluation and training rewards: e.g., best BLEU2 are with small $\lambda$. Lastly, as per (Izmailov et al., 2018) and Lemma 4, Figure 4(c) suggests that RS succeeds because WI approximates deep ensembling (Hansen & Salamón, 1990; Lakshminarayanan et al., 2017), interpolating the predictions rather than the weights. Actually, ensembling performs better, but it cannot be fairly compared as its inference cost is doubled.
3.3. Text-to-image: diffusion models with RLHFs

Beyond text generation, we now apply RS to align text-to-image generation with human feedbacks, as previously done in recent papers (Lee et al., 2023; Wu et al., 2023b; Xu et al., 2023). This alignment is expected to improve specific visual control signals like colors, counts, and backgrounds. Notably, diffusion models can be fine-tuned to match human aesthetic preferences. Our network is a diffusion model (Ho et al., 2020) with 2.2B parameters, pre-trained on an internal dataset of 300M images; it reaches similar quality as Stable Diffusion (Rombach et al., 2022), which was not used for copyright reasons. As for any subjective metric, there is a variety of reward models that can capture different aspects of aesthetic preference. In our experiments, we employ \( N = 2 \) open-source reward models: \textit{ava}, trained on the AVA dataset (Murray et al., 2012), and \textit{cafe}, trained on a mix of real-life and manga images. These two models are trained in a supervised setting to match human quality ratings collected. We first generate 10000 images; then, for each reward, we remove half of the images with the lowest reward’s score, and fine-tune 10% of the parameters (Xie et al., 2023) on the reward-weighted negative log-likelihood (Lee et al., 2023). More details are in Appendix E.
We now consider visual grounding (VG) (Yu et al., 2016): the task is to predict the bounding box of the region described by an input text. We use a seq-to-seq unified model predicting the box auto-regressively as a sequence of location tokens (Wang et al., 2022a). This model is pre-trained on a large image-text dataset, then fine-tuned with cross-entropy for VG; finally, we use a weighted loss between the cross-entropy and REINFORCE in the RL stage. As the main evaluation metric for VG is the accuracy (i.e., intersection over union (IoU) > 0.5), we consider 3 non-differentiable rewards: the accuracy on small, medium, and large objects. We design this experimental setup because improving results on all sizes simultaneously is challenging, as shown in Figure 5(c), where MORL performs similarly to the initialization. The results in Figure 5(b) confirm that optimizing for small objects degrades performance on large ones; fortunately, interpolating can trade-off. In conclusion, we can adapt to users’ preferences at test time by adjusting $\lambda$, which in turn changes the object sizes that the model effectively handles. On the one hand, if focusing on distant and small objects, a large coefficient should be assigned to $\theta_{Small}$. On the other hand, to perform well across all sizes, we can recover initialization’s performances by averaging uniformly (in Figure 5(c)). More details are in Appendix F.

### 3.5. Locomotion with diverse engineered rewards

Teaching humanoids to walk in a human-like manner (Duan et al., 2016) serves as a benchmark to evaluate RL strategies (Ng et al., 1999) for continuous control. One of the main challenges is to shape a suitable proxy reward (Dorigo & Colombetti, 1994; Dewey, 2014), given the intricate coordination and balance involved in human locomotion. It is standard (Todorov et al., 2012) to consider dense rewards of the
form \( R = velocity - \alpha \times \sum a^2 \), controlling the agent’s velocity while regularizing the actions \( \{a_t\} \) taken over time. Yet, the penalty coefficient \( \alpha \) is challenging to set. To address this, we devised two rewards in the Brax physics engine (Freeman et al., 2021): a risky \( R_1 \) with \( \alpha = 0 \), and a more cautious \( R_2 \) with \( \alpha = 1 \).

Like in all previous tasks, RS’s front in Figure 6 exceeds the interpolated rewards, as per Hypothesis 1. Moreover, RS’s front indicates an effective balance between risk-taking and cautiousness, providing empirical support for Hypothesis 2, although MORL with \( \mu = 0.5 \) (i.e., \( \alpha = 0.5 \)) slightly surpasses RS’s front. For a more qualitative and intuitive assessment, we provide animations of our RL agent’s locomotion on our anonymized website. More details are available in Appendix G.

### 4. Related work

Our RS approach leans on two key components from traditional DRL. The first is proxy rewards, whose design is challenging. Statistical metrics, the standard in captioning (Rennie et al., 2017) or language translation (Ranzato et al., 2016), are not practical to measure human concepts (Kwon et al., 2023) such as helpfulness (Bai et al., 2022a; Askell et al., 2021). Reward models can be trained via inverse DRL (Ng et al., 2000; Abbeel & Ng, 2004) when supervision from experts is available, otherwise from prediction comparison in recent RLHF works (Stiennon et al., 2020; Ouyang et al., 2022; Christiano et al., 2017). The latest (Kwon et al., 2023; Bai et al., 2022b; Madaan et al., 2023; Scheurer et al., 2023; Sun et al., 2023) further reduce the labeling costs by using the in-context abilities of LLMs.

Second, RS relies on existing RL algorithms to maximize the given rewards. RS succeeds with variants of two of the most common, REINFORCE (Williams, 1992) and PPO (Schulman et al., 2017), suggesting it could be applied to others (Go et al., 2023; Yuan et al., 2023). Among the ensembling-like RL strategies (Wang et al., 2010; Mordatch et al., 2015; Rajeswaran et al., 2017) handling multiple policies, some (Parker-Holder et al., 2020; Osa et al., 2022) aim to explicitly increase the diversity, yet never with foundation models nor weight interpolation. Moreover, pre-training could address stability and exploration issues (Xie et al., 2022; Yang et al., 2023; Sekar et al., 2020).

When dealing with multiple objectives in deep learning, the common approach is to combine them into a single reward (Roijers et al., 2013; Rădulescu et al., 2020); (Glaese et al., 2022) multiply the predictions of a preference RM (evaluating factfulness) and a rule RM (detecting rules breaking). The multi-policy alternatives (Barrett & Narayanan, 2008; Li et al., 2020; Tanaka & Yamamura, 2003; Van Moffaert & Nowé, 2014) are usually more costly. To reduce the cost, (Won et al., 2020; Yang et al., 2020) build experts and then train a new network to combine them; (Mossalam et al., 2016; Wilson et al., 2007; Nguyen et al., 2020) share weights across experts; (Castelletti et al., 2013; Yang et al., 2019; Abels et al., 2019; Peschl et al., 2021) directly train a single model; the recent and more similar (Hua et al., 2023) learns one linear embedding per (locomotion) task that can be interpolated. Yet, these works are mostly for academic benchmarks (Todorov et al., 2012; Vamplew et al., 2011); adapting them to larger tasks (e.g., RLHF for foundation models with PPO) is challenging as they modify the training procedure. Finally, we relate to multitask learning (Caruana, 1997), where predictions are evaluated for multiple tasks; in contrast, we have a single prediction evaluated by multiple rewards.

Recent works extended the linear mode connectivity when fine-tuning on different tasks (Ilharco et al., 2022; Don-Yehiya et al., 2022; Ramé et al., 2023; Wu et al., 2023a) or with different losses (Ramé et al., 2022; Croce et al., 2023), while (Juneja et al., 2023) highlighted some failures in NLP for classification. In contrast, we investigate the LMC in RL. The most similar works are for control system tasks: (Lawson & Qureshi, 2023) averaging decision transformers and (Gaya et al., 2022) explicitly enforcing connectivity in subspaces of policies trained from scratch on a single reward. When the LMC holds, combining networks in weights combines their abilities (Ilharco et al., 2023; Daheim et al., 2023); e.g., averaging an English summarizer and an English-to-French translator can summarize in French (Jang et al., 2023). In domain generalization, (Wortsman et al., 2022a; Ramé et al., 2022; Arpit et al., 2021) showed that WI reduces model misspecification (D’Amour et al., 2020); by analogy, we show that RS reduces reward misspecification.

### 5. Conclusion

As AI systems are increasingly applied to crucial real-world tasks, there is a pressing issue to align them to our specific and diverse needs, while making the process more transparent and limiting the cultural hegemony of a few individuals. In this paper, we proposed rewarded soup, a strategy that efficiently yields Pareto-optimal solutions through weight interpolation after training. Our experiments have consistently validated our working hypotheses for various significant large-scale learning tasks, demonstrating that rewarded soup can mitigate reward misspecification. We hope to inspire further research in exploring how the generalization literature in deep learning can help for alignment, to create AIs that benefit society as a whole.
References


Barrett, L. and Narayanan, S. Learning all optimal policies with multiple criteria. In ICML, 2008. (pp. 3, 8)

Becker, S. and Le Cun, Y. Improving the convergence of back-propagation learning with second order methods. In Connectionist models summer school, 1988. (p. 20)


Chen, S., Zhang, F., Sone, K., and Roth, D. Improving Faithfulness in Abstractive Summarization with Contrast Candidate Generation and Selection. In NAACL, 2021. (pp. 4, 23, and 24)


Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018. (pp. 5, 17, and 23)


Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. In *ICLR*, 2023. (pp. 1 and 8)


LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural Networks*. 2012. (p. 20)


Lin, C.-Y. and Hovy, E. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL*, 2003. (pp. 1, 2, and 5)


Matena, M. and Raffel, C. Merging models with Fisher-weighted averaging. In *NeurIPS*, 2022. (pp. 2 and 22)


Neyshabur, B., Sedghi, H., and Zhang, C. What is being transferred in transfer learning? In *NeurIPS*, 2020. (pp. 2, 3, and 4)


Ngo, R. The alignment problem from a deep learning perspective. *arXiv preprint*, 2022. (pp. 1 and 17)


Smith, B. A brief review of the reasons multi-objective RL could be important in AI Safety Research. https://www.alignmentforum.org/posts/i5dLf i6m6FCexReK9/a-brief-review-of-the-reasons-multi-objective-rl-could-be, 2021. (p. 17)


Tetlock, P. E. A value pluralism model of ideological reasoning. JSPS, 1986. (p. 18)


Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In IROS, 2012. (pp. 7 and 8)


Vamplew, P., Dazeley, R., Foale, C., Firmin, S., and Mumery, J. Human-aligned artificial intelligence is a multi-objective problem. Ethics and Information Technology, 2018. (pp. 2, 17, and 18)

Van Moffaert, K. and Nowé, A. Multi-objective reinforcement learning using sets of pareto dominating policies. JMLR, 2014. (pp. 2 and 8)


Völks, M., Potthast, M., Syed, S., and Stein, B. Tl; dr: Mining reddit to learn automatic summarization. In ACL Workshop, 2017. (pp. 4 and 24)
Yen, G. G. and He, Z. Performance metric ensemble for multiobjective evolutionary algorithms. *TEVC*, 2013. (pp. 4 and 5)


Yu, L., Poirson, P., Yang, S., Berg, A. C., and Berg, T. L. Modeling context in referring expressions. In *ECCV*, 2016. (pp. 7, 28, and 29)


Rewarded soups: towards Pareto-optimality by interpolating weights fine-tuned on diverse rewards

Supplementary material

This supplementary material is organized as follows:

- Appendix A further discusses the societal impacts of our RS strategy.
- Appendix B details some theoretical aspects of our RS strategy.
- Appendix C details our experiments in RLHF with LLaMA for text-to-text generation.
- Appendix D details and enriches our experiments in image captioning.
- Appendix E details and enriches our experiments in image generation.
- Appendix F details and enriches our experiments in visual grounding.
- Appendix G details and enriches our locomotion experiments.

The shareable code will be released on this anonymized url page. Moreover, you can find additional qualitative results of our experiments on our anonymized website.

A. Discussion

A.1. Limitations and societal impacts

The recent and rapid scaling of networks presents both opportunities and major concerns (Amodei et al., 2016; Hendrycks & Mazeika, 2022; Hendrycks, 2023). Our approach is a step towards better empirical alignment (Taylor et al., 2016; Ngo, 2022). Yet, reward misspecification is only one of the many challenges inherited from the RL paradigm. First, proxy rewards may lack robustness (Gao et al., 2022) or be hacked (Skalse et al., 2022) via adversarial exploitation, making them unreliable. Second, RL algorithms may cause overfitting, leading to poor generalization in test, with a risk of goal misgeneralization (Shah et al., 2022; Di Langosco et al., 2022). Third, RLHF has drawbacks, such as harming calibration (OpenAI, 2023). Our a posteriori multi-policy strategy could alleviate the impact of some badly shaped proxy rewards and some failed optimizations, as well as tackling Goodhart’s law (Smith, 2021). Yet, without constraint on the test distribution, complete alignment may be impossible (Wolf et al., 2023), for example for LLMs with prompts of arbitrary (long) length. Therefore, new training paradigms (Keskar et al., 2019; Korbak et al., 2023) beyond RL may be required.

Theoretical guarantees for alignment are also needed (Rodriguez-Soto et al., 2021). Yet, RS relies on an empirical finding: the LMC (Frankle et al., 2020), which currently lacks full theoretical guarantees, even in the simplest case of moving averages (Izmailov et al., 2018). The best existing explanation (Ramé et al., 2022; Izmailov et al., 2018) relies on the similarities between weight interpolation and functional ensembling (Hansen & Salamon, 1990; Lakshminarayanan et al., 2017) when weights remain close, as recalled in Lemma 4. Moreover, assuming the LMC, Lemma 1 theoretically fixes issues only for $\hat{R}$ linear over the proxy rewards. Yet, such linearization cannot encapsulate all types of (human) preferences (Vamplew et al., 2018; 2008). Thus, considering more complex combinations (Gábor et al., 1998; An et al., 2021; Van Moffaert et al., 2013; Smith et al., 2021) is a promising direction. We may empirically overcome this limitation within RS by continually adjusting and adding new proxy rewards, such that their linear mixtures have increasingly good coverage. Indeed, RS is flexible and was shown to handle variable numbers of rewards.

Finally, our a posteriori alignment with users facilitates personalization (Salemi et al., 2023) of models. As discussed in Appendix A.2.1 and in (Kirk et al., 2023), this could increase usefulness by providing tailored generation, notably to under-represented groups. Moreover, the distributed nature of RS makes it parallelizable thus practical in a federated
learning setup (McMahan et al., 2017) where data must remain private. Yet, this personalization comes with risks for individuals of “reinforcing their biases [...] and narrowing their information diet” (Kirk et al., 2023). This may worsen the polarization of the public sphere. Under these concerns, we concur with the notion of “personalization within bounds” (Kirk et al., 2023), with these boundaries potentially set by weights fine-tuned on diverse and carefully inspected rewards.

### A.2. Benefits of our approach

In this section we discuss the benefits of our rewarded soup (RS) approach with respect to the two families of strategies: the single-policy and the multi-policy approaches.

#### A.2.1. Compared to single-policy approaches

The main reason why single-policy approaches are not suitable is because they optimize over a single set of preferences. In contrast, we build a coverage set of Pareto-optimal policies. This is important for the following reasons, mostly first discussed in Hayes et al. (Hayes et al., 2022) and in Kirk et al. (Kirk et al., 2023).

Indeed, the user’s true reward is highly uncertain before training. This “semi-blind” (Hayes et al., 2022) manual process forces a priori and uncertain decisions about the required trade-offs. It shifts the responsibility from the problem stakeholders to the system engineers, who need to anticipate the impact of their choices on the final performance. Critically, the RLHF process may cause the “tyranny of the crowdworker” (Kirk et al., 2023), as models are “tailored to meet the expectations of [...] a small number of crowdworkers primarily based in the US, with little to no representation of broader human cultures, geographies or languages.” (Kirk et al., 2023). Moreover, these “biases are exacerbated by a lack of [...] documentation” (Kirk et al., 2023). Thus (Kirk et al., 2023) argue that the personalization should be explicit—rather than implicitly caused by hidden and chaotic engineering choices. In contrast, our strategy could support decision-making to find a good balance between (potentially conflicting) parties’ interests. This value pluralism (Tetlock, 1986) can lead to fairer and more equitable outcomes (Vamplew et al., 2018; Siddique et al., 2020). Single-policy cannot adapt to test time requirements; in contrast, RS facilitates personalized assistances (Salemi et al., 2023), with fewer prompts/inputs to the model, as we only need to adapt interpolating coefficients and not the full network. This is all the more important as human preferences change from time to time: in this dynamic utility function scenario, RS can quickly adapt by adjusting the λ to match new preferences. Finally, RS could also improve the interpretability and explainability of the decisions. Letting the users decide could make the process more transparent (Gabriel & Ghazavi, 2021), which is essential to ensure that the development process is fair, unbiased, and inclusive (Abadi et al., 2016).

#### A.2.2. Compared to multi-policy approaches

The main reason why other multi-policy approaches through multitasking are not suitable is because of their computational costs required to learn a dense set of policies. In contrast, RS only trains the proxy rewards independently, and enables the selection of the interpolating coefficient a posteriori. This is especially useful with large number of rewards and thus growing number of combinations. Second, multitask (Caruana, 1997) is challenging; for example, even if the true reward is actually a linear weighted sum of some proxy rewards and those coefficients are known, using those preferences during training can lead to suboptimal results (Van Moffaert et al., 2014), because of conflicting gradients (Yu et al., 2020; Liu et al., 2021) or different variance scales (Espeholt et al., 2018; Teh et al., 2017). This has been tackled in RL, but so far mostly for games such as ATARI (Bellemare et al., 2013). Third, our strategy is compatible with the inherent iterative engineering process of alignment. Indeed, RS can continually include adjusted opinions while preventing forgetting of the old behaviours. This relates to the continual learning challenge, and the empirical observations that weight averaging can reduce catastrophic forgetting (Stojanovski et al., 2022; Eeckt et al., 2022). Moreover, as shown in (Ilharco et al., 2023) and confirmed in Figure 10(c), negative editing by weight interpolation can fix and force the removal of some behaviours. Finally, RS is computationally effective, requiring no communication across servers, thus enabling “embarrassingly simple parallelization” (Li et al., 2022). This facilitates its use in federated learning scenario (McMahan et al., 2017) where the data should remain private. Actually, RS follows the updatable machine learning paradigm (Raffel, 2021), “allowing for the collaborative creation of increasingly sophisticated AI system” (Ramé et al., 2023). In the future, we may develop open-source personalized models, rewarded on decentralized private datasets, and combine them continuously.
B. Theoretical insights

B.1. Proof of Lemma 1

Proof. Considering $\theta$ maximizing $\hat{R}$, we first show that $\theta$ is on the PF of $\{R_i\}_i$. Otherwise, considering $\theta' > N \theta$ and as $\forall i, \hat{\mu}_i \geq 0$, we have $\sum_i \hat{\mu}_i R_i(\theta') > \sum_i \hat{\mu}_i R_i(\theta)$. This implies that $\theta'$ would produce a better policy than $\theta$ for $\hat{R} = \sum_i \hat{\mu}_i R_i$ and thus the contradiction. Finally, as $\theta$ is on the PF and by definition of a PCS, there exists $\lambda$ s.t. $\forall k, R_k(\sum_i \lambda_i \cdot \theta_i) = R_k(\theta)$. □

B.2. Theoretical guarantees with quadratic rewards

In this section, we provide theoretical guarantees for the near-optimality of RS when considering quadratic rewards. This simplification amounts to replacing the rewards by their second-order Taylor approximation, which is a realistic assumption when the weights remain within a small neighborhood.

B.2.1. Simple case with Hessians proportional to the Identity matrix

For the first Lemma 2, we make the following simplifying Assumption 1.

Assumption 1 (Hessians proportional to the Identity matrix.). Every reward $R_i$ is quadratic, with Hessians proportional to $I_d$. Specifically, let $\Theta \subset \mathbb{R}^d$ be the set of possible weights, and let $\{R_i\}_{i=1}^N$ be the $N$ rewards, we can write for $i \in \{1, ..., N\}$:

$$\forall \theta \in \Theta, \quad R_i(\theta) = R_i(\hat{\theta}_i) - \eta_i \|\theta - \hat{\theta}_i\|^2 \quad (1)$$

where $\eta_i \in \mathbb{R}^+ \setminus \{0\}$ and $\hat{\theta}_i$ is the global maximum for reward $R_i$.

Lemma 2. Let $\hat{\mu} = (\hat{\mu}_1, ..., \hat{\mu}_N) \in \Delta_N$. Then, under Assumption 1, the reward $R_{\hat{\mu}} = \sum_i \hat{\mu}_i \times R_i$ is maximized on the convex hull of $\{\theta_1, ..., \theta_N\}$.

Proof. The function $R_{\hat{\mu}}$ is quadratic thus has an unique global maximum $\hat{\theta}$, that we find analytically:

$$\nabla_{\theta} R_{\hat{\mu}}(\hat{\theta}) = 0 \implies \sum_{i=1}^N \mu_i \eta_i \cdot (\hat{\theta} - \hat{\theta}_i) = 0 \implies \hat{\theta} = \frac{\sum_{i=1}^N \hat{\mu}_i \eta_i \cdot \hat{\theta}_i}{\sum_{i=1}^N \hat{\mu}_i \eta_i}$$

Since all the $\hat{\mu}_i \eta_i$ are positive or zero, and at least one is greater than zero, $\hat{\theta}$ is indeed in the convex hull of $\{\theta_1, ..., \theta_N\}$. □

Remark 3. Under Assumption 1, the reward functions are concave; thus we can reasonably assume that each fine-tuning procedure for $R_i$ reaches its global optimum $\theta_i$ for $i \in \{1, ..., N\}$. Then, Lemma 2 tells us that the maximum value for linear user’s reward $R_{\hat{\mu}}$ is obtainable by weight interpolation between the $\{\theta_i\}_{i=1}^N$: the interpolating coefficients in $\Delta_N$ such that $\lambda_i \propto \hat{\mu}_i \eta_i$ make rewarded soups optimal.

B.2.2. Advanced case with diagonal Hessians

We now consider the more complex case with the relaxed Assumption 2. For simplicity, we only consider $N = 2$ rewards $R_1$ and $R_2$.

Assumption 2 (Diagonal Hessians). The rewards are quadratic, with Hessians diagonal negative definite. Specifically, we can write for $i \in \{1, 2\}$:

$$\forall \theta = (\theta^1, ..., \theta^d) \in \Theta, \quad R_i(\theta) = R_i(\hat{\theta}_i) - \sum_{j=1}^d \eta^j_i (\theta^j - \theta^j_i)^2, \quad (2)$$

where $(\eta^1, ..., \eta^d) \in (\mathbb{R}^+)^d$ and $\hat{\theta}_i = (\theta^1_i, ..., \theta^d_i)$ is the global maximum for reward $R_i$. 

Remark 4. This diagonal Assumption 2 of the Hessian is common: for example in optimization (LeCun et al., 2012; Kingma & Ba, 2015), to prune networks (LeCun et al., 1990) or in out-of-distribution generalization (Rame et al., 2022). This strong assumption is supported by the empirical observation (Becker & Le Cun, 1988) that Hessians are diagonally dominant, in particular at the end of training. Also, we note that our findings remain valid assuming only that the Hessians are co-diagonalizable.

Lemma 3. We consider the user’s reward $R_{\hat{\mu}} = (1 - \hat{\mu}) \times R_1 + \hat{\mu} \times R_2$ with $\hat{\mu} \in [0, 1]$, and

$$\Delta R_{\hat{\mu}} = \max_{\theta \in \Theta} R_{\hat{\mu}}(\theta) - \max_{\lambda \in [0, 1]} R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2).$$

$\Delta R_{\hat{\mu}}$ corresponds to the difference in terms of $R_{\hat{\mu}}$ between the global maximum and the maximum reachable by weight interpolation through rewarded soups (with a single interpolating coefficient for all dimensions). Then, under Assumption 2, we have:

$$\Delta R_{\hat{\mu}} \leq \frac{\hat{\mu}^2(1 - \hat{\mu})^2(M \Delta_1 - \Delta_2)(M \Delta_2 - \Delta_1)}{(\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M)((1 - \hat{\mu})\Delta_1 + \hat{\mu}\Delta_2)},$$

where $M = \max_{j \in \{1, \ldots, d\}} \max\left(\frac{\eta_{j1}}{\eta_{j1}\eta_{j2}}, \frac{\eta_{j2}}{\eta_{j1}\eta_{j2}}\right)$ is the maximum of eigenvalues ratio, $\Delta_1 = R_1(\theta_2) - R_1(\theta_1)$ and $\Delta_2 = R_2(\theta_1) - R_2(\theta_2)$.

When $\Delta_1 = \Delta_2$, the bound simplifies into:

$$\Delta R_{\hat{\mu}} \leq \frac{\hat{\mu}^2(1 - \hat{\mu})^2(M - 1)^2}{\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M} \Delta_1$$

Furthermore, when the Hessians are equal, then $M = 1$ and $\Delta R_{\hat{\mu}} = 0$: RS is optimal.

Proof. This novel proof is in three steps. First, we find $\hat{\lambda}$ maximizing $R_{\hat{\mu}}(\theta)$ for $\theta$ on the full set of weights $\Theta$. Second, we find $\hat{\lambda}$ maximizing $R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2)$ for $\lambda \in [0, 1]$ and thus defining the best interpolation between the expert weights. Finally, we bound $\Delta R_{\hat{\mu}}$, the differences between their rewards, by applying the Bhatia-Davis inequality.

First step. Let’s first find the maximum of $R_{\hat{\mu}}$ on $\Theta$. Denoting $S = (1 - \hat{\mu}) \times R_1(\theta_1) + \hat{\mu} \times R_2(\theta_2)$, we have for all $\theta \in \Theta$:

$$R_{\hat{\mu}}(\theta) = S - \sum_{j=1}^{d} \left( (1 - \hat{\mu})\eta_{j1}(\theta_{j1} - \theta_1) + \hat{\mu}\eta_{j2}(\theta_{j2} - \theta_2) \right)^2$$

Since $R_{\hat{\mu}}$ is a sum of concave quadratic functions, it has a unique global maximum reached at a point we note $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_d)$. The global maximum can be computed by differentiating $R_{\hat{\mu}}$ with respect to each variable $\theta_j$, which gives:

$$\hat{\theta}_j = \left( 1 - \hat{\lambda}_j \right) \cdot \theta_1 + \hat{\lambda}_j \cdot \theta_2$$

where the interpolating coefficients per dimension $\hat{\lambda}_j$ are defined for $j \in \{1, \ldots, d\}$ as:

$$\hat{\lambda}_j = \frac{\hat{\mu}\eta_{j2}}{(1 - \hat{\mu})\eta_{j1} + \hat{\mu}\eta_{j2}} \in [0, 1].$$

Second step. With $\lambda \in [0, 1]$ and $\theta = (1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2$, we can write $R_{\hat{\mu}}(\theta)$ as a function of $\lambda$:

$$R_{\hat{\mu}}(\theta) = S - \sum_{j=1}^{d} \left( (1 - \hat{\mu})\eta_{j1}(\lambda - \hat{\lambda}_j) + \hat{\mu}(1 - \hat{\mu})\eta_{j2}(\lambda - \hat{\lambda}_j) \right)^2$$

$$= R_{\hat{\mu}}(\hat{\theta}) - \sum_{j=1}^{d} p_j \left( \lambda - \hat{\lambda}_j \right)^2$$

20
where \( p_j \) is defined as

\[
p_j = \left( (1 - \hat{\mu}) \eta_1^j + \hat{\mu} \eta_2^j \right) \left( \theta_1^j - \theta_2^j \right)^2.
\]

From Equation (8), we can compute the maximum reward obtainable for weight averaging \( \max_{\lambda \in [0, 1]} R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2) \). Since the function \( \lambda \mapsto R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2) \) is a concave quadratic function, there is a unique value \( \lambda \) maximizing \( R_{\hat{\mu}} \) equal to

\[
\lambda = \frac{\sum_{j=1}^d p_j \hat{\lambda}^j}{\sum_{j=1}^d p_j}.
\]

(9)

Since all \( p_j \) are positive and all \( \hat{\lambda}^j \) are between 0 and 1, \( \lambda \) is also between 0 and 1. Therefore, \( R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2) \) is indeed the maximum reward for rewarded soups.

**Third step.** Applying Equation (8) to \( \lambda \) gives:

\[
\lambda R_{\hat{\mu}} = R_{\hat{\mu}}(\hat{\theta}) - R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2)
\]

\[
= \sum_{j=1}^d p_j \left( \lambda - \hat{\lambda}^j \right)^2
\]

\[
= \left( \sum_{j=1}^d \frac{p_j}{\sum_{i=1}^n p_i} \left( \lambda - \hat{\lambda}^j \right)^2 \right) \left( \sum_{j=1}^d p_j \right)
\]

(12)

The second term in Equation (12) can be simplified as:

\[
\sum_{j=1}^d p_j = (1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2.
\]

(13)

The core component of this proof is the upper bounding of the first term in Equation (12). The key idea is to recognize the variance of a discrete random variable \( \Lambda \) with \( \mathbb{P}(\Lambda = \lambda_i) = \frac{p_i}{\sum_{j=1}^d p_j} \); then, \( \lambda \) from Equation (9) is actually the expectation of \( \Lambda \). Then, we can apply the **Bhatia-Davis inequality**, as recalled in Equation (14), on the variance of a bounded random variable \( a \leq \Lambda \leq b \):

\[
\text{Var}(\Lambda) \leq (b - \mathbb{E}(\Lambda))(\mathbb{E}(\Lambda) - a)
\]

(14)

Therefore Equation (12) is bounded by:

\[
\Lambda R_{\hat{\mu}} \leq \left( \max_{1 \leq j \leq d} \hat{\lambda}^j \right) \left( \lambda - \min_{1 \leq j \leq d} \hat{\lambda}^j \right) ((1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2).
\]

(15)

Now, we bound the variables \( \hat{\lambda}^j \), since \( 1/M \leq \eta_1^j / \eta_2^j \leq M \). Then for all \( j \) we have:

\[
\frac{\mu}{(1 - \hat{\mu})M + \hat{\mu}} \leq \hat{\lambda}^j \leq \frac{\mu}{M - (1 - \hat{\mu})M}.
\]

(16)

and thus:

\[
\Lambda R_{\hat{\mu}} \leq \left( \frac{\hat{\mu}M}{(1 + \hat{\mu}(M - 1))} - \hat{\lambda} \right) \left( \lambda - \frac{\hat{\mu}}{M - (1 - \hat{\mu})M} \right) ((1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2).
\]

(17)

Finally, noting that \( \Delta_1 = \sum_{j=1}^d p_j (\theta_1^j - \theta_2^j)^2 \), we deduce from Equation (9) that \( \hat{\lambda} = \frac{\hat{\mu} \Delta_2}{(1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2} \). Replacing this in the previous Equation (17) gives the final Equation (4), concluding the proof.
**Remark 5.** As a final remark, please note that the suboptimality of RS comes from the need of having one single interpolating coefficient $\bar{\lambda}$ for all $d$ parameters $(\theta^1, \ldots, \theta^d)$ of the network. Yet, the advanced merging operations in (Matena & Raffel, 2022) remove this constraint, with interpolating coefficients proportional to the eigenvalues of the Fisher matrices (Fisher, 1922), which actually approximate the eigenvalues of the Hessian (Schraudolph, 2002; Thomas et al., 2020). Combining (Matena & Raffel, 2022) and our RS is a promising research direction, the key issue being the computation of the Fisher matrices (Kunstner et al., 2019) for networks with billions of parameters.

**B.2.3. Bound visualization**

We visualize in Figure 7 the bound given by Lemma 3. We show that for small values of $M$ like $M = 2$, the value of $R_\hat{\mu}$ for RS is quite close to the global optimum. Also, recall that RS theoretically matches this upper bound when $M = 1$. For larger values like $M = 10$, the bound is less tight, and we note that the maximum value of $R_\hat{\mu}$ approaches the constant function 1 as $M \to \infty$.

![Figure 7: Illustration of the bound given by Lemma 3 under Assumption 2. For simplicity, we showcase the case where $R_1(\theta_1) = R_2(\theta_2) = 1$, $R_1(\theta_2) = R_2(\theta_1) = 0$, thus $\Delta_1 = \Delta_2 = 1$. In green, we plot the rewards obtained with rewarded soups for the optimal $\lambda$, i.e., $R_\hat{\mu} \left(1 - \hat{\mu}\right) \cdot \theta_1 + \hat{\mu} \lambda \cdot \theta_2$, whose value is independent of $M$ in this case. In blues, we plot the maximum value of $R_\hat{\mu}$ given by Equation (5) in Lemma 3, for $M = 2$ and $M = 10$. For reference, we also plot the values for the lower bound in the LMC Hypothesis 1, i.e., equal to $(1 - \hat{\mu})(1 - \lambda) R_1(\theta_1) + \hat{\mu} \lambda R_2(\theta_2)$. As RS outperforms this lower bound, it validates Hypothesis 1 in this case.](image-url)
We summarize the key implementation details of our text-to-text generation experiments in Table 1. The pre-trained network θ \(\theta\) we obtain the results when \(R\) (Wolf et al., 2020). For example in summarization tasks, then \(\theta\) was open-sourced in an effort to reproduce the Summarize 1 \(\theta\), \(\theta\) was fine-tuned on Alpaca (Taori et al., 2023) to follow instructions. We eventually fine-tune via PPO on the different considered tasks. Our code is adapted from (Beeching et al., 2023); we kept most of their hyperparameter values, only dividing by 2 the batch size to fit in our GPU and extending the output length. For each considered task, we downloaded the reward models from HuggingFace (Wolf et al., 2020). For example in summarization tasks, \(R_1\) was open-sourced in an effort to reproduce the Summarize from Human Feedback paper (Stiennon et al., 2020), while \(R_2\) (Chen et al., 2021) aimed at improved “faithfulness in abstractive summarization with contrast candidate generation”. For the assistant task, we rely on different reward models from OpenAssistant (Köpf et al., 2023). Though they all aim at evaluating whether an answer is adequate given a question, they differ in their predictions due to differences in their architecture and training procedures. In practice, we simply leverage them as block-box classification pipelines, implemented in the transformers library (Wolf et al., 2020).

B.3. Similarity between weight interpolation and functional ensembling

**Lemma 4** (\(λ\)-interpolation of weights approximates the \(λ\)-ensembling of predictions. Adapted from (Wortsman et al., 2022; Ramé et al., 2022; Izmailov et al., 2018).) Given \(\theta_1\) and \(\theta_2\) optimized for \(R_1\) and \(R_2\) s.t. they remain close, i.e., \(\|\theta_1 - \theta_2\|^2 \approx 0\). Denoting \(\theta_λ\) the interpolated weights \(\theta_λ = (1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2\) and \(f_λ\) the ensembling of predictions

\[
f(\cdot, \theta_λ) \approx f_λ(\cdot)
\]

and for \(k \in \{1, 2\}\):

\[
R_k(f(\cdot, \theta_λ)) \approx R_k(f_λ(\cdot))
\]

**Proof.** This proof follows (Ramé et al., 2022) and has two components.

**Functional approximation.** First, we perform a Taylor expansion at the first order of the models’ predictions w.r.t. parameters \(\theta\) for \(x \in T\):

\[
f(x, \theta_1) = f(x, \theta_λ) + \nabla_\theta f(x, \theta_λ)^T(\theta_1 - \theta_λ) + \mathcal{O}(\|\theta_1 - \theta_λ\|^2)
\]

\[
= f(x, \theta_λ) + \nabla_\theta f(x, \theta_λ)^T(\lambda \cdot \theta_1 - \lambda \cdot \theta_2) + \mathcal{O}(\|\theta_1 - \theta_2\|^2)
\]

and similarly:

\[
f(x, \theta_2) = f(x, \theta_λ) + \nabla_\theta f(x, \theta_λ)^T((\lambda - 1) \cdot \theta_1 + (1 - \lambda) \cdot \theta_2) + \mathcal{O}(\|\theta_1 - \theta_2\|^2)
\]

Then by \(λ\)-weighted sum over \(i\), the term multiplying \(\nabla_\theta f(x, \theta_λ)^T\) cancels out and we obtain:

\[
f_λ(x) = (1 - \lambda) \cdot f(x, \theta_1) + \lambda \cdot f(x, \theta_2) = f(x, \theta_λ) + \mathcal{O}(\|\theta_1 - \theta_2\|^2).
\] (18)

**Reward approximation.** Second, we obtain the reward approximation with a Taylor expansion at the zeroth order of the reward \(R_k\) for \(k \in \{1, 2\}\) and injecting Equation (18):

\[
R_k(f_λ(x)) = R_k(f(x, \theta_λ)(x)) + \mathcal{O}(f_λ(x) - f(x, \theta_λ))_2)
\]

\[
= R_k(f(x, \theta_λ)(x)) + \mathcal{O}(\|\theta_1 - \theta_2\|^2).
\]

We obtain the results when \(\theta_1\) and \(\theta_2\) remain close, i.e., when we can ignore the \(\mathcal{O}\) term.

C. Text-to-text: LLaMA with diverse RLHFs

We summarize the key implementation details of our text-to-text generation experiments in Table 1. The pre-trained network is LLaMA-7b (Touvron et al., 2023); then low-rank adapters (Hu et al., 2022a) were fine-tuned on Alpaca (Taori et al., 2023) to follow instructions. We eventually fine-tune via PPO on the different considered tasks. Our code is adapted from (Beeching et al., 2023); we kept most of their hyperparameter values, only dividing by 2 the batch size to fit in our GPU and extending the output length. For each considered task, we downloaded the reward models from HuggingFace (Wolf et al., 2020). For example in summarization tasks, \(R_1\) was open-sourced in an effort to reproduce the Summarize from Human Feedback paper (Stiennon et al., 2020), while \(R_2\) (Chen et al., 2021) aimed at improved “faithfulness in abstractive summarization with contrast candidate generation”. For the assistant task, we rely on different reward models from OpenAssistant (Köpf et al., 2023). Though they all aim at evaluating whether an answer is adequate given a question, they differ in their predictions due to differences in their architecture and training procedures. In practice, we simply leverage them as block-box classification pipelines, implemented in the transformers library (Wolf et al., 2020).
Table 1: LLaMA with RLHF experiments: key implementation details.

<table>
<thead>
<tr>
<th>Model</th>
<th>Transformer (Vaswani et al., 2017)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-training</td>
<td>LLaMA-7b (Touvron et al., 2023)</td>
</tr>
<tr>
<td>Instruction FT</td>
<td>Alpaca (Taori et al., 2023)</td>
</tr>
</tbody>
</table>

**RL procedure**

- **Fine-tuning strategy**: LoRA (Hu et al., 2022a) following Alpaca-LoRA (Wang, 2023)
- **LoRA alpha**: 16
- **LoRA dropout**: 0.05 following trl-peft (von Werra et al., 2020; Beeching et al., 2023)
- **Optimizer**: Adam (Kingma & Ba, 2015)
- **Learning rate**: 1.41e-5
- **Batch size**: 128
- **Output length**: Between 16 and 32
- **KL PPO**: 0.05 for summary tasks else 0.2
- **Epochs**: 2 for Reuter summary else 1
- **Hardware**: NVIDIA RTX A6000 49 Go
- **Compute budget**: 4000 GPUh

**Task name**  | **Description**  | **Prompt** | **Dataset** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reuter summary</strong></td>
<td>Generate a concise and clear summary of newspaper articles from Reuters.</td>
<td>“Generate a one-sentence summary of this post.”</td>
<td>Reuter news from (Ahmed, 2017; Ahmed et al., 2018) from news-summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>gpt2-reward-summarization trained here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bart-faithful-summary-detector (Chen et al., 2021)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Figure 2(a)</td>
</tr>
<tr>
<td><strong>Reddit summary</strong></td>
<td>Generate a concise and clear summary of posts from Reddit across a variety of topics (subreddits).</td>
<td>“Generate a one-sentence summary of this post.”</td>
<td>Reddit crawl from the TL;DR dataset (Völks et al., 2017) from summarize-from-feedback (Stiennon et al., 2020)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>gpt2-reward-summarization trained here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bart-faithful-summary-detector (Chen et al., 2021)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Figure 2(b)</td>
</tr>
<tr>
<td><strong>Helpful assistant</strong></td>
<td>Provide helpful and harmless answers to potentially complex and sensitive questions.</td>
<td>No prompt, only users’ questions.</td>
<td>Helpfulness and Harmlessness datasets (Bai et al., 2022a) from hh-rlhf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reward-model-deberta-v3-large-v2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reward-model-electra-large-discriminator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reward-model-deberta-v3-base-v2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reward-model-deberta-v3-base</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Figures 2(c) and 2(d)</td>
</tr>
</tbody>
</table>
D. Image-to-text: captioning with diverse rewards

D.1. Experimental details

We summarize the key implementation details of our captioning experiments in Table 2. In short, we took the state-of-the-art with a Swin Transformer (Liu et al., 2022) visual encoder and a block static expansion for efficiency. We investigate whether other rewards. That’s why in Figure 3(c) we consider the initialization as the network parametrization optimized for CIDEr. Thus optimizing CIDEr a second time does not help, nor in CIDEr neither in METEOR.

We summarize the key implementation details of our captioning experiments in Table 2. In short, we took the state-of-the-art with a Swin Transformer (Liu et al., 2022) visual encoder and a block static expansion for efficiency. We investigate whether other rewards. That’s why in Figure 3(c) we consider the initialization as the network parametrization optimized for CIDEr. Thus optimizing CIDEr a second time does not help, nor in CIDEr neither in METEOR.

<table>
<thead>
<tr>
<th>Model</th>
<th>ExpansionNetv2 (Hu et al., 2022b)</th>
<th>Swin Transformer (Liu et al., 2022)</th>
<th>ImageNet 22k (Deng et al., 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual encoder</td>
<td>ExpansionNetv2 (Hu et al., 2022b)</td>
<td>Swin Transformer (Liu et al., 2022)</td>
<td>ImageNet 22k (Deng et al., 2009)</td>
</tr>
<tr>
<td>Fine-tuning strategy</td>
<td>Cross-entropy then CIDEr RL (Rennie et al., 2017) on COCO (Lin et al., 2014)</td>
<td>Cross-entropy then CIDEr RL (Rennie et al., 2017) on COCO (Lin et al., 2014)</td>
<td>Cross-entropy then CIDEr RL (Rennie et al., 2017) on COCO (Lin et al., 2014)</td>
</tr>
<tr>
<td>RL algorithm</td>
<td>Usually frozen visual backbone, but end-to-end in Figure 10(d)</td>
<td>Usually frozen visual backbone, but end-to-end in Figure 10(d)</td>
<td>Usually frozen visual backbone, but end-to-end in Figure 10(d)</td>
</tr>
<tr>
<td>Dataset</td>
<td>COCO (Lin et al., 2014) and Karpathy split (Karpathy &amp; Fei-Fei, 2015)</td>
<td>COCO (Lin et al., 2014) and Karpathy split (Karpathy &amp; Fei-Fei, 2015)</td>
<td>COCO (Lin et al., 2014) and Karpathy split (Karpathy &amp; Fei-Fei, 2015)</td>
</tr>
<tr>
<td>Rewards</td>
<td>BLEU (with 1-gram or 4-grams), ROUGE, METEOR, CIDEr</td>
<td>BLEU (with 1-gram or 4-grams), ROUGE, METEOR, CIDEr</td>
<td>BLEU (with 1-gram or 4-grams), ROUGE, METEOR, CIDEr</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-5</td>
<td>1e-5</td>
<td>1e-5</td>
</tr>
<tr>
<td>Batch size</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Gradient accumulation</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Warmup</td>
<td>Anneal 0.8 during 1 epoch</td>
<td>Anneal 0.8 during 1 epoch</td>
<td>Anneal 0.8 during 1 epoch</td>
</tr>
<tr>
<td>Epochs</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Hardware</td>
<td>GPU V100 32G</td>
<td>GPU V100 32G</td>
<td>GPU V100 32G</td>
</tr>
<tr>
<td>Compute budget</td>
<td>1500 GPUh</td>
<td>1500 GPUh</td>
<td>1500 GPUh</td>
</tr>
</tbody>
</table>

D.2. Additional results

Figure 8: Additional results in captioning with more rewards, complementing Figure 3. Specifically, Figure 8(a) uses $R_1 = \text{BLEU4}$ and $R_2 = \text{ROUGE}$; then, with $R_1 = \text{BLEU1}$, Figure 8(b) uses $R_2 = \text{METEOR}$ and Figure 8(c) uses $R_2 = \text{CIDEr}$. In particular, the latter shows the failure when optimizing CIDEr; indeed, let’s recall that the pre-trained initialization (Hu et al., 2022b) has already been trained by optimizing CIDEr (Rennie et al., 2017). Thus optimizing CIDEr a second time does not help, nor in CIDEr neither in other rewards. That’s why in Figure 3(c) we consider the initialization as the network parametrization optimized for CIDEr.
(b) Results variances.

(c) \( \lambda \) outside of \([0, 1]\).

(d) End-to-end training.

Figure 10: Additional results in captioning with \( R_1 = BLEU_1 \) and \( R_2 = ROUGE \). In Figure 10(a), we investigate interpolating the fine-tuned networks with the pre-trained initialization as in WiSE (Wortsman et al., 2022b); this only reveals a small portion of the front. In contrast, the interpolation with \( \theta_{MORL} (\mu = 0.5) \) solution improves RS’s front: this highlights some limitations in Hypothesis 2 and strict Pareto optimality of RS. Adding the MORL solutions as intermediate weights may help interpolate between two weights too distant. This suggests some practical complementarity between RS and MORL: given a training budget larger than the number of rewards, one may learn a few MORL for varying \( 0 \leq \mu \leq 1 \), and then interpolate the obtained solutions. Figure 10(b) shows results’ variance with two RL trainings for BLEU, and two for ROUGE, each time with a different seed defining the data ordering and augmentations. Though we observe some randomness, the Hypothesis 1 is consistently validated. Moreover, it presents the fronts described when we interpolate weights fine-tuned on a shared reward, as in model soups (MS) (Wortsman et al., 2022a; Rame et al., 2022). This also only reveals a small portion of the spectrum of preferences, validating the need for diverse rewards to reveal the full Pareto front. In Figure 10(c), we report the front of the costly ensembling (Hansen & Salamon, 1990) of predictions (rather than of weights). Actually, ensembling performs better, but it cannot be fairly compared as its inference cost is doubled. Finally, Figure 10(d) shows the results when the networks are trained end-to-end, rather than keeping the backbone frozen. This validates the efficiency of rewarded soups in a new more general setting where all layers are trainable.
E. Text-to-image: diffusion models with RLHFs

E.1. Experimental details

Several works have studied the problem of aligning the output of diffusion models with human feedbacks (Lee et al., 2023; Wu et al., 2023b; Xu et al., 2023). Models are expected to understand specific visual control signals like colors, counts, and backgrounds more accurately after alignment. Notably, diffusion models can be fine-tuned to match human aesthetics preferences. As for any subjective metric, there is a variety of reward models that capture different aspects of aesthetic preference. These models are trained in a supervised setting to match human quality ratings collected on large image datasets (Xu et al., 2023), like the AVA dataset (Murray et al., 2012). As in the previous sections, RS allows to efficiently optimize multiple aesthetics reward models at test time, which allows adapting to the preferences of a single user.

We consider three metrics as rewards models: The cafe aesthetics model\(^1\), trained on 3500 real-life and anime/manga images; An aesthetic score predictor based on CLIP features\(^2\), trained on 250 000 images from the AVA dataset (Murray et al., 2012); we also experiment with a CLIP-based NSFW detector that estimates the probability of an image being "safe" by computing the cosine similarity with the embeddings of a set of "unsafe" words. The last two reward models are used to filter the LAION dataset (Schuhmann et al., 2021).

To fine-tune a diffusion model on a reward model \(R\), we first generate 10000 images with the pre-trained diffusion model and compute the rewards for every generated image. Then, we fine-tune the diffusion model on the reward-weighted negative log-likelihood (Lee et al., 2023):

\[
L = \mathbb{E}_{(x_0,Q) \in D, \epsilon \sim N(0,1), t \sim \text{Uniform}(0,T)} r(x_0) \| \epsilon_{\theta}(x_t, t, Q) - \epsilon \|^2
\] (19)

where \(\epsilon_{\theta}\) is the noise estimation network, \(T\) is the total number of training steps, \(r(x_0)\) is the reward of image \(x_0\) and \(Q\) is the text associated to image \(x_0\).

On-policy Reinforcement Learning would normally require to perform loops of image generation and model fine-tuning (Dong et al., 2023), but we only perform a single optimization loop for simplicity.

Implementation details. We use a 2.2B parameters diffusion model trained on an internal dataset of 300M images, which reaches similar generation quality as Stable Diffusion (Rombach et al., 2022) in terms of CLIP alignment and FID scores. For efficient finetuning, we only fine-tune 10% of the diffusion model’s weights (Xie et al., 2023) corresponding to the cross-attention layers and the bias/scaling parameters. For computational efficiency, we remove the 50% images with the worse scores, and rescale rewards linearly so that \(\min_{x_0 \in D'} r(x_0) = 0\) and \(\frac{1}{|D'|} \sum_{x_0 \in D'} r(x_0) = 1\). All models are fine-tuned with Adam (Kingma & Ba, 2015) for 4000 steps with a batch size of 32 and learning rate 5e-6. Fine-tuned checkpoints and checkpoints interpolated with RS are evaluated on 1000 images.

Table 3: Image generation experiments: key implementation details.

<table>
<thead>
<tr>
<th>Model</th>
<th>GLIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Fine-tuning cross-attention layers and biasSCALE parameters</td>
</tr>
<tr>
<td>Fine-tuning objective</td>
<td>Reward-weighted Diffusion Loss</td>
</tr>
<tr>
<td>Fine-tuning strategy</td>
<td>Adam (Kingma &amp; Ba, 2015)</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Generated with COCO prompts</td>
</tr>
<tr>
<td>Dataset</td>
<td>ava (Murray et al., 2012) and cafe</td>
</tr>
<tr>
<td>Rewards</td>
<td></td>
</tr>
<tr>
<td>Learning rate</td>
<td>5e-6</td>
</tr>
<tr>
<td>Batch size</td>
<td>64</td>
</tr>
<tr>
<td>Epochs</td>
<td>25</td>
</tr>
<tr>
<td>Hardware</td>
<td>Single GPU V100 32G</td>
</tr>
<tr>
<td>Compute budget</td>
<td>500 GPUh</td>
</tr>
</tbody>
</table>

\(^1\)available at https://huggingface.co/cafeai/cafe_aesthetic
\(^2\)available at https://github.com/christophschuhmann/improved-aesthetic-predictor/
E.2. Additional results

We show in Figure 11 the spider map when computing MORL and RS on all three metrics: ava, cafe and the nsfw detector. In this case, MORL has higher scores than RS on the ava and cafe scores. We speculate that this is because the nsfw is very different from aesthetics preferences and that it can be inversely correlated with image quality: we have indeed noticed that lower quality images result in higher scores for the nsfw metric, being less often flagged as unsafe.

F. Text-to-box: visual grounding

F.1. Experimental details

We show the implementation details in Table 4. We use an internal unified model (Wang et al., 2022a; Lu et al., 2022) which will be released soon. The model is pre-trained solely on public benchmarks, to solve a variety of multimodal tasks such as VQA, visual grounding and image captioning. It is then fine-tuned on RefCOCO+ dataset for visual grounding. During the last fine-tuning phase, we complement the cross-entropy loss with an additional REINFORCE (Williams, 1992) term rewarding accuracy when the object is of the considered size. This means that the loss for $\theta_{small}$ is $-(\log(\hat{y}) + 5 \times 1_{\text{area}(\hat{y}) \text{ is small}}) \times 1_{AUC(y, \hat{y}) > 0.5} \times \log(y)$ for an object with ground-truth box $\hat{y}$ and prediction $y$. The image is discretized into $1000 \times 1000$ bins before calculating the box areas. The task is illustrated in Figure 12.

Figure 12: Illustration of the Visual Grounding task. The RS model results from the average of $N = 3$ weights specialized to detect respectively small, medium and large objects. The model takes the text (one at a time) as input and outputs the bounding box (i.e., colored predictions) in the image region described by the text (the ground truths are shown in green). We show an example of small, medium and large predictions. The texts and image input are from the validation set of RefCOCO+ (Yu et al., 2016).
Table 4: Visual grounding experiments: key implementation details.

<table>
<thead>
<tr>
<th>Model</th>
<th>Unified Model (ResNet-101+BART (Lewis et al., 2020))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual encoder</td>
<td>ResNet-101</td>
</tr>
<tr>
<td>Pretraining</td>
<td>Cross-Entropy on Public datasets (VQA, VG, Captioning)</td>
</tr>
<tr>
<td>Finetuning</td>
<td>Cross-Entropy on RefCOCO+ (Yu et al., 2016)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RL procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuning strategy</td>
</tr>
<tr>
<td>Dataset</td>
</tr>
<tr>
<td>RL algorithm</td>
</tr>
<tr>
<td>Reward Small</td>
</tr>
<tr>
<td>Reward Medium</td>
</tr>
<tr>
<td>Reward Large</td>
</tr>
<tr>
<td>Optimizer</td>
</tr>
<tr>
<td>Learning rate</td>
</tr>
<tr>
<td>Batch size</td>
</tr>
<tr>
<td>Epochs</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
<tr>
<td>Compute budget</td>
</tr>
</tbody>
</table>

F.2. Additional results

Figure 13: Results in visual grounding on RefCOCO+ (Yu et al., 2016). We use REINFORCE (Williams, 1992) to improve directly the non-differentiable accuracy, i.e., predict boxes with IoU $> 0.5$ w.r.t. the ground-truth. Trainings are specialized on either small, medium or large objects. These experiments complement Figures 5(b) and 5(c). Finally, Figure 13(c) compares between cross-entropy (CE) supervised fine-tuning (with Cross-entropy CE) and REINFORCE RL fine-tuning, using RS and MORL.
G. Locomotion with diverse engineered rewards

G.1. Experimental details

Setup and task. This experiment consists in fine-tuning a policy that has already learned how to make an humanoid run on Brax physics engine (Freeman et al., 2021).

Pre-training. We used the Brax implementation of PPO (Schulman et al., 2017) algorithm to pre-train the base policy used fine-tuning (see Table 5). The goal task used for pre-training is to make a Humanoid run with the default dense reward implemented in Brax: \( R = \text{velocity} - 0.5 \cdot a_t^2 a_t \). This phase is also used to collected statistics about observations and normalize them before inputting to the model, which helps training a lot.

Fine-tuning. The pre-trained policy is saved while the value function is discarded. We use the normalization procedure inherited from the pre-training but freeze it. We keep the same environment. Two reward functions are designed: a risky one for \( R_1(t) = \text{velocity} \) and a cautious one where \( R_2(t) = \text{velocity} - a_t^2 a_t \). We make a grid-search on a few hyperparameters over 3 seeds (see the values between brackets in Table 5).

Table 5: Locomotion experiments: key implementation details.

<table>
<thead>
<tr>
<th>PPO Pre-training</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>5e8</td>
</tr>
<tr>
<td>Reward Scaling</td>
<td>1.0</td>
</tr>
<tr>
<td>Episode Length</td>
<td>1000</td>
</tr>
<tr>
<td>Normalize observations</td>
<td>True</td>
</tr>
<tr>
<td>Unroll Length</td>
<td>10</td>
</tr>
<tr>
<td>Discounting</td>
<td>0.99</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>5e-5</td>
</tr>
<tr>
<td>Entropy Cost</td>
<td>1e-3</td>
</tr>
<tr>
<td>Number of environments in parallel</td>
<td>4096</td>
</tr>
<tr>
<td>Batch Size</td>
<td>1024</td>
</tr>
<tr>
<td>Hardware</td>
<td>1GPU Tesla V100-SXM2-16GB</td>
</tr>
<tr>
<td>Runtime per experiment</td>
<td>80min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PPO Fine-tuning</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>1e8</td>
</tr>
<tr>
<td>Reward Scaling</td>
<td>1.0</td>
</tr>
<tr>
<td>Normalize observations</td>
<td>True</td>
</tr>
<tr>
<td>Unroll Length</td>
<td>10</td>
</tr>
<tr>
<td>Discounting</td>
<td>[0.97, 0.99, 0.999]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>(1e-5, 3e-5, 1e-4)</td>
</tr>
<tr>
<td>Entropy Cost</td>
<td>1e-3, 3e-3, 1e-2</td>
</tr>
<tr>
<td>Number of environments in parallel</td>
<td>4096</td>
</tr>
<tr>
<td>Batch Size</td>
<td>1024</td>
</tr>
<tr>
<td>Hardware</td>
<td>1GPU Tesla V100-SXM2-16GB</td>
</tr>
<tr>
<td>Runtime per experiment</td>
<td>20min</td>
</tr>
</tbody>
</table>

Model architecture

<table>
<thead>
<tr>
<th>Policy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>MLP</td>
</tr>
<tr>
<td>Nb of Layers</td>
<td>6</td>
</tr>
<tr>
<td>Hidden Size</td>
<td>512</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>MLP</td>
</tr>
<tr>
<td>Nb of Layers</td>
<td>5</td>
</tr>
<tr>
<td>Hidden Size</td>
<td>256</td>
</tr>
</tbody>
</table>
G.2. Additional results

Figure 14: Some other runs for the locomotion task when varying the seed / hyperparameters.