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 practical benefits of rewarded soups.
- Appendix B details some theoretical guarantees.
- Appendix C details our text-to-text generation experiments.
- Appendix D enriches our image captioning experiments.
- Appendix E enriches our image generation experiments.
- Appendix F enriches our visual grounding experiments.
- Appendix G 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

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.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 Kirk *et al.* [50] and in Hayes *et al.* [52].

Indeed, the user's true reward is highly uncertain before training. This "semi-blind" [52] 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" [50], 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." [50]. Moreover, biased are caused by chaotic engineering choices, and "are exacerbated by a lack of […] documentation" [50]. In contrast, our approach makes **personalization explicit**, as argued by [50]. Moreover, we could **support decision-making** to find a good balance between (potentially conflicting) parties' interests. This value pluralism [163] can lead to **fairer** and more equitable outcomes [53, 164]. Single-policy cannot adapt to test time requirements; in contrast, RS facilitates personalized assistances [161]. This is all the more important as human preferences change from time to time. In this **dynamic utility function** scenario, RS can quickly adapt with fewer data, by simply adjusting the λ to match new preferences (rather than the full network). Finally, RS could also improve the **interpretability** and **explainability** of the decisions. Letting the users decide would make the process more **transparent** [165], which is essential to ensure that the development process is fair, unbiased, and inclusive [166].

A.2 Compared to multi-policy approaches

The main reason why existing 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 [135] 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 [167], because of conflicting gradients [168, 169] or different variance scales [170, 171]. This has been tackled in RL, but so far mostly for games such as ATARI [172]. 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 [173, 174]. Moreover, as shown in [141] 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" [175]. This facilitates its use in **federated learning** scenario [162] where the data should remain private. Actually, RS follows the **updatable machine learning paradigm** [176], "allowing for the collaborative creation of increasingly sophisticated AI system" [67]. 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 θ maximizing $\hat{R}$, we first show that θ 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 θ' would produce a better policy than θ for $\hat{R} = \sum_i \hat{\mu}_i R_i$ and thus the contradiction. Finally, as θ is on the PF and by definition of a PCS, there exists λ s.t. $\forall k, R_k(\sum_i \lambda_i \cdot \theta_i) = R_k(\theta)$. $\square$

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 $\mathbb{I}_d$. 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 $i \in \{1, \dots, N\}$:*

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

where $\eta_i \in \mathbb{R}_+^*$ and θ_i is the global maximum for reward R_i.

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

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

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

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 $\{\theta_1, \dots, \theta_N\}$. $\square$

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 θ_i for $i \in \{1, \dots, N\}$. 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 Δ_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* $i \in \{1, 2\}$*:*

$$\forall \theta = (\theta^1, \ldots, \theta^d) \in \Theta, \quad R_i(\theta) = R_i(\theta_i) - \sum_{j=1}^{d} \eta_i^j (\theta^j - \theta_i^j)^2, \tag{2}$$

where $(\eta_i^1, \ldots \eta_i^d) \in \{\mathbb{R}_+^*\}^d$ *and* $\theta_i = (\theta_i^1, \ldots, \theta_i^d)$ *is the global maximum for reward* R_i*.*

Remark 4. *This diagonal Assumption 2 of the Hessian is common: for example in optimization [177, 178], to prune networks [179] or in out-of-distribution generalization [180]. This strong assumption is supported by the empirical observation [181] 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*

$$\Delta R_{\hat{\mu}} = \max_{\theta \in \Theta} R_{\hat{\mu}}(\theta) - \max_{\lambda \in [0,1]} R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2). \tag{3}$$

$\Delta R_{\hat{\mu}}$ *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:*

$$\Delta R_{\hat{\mu}} \leq \frac{\hat{\mu}^2 (1 - \hat{\mu})^2 (M\Delta_1 - \Delta_2)(M\Delta_2 - \Delta_1)}{(\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M)((1 - \hat{\mu})\Delta_1 + \hat{\mu}\Delta_2)}, \tag{4}$$

where $M = \max_{j \in \{1, \ldots, d\}} \max\left(\frac{\eta_1^j}{\eta_2^j}, \frac{\eta_2^j}{\eta_1^j}\right)$ *is the maximum of eigenvalues ratio,* $\Delta_1 = R_1(\theta_1) - R_1(\theta_2)$ *and* $\Delta_2 = R_2(\theta_2) - R_2(\theta_1)$*.*

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

$$\Delta R_{\hat{\mu}} \leq \frac{\hat{\mu}^2 (1 - \hat{\mu})^2 (M - 1)^2}{\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M} \Delta_1 \tag{5}$$

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 $\hat{\theta}$ maximizing $R_{\hat{\mu}}(\theta)$ for θ on the full set of weights Θ. Second, we find $\bar{\lambda}$ 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 Θ. Denoting $S = (1 - \hat{\mu}) \times R_1(\theta_1) + \hat{\mu} \times R_2(\theta_2)$, we have for all $\theta \in \Theta$:

$$R_{\hat{\mu}}(\theta) = S - \sum_{j=1}^{d} \left((1 - \hat{\mu}) \eta_1^j \left(\theta^j - \theta_1^j \right)^2 + \hat{\mu} \eta_2^j \left(\theta^j - \theta_2^j \right)^2 \right) \tag{6}$$

Since $R_{\hat{\mu}}$ is a sum of concave quadratic functions, it has a unique global maximum reached at a point we note $\hat{\theta} = \left(\hat{\theta}^1, \ldots, \hat{\theta}^d\right)$. The global maximum can be computed by differentiating $R_{\hat{\mu}}$ with respect to each variable θ^j, which gives:

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

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

$$\hat{\lambda}^j = \frac{\hat{\mu} \eta_2^j}{(1 - \hat{\mu}) \eta_1^j + \hat{\mu} \eta_2^j} \in [0, 1]. \tag{7}$$

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 λ:

$$\begin{aligned} R_{\hat{\mu}}(\theta) &= S - \sum_{j=1}^{d}\left(\left((1-\hat{\mu})\eta_1^j + \hat{\mu}\eta_2^j\right)\left(\lambda - \hat{\lambda}^j\right)^2 + \frac{\hat{\mu}(1-\hat{\mu})\eta_1^j\eta_2^j}{(1-\hat{\mu})\eta_1^j + \hat{\mu}\eta_2^j}\right)\left(\theta_1^j - \theta_2^j\right)^2 \\ &= R_{\hat{\mu}}(\hat{\theta}) - \sum_{j=1}^{d} p_j\left(\lambda - \hat{\lambda}^j\right)^2 \end{aligned} \tag{8}$$

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 $\bar{\lambda}$ maximizing $R_{\hat{\mu}}$ equal to

$$\bar{\lambda} = \frac{\sum_{j=1}^{d} p_j \hat{\lambda}^j}{\sum_{j=1}^{d} p_j}. \tag{9}$$

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

Third step. Applying Equation (8) to $\bar{\lambda}$ gives:

$$\Delta R_{\hat{\mu}} = R_{\hat{\mu}}(\hat{\theta}) - R_{\hat{\mu}}\big((1-\bar{\lambda})\cdot\theta_1 + \bar{\lambda}\cdot\theta_2\big) \tag{10}$$

$$= \sum_{j=1}^{d} p_j\left(\bar{\lambda} - \hat{\lambda}^j\right)^2 \tag{11}$$

$$= \left(\sum_{j=1}^{d} \frac{p_j}{\sum_{i=1}^{n} p_i}\left(\bar{\lambda} - \hat{\lambda}^j\right)^2\right)\left(\sum_{j=1}^{n} p_j\right) \tag{12}$$

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

$$\sum_{j=1}^{d} p_j = (1-\hat{\mu})\Delta_1 + \hat{\mu}\Delta_2. \tag{13}$$

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 Λ with $\mathbb{P}(\Lambda = \hat{\lambda}_i) = \frac{p_i}{\sum_{j=1}^{n} p_j}$; then, $\bar{\lambda}$ from Equation (9) is actually the expectation of Λ. Then, we can apply the **Bhatia-Davis inequality**, as recalled in Equation (14), on the variance of a bounded random variable $a \leq \Lambda \leq b$:

$$Var(\Lambda) \leq (b - \mathbb{E}(\Lambda))(\mathbb{E}(\Lambda) - a) \tag{14}$$

Therefore Equation (12) is bounded by:

$$\Delta R_{\hat{\mu}} \leq \left(\max_{1\leq j\leq d} \hat{\lambda}^j - \bar{\lambda}\right)\left(\bar{\lambda} - \min_{1\leq j\leq d} \hat{\lambda}^j\right)((1-\hat{\mu})\Delta_1 + \hat{\mu}\Delta_2). \tag{15}$$

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:

$$\frac{\hat{\mu}}{(1-\hat{\mu})M + \hat{\mu}} \leq \hat{\lambda}^j \leq \frac{\hat{\mu}M}{(1-\hat{\mu}) + \hat{\mu}M}, \tag{16}$$

and thus:

$$\Delta R_{\hat{\mu}} \leq \left(\frac{\hat{\mu}M}{1+\hat{\mu}(M-1)} - \bar{\lambda}\right)\left(\bar{\lambda} - \frac{\hat{\mu}}{M - \hat{\mu}(M-1)}\right)((1-\hat{\mu})\Delta_1 + \hat{\mu}\Delta_2). \tag{17}$$

Finally, noting that $\Delta_i = \sum_{j=1}^{d} \eta_i^j\left(\theta_2^j - \theta_1^j\right)^2$, we deduce from Equation (9) that $\bar{\lambda} = \frac{\hat{\mu}\Delta_2}{(1-\hat{\mu})\Delta_1 + \hat{\mu}\Delta_2}$. Replacing this in the previous Equation (17) gives the final Equation (4), concluding the proof. $\square$

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 [64] remove this constraint, with interpolating coefficients proportional to the eigenvalues of the Fisher matrices [182], which actually approximate the eigenvalues of the Hessian [183, 184]. Combining [64] and our RS is a promising research direction, the key issue being the computation of the Fisher matrices [185] 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 $M \to \infty$.

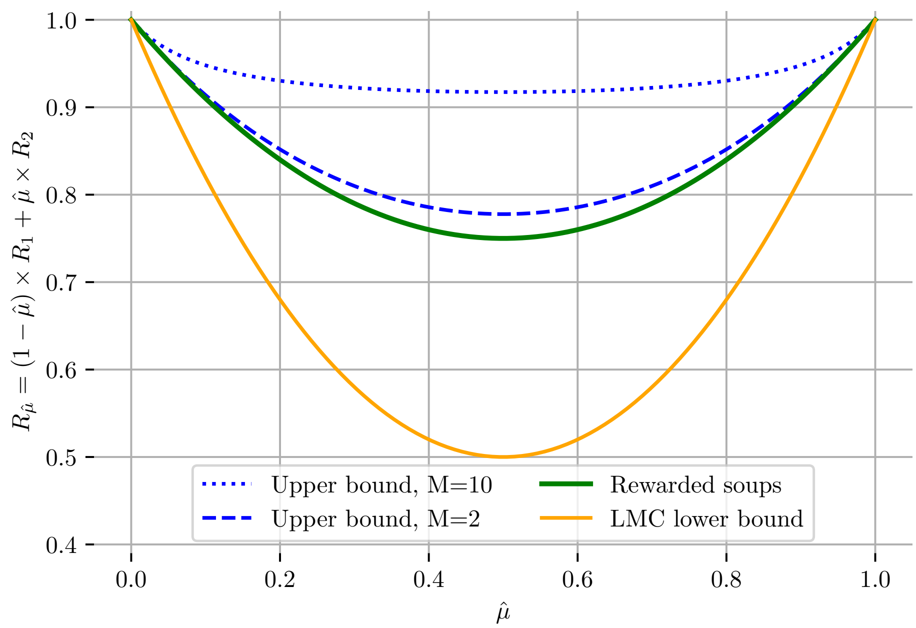


Figure 7: Illustration of the bound given by Lemma 3 under Assumption 2. For simplicity, we showcase the case where $R_1(\theta_1) = R_2(\theta_2) = 1$, $R_1(\theta_2) = R_2(\theta_1) = 0$, thus $\Delta_1 = \Delta_2 = 1$. In green, we plot the rewards obtained with rewarded soups for the optimal $\bar{\lambda}$, i.e., $R_{\hat{\mu}}\big((1-\bar{\lambda}) \cdot \theta_1 + \bar{\lambda} \cdot \theta_2\big)$, whose value is independent of M in this case. In blues, we plot the maximum value of $\mathcal{R}_{\hat{\mu}}$ 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 $(1-\hat{\mu})(1-\bar{\lambda})R_1(\theta_1) + \hat{\mu}\bar{\lambda}R_2(\theta_2)$. As RS outperforms this lower bound, it validates Hypothesis 1 in this case.

B.3 Similarity between weight interpolation and functional ensembling

Lemma 4 (λ-interpolation of weights approximates the λ-ensembling of predictions. Adapted from [62, 63, 94].)**.** *Given θ_1 and θ_2 optimized for R_1 and R_2 s.t. they remain close, i.e., $\|\theta_1 - \theta_2\|_2 \approx 0$. Denoting θ_λ the interpolated weights $\theta_\lambda = (1-\lambda)\cdot\theta_1 + \lambda\cdot\theta_2$ and f_λ the ensembling of predictions $f_\lambda(\cdot) = (1-\lambda)\cdot f(\cdot,\theta_1) + \lambda\cdot f(\cdot,\theta_2)$:*

$$f(\cdot,\theta_\lambda) \approx f_\lambda(\cdot)$$

and for $k \in \{1,2\}$:

$$R_k(f(\cdot,\theta_\lambda)) \approx R_k(f_\lambda(\cdot))$$

Proof. This proof follows [63] and has two components.

Functional approximation. First, we perform a Taylor expansion at the first order of the models' predictions w.r.t. parameters θ for $x \in T$:

$$\begin{aligned} f(x,\theta_1) &= f(x,\theta_\lambda) + \nabla_\theta f(x,\theta_\lambda)^\intercal(\theta_1 - \theta_\lambda) + \mathcal{O}\Big(\|\theta_1-\theta_\lambda\|_2^2\Big) \\ &= f(x,\theta_\lambda) + \nabla_\theta f(x,\theta_\lambda)^\intercal(\lambda\cdot\theta_1 - \lambda\cdot\theta_2) + \mathcal{O}\Big(\|\theta_1-\theta_2\|_2^2\Big) \end{aligned}$$

and similarly:

$$f(x,\theta_2) = f(x,\theta_\lambda) + \nabla_\theta f(x,\theta_\lambda)^\intercal((\lambda-1)\cdot\theta_1 + (1-\lambda)\cdot\theta_2) + \mathcal{O}\Big(\|\theta_1-\theta_2\|_2^2\Big)$$

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

$$f_\lambda(x) = (1-\lambda)\cdot f(x,\theta_1) + \lambda\cdot f(x,\theta_2) = f(x,\theta_\lambda) + \mathcal{O}\Big(\|\theta_1-\theta_2\|_2^2\Big). \tag{18}$$

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):

$$\begin{aligned} R_k(f_\lambda(x)) &= R_k(f(x,\theta_\lambda)(x)) + \mathcal{O}(\|f_\lambda(x) - f(x,\theta_\lambda)\|_2) \\ &= R_k(f(x,\theta_\lambda)(x)) + \mathcal{O}\Big(\|\theta_1-\theta_2\|_2^2\Big). \end{aligned}$$

We obtain the results when θ_1 and θ_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 [45]; then low-rank adapters [81] were fine-tuned on Alpaca [22] to follow instructions. We eventually fine-tune via PPO on the different considered tasks. Our code is adapted from [80]; 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 [76]. For example in summarization tasks, R_1 was open-sourced in an effort to reproduce the Summarize from Human Feedback paper [12], while R_2 [85] aimed at improved "faithfulness in abstractive summarization with contrast candidate generation". For other dialog tasks, we mostly rely on different reward models from OpenAssistant [86]. 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 [76].

Table 1: LLaMA with RLHF experiments: key implementation details.

	Model
Architecture	Transformer [70]
Pre-training	LLaMA-7b [45]
Instruction FT	Alpaca [22]
	RL procedure
Fine-tuning strategy	LoRA [81] *following Alpaca-LoRA [186]*
LoRA alpha	16
LoRA dropout	0.05 *following trl-peft [79, 80]*
Optimizer	Adam [178]
Learning rate	1.41e-5
Batch size	128
Output length	Uniformly sampled between 16 and 32
RL algorithm	PPO [78]
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	**Reuter summary**
Description	Generate a concise and clear summary of newspaper articles from Reuters.
Prompt	"Generate a one-sentence summary of this post."
Dataset	Reuter news from [82, 187] from news-summary
R_1	gpt2-reward-summarization trained here.
R_2	bart-faithful-summary-detector [85]
Figure	Figures 1(b) and 2(a)
Task name	**Reddit summary**
Description	Generate a concise and clear summary of posts from Reddit across a variety of topics (subreddits).
Prompt	"Generate a one-sentence summary of this post."
Dataset	Reddit crawl from the TL;DR dataset [83] from summarize-from-feedback [12]
R_1	gpt2-reward-summarization trained here.
R_2	bart-faithful-summary-detector [85]
Figure	Figure 2(b)
Task name	**Stack Exchange**
Description	Answer accurately to technical questions from Stack Exchange.
Prompt	No prompt, only users' questions.
Dataset	Q&A from Stack Exchange [84, 188] from stack-exchange-preferences
R_1	reward-model-deberta-v3-base
R_2	reward-model-electra-large-discriminator
Figure	Figure 2(c)
Task name	**Movie review**
Description	Generate movie reviews that accurately describe a movie.
Prompt	"Generate a movie review."
Dataset	IMDB reviews [189] from IMDB
R_1	reward-model-deberta-v3-base
R_2	reward-model-electra-large-discriminator
Figure	Figure 2(d)
Task name	**Helpful assistant**
Description	Provide helpful and harmless answers to potentially complex and sensitive questions.
Prompt	No prompt, only users' questions.
Dataset	Helpfulness and harmlessness datasets [41] from hh-rlhf
R_1	reward-model-deberta-v3-large-v2
R_2	reward-model-electra-large-discriminator
R_3	reward-model-deberta-v3-base-v2
R_4	reward-model-deberta-v3-base
Figure	Figures 2(e) and 2(f)

D Image-to-text: captioning with diverse statistical rewards

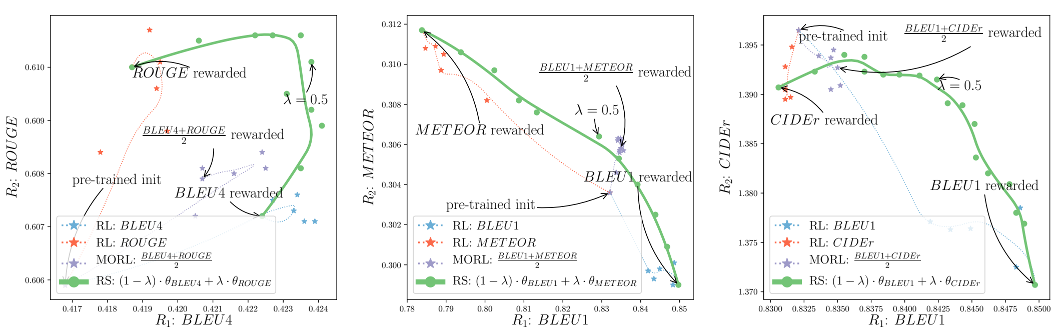
D.1 Experimental details

We summarize the key implementation details of our captioning experiments in Table 2. In short, we took the state-of-the-art network [90] for captioning on COCO, fine-tuned with their code and only changed the reward. In more details, since the *self-critical* paper [24] (a variant of REINFORCE [92] with a specific estimation of the baseline score) it is now common in captioning to optimize the CIDEr reward [31] after a first step of supervised fine-training. The recent ExpansionNetv2 [90] follows this strategy to reach state-of-the-art results, with a Swin Transformer [91] visual encoder and a block static expansion for efficiency. We investigate whether additional RL trainings on the other traditional statistical metrics can help. We use the code from [90] and their hyperparameters, only reducing the batch size from 24 to 18 to fit in our GPUs and consequently adapting the learning rate.

Table 2: Captioning experiments: key implementation details.

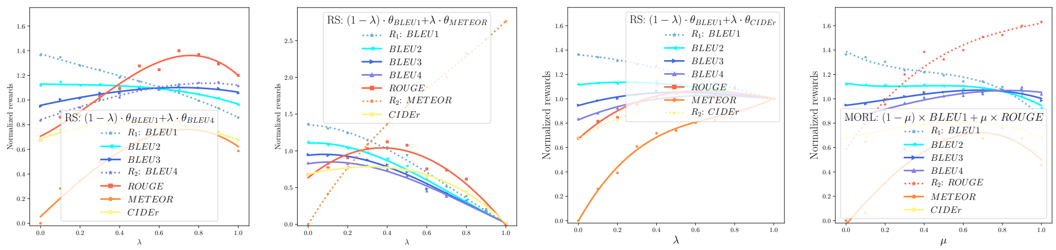
Model	
Architecture	ExpansionNetv2 [90]
Visual encoder	Swin Transformer [91]
Visual encoder pre-training	ImageNet 22k [190]
Fine-tuning	Cross-entropy then CIDEr RL [24] on COCO [88]
RL procedure	
Fine-tuning strategy	Usually frozen visual backbone, but end-to-end in Figure 10(d)
RL algorithm	Self-critical [24], a variant of REINFORCE [92]
Optimizer	Radam [191]
Dataset	COCO [88] and Karpathy split [93]
Rewards	BLEU [29] (with 1-gram or 4-grams), ROUGE [30], METEOR [89], CIDEr [31]
Learning rate	1e-5
Batch size	18
Gradient accumulation	2
Warmup	Anneal 0.8 during 1 epoch
Epochs	6
Hardware	GPU V100 32G
Compute budget	1500 GPUh

D.2 Additional results



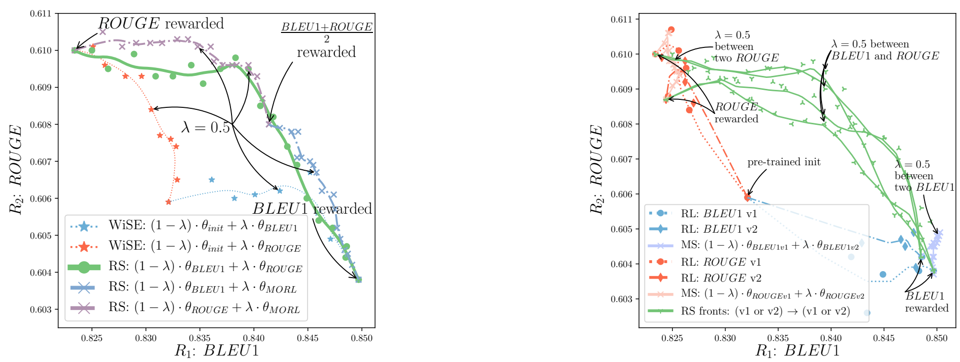
(a) BLEU4 and ROUGE. (b) BLEU1 and METEOR. (c) BLEU1 and CIDEr.

Figure 8: Additional results in captioning with more rewards, complementing Figure 3. Specifically, Figure 8(a) uses $R_1 = BLEU4$ and $R_2 = ROUGE$; then, with $R_1 = BLEU1$, Figure 8(b) uses $R_2 = METEOR$ and Figure 8(c) uses $R_2 = CIDEr$. In particular, the latter shows the failure when optimizing CIDEr; indeed, let's recall that the pre-trained initialization [90] has already been trained by optimizing CIDEr [24]. Thus optimizing CIDEr a second time does not help, neither in CIDEr nor in other rewards. That's why in Figure 3(c) we consider the initialization as the network parametrization optimized for CIDEr.

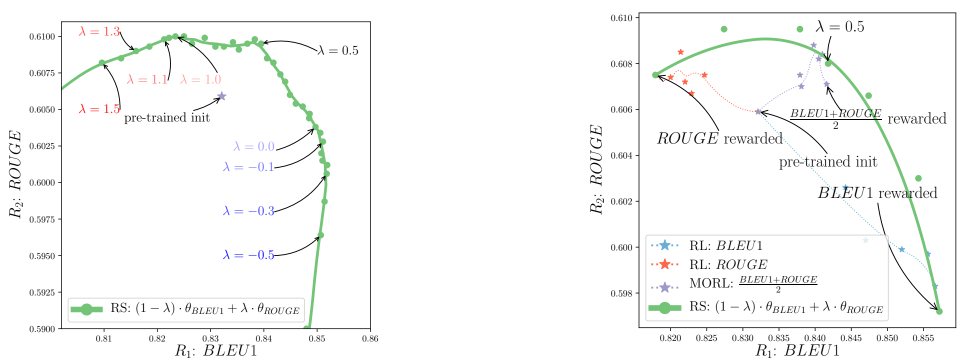


(a) $R_2 = BLEU4$. (b) $R_2 = METEOR$. (c) $R_2 = CIDEr$. (d) MORL.

Figure 9: Additional results in captioning when measuring performances on all rewards and varying the interpolating coefficients, complementing Figure 4(b). In Figures 9(a) to 9(c), we extend the results for RS with $R_1 = BLEU1$ and for varying R_2; the optimal λ depends on the similarity between the evaluation metric and R_1 and R_2. We also see in Figure 9(c) that all rewards are normalized to 1 for the CIDEr-initialization. In Figure 9(d), we perform the same analysis for MORL while varying the weighting μ over the proxy rewards $R_1 = BLEU1$ and $R_2 = ROUGE$; we recover similar curves than in Figure 4(b) for RS.



(a) Exploring new WI strategies. (b) Results variances and model soups (MS).



(c) Extrapolation with λ outside of $[0, 1]$. (d) End-to-end training.

Figure 10: Additional results in captioning with $R_1 = BLEU1$ and $R_2 = ROUGE$. In Figure 10(a), we investigate interpolating the fine-tuned networks with the pre-trained initialization as in WiSE [192]; this only reveals a small portion of the front. In contrast, the interpolation with θ_{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) [62, 63]. This also only reveals a small portion of the spectrum of preferences, validating the need of diverse rewards to satisfy all users' preferences. Figure 10(c) presents the extrapolation results when λ goes outside of $[0, 1]$. This suggests that we can artificially reduce a reward with negative coefficients, as studied in [141]. 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 diverse RLHFs

E.1 Experimental details

Task description. Several works have studied the problem of aligning the output of diffusion models with human feedbacks [25, 26, 33]. Notably, diffusion models can be fine-tuned to match human aesthetic perception. As for any subjective metric, there is a variety of reward models capturing different aesthetics. In our experiments, the two first reward models were trained in a supervised setting to match human quality ratings collected on large image datasets. Specifically, the first R_1 is the *ava* aesthetic model, available here, trained on 250.000 images from the AVA dataset [97], based on CLIP features. The second R_2 is the *cafe* aesthetic model, available here, trained on 3500 real-life and anime/manga images. Moreover, in Figure 11, we also consider a *nsfw* detector, estimating the probability of an image being *safe* by computing the cosine similarity with the CLIP embeddings of a set of *unsafe* words, as already done to filter the LAION dataset [193].

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 [96] in terms of CLIP alignment and FID scores on prompts from the 5000 images of the COCO test dataset (CLIPScore 30.0 vs 30.2 for Stable Diffusion, FID 19.0 vs 19.1 for Stable Diffusion). Given a reward model R, we first generate 10000 images with the pre-trained diffusion model on prompts from the COCO dataset, and compute the rewards for every generated image. For computational efficiency, we keep only a dataset $\mathcal{D}'$ containing the 50% images with the best scores, and rescale rewards R linearly into r so that $\min_{\mathbf{x}_0 \in \mathcal{D}'} r(x_0) = 0$ and $\frac{1}{|\mathcal{D}'|} \sum_{\mathbf{x}_0 \in \mathcal{D}'} r(x_0) = 1$. Then, we **fine-tune the diffusion model** on the reward-weighted negative log-likelihood [25]:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x}_0, Q) \in \mathcal{D}, \epsilon \sim \mathcal{N}(0,1), t \sim Uniform(0,T)} \quad r(\mathbf{x}_0) \times \|\epsilon_\theta(\mathbf{x}_t, t, Q) - \epsilon\|^2, \tag{19}$$

where ϵ_θ is the noise estimation network, T is the total number of training steps, $r(\mathbf{x}_0)$ is the rescaled reward of image $\mathbf{x}_0$ and Q is the text associated to image $\mathbf{x}_0$. As a side note, on-policy RL would require performing loops of image generations and model fine-tunings [194], but we only perform a single *offline* iteration for simplicity. Moreover, for efficiency, we only fine-tune 10% of the diffusion model's weights [98] corresponding to the cross-attention layers and the bias/scaling parameters. As further described in Table 3, we apply the Adam [178] optimizer for 4000 steps with a batch size of 64 and a learning rate of 5e-6. To report results for each model (fine-tuned or interpolated via RS), we generate 1000 images from a held-out set of COCO prompts and then we average the scores given by the reward models. To reduce the variance in image generation, each prompt has a unique seed for all models, so that the input noise given to the diffusion model only depends on the text prompt.

Table 3: Image generation experiments: key implementation details.

Model	
Architecture	GLIDE (2.2B parameters)
Pre-training	Internal dataset of 300M captioned images
RL Procedure	
Fine-tuning objective	Reward-weighted diffusion loss
Fine-tuned parameters	Cross-attention layers and bias/scale
Optimizer	Adam [178]
Dataset	Generated with COCO prompts
Rewards	*ava* [97] and *cafe* and *nsfw*
Learning rate	5e-6
Batch size	64
Epochs	25
Hardware	Single GPU V100 32G
Compute budget	500 GPUh

E.2 Additional results

RS can trade-off between the two aesthetic rewards in Figure 5(a), allowing adaptation to the user's preferences at test time. Yet, we show some limitations in the spider map of Figure 11, when computing MORL and RS on all three rewards: *ava*, *cafe* and also the *nsfw*. In this case, MORL has higher scores than RS. We speculate this is because the *nsfw* is very different from aesthetic preferences. Actually, the *nsfw* is inversely correlated with image quality: lower quality images result are less flagged as *unsafe*. This shows some limitations of weight interpolation when combining antagonist rewards. An improved strategy would first learn the MORL of the $N = 3$ rewards, and then optimize each reward independently from this improved initialization, before applying RS.

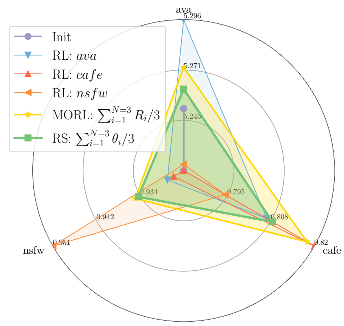


Figure 11: Image generation: spider map, with *ava*, *cafe* and *nsfw* reward models.

E.3 Visualization of generated images from interpolated models

We show in Appendix E.3 images generated by rewarded soups when varying the interpolation coefficient λ between the two models fine-tuned for the *ava* and the *cafe* reward models. You can find additional qualitative results of this experiment on our anonymized website.

Prompt	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1.0$
a dog stands inside of a boat as it stares at a camera						
A family room with wood floor and beige walls and a mattress leaning against a stone wall.						
A man sitting on top of a chair holding up a cell phone.						
A little child standing next to a toy dump truck.						
A man sitting underneath an umbrella and other structures.						

Figure 12: Visualization of images generated with rewarded soups for a varying interpolation coefficient λ between the two models fine-tuned for the *ava* (corresponding to $\lambda = 0$) and *cafe* (corresponding to $\lambda = 1$) reward models. We can see that all interpolated models produce images of similar quality compared to finetuned models, demonstrating linear mode connectivity between the two fine-tuned models.

F Text-to-box: visual grounding of objects with diverse sizes

F.1 Experimental details

We show the implementation details in Table 4. We use an internal unified model [100, 195] 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 [92] term rewarding accuracy when the object is of the considered size. This means that the loss for θ_{Small} is $-\big(log(\hat{y}) + 5 \times 1_{\{\text{area}(\hat{y}) \text{ is small}\}} \times 1_{AUC(y,\hat{y})>0.5} \times log(y)\big)$ for an object with ground-truth box $\hat{y}$ and prediction y. The image is discretized into 1000×1000 bins before calculating the box areas. The task is illustrated in Figure 13.

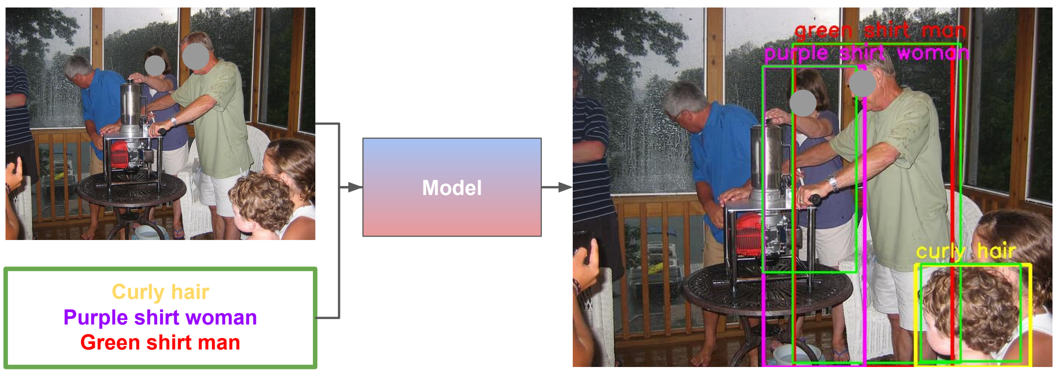
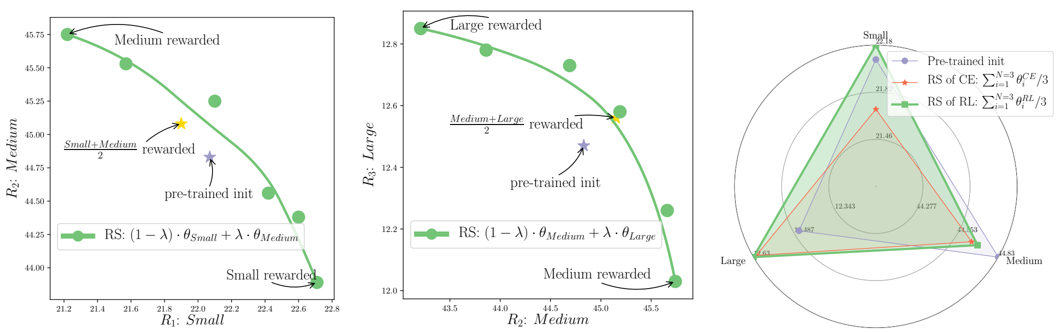


Figure 13: Illustration of the visual grounding task. The RS model results from the average of $N = 3$ weights specialized to detect respectively small, medium and large objects. The model takes a text (one description at a time) as input and outputs the bounding box in the corresponding region of the image. We show an example of small, medium and large predictions, and the associated ground truths in green. These texts and image are from the validation set of RefCOCO+ [99].

Table 4: Visual grounding experiments: key implementation details.

Model	
Architecture	Unified Model (ResNet-101+BART [196])
Visual encoder	ResNet-101
Pre-training	Cross-Entropy on Public datasets (VQA, VG, Captioning)
Supervised fine-tuning	Cross-Entropy on RefCOCO+ [99]
RL procedure	
Fine-tuning strategy	end-to-end
Dataset	RefCOCO+ [99]
RL algorithm	Cross-entropy + $5\times$ REINFORCE
Reward Small	IoU>0.5 for object with area < 30000
Reward Medium	IoU>0.5 for object with $30000 \leq$ area < 100000
Reward Large	IoU>0.5 for object with $100000 \leq$ area
Optimizer	Adam
Learning rate	3e-5
Batch size	256
Epochs	10
Hardware	8 GPU 60GB
Compute budget	800 GPUh

F.2 Additional results



(a) Small and Medium. (b) Medium and Large. (c) CE vs. RL.

Figure 14: Results in visual grounding on RefCOCO+ [99]. We use REINFORCE [92] to improve directly the non-differentiable accuracy, i.e., predict boxes with IoU>0.5 w.r.t. the ground-truth. Fine-tunings are specialized on either small, medium, or large objects. These experiments complement Figures 5(b) and 5(c). Finally, Figure 14(c) motivates the use of RL to fine-tune on different sizes. Indeed, the results for (the proposed) RS of RL are significantly better than the results for RS of CE, where we average weights specialized on different sizes by fine-tuning with cross-entropy (rather than with REINFORCE).

G Locomotion with diverse engineered rewards

Task description. This experiment takes on the intricate challenge of controlling a running humanoid in the Brax [106] physics engine. The complexities involved in achieving natural or fast movement in continuous control environments serve as a testament to the robustness of our approach. The fine-tuning procedure is carried out on two distinct reward functions, with the aim of refining the running behavior of the humanoid, potentially resulting in smoother motion patterns. You can find qualitative results of this experiment on our anonymized website.

Pre-training. According to Remark 1, the LMC requires pre-training the base policy before fine-tuning. Thus, as the pre-training task, we use the default dense reward implemented in Brax: $R = velocity - 0.1 \times \sum_t a_t^2$. This pre-training phase also serves to collect statistics about observations and normalize them before inputting to the model (as it facilitates training). We used the Brax implementation of PPO [78]. The pre-trained policy is saved while the value function is discarded.

Fine-tuning. We keep the same environment as in pre-training. We also use the normalization procedure inherited from pre-training but freeze the statistics. Two reward functions are designed: a *risky* one for $R_1 = velocity$ and a *cautious* one where $R_2 = velocity - \sum_t a_t^2$. We tried a few hyperparameters (see the values in brackets in Table 5) but results (see Figure 15) remain close and consistently validate our working hypotheses.

Table 5: Locomotion experiments: key implementation details.

PPO Pre-training	
Interactions	5e8
Reward Scaling	1.0
Episode Length	1000
Unroll Length	10
Discounting	0.99
Learning Rate	5e-5
Entropy Cost	1e-3
Number of environments in parallel	4096
Batch Size	1024
Hardware	1GPU Tesla V100-SXM2-16GB
Runtime per experiment	80min
PPO Fine-tuning	
Interactions	1e8
Reward Scaling	1.
Normalize observations	True
Unroll Length	10
Discounting	{0.97, 0.99, 0.999}
Learning Rate	{1e-5, 3e-5, 1e-4}
Entropy Cost	{1e-3, 3e-3, 1e-2}
Number of environments in parallel	4096
Batch Size	1024
Hardware	1GPU Tesla V100-SXM2-16GB
Runtime per experiment	20min
Model architecture	
Policy	
Architecture	MLP
Nb of Layers	6
Hidden Size	512
Value	
Architecture	MLP
Nb of Layers	5
Hidden Size	256

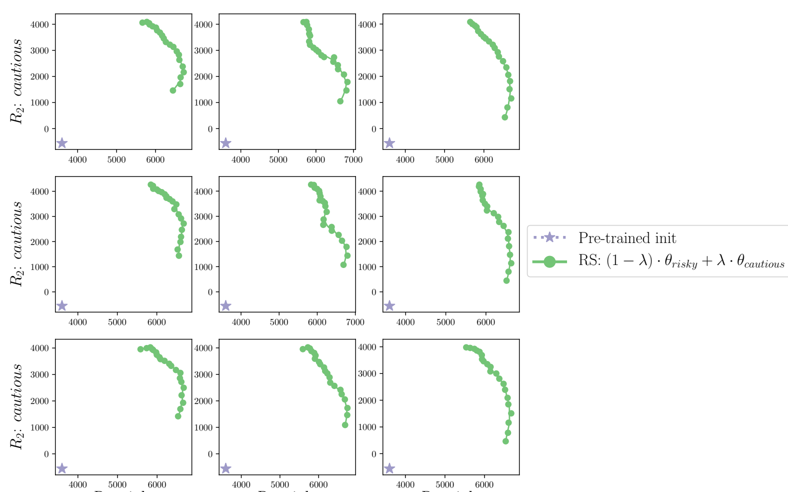


Figure 15: Analysis of results' variance for the locomotion task when varying the hyperparameters. Each column i corresponds to the i-th θ_{risky}, interpolated in case (i, j) towards the j-th $\theta_{cautious}$. The Figure 6 is actually the plot from case $(1, 1)$.