EVarEst: Error-Variance penalized Estimation for Deep Reinforcement Learning

Yann Berthelot
yann.berthelot@inria.fr
Saint-Gobain Research
UMR 9189 - CRIStAL
Université de Lille
Inria, CNRS, Centrale Lille
Univ. Lille, CRIStAL, CNRS

Timothée Mathieu
timothee.mathieu@inria.fr
UMR 9189 - CRIStAL
Université de Lille
Inria, CNRS, Centrale Lille

Riad Akrour
riad.akrour@inria.fr
UMR 9189 - CRIStAL
Université de Lille
Inria, CNRS, Centrale Lille

Philippe Preux
philippe.preux@inria.fr
UMR 9189 - CRIStAL
Université de Lille
Inria, CNRS, Centrale Lille

Abstract

Modern reinforcement learning (RL) algorithms often rely on estimating the value of state-action pairs for a given policy, typically using neural networks to model this value. However, this estimation is hindered by intrinsic policy non-stationarity during training. This leads to increasing errors as the model ability to adapt to new policies degrades over time. To address this issue, we introduce EVarEst, a novel objective function that enhances the standard mean squared error (MSE) by incorporating a weighted penalty on the variance of residual errors. Unlike traditional penalties, this variance term reweights the bias-variance decomposition of the MSE without adding extra terms. EVarEst encourages the value network to generalize better across successive policies by promoting consistent prediction errors across a broader range of states and actions. EVarEst can be used in place of the value network objective in any RL algorithm with minimal modification to the algorithm and source code. Empirically we show that the traditional MSE objective is generally under-performing when compared to some version of EVarEst in terms of policy performance, illustrating the benefits of the flexibility offered by EVarEst. Additionally, we offer insights into how this new objective enhances performance, specifically by improving adaptability to policy non-stationarity.

1 Introduction

Value is a key notion in Markov decision problems (MDP), and in particular reinforcement learning (RL). Informally, the value of a state $V(s)$ tells how good, or bad, it is to be in a certain state s to solve an MDP. Likewise, the value (or quality) $Q(s, a)$ of a state-action pair (s, a) tells how good, or bad, it is to perform action a in state s to solve an MDP.

Value Estimation in RL. As the knowledge of the optimal value function is enough to derive the optimal policy (Bertsekas and Tsitsiklis [1996]), it is clear why most RL algorithms aim at estimating value functions in a way or another. With full knowledge of the MDP and assuming the set of states and actions are not too large, this optimal value can be computed through value iteration [Bellman,

18th European Workshop on Reinforcement Learning (EWRL 2025).

1958]. Without this full knowledge, in simple environments where the number of state and actions is finite and reasonably low, it is still possible to compute the optimal policy by relying on experience collected by interacting with the environment (e.g. Q-Learning [Watkins and Dayan, 1992]).

In more complex tasks, where the number of states and actions is too large or infinite, we resort to approximating the value functions using a function approximator (usually a neural network). Although technically RL can be done entirely without a value estimator (e.g. REINFORCE [Williams, 1992]), it is highly impractical and inefficient. Modern RL algorithms incorporate some form of value estimation, may it be (1) directly estimating the value function to derive a policy (e.g. Deep-Q-Networks (DQN) Mnih et al. [2013]), or (2) combining it with policy gradient methods [Sutton et al., 1999] in the actor-critic setting (e.g. Proximal Policy Optimization (PPO) Schulman et al. [2017b], Soft Actor-Critic (SAC) Haarnoja et al. [2018] amongst the most popular).

Neural networks are trained to represent value functions using the Fitted-Q Iteration (FQI, Ernst et al. [2005]) scheme. FQI turns value estimation into a sequence of regression problems where the goal is to minimize the temporal difference error (TD error). However this sequential supervised learning setting is cursed by **non-stationarity**: as the agent learns, the policy evolves, which implies a shift in data (states and actions encountered by the agent) and the values to predict.

Consequences of the policy non-stationarity. This non-stationarity introduces the need for the neural network to adapt to new sample distributions along time. In practice, value networks lack robustness to this perturbation (Lyle et al. [2022], Kumar et al. [2021], Nikishin et al. [2022], Dohare et al. [2024]), they may fit the earlier examples correctly but fail to correctly predict newer ones. Since value prediction is such a core component of Deep RL, improving the robustness to this intrinsic non-stationarity would benefit a wide range of algorithms.

Bias-variance weighting in RL as an answer to the problem of non-stationarity. In this work, we design a value prediction objective better suited to mitigate the aforementioned incapacity to adapt to new policies. The traditional MSE can be expressed as a combination of a bias and a variance [Geman et al., 1992]. In this decomposition, we propose to weigh the two terms with a hyperparameter α. The resulting objective is no longer the MSE but a quantity that boils down to mean squared error (MSE) for a certain value of α ($\alpha = 1/2$). We name this objective EVarEst. The idea is that in some situations, it is more sensitive to emphasize the variance than the bias. In particular, the non-stationarity situation in which RL algorithms learn is such a situation. This results in a better ability of the neural network to adapt to successive updates of the policy. In Section 4.1, we motivate this weighting between the variance and bias terms. More specifically in Section 5, we provide intuitions and experimental results about the superiority of putting more weight on the variance of the residual errors than on the bias. This can also be seen as a penalization of the MSE as explained in Section 4.2. In Section 5, we compare this new objective to two baseline objectives, (1) the MSE and (2) AVEC [Flet-Berliac et al., 2021]. AVEC is an objective designed with similar motivations that only considers the variance of residual errors and removes the bias. AVEC has shown promising performances on MuJoCo [Flet-Berliac et al., 2021] and soft-robotics [Schegg et al., 2023]. MSE and AVEC can be seen as extreme-cases of EVarEst.

Contributions. In this paper, we make the following set of contributions:

• We motivate and introduce **EVarEst**, a new objective function allowing the control of the relative weight of bias and residual-error-variance using a hyperparameter α, that (1) requires minimal modifications in the code and (2) is compatible with any algorithm using a variation of the MSE as its objective function for value estimation: **EVarEst** only requires changing the value network objective function, and does not use any new term.
• We provide experimental results on the MuJoCo and AdroitHand continuous control benchmark, testing a wide variety of α values for EVarEst coupled to SAC and PPO. We experimentally demonstrate that EVarEst with the proper α value coupled to SAC leads to significant improvements over using the standard MSE value objective or AVEC.

Before delving into EVarEst, we present the related work and introduce the necessary background in the next 2 sections.

2 Related works

In reinforcement learning, value estimation is a form of non-stationary online regression [Raj et al., 2020]. Here, the non-stationarity arises from the evolution of the policy π during training, which alters the data distribution: the inputs are $(s, a) \sim \pi$ and the targets are $Q^\pi(s, a)$. As the evolution of the policy is an inherent aspect of RL, and its evaluation is central to solving an RL problem, various works have examined the implications of this non-stationarity and ways to mitigate its effects.

Understanding and mitigating policy non-stationarity consequences. A subset of studies focuses on understanding how policy non-stationarity impacts value estimation. The prevailing consensus is that this transient non-stationarity [Igl et al., 2021] (due to the policy expected convergence) leads to over-emphasize the early policies encountered during training, resulting in an increasing prediction error as training progresses. This phenomenon is described using different terms that highlight different aspects of the issue: (1) primacy bias [Nikishin et al., 2022] which underscores the importance of early policies in the performance of the value estimator, (2) capacity loss [Lyle et al., 2022] which refers to the diminishing ability of the network to accommodate new examples during training with significant effects in sparse reward environments, and (3) implicit sub-parameterization [Kumar et al., 2021], which points out that training a value estimator on non-stationary data sequentially leads to higher error rates compared to training it on the entire dataset at once. Consequently, the same data requires more complex estimators to achieve the same performance when subjected to sequential non-stationarity. Each of these perspectives proposes solutions centered on preserving the model initial capacity to learn from samples: (1) through periodic resetting of parts of the agent, (2) by incentivizing the model to align its predictions with its initial ones, and (3) by introducing a feature rank penalty in the objective function.

Exogenous vs. Endogenous non-stationarity. It is important to distinguish between what we call "endogenous" non-stationarity, and what we call "exogenous" non-stationarity. Endogenous non-stationarity originates from within the RL process itself due to the non-stationarity in policy. Exogenous non-stationarity comes from the environment non-stationarity, which can come from non-stationary dynamics[Pourshamsaei and Nobakhti, 2024], or from continual learning (CL) [Wang et al., 2024] where the goal is to solve a sequence of different tasks using a single estimator. In RL, both endogenous and exogenous non-stationarity can occur simultaneously, particularly in the context of generalization, where the objective is to train an agent to solve multiple tasks. While aspects of non-stationarity in RL and continual learning have been studied [Dohare et al., 2024], the causes and solutions differ. A key challenge in the continual learning setting is catastrophic forgetting (McCloskey and Cohen [1989], Atkinson et al. [2021]), where model performance on previously learned tasks deteriorates when new tasks are introduced. This issue does not arise in the endogenous RL setting as we only have a single tasks. While we do have multiple policies, the value network does not need to retain prediction performance on past policies but must maximize prediction performance on the current policy while remaining adaptable for future policies.

This paper focuses exclusively on endogenous non-stationarity.

Reducing policy non-stationarity. Finally, rather than attempting to mitigate the effects of policy non-stationarity, some studies explore ways to reduce its impact by removing it to some extent. Notable examples include TRPO [Schulman et al., 2017a] and PPO [Schulman et al., 2017b] which limit the difference between successive policies using the Kullback-Leibler-divergence (KL divergence). TRPO does this explicitly, while PPO applies a similar constraint implicitly. Both approaches enable smoother policy evolution and reduce the amount of non-stationarity during training.

Similar methods. EVarEst is a penalization of the MSE using a data-dependent penalization term (instead of a term dependent on the model itself like in regularization). It is also equivalent to a reweighting of the bias and variance of residual errors terms of the MSE decomposition. Similar methods have already been explored in supervised classification, like reweighting the asymmetry in ROC [Bach et al., 2006], or composite losses [Reid and Williamson, 2009].

However in regression this is a less frequent method. After identifying the penalization formulation of EVarEst, only Sample Variance Penalization (SVP) [Maurer and Pontil, 2009] comes close to ours. SVP penalizes using the square-root of the variance of the squared residual errors.

Although this method is similar to our proposed solution, there are some noteworthy differences: (1) SVP is aimed at improving generalization and not mitigating non-stationarity related issues. (2) The use of the squared residual errors instead of the residual errors makes underestimation and overestimation indistinguishable, e.g. if $\widehat{f} = f \pm 1$ the SVP penalization is 0 while this error remain penalized in EVarEst. For these reasons, although SVP is well studied and comes with theoretical guarantees, we still think EVarEst is better suited for the RL setting.

3 Background and notations

3.1 Reinforcement Learning

Reinforcement Learning consists in solving a Markov Decision Problem. In this work we consider infinite-horizon MDPs. The state space $\mathcal{S} \subseteq \mathbb{R}^{\text{state dimension}}$ is continuous and s denotes a state. Likewise, the action space $\mathcal{A} \subseteq \mathbb{R}^{\text{action dimension}}$ is continuous and a denotes an action. $t \in \mathbb{N}$ denotes the times of decision making. The transition function is denoted $P(s_{t+1}|s_t, a_t) : \mathcal{S}^2 \times \mathcal{A} \to [0, 1]$ (with a slight abuse of notation as we denote probabilities as a function, as in Sutton and Barto [2020, (3.4) Chapter 3.1 p 49]) and the reward function $R(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$. $\pi_\theta(a|s) : \mathcal{S} \times \mathbb{R}^{|\theta|} \to \mathcal{P}(\mathcal{A})$ denotes a stochastic policy parameterized by θ ($\pi_\theta(a|s) = \pi(a|s, \theta)$). The agent interacts with the environment by sampling actions $a_t \sim \pi_\theta(.|s_t)$, and observing rewards $r_t = R(s_t, a_t, s_{t+1})$ and new states $s_{t+1} \sim P(.|s_t, a_t)$. The objective is to find the parameters θ^* of a policy π_{θ^*} that maximizes the sum of discounted returns $J(\pi_\theta) \triangleq \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$, where $\gamma \in [0, 1)$ is the discount factor. Throughout this paper $\mathbb{E}_{\tau\sim\pi}[X]$ denotes the expectation of X under samples from trajectory τ generated by policy π in the current MDP. This means that a_t, s_t, and r_t are those visited along a trajectory sampled from the environment using π_θ through repeated sampling of $a_t \sim \pi_\theta(.|s_t)$, $s_{t+1} \sim P(.|a_t, s_t)$ and $r_t = R(s_t, a_t, s_{t+1})$, $\forall t \in \mathbb{N}^+$. We denote the value of a state s for a policy π as $V^\pi(s) \triangleq \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right]$ and the value of an action a in state s for a policy π as $Q^\pi(s, a) \triangleq \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{\infty} \gamma^t r_{t+k+1}|s_t = s, a_t = a\right]$.

3.2 Value estimation in Deep Reinforcement Learning

DQN or an actor-critic algorithm represents the value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (or V^π) under policy π using the value network f_ϕ. This is done by training the network to minimize an objective function J, most commonly the mean squared error (MSE), between its predictions $f_\phi(s, a)$ and $\hat{Q}^\pi(s, a)$, a surrogate for $Q^\pi(s, a)$:

$$\text{MSE}(f_\phi, \hat{Q}^\pi) = E_{(s,a)\sim\pi}[(f_\phi(s, a) - \hat{Q}^\pi(s, a))^2]. \tag{1}$$

The value surrogate $\hat{Q}^\pi$ is usually a variation of n-step return (see Sutton and Barto [2020, chapter 7.1 p. 143]): $\hat{Q}^\pi(s_t, a_t) = \sum_{k=0}^{n-1} \gamma^k r_{k+t+1} + \gamma^n \hat{V}^\pi(s_{t+n}, a_{t+n})$.

3.3 Non-stationarity in Value estimation in Deep RL

Value prediction is a sequence of regression problems, however it differs from classic supervised learning because of the non-stationarity introduced by policy changes during training. As the agent learns, the policy π is updated. These changes in π modify the distributions of (1) targets $\hat{Q}^\pi$ and (2) samples $(s, a) \sim \pi$, which makes the objective J (see eq. (1)) non-stationary during training.

Instead of wanting our model f_ϕ to minimize the objective [1] J, we are instead trying to minimize the expected cumulative error [Raj et al., 2020] over time:

$$\sum_{i=0}^{n} \mathbb{E}_{(s,a)\sim\pi_i}\left[(\hat{y}_i - \hat{Q}^{\pi_i})^2\right], \tag{2}$$

with n the current number of policy updates. In order to train a model to minimize J_n over time, this model needs to be able to adapt to policy updates. In practice, the opposite effect is observed and the

[1] We use the term objective J instead of the risk R to avoid confusion with the reward function R. As we are minimizing the objective, both terms are equivalent here.

value model at a step t often overfit the policy at a step t. This issue is known under various names in the literature: capacity loss [Lyle et al., 2022], primacy bias [Nikishin et al., 2022], or plasticity loss [Dohare et al., 2024]. Overall the underlying trouble is the same: the model tends to fit the earlier samples but loses the capacity to adapt to newer samples over time, which leads to larger and larger cumulative error over time.

The capacity of a model to adapt to a new data distribution is called robustness. A robust model has a small loss change when subject to small variations in distribution. To formalize this characteristic, we refer to the early work on robustness in Hampel [1971]. In the context of learning with data sampled from a non-stationary distribution P_t that can change from one iteration t to the next, if some distance between distributions $d(P_t, P_{t+1})$ is small then an estimator is robust to this distance if its value on P_t is not far from its value on P_{t+1}, this is the type of robustness we target in this article.

4 EVarEst

4.1 Motivation

Value prediction is a core component of RL, and this is why mitigating the issues that arise from its non-stationary setting is crucial. We would like to improve the capacity of the value network to adapt to successive distributions from successive policies in order to get consistent and low prediction errors along the training. Our idea to mitigate these issues stems from observing the most common objective for value prediction, the MSE. The MSE can be decomposed in terms of bias and variance of residual errors [Geman et al., 1992]:

Lemma 4.1. *For any estimator $\hat{f}$ and its target f, we have:*

$$MSE(\hat{f}, f) = Bias(\hat{f}, f)^2 + Var(\hat{f} - \hat{f}) \tag{3}$$

where: Bias$(\hat{f}, f) = \mathbb{E}_x[\hat{f}(x) - f(x)]$ *and Var*$(\hat{f} - f) = \mathbb{E}_x\left[(\hat{f}(x) - f(x) - \mathbb{E}_x[\hat{f}(x) - f(x)])^2\right]$.

The proof is postponed to Appendix B.1.

In this decomposition, the bias and the variance are equally weighted. In order to adapt to future policies, we argue that we should focus on the minimization of the variance of residual errors rather than on the minimization of the bias. The intuition is that by focusing on minimizing the variance of residual errors, we incentivize the model to learn to reduce errors in a consistent way over all samples, rather than focusing on predicting well a subset of samples, and badly predicting the others. We hypothesize that this way, the network will learn features that are more "generalizable"[2] across policies. In addition, other works (Ilyas et al. [2020],Flet-Berliac et al. [2021]) seem to converge to the fact that variance is a bigger problem than bias when it comes to value prediction.

Moreover, although reaching a global optimum of the MSE would mean perfectly predicting the values, in practice we will most likely not converge to a global optimum and hence suffer from a bias. Worse, depending on the number of samples observed before policy update, we may not even converge to a local optimum. The bias due to the non-convergence is most likely larger than the asymptotic bias we would have observed if we had converged to a local optimum. This leads us to consider increasing this asymptotic bias to reach a lower residual error variance instead. In practice, we do not suffer from the asymptotic bias. The intended result is an overall reduction in prediction error.

As a consequence we think that the value estimation objective should reflect this non-stationary setting by promoting (1) low variance of the error of the estimator at the expense of some (transient) bias, and (2) robustness to non-stationarity. If criteria (2) is not met, the model may learn to perfectly fit the samples from the first policies but fail to adapt to a new one. If criteria (1) is not met, the neural network may be able to adapt to new policies but may fail to optimize the objective function enough before the sample distribution changes. As explained above, we hypothesize that both objectives could be achieved through weighting bias and variance of residual errors. We therefore think that there exists better value prediction objectives than the MSE, and we think they take the form of different weightings of bias and variance of residual errors

[2]This is not the usual notion of generalization as we consider the performance across different distributions instead of different samples from the same distributions.

4.2 Method

In order to study the weighting of bias and variance of residual errors as alternative value network objectives, we introduce a weight α into the MSE bias-variance decomposition (eq. (3)):

$$\mathcal{L}(\alpha) \triangleq \alpha Bias(\hat{f}, f)^2 + (1-\alpha)Var(\hat{f} - f) \tag{4}$$

with $\alpha \in [0, 1]$, the bias-variance weighting parameter,

Setting $\alpha = 0.5$ is equivalent to the MSE (multiplied by $\frac{1}{2}$. Setting $\alpha = 0$ only considers the variance of residual errors and removes the bias of the equation. This is equivalent to a previously proposed method: AVEC [Flet-Berliac et al., 2021]. Setting $\alpha = 1$ only considers the bias.

This objective can also be expressed as a penalization of the MSE by the variance of residual errors, leading to a novel objective function, the **Error-Variance penalized Estimation (EVarEst)**. For any estimator $\hat{f}$ and its target f, we have, for $\alpha \in (0, 1]$,

$$\mathcal{L}_{\text{EVarEst}}(\alpha) \triangleq \frac{\mathcal{L}(\alpha)}{\alpha} = \text{MSE}(\hat{f}, f) + \frac{(1-2\alpha)}{\alpha} Var(\hat{f} - f). \tag{5}$$

The formulation expressed in equation (5) emphasizes the difference in nature between a penalization of the MSE that changes the relative weight of bias and variance of the residual errors, and an objective focusing only on the variance of the residual errors as it is done in AVEC [Flet-Berliac et al., 2021]. Then, because AVEC is not a penalization of the MSE, it does not enjoy the theoretical guarantees of the MSE. The same remark holds for $\alpha = 1$, which only considers the bias.

We emphasize that unlike other penalizations, we are not adding any new term such as a regularization or any auxiliary objective. Rather, we only re-weight bias and variance in the MSE decomposition.

Finally, applying the EVarEst objective to value estimation in RL we get

$$\mathcal{L}_{\text{EVarEst}}(\alpha) \triangleq \text{MSE}(f_\phi, \hat{Q}^\pi) + \frac{(1-2\alpha)}{\alpha} Var(f_\phi - \hat{Q}^\pi) \tag{6}$$

with f_ϕ the parameterized value function, and $\hat{Q}^\pi$ the surrogate value function. This objective can then be used in place of the value network objective for any RL algorithm making use of such network.

Remark For $\alpha \leq 1/2$, unlike other penalizations (SVP, ℓ_1 regularization, *etc.*), our objective function is upper and lower bounded by a constant times the MSE: $\text{MSE}(f_\phi, \hat{Q}^\pi) \leq \mathcal{L}_{\text{EVarEst}}(\alpha) \leq \left(1 + \frac{1-2\alpha}{\alpha}\right) \text{MSE}(f_\phi, \hat{Q}^\pi)$. In particular, for α close to $1/2$, the optimization done by EVarEst remains close to minimizing the MSE.

5 Experimental study

To measure EVarEst performance, we embed it into two popular RL algorithms, SAC [Haarnoja et al., 2018] and PPO [Schulman et al., 2017b], creating EVarEst-SAC and EVarEst-PPO. SAC is an off-policy actor-critic algorithm that extends DDPG with entropy maximization, while PPO is an on-policy actor-critic algorithm based on TRPO, improving sample efficiency through policy clipping. Both use a value/critic network for estimating values (soft action-value $\hat{Q}$ for SAC, state-value V for PPO), but differ in how much the critic influences policy updates. SAC directly derives updates from the critic, while PPO uses the advantage function, computed using Generalized Advantage Estimation (GAE), which blends actual returns and value predictions controlled by the hyperparameter λ. A typical λ value for PPO is 0.95, emphasizing returns over predictions. We chose SAC and PPO to represent different aspects of deep RL and demonstrate the general applicability of EVarEst.

For evaluation, we use the MuJoCo [Todorov et al., 2012] and AdroitHand [Rajeswaran et al., 2017] tasks, which involve locomotion and object manipulation in complex environments.These tasks are widely used benchmarks, providing sufficient complexity to differentiate between algorithms and allowing for meaningful comparisons. All algorithms performance were evaluated over 10 seeds. Experiments were made using stable-baselines3 [Raffin et al., 2021] implementations of SAC and

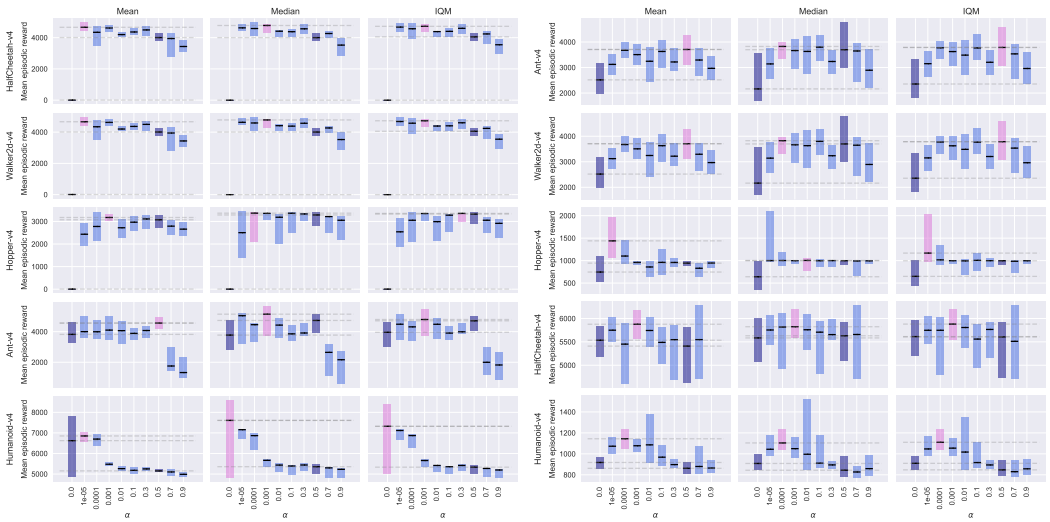

Figure 1: Average performance over the last 10^5 training timesteps for EVarEst-SAC (left) and EVarEst-PPO (right) on Mujoco with different α values. Y-axis: Average episodic returns. Black lines represent the metric specified at the top of the column over 10 seeds. Bars represent 95% bootstrap confidence intervals. Pink is the best performing α, dark blue are the baselines.

PPO and applying modifications for the EVarEst loss and the logging of the metrics reported in the paper. For more details about the experimental setup, see appendix E. In order to compare the performance of the different values of α, we follow the methodology from Agarwal et al. [2021] which promotes the use of mean, median and inter-quartile mean (IQM), as well as confidence intervals.

5.1 EVarEst policy performance

5.1.1 Continuous control on MuJoCo

In figure 1 we observe that, for SAC, α close to 0 leads to equal or better performance than larger values, notably the MSE ($\alpha = 0.5$), except for Ant-v4 where the differences are limited. For PPO, the relation is more dependent on the environment, as smaller values (Hopper, Humanoid) and larger values (Walker2d) can both lead to the best results. Interestingly, the behavior can vary dramatically when using AVEC (equivalent to EVarEst with $\alpha = 0$): we observe that on two tasks, Hopper and Walker2d, it instead leads to the worst performance[3], seemingly not learning at all for SAC.

5.1.2 Continuous control on AdroitHand

In figure 1 we can observe how AdroitHand tasks differ from Mujoco tasks, as a smaller α is no longer correlated with better performance. Most interestingly, on AdroitHandPen the relation seems to be the opposite with a higher bias leading better results. In this task, the robot hand needs to manipulate a pen to match a given orientation and location. However the pen can be dropped leading to important negative rewards until the end of the fixed length episode. We slightly modified the task so that when the robot drops the pen, the task is ended. With this modification of the task, the anomaly is reduced (see appendix C.2). This illustrates that task design has significant impact on results, and that allowing for early termination in this case is beneficial. Indeed, if the task is not stopped, then the agent observes bad returns which are essentially meaningless for the task since the robot dropped the pen, so that any action of the robot is simply useless. Actually, this is a key remark concerning the design of the task: there is no hope to learn anything if the return signal does not contain any useful information.

[3] In the original AVEC paper [Flet-Berliac et al., 2021] the algorithm fails neither on Walker2d for SAC nor on HalfCheetah for PPO, however we failed to reproduce their results. Hopper is not considered in their work.

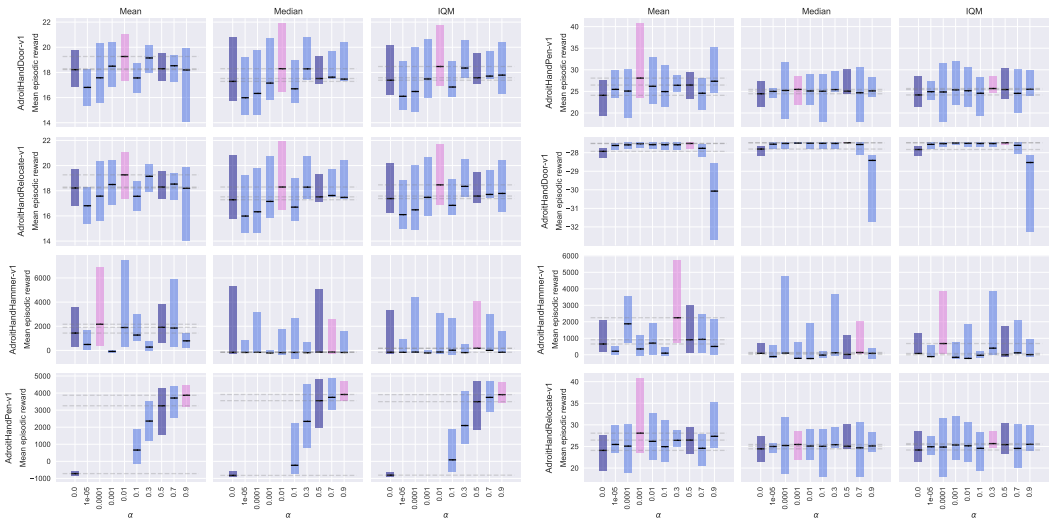

Figure 2: Average performance over the last 10^5 training timesteps for EVarEst-SAC (left) and EVarEst-PPO (right) on AdroitHand with different α values. Y-axis: Average episodic returns. Black lines represent the metric specified at the top of the column over 10 seeds. Bars represent 95% bootstrap confidence intervals. Pink is the best performing α, dark blue are the baselines.

5.2 Overall improvements

Environment	**SAC** ($\alpha = 0.5$)	**AVEC-SAC** ($\alpha = 0$)	**EVarEst-SAC** (α^*)
Ant-v4	**4563.3 ± 1783.8**	3821.9 ± 851.4 (-16.2%)	4102.6 ± 2216.7 (-10.1%), $\alpha = 10^{-3}$
HalfCheetah-v4	10730.1 ± 799.8	11346.6 ± 464.3 (5.7%)	**11405.9 ± 536.1 (6.3%)**, $\alpha = 10^{-5}$
Hopper-v4	3074.5 ± 11.6	4.4 ± 655.2 (-99.9%)	**3175.3 ± 504.3 (3.3%)**, $\alpha = 10^{-3}$
Humanoid-v4	5154.2 ± 2874.4	6624.5 ± 790.4 (28.5%)	**6852.2 ± 1127.8 (32.9%)**, $\alpha = 10^{-5}$
Walker2d-v4	4006.5 ± 68.4	5.8 ± 663.4 (-99.9%)	**4664.2 ± 773.4 (16.4%)**, $\alpha = 10^{-5}$
AdroitHandDoor-v1	**46± 641.8**	217.0 ± 413.9 (-91.9%)	2001.4 ± 1131.0 (-25.0%), $\alpha = 0.1$
AdroitHandHammer-v1	1910.7 ± 3948.8	1435.0 ± 3802.5 (-24.9%)	**2164.9 ± 5013.0 (13.3%)**, $\alpha = 10^{-4}$
AdroitHandPen-v1	3255.3 ± 348.6	-719.5 ± 2517.1 (-122.1%)	**3870.8 ± 1975.2 (18.9%)**, $\alpha = 0.9$
AdroitHandRelocate-v1	18.3 ± 6.1	18.2 ±6.7 (-0.5%)	**19.3 ± 6.7 (5.5%)**, $\alpha = 0.01$
Average Improvement		5.43%	**13.29%**

Table 1: Average performance (± std) over the last 10^5 timesteps. Comparison of SAC, AVEC-SAC, and EVarEst-SAC (α^*: the best performing α)

In tables 1 and 2, we synthesize the performance of EVarEst-SAC and EVarEst-PPO, and we report the average improvements over their respective baselines (MSE and AVEC) for all tasks.

5.3 EVarEst impact on learning – Adaptability to non-stationarity

Having observed that EVarEst generally outperforms both the MSE and AVEC objectives when tuned with the appropriate α value, we now turn to an experimental investigation of how EVarEst affects the learning process. Specifically, we aim at assessing whether EVarEst influences the model adaptability to new samples over the course of training. In order to quantify how much EVarEst impacts the adaptability of the estimator with regards to new samples from successive policies we measure the feature rank [Lyle et al., 2022] of the model throughout training. The feature rank is computed as the number of singular values of the feature matrix that are greater than a given threshold ε. For SAC this threshold is kept at $\varepsilon = 0.01$ as in Lyle et al. [2022], for PPO however we increased it to $\varepsilon = 0.1$ as the initial value did not allow to observe any difference between α values. We focus on SAC here for conciseness; results for PPO can be found in Appendix C.4.

In figure 3 we can note that lower values of α seem to promote higher feature ranks, at the exception of Walker2d and Hopper, where AVEC fails. We can also observe that the feature rank evolves similarly to performance. This is especially visible for the environments where AVEC ($\alpha = 0$) fails

Environment	**PPO** ($\alpha = 0.5$)	**AVEC-PPO** ($\alpha = 0$)	**EVarEst-PPO** (α^*)
Ant-v4	2238.0 $\pm$ 827.2	2381.9 $\pm$ 540.7 (6.4%)	**2486.2 $\pm$ 722.6 (11.1%)**, $\alpha = 0.9$
HalfCheetah-v4	5409.4 $\pm$ 620.0	5534.9 $\pm$ 852.7 (2.3%)	**5877.7 $\pm$ 575.8 (8.7%)**, $\alpha = 10^{-3}$
Hopper-v4	943.2 $\pm$ 562.2	745.5 $\pm$ 155.6 (-21.0%)	**1439.3 $\pm$ 786.3 (52.6%)**, $\alpha = 10^{-5}$
Humanoid-v4	862.0 $\pm$ 173.0	917.8 $\pm$ 154.0 (6.5%)	**1144.1 $\pm$ 285.1 (32.7%)**, $\alpha = 10^{-4}$
Walker2d-v4	**3706.7 $\pm$ 1097.5**	2516.9 $\pm$ 1147.8 (-32.1%)	3673.8 $\pm$ 673.4 (-0.9%), $\alpha = 10^{-4}$
AdroitHandDoor-v1	**-27.5 $\pm$ 0.4**	-27.9 $\pm$ 0.3 (-1.5%)	-27.5 $\pm$ 0.3 (0.0%), $\alpha = 10^{-3}$
AdroitHandPen-v1	1449.7 $\pm$ 924.9	1056.2 $\pm$ 1119.3 (-27.1%)	**1485.8 $\pm$ 1093.9 (2.5%)**, $\alpha = 0.7$
AdroitHandHammer-v1	911.1 $\pm$ 1768.6	653.0 $\pm$ 2376.7 (-28.3%)	**2246.8 $\pm$ 4065.6 (146.6%)** - $\alpha = 0.3$
AdroitHandRelocate-v1	26.5 $\pm$ 8.9	24.1 $\pm$ 7.1 (-9.1%)	**28.1 $\pm$ 13.0 (6.0%)** - $\alpha = 0.001$
Average Improvement		-11.65%	**28.7%**

Table 2: Average performance ($\pm$ std) over the last 10^5 timesteps. Comparison of PPO, AVEC-PPO, and EVarEst-PPO (α^*: the best performing α)

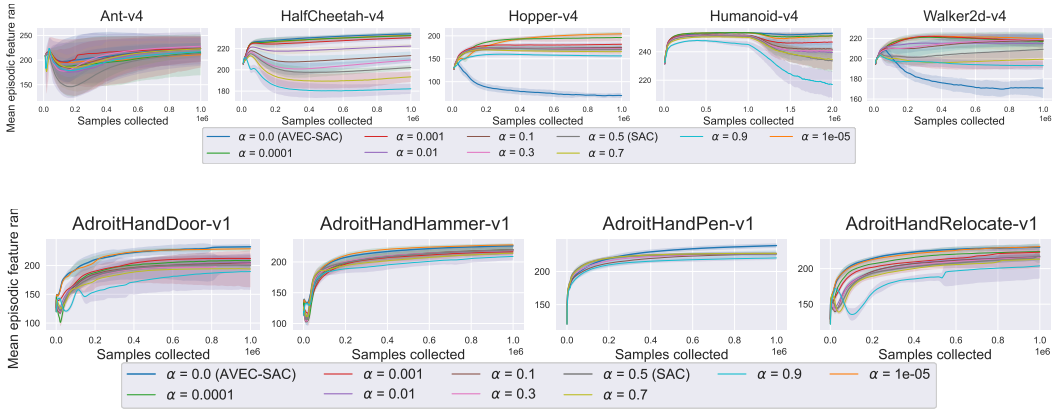

Figure 3: Feature rank of EVarEst-SAC evolution during training (Top: Mujoco, Bottom: AdroitHand)

(Walker2d and Hopper). However, on AdroitHandPen we can observe that the highest feature rank is for AVEC, which also has the worst performance. We think this might be due to the model trying to fit a complex signal and failing to do so (see appendix C.2). As a consequence estimators with lower feature rank, and therefore most likely less complex, are actually less affected by the 'noisy' returns and end up having better performance thanks to their simpler models.

6 Conclusion

In this work, we introduced, evaluated, and analyzed a new value estimation objective, EVarEst, which is designed to better address the inherent non-stationarity in reinforcement learning compared to the traditional MSE objective. By incorporating a hyperparameter, α, to balance the bias-variance trade-off of the MSE, EVarEst effectively penalizes the classical MSE, allowing for adaptive control over the variance of residual errors. Importantly, this requires minimal modifications to existing algorithms.

We demonstrated that EVarEst, with α adapted to each task, significantly improves performance across a range of tasks, achieving an average improvement over the MSE of 13.29% on AdroitHand and Mujoco tasks for SAC and 28.7% for PPO. This suggests that the classical MSE value estimation objective is often suboptimal, and that task-specific adaptation of α leads to better performance in reinforcement learning.

To deepen our understanding of EVarEst, we conducted an experimental study on its impact on adaptability, robustness, and critic estimation error. We found that lower values of α increased adaptability at the cost of robustness. While this differs from our initial expectations, we conclude that adaptability, rather than robustness, may be a more desirable property for tackling non-stationarity in deep RL. Moreover, for lower α values, EVarEst tended to reduce the variance of residual errors, albeit at the cost of increased bias.

We argue that the standard MSE objective, with its arbitrary equal weighting of bias and variance, should be replaced with an objective tailored for reinforcement learning. EVarEst is a promising candidate, as it maintains the theoretical properties of the classical MSE while offering the flexibility for adaptation—requiring only lightweight modifications to existing algorithms. We believe that further exploration of the relationship between environment properties and the optimal value of α will facilitate efficient hyperparameter selection and improve the performance of a wide range of RL algorithms. Additionally, we suggest that the potential of EVarEst as a more general objective for supervised learning in non-stationary settings is further investigated.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, March 2021. ISSN 0925-2312.

Francis R. Bach, David Heckerman, and Eric Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7(63):1713–1741, 2006.

Richard Bellman. Dynamic programming and stochastic control processes. *Information and Control*, 1(3):228–239, September 1958.

Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*, volume 27. Athena Scientific, January 1996. ISBN 978-0-387-74758-3.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, August 2024. ISSN 1476-4687.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Yannis Flet-Berliac, Reda Ouhamma, Odalric-Ambrym Maillard, and Philippe Preux. Learning value functions in deep policy gradients using residual variance. In *Proc. ICLR*, 2021.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. ICML*, pages 1861–1870. PMLR, 2018.

Frank R. Hampel. A General Qualitative Definition of Robustness. *The Annals of Mathematical Statistics*, 42(6):1887–1896, December 1971. Publisher: Institute of Mathematical Statistics.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient Non-Stationarity and Generalisation in Deep Reinforcement Learning. In *Proc. ICLR*, 2021.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A Closer Look at Deep Policy Gradients. In *Proc. ICLR*, 2020.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning. In *Proc. ICLR*, 2021.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and Preventing Capacity Loss in Reinforcement Learning. In *Proc. ICLR*, 2022.

Andreas Maurer and Massimiliano Pontil. Empirical Bernstein Bounds and Sample Variance Penalization, July 2009. arXiv:0907.3740.

Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, January 1989.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. arXiv:1312.5602 [cs].

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The Primacy Bias in Deep Reinforcement Learning. In *Proc. ICML*, pages 16828–16847. PMLR, June 2022.

Hossein Pourshamsaei and Amin Nobakhti. Predictive reinforcement learning in non-stationary environments using weighted mixture policy. *Applied Soft Computing*, 153:111305, 2024.

Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Anant Raj, Pierre Gaillard, and Christophe Saad. Non-stationary Online Regression, November 2020. arXiv:2011.06957 [cs].

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2017.

Mark D. Reid and Robert C. Williamson. Composite Binary Losses, December 2009. arXiv:0912.3301 [stat].

P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, Ph. Preux, and Ch. Duriez. Sofagym: An open platform for machine learning based on soft robot simulations. *Soft Robotics*, 10:410–430, 4 2023.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization, April 2017a. arXiv:1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017b.

Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts London, England, second edition, 2020. ISBN 978-0-262-03924-6.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proc. NIPS*, volume 12. MIT Press, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

A Reproducibility

The code to reproduce the experiments and figures can be found at https://anonymous.4open.science/r/EVarEstRLC.

B Proof of lemmas

B.1 Proof of lemma 4.1

We have:

$$\begin{aligned}
MSE(\hat{y}, y) &= \mathbb{E}\left[(\hat{y} - y)^2\right] \\
&= \mathbb{E}\left[(\hat{y} - y - \mathbb{E}[\hat{y} - y] + \mathbb{E}[\hat{y} - y])^2\right] \\
&\overset{(a)}{=} \mathbb{E}\left[(\hat{y} - y - \mathbb{E}[\hat{y} - y])^2\right] + \mathbb{E}[\hat{y} - y]^2 + 2\mathbb{E}[\hat{y} - y]\mathbb{E}\left[\hat{y} - y - \mathbb{E}[\hat{y} - y]\right] \\
&\overset{(b)}{=} \mathbb{E}\left[(\hat{y} - y - \mathbb{E}[\hat{y} - y])^2\right] + \mathbb{E}[\hat{y} - y]^2 \\
&= var\left[\hat{y} - y\right] + \mathbb{E}[\hat{y} - y]^2
\end{aligned}$$

Where (a) follows from expansion of the square and (b) by linearity of expectation showing that the last term is zero. Then, because the covariance is bilinear,

$$MSE(\hat{y}, y) = var\left[\hat{y}\right] + var\left[y\right] - 2Covar(y, \hat{y}) + Bias(\hat{y}, y)^2.$$

B.2 Proof of equation (5)

We start from eq (4):

$$\begin{aligned}
\mathcal{L}(\alpha) &= \alpha Bias(f_\phi, \hat{Q}^\pi) + (1-\alpha)Var(f_\phi - \hat{Q}^\pi) \\
&= \alpha Bias(f_\phi, \hat{Q}^\pi) + \alpha Var(f_\phi - \hat{Q}^\pi) - \alpha Var(f_\phi - \hat{Q}^\pi) \\
&\quad + (1-\alpha)Var(f_\phi - \hat{Q}^\pi) \\
&= \alpha MSE(f_\phi, \hat{Q}^\pi) + (1-2\alpha)Var(f_\phi - \hat{Q}^\pi) \\
\frac{\mathcal{L}(\alpha)}{\alpha} &= MSE(f_\phi, \hat{Q}^\pi) + \frac{(1-2\alpha)}{\alpha}Var(f_\phi - \hat{Q}^\pi) \\
\mathcal{L}_{\text{EVarEst}}(\alpha) &= MSE(f_\phi, \hat{Q}^\pi) + \frac{(1-2\alpha)}{\alpha}Var(f_\phi - \hat{Q}^\pi)
\end{aligned}$$

C Additional figures

C.1 Mean and variance of value network predictions

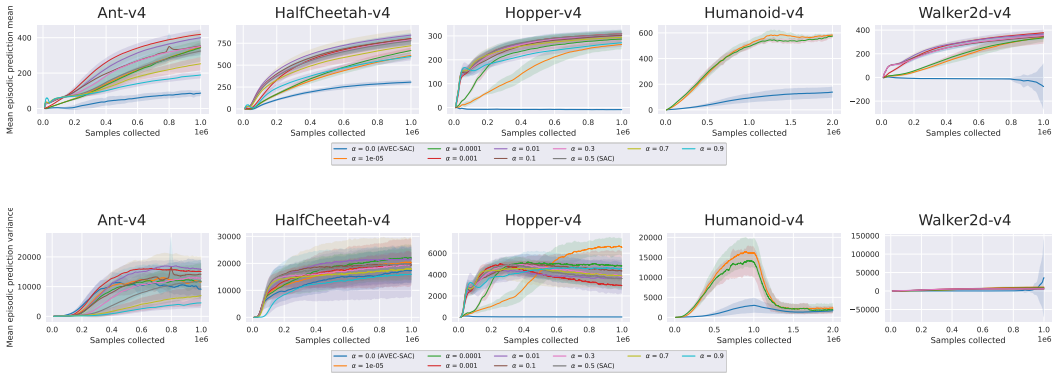

Figure 4: EVarEst-SAC critic predictions evolution during training for different α values (Top: prediction mean, Bottom: prediction variance)

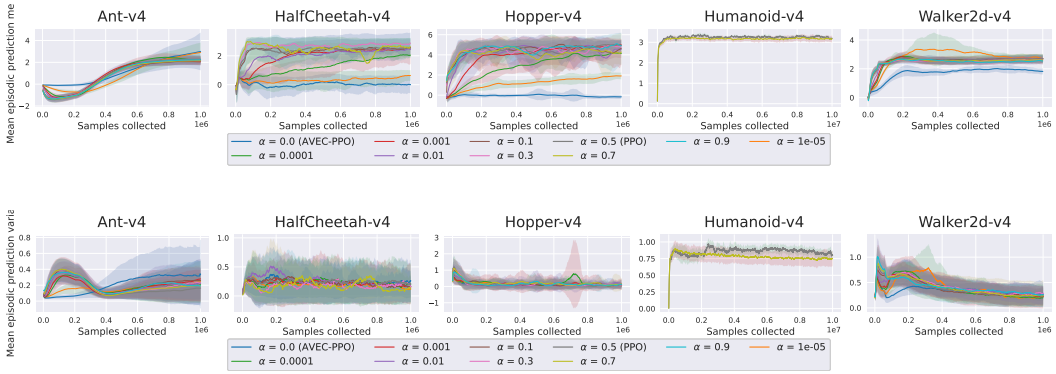

Figure 5: EVarEst-PPO critic predictions evolution during training for different α values (Top: prediction mean, Bottom: prediction variance)

C.2 AdroitHandPen early termination

We take a deeper look at AdroitHandPen where the policy returns seem to be correlated to a high α value instead of a low α value like on MuJoCo. In this task the agent controls a robotic hand that has to manipulate a pen to match a given orientation and position. Looking at the specificities of this task we notice that although it has a fixed episode length like all other AdroitHand tasks, it is possible for the hand to drop the pen. If this happens the agent suffers from a constant negative reward value without being able to pick up the pen, this leads the value estimator to try and learn from a signal that is independent from its actions. Our hypothesis is that this is the property that separates it from other tasks in our benchmark. To verify this we compare the results of EVarEst-SAC on AdroitHandPen and AdroitHandPen with early termination where the episode ends if the pen drops. The agent is train the agent on this variation, while it still evaluated on the fixed length task for comparison.

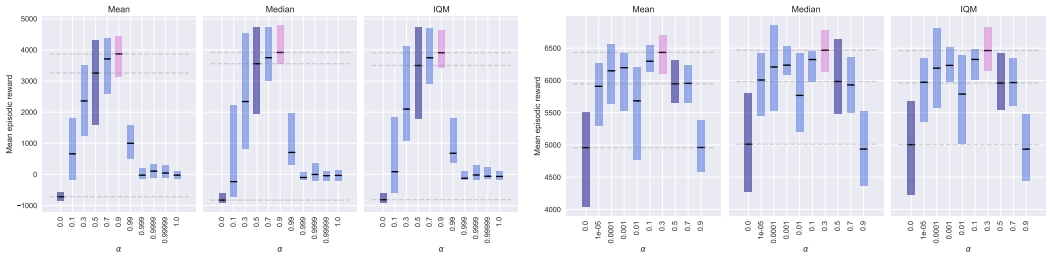

Figure 6: left without early termination, right with early termination.

In figure 6 we can observe that by adding early termination we observe a relation between α and policy performance much closer to what we observed on MuJoCo. We also notice that on the normal task without early termination, although putting more weight on bias seems to be correlated with better performance, it has a limit where $\alpha > 0.9$ leads an important drop in performance.

Our hypothesis about these results is that if the bias is to be more important that the variance of the residual errors, it likely means that the variance of the residual errors is too hard to reduce and spending representation capacity on this aspect of the MSE is a bad investment. The possibility for the pen to drop and have negative rewards there onward leads to an important variance in episodic returns that is hard for the model to estimate. Therefore avoiding to try and estimate it to some degree seem to be beneficial as it is almost noise-like from the estimator perspective.

C.3 SAC Critic performance

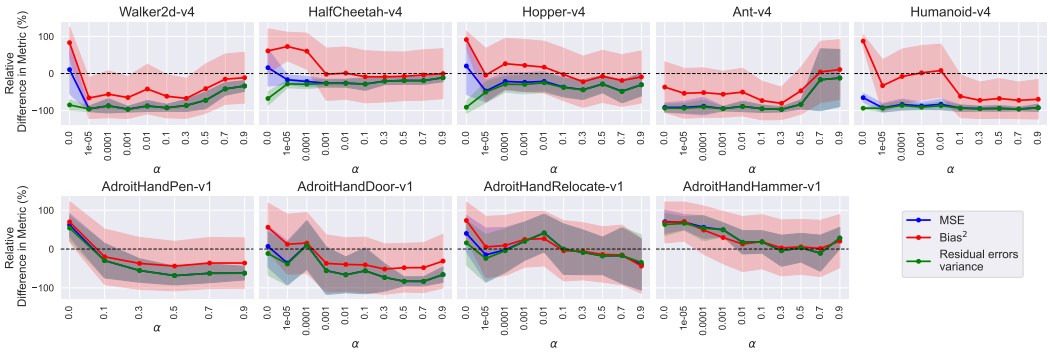

Figure 7: Relative difference between EVarEst-SAC and SAC for value estimation metrics (MSE, bias^2 and variance of residual errors) averaged on the last 100,000 timesteps with different α values (SAC).

C.4 PPO Feature rank

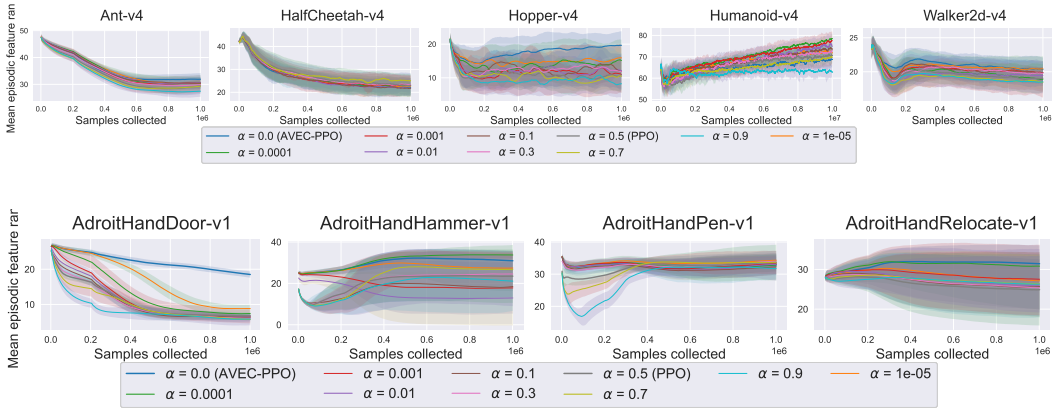

Figure 8: Feature rank of EVarEst-PPO evolution during training (Top: Mujoco, Bottom: AdroitHand).

C.5 Robustness on AdroitHand

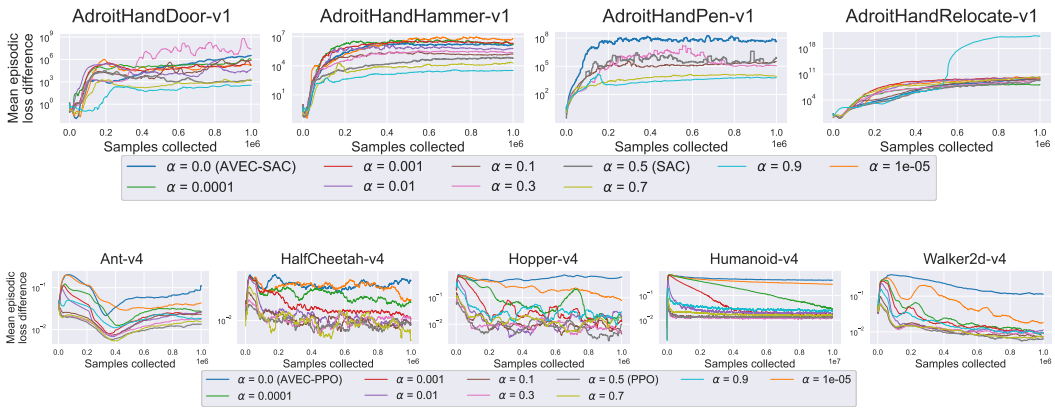

Figure 9: Relative difference between EVarEst and baseline MSE for value loss over training for different α values on AdroitHand tasks (Top: SAC, Bottom: PPO).

C.6 Performance during training

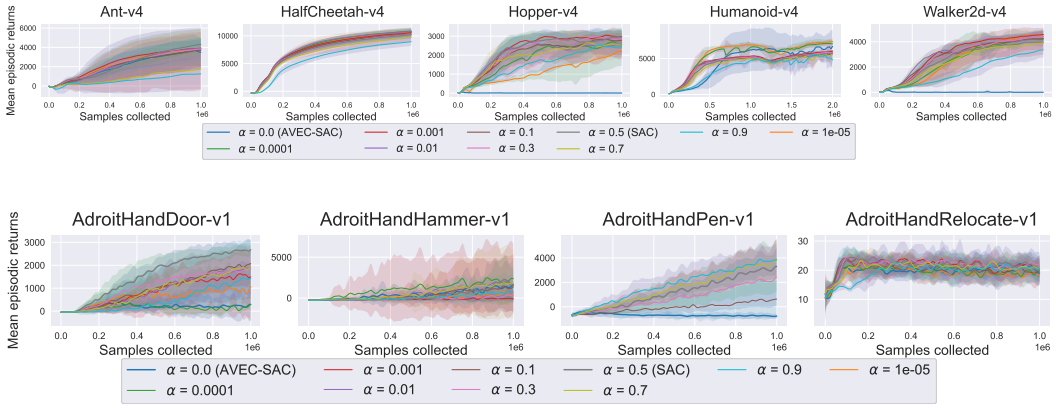

Figure 10: Mean episodic return of EVarEst-SAC during training for different α values(Top: Mujoco tasks, Bottom: AdroitHand tasks).

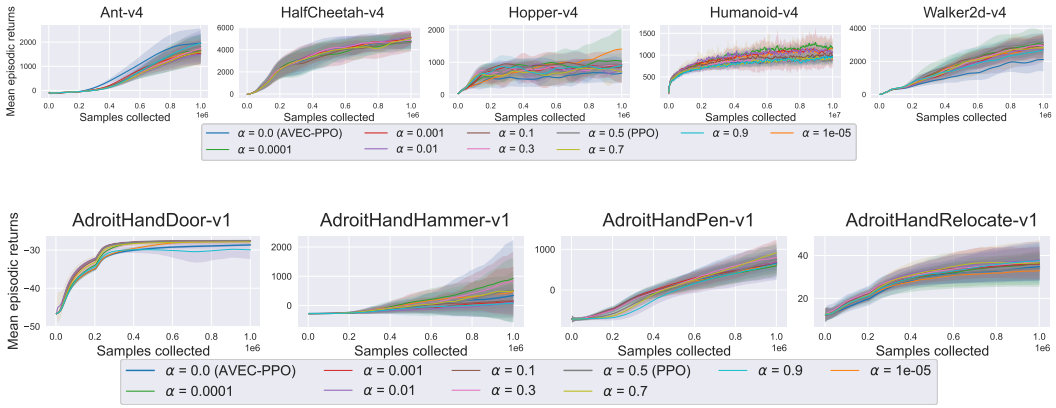

Figure 11: Mean episodic return of EVarEst-PPO during training for different α values(Top: Mujoco tasks, Bottom: AdroitHand tasks).

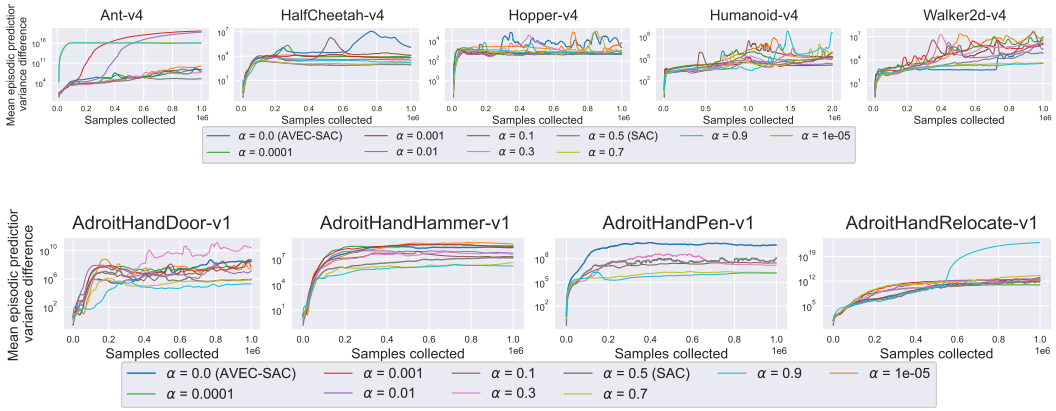

Figure 12: Mean episodic length of EVarEst-SAC during training for different α values(Top: Mujoco tasks, Bottom: AdroitHand tasks).

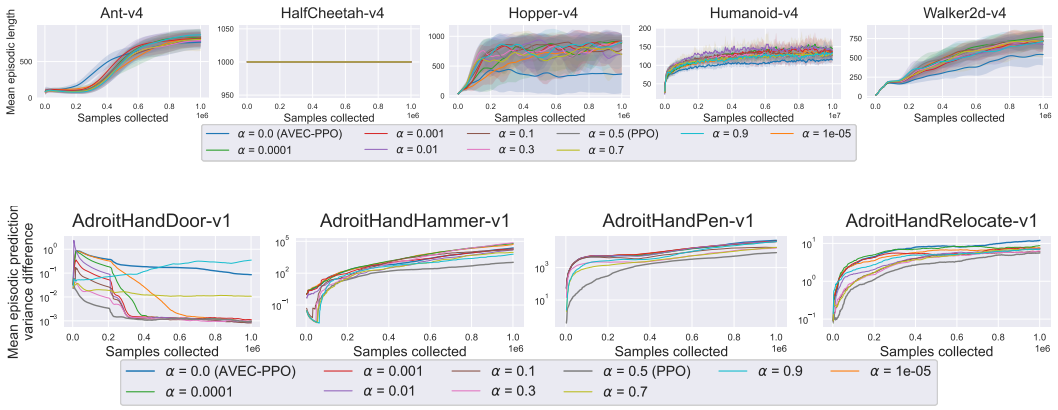

Figure 13: Mean episodic length of EVarEst-PPO during training for different α values(Top: Mujoco tasks, Bottom: AdroitHand tasks).

D Critic comparison protocol

In order to provide meaningful comparison between value networks, we argue that they should be trained and evaluated on the same data. The data should be representative of our setting to provide insightful results. As a consequence we chose to use policies obtained from the corresponding algorithms to generate our data (states visited, actions taken during rollout of the policy, and value surrogates of the policy).

In order to reduce experimental load, we achieve this during the training of the agent by training an additional value network, disconnected from the policy. Both networks function like a standard value network, observing the data generated by the actor, estimating its value, and learning from the policy returns; however only the first critic is used for policy updates.

E Hyperparameters

In Tables 3 and 5 we report the list of hyperparameters common to all continuous control experiments. All environments use the default hyperparameters unless specified otherwise in Table 4, 6, and 7. Hyperparameters are taken from stable-baselines3 [Raffin et al., 2021] default parameters, and rl-zoo3 [Raffin, 2020] when they differ from the default stable-baselines3 value. rl-zoo3 provides optimized hyperparemeters for different agents and environments. Note that the parameters were optimized for the original algorithms (PPO and SAC) and not for they modified versions (EVarEst-PPO and EVarEst-SAC). Hence they should favor the original algorithms.

Table 3: Default hyperparameters for both SAC and EVarEst-SAC.

Parameter	Value
Number of training steps	10^6
Adam stepsize	$3 \cdot 10^{-4}$
Discount (γ)	0.99
Replay buffer size	10^6
Batch size	256
Nb. hidden layers	2
Nb. hidden units per layer	256
Nonlinearity	ReLU
Target smoothing coefficient (τ)	0.005
Target update interval	1
Gradient steps	1
Learning starts	10^4

Table 4: Environment specific hyperparameters for SAC and EVarEst-SAC.

Environment	Parameter	Value
Humanoid-v4	Number of training steps	$2 \cdot 10^6$

Table 5: Default hyperparameters for both PPO and EVarEst-PPO.

Parameter	Value
Number of training steps	10^6
Horizon (T)	2048
Adam stepsize	$3 \cdot 10^{-4}$
Batch size	64
Nb. epochs	10
Nb. hidden layers	2
Nb. hidden units per layer	64
Nonlinearity	tanh
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2
Maximum gradient norm	0.5
State and reward normalization	True
Nb. environments	1
Value function loss coefficient	0.5
Initialization log standard deviation	0.0
Orthogonal initialization	True

Table 6: Environment specific hyperparameters for PPO and EVarEst-PPO.

Environment	Parameter	Value
HalfCheetah-v4	Discount (γ)	0.98
	Horizon (T)	512
	Adam stepsize	$2.0633 \cdot 10^{-5}$
	Entropy coefficent	0.000401762
	Clipping parameter (ϵ)	0.1
	Nb. epochs	20
	GAE parameter (λ)	0.92
	Maximum gradient norm	0.8
	Value function loss coefficient	0.58096
	Initialization log standard deviation	-2.0
	Orthogonal initialization	False
	Nb. hidden units per layer	256
	Nonlinearity	ReLU
Hopper-v4	Discount (γ)	0.999
	Horizon (T)	512
	Adam stepsize	$9.80828 \cdot 10^{-5}$
	Batch size	32
	Entropy coefficent	0.00229519
	Nb. epochs	5
	GAE parameter (λ)	0.99
	Maximum gradient norm	0.7
	Value function loss coefficient	0.835671
	Initialization log standard deviation	-2.0
	Orthogonal initialization	False
	Nb. hidden units per layer	256
	Nonlinearity	ReLU
Humanoid-v4	Number of training steps	10^7
	Discount (γ)	0.95
	Horizon (T)	512
	Adam stepsize	$3.56987 \cdot 10^{-5}$
	Batch size	256
	Entropy coefficent	0.00229519
	Clipping parameter (ϵ)	0.3
	Nb. epochs	5
	GAE parameter (λ)	0.9
	Maximum gradient norm	2.0
	Value function loss coefficient	0.431892
	Initialization log standard deviation	-2.0
	Orthogonal initialization	False
	Nb. hidden units per layer	256
	Nonlinearity	ReLU

F Additional discussions on the impact of EVarEst on learning

F.0.1 Critic performance

Finally we evaluate the value network performance in terms of prediction error (MSE) between the predictions f_ϕ and the targets $\hat{Q}$. We also measure and study the fluctuation of the bias and the variance. In order to measure how α values influence the critic performance, we cannot just compare the values of our metrics across different α values. Indeed, as α modifies the value network, it modifies the policy, which modifies the distribution used to train the critic. As a consequence, a value network that leads to a better performing policy may have a worse error than a value network that leads to poor performance as they are evaluated on different trajectories. In order to compare the

Table 7: Environment specific hyperparameters for PPO and EVarEst-PPO.

Environment	Parameter	Value
Walker2d-v4	Discount (γ)	0.99
	Horizon (T)	512
	Adam stepsize	$5.05041 \cdot 10^{-5}$
	Batch size	32
	Entropy coefficent	0.00229519
	Clipping parameter (ϵ)	0.1
	Nb. epochs	20
	GAE parameter (λ)	0.95
	Maximum gradient norm	1
	Value function loss coefficient	0.871923

EVarEst critic with the baseline MSE critic, we instead evaluate and train them on the same dataset. To do so, we use EVarEst for the usual value network, and we add an additional value network in parallel, which has not impact on the policy, trained using the MSE on the same data as the first critic. We can then compare the metrics for both models while they have been exposed to the same data and are evaluated on the same data. We are comparing relative difference to the baseline critic. For more details, please refer to appendix D. It is important to note that using this protocol, two critics with the same objective may not have the same performance as (1) they are initialized independently, and (2) the EVarEst critic is used to update the actor and the MSE critic is not.

To study how using EVarEst instead of the MSE changed the value network performance in terms of prediction error, we compute the relative difference between a value network trained using EVarEst and another trained using the MSE objective, for each of the 3 previously mentioned metrics: MSE, Bias^2 and Variance of residual errors. The relative difference for a given metric is computed as follows:

$$\Delta_{relative}(metric, \alpha) = \frac{1}{N}\sum_{t=1}^{N} 100. \frac{\text{metric}\left(f_\phi^{\text{EVarEst}(\alpha)}(s_t,a_t), \hat{Q}(s_t,a_t)\right) - \text{metric}\left(f_\phi^{\text{MSE}}(s_t,a_t), \hat{Q}(s_t,a_t)\right)}{|\text{metric}\left(f_\phi^{\text{EVarEst}(\alpha)}(s_t,a_t), \hat{Q}(s_t,a_t)\right)| + |\text{metric}\left(f_\phi^{\text{MSE}}(s_t,a_t), \hat{Q}(s_t,a_t)\right)|}$$

A positive value of the relative difference means that EVarEst leads to a value network with higher metric (MSE, bias^2 or variance of the residual errors) than a value network trained using the MSE, a negative value is the opposite. We concentrate on SAC in this section, while results for PPO can be found in Appendix C.3.

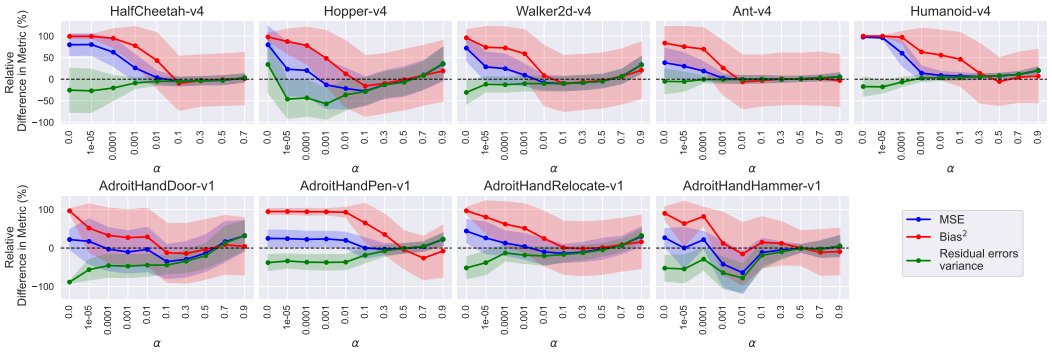

Figure 14: Relative difference between EVarEst-PPO and baseline MSE value networks for value estimation metrics (MSE, bias^2 and variance of residual errors) averaged on the last 10^5 timesteps with different α values

On figure 14 we can observe that on the Mujoco tasks, a lower α value seems slightly correlated with a lower variance of the residual errors for value networks trained with EVarEst than for networks trained with MSE, however it is at the expense of more bias and therefore a larger MSE. On Mujoco we can conclude that lower values of α are indeed reducing the variance of residual errors more than larger values, however on AdroitHand the trend is almost the opposite.

On figures 4 and 5, we can observe the impact of α on the mean and on the variance of the predictions. We can observe that lower values of α produce lower prediction mean and lower variances of

predictions than larger values. In the tasks where AVEC fails we can also state that the value is almost zero for Hopper, and on Walker2d can even start to drift in the negatives with an explosion of variance.

F.0.2 Robustness

Finally, in order to study the robustness of our method with respect to the policy non-stationarity, we want to measure how much a change in policy will impact value estimations. To do so, we consider difference between successive loss values. If, for the same variation in policy, one objective leads to a smaller difference in loss, then we say it is more robust to policy variation than another. We follow the same methodology as in section $F.0.1$ with parallel value networks to compare different value objectives. We compare the difference in successive loss between a value network with EVarEst objective and another with the MSE objective. This allows both estimators to share the same policy non-stationarity, removing the need to compute the distance between successive data distributions generated by the policy. We analyze Mujoco tasks in this section, AdroitHand tasks can be found in appendix C.5.

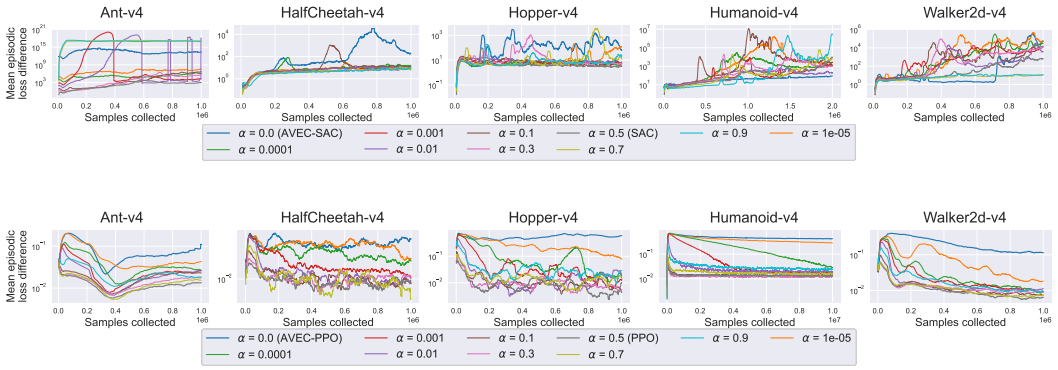

Figure 15: Relative difference between EVarEst and baseline MSE for value loss over training for different α values on Mujoco tasks (Top: SAC, Bottom: PPO)

In figure 15 we can observe that lower values of α tends to lead to higher variations in loss and thus lower robustness with respect to policy non-stationarity. This is the opposite effect of what we intended when designing EVarEst.