58th IEEE Conference on Decision and Control (CDC 2019)

# Deep Reinforcement Learning with Feedback-based Exploration

Jan Scholten, Daan Wout, Carlos Celemin, and Jens Kober

***Abstract*— Deep Reinforcement Learning has enabled the control of increasingly complex and high-dimensional problems. However, the need of vast amounts of data before reasonable performance is attained prevents its widespread application. We employ binary corrective feedback as a general and intuitive manner to incorporate human intuition and domain knowledge in model-free machine learning. The uncertainty in the policy and the corrective feedback is combined directly in the action space as probabilistic conditional exploration. As a result, the greatest part of the otherwise ignorant learning process can be avoided. We demonstrate the proposed method, Predictive Probabilistic Merging of Policies (PPMP), in combination with DDPG. In experiments on continuous control problems of the OpenAI Gym, we achieve drastic improvements in sample efficiency, final performance, and robustness to erroneous feedback, both for human and synthetic feedback. Additionally, we show solutions beyond the demonstrated knowledge.**

## I. Introduction

Contemporary control engineering is adopting the data-driven domain where high-dimensional problems of increasing complexity are solved, even if these are intractable from a classic control perspective. Learning algorithms, in particular Reinforcement Learning (RL) [1], already enable innovations in robotic, automotive and logistic applications [2]–[4] and are on the verge of broad application now that data becomes ubiquitous [5]. There are many applications also beyond the classical control engineering domain, such as HIV [6] and cancer treatment schedules [7]. A possibley extension to diabetes treatment could have great impact [8]. In contrast to model-based control, RL is able to retain optimality even in a varying environment, and modelling of dynamics or control design is not needed.

This study concerns deep RL (DRL), the leading approach for high-dimensional problems that uses neural networks to generalise from observations to actions. DRL can greatly outperform humans [9] in virtue of machine precision and reaction time. However, DRL requires extensive interaction with the problem before achieving final performance. For real-world systems that have restrictions on interaction, the sample efficiency can be decisive for the feasibility of the intended application [10]. Improving sample efficiency is thus essential to the development of DRL and its applications.

In contrast to autonomous learning algorithms, humans are very effective in identifying strategies when faced with new problems. In many cases we achieve decent performance in the first try, despite poorer precision and reaction time that limit final performance. Indeed, from the sample efficiency perspective, human and RL performance are complementary and incorporating human insight into a learning algorithm is a great way to accelerate it.

All authors are with Cognitive Robotics Department, Faculty of 3mE, Delft University of Technology, The Netherlands and reachable via jan@jjscholten.com, daan090@gmail.com, {J.Kober, C.E.CeleminPaez}@tudelft.nl

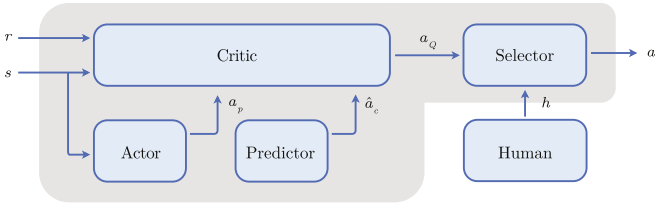


Fig. 1. In the suggested approach, human feedback is combined either with the policy or a prediction of the corrected action. The critic discriminates these with respect to the estimated value of the action. The magnitude of the correction is estimated in the *Selector* and proportional to the estimation variance of the policy. The elements in the grey area constitute an autonomous learner.

Some existing RL methods update the policy with additional human feedback, which is provided occasionally [11], [12]. In contrast, we propose to keep the original RL process and use the human feedback for exploration, as Nair et al. do with demonstrations [13]. Focussing on DRL, there are methods that learn from a priori demonstrations [14] or intermittently collect those (dataset aggregation) [15]. In contrast to corrective feedback, demonstrations are not always available or even possible (there may be limitations in the interface or expertise), besides that they may require manual processing [14] or simulation [13]. Likewise, methods that receive preferences between trajectories can be powerful [16] but they assume the availability of simulation, which is not generally realistic. As a general measure, there is evaluative feedback [2] but we believe that there currently is no method that uses (binary) corrective feedback [12] to accelerate DRL. Yet for the purpose of conditioning the exploration of continuous control problems this would be a natural choice. Moreover, corrective feedback promises to be more effective than evaluative rewards especially in larger action spaces [17].

We present a pioneering combination of DRL with corrective human feedback for exploration, to efficiently solve continuous control problems (Fig. 1). We revisit the question of how the current estimate of the policy is best combined with feedback, and subsequently derive a probabilistic algorithm named Predictive Probabilistic Merging of Policies (PPMP) that improves the state-of-the art in sample efficiency, is robust to erroneous feedback, and feedback efficient. Whilst the proposed assumptions remain realistic, the introduced techniques are moreover generic and should apply to many deep actor-critic (off-policy) methods in the field.

Our approach is motivated by four ideas:

*Action Selection:* After first evidence by Knox & Stone [18], it later were Griffith et al. [19] who made a strong case for how human feedback is given with respect to the action (sequence) and it is most effective to directly adjust the actions when feedback is obtained. Their algorithm outperformed other evaluative feedback methods that instead affect the actions indirectly (modification of the policy).

*Significant Error:* If we consider the early learning phase, where the policy is useless but the human feedback most valuable, we believe feedback is received in case of a significant error as to help the agent develop a notion of the task rather than to communicate refinements. Indeed, a recent study demonstrated that vigorous initial exploration is beneficial for sample efficiency [20]. Moreover, we argue that the instantaneous precision of human feedback is then rather coarse (in contrast, corrections for steady-state errors of an almost converged policy may be smaller). Accordingly, this limitation is quantified by defining a precision $d$ expressing a region of indifference per dimension.

*RL for Fine-stage Optimisation:* Reinforcement learning is superior in final performance due to its precision and reaction time [21]. From our point of view, it should therefore be allowed to autonomously optimise during the later learning phases, such that local optima are identified (e.g. using gradients) independent from past feedback.

*Probabilistic Approach:* Griffith et al. [19] proposed a probabilistic combination of the policy and feedback distributions to determine the action. Because the policy and feedback estimates are balanced by their respective accuracy, such approaches are very effective and robust. In the words of Losey & O'Malley: *'When learning from corrections ... [the agent] should also know what it does not know, and integrate this uncertainty as it makes decisions'* [22]. We subscribe to this point of view and furthermore emphasise past success of using uncertainty in other fields, such as the Kalman filter or localisation algorithms [23]. However, whereas Losey & O'Malley estimate the variance in the corrections [22], we consider the feedback (co)variance fixed ($d$ in *Significant Error*) and argue that the correction size is inversely proportional to the performance of the agent.

It immediately becomes apparent that some of these ideas align, e.g., that corrections are inaccurate (*Significant Error*), but do not need to be accurate, since RL will efficiently identify local optima (*RL for Fine-stage Optimisation*). However, before connecting the dots, let us complete this motivation with the assumption that, given the assumed area of indifference of *Significant Error* expressed by $d$ (Fig. 2), RL is able to identify the local optimum. In other words, *Significant Error* and *RL for Fine-stage Optimisation* concern overlapping regions and the global optimum is attained if the feedback brings us in proximity.

As a corollary of the above statements, we develop a learning method where actions are obtained by significant modification of the policy in direction of the obtained binary feedback. A probabilistic manner that reflects the current abilities of the agent determines the magnitude of correction. This method strongly reduces the need for interaction and furthermore improves final performance. Autonomy and optimality are furthermore preserved, since there will be no feedback when the performance is deemed satisfactory, and our method then resorts to its RL principles.

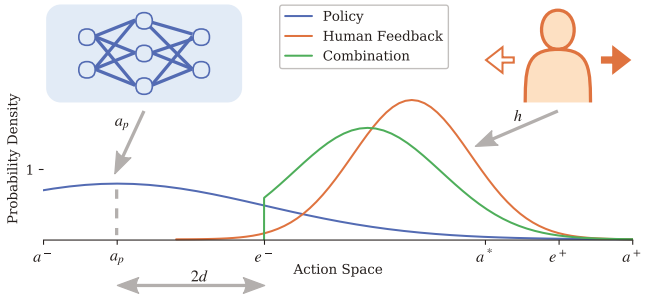


Fig. 2. Using respective covariances, the policy is combined with human feedback in the action space. The resulting distribution on the action that is selected, is truncated such that corrections always have significant effect and the given information cannot dissipate in case of an overconfident policy.

## II. Background

This study is defined in a sequential decision making context, in which the Markov decision process serves as a mathematical framework by defining the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. This set consists of an observable state-space $\mathcal{S}$, action-space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S}\times\mathcal{S}\times\mathcal{A} \mapsto [0, 1]$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, and constant discount rate $\gamma$ [1].

The computational agent interacts with an environment by taking actions $a_k$ (where convenient, we omit the time index $k$) based on its current state $s_k$ and will then end up in a new state $s_{k+1}$ and receive a reward $r_k$ and human feedback $h_k \in \{-1, 0, 1\}$ to indicate an advice of the direction in the action space, wherein the agent could explore. The objective of the agent is to learn the optimal policy $\pi^*(s)$ that maximises the accumulated discounted reward $R = \sum_{i=0} \gamma^i r_i$ And we assume the feedback aligns with this goal. Along with the policy contained in a neural network called the *actor*, the agent will have a network called *critic* which learns to predict the *value* of a state-action pair, i.e., the $Q$-function $Q(s, a) = \mathbb{E}[R \mid \pi, s_k = s, a_k = a]$. This deep actor-critic approach was introduced in [24]. Their work is the basis of the RL-functionalities used here, such as target networks $\pi'$ and $Q'$, parameter initialisation, and replay buffers $B$, although we consider our scheme could also be applied to other RL algorithms.

## III. Predictive Probabilistic Merging of Policies

The aforementioned point of view materialises in our new learning algorithm PPMP, of which we will discuss each element in one of the following subsections.

### A. Combining Policy Information in the Selector

For the sake of this explanation, let us temporally assume non-erroneous corrective feedback $h$ on policy $a_p$ that indicates the relative location of optimal action $a^*$ (all scalar, as in Fig. 2). With reference to the *Action Selection* statement, our approach is to immediately alter the actors suggested $a_p$

in the direction of $h$, such that the eventually selected action $a = a_p + \hat{e}h$ (the orange distribution in Fig. 2) and $\hat{e}$ being an estimate of the absolute error $|a_p - a^*|$.

Deriving from the Kalman filter, the unknown magnitude of the error is estimated using the covariance of the policy (the prediction) and the feedback (an observation) in a module that we call the *Selector*. It is assumed that the magnitude of the error will diminish over time along with the covariance of the policy $\Sigma_{a_p a_p}$. With the covariance of the feedback $\Sigma_{hh}$ as a known constant and $\Sigma_{a_p a_p}$ obtained as described in Sec. III-C, let $\hat{e} = G\,\mathrm{diag}(c_s) + \mathbf{1}c_o^T$, (lines 9-11 in Algorithm 1) where the constant vectors $c$ set the bounds on $\hat{e}$ as described in the last two paragraphs of this section, $\mathbf{1} = [1, 1, \ldots, 1]^T$ and $G = \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$. Note that $G \in (0, 1)$ (all $\Sigma$ are positive definite by definition) is analogue to the Kalman gain as a dimensionless trade-off measure. When the policy shows large covariance, the corrections will have larger effect and facilitate vigorous exploration. And inversely, corrections will be more subtle upon convergence of the policy. Besides that, the exploration is automatically annealed over time by the decrease of $\Sigma_{a_p a_p}$, its effect is state-dependent and tailored for every action channel, respecting correlations.

The relation of $G$ to $\hat{e}$ (line 11) is defined using two vectors of which the length equals the dimensionality of the action space. First, let us discuss the relevance of offset $c_o$. With reference to *Significant Error*, a lower bound on $e$ is $e^- = a_p + hd$. Moreover, with effective exploration in mind, note that $a$ is guaranteed to be closer to $a^*$ than $a_p$ even if $\hat{e} = 2d$. Accordingly, $c_o = 2d$ acts as a lower bound on the corrections. Besides the optimisation perspective, it is always desired to apply a significant correction in case feedback is provided. First, it will avoid frustration of the user, as very subtle corrections may not be noticed and experienced as if the algorithm ignores the feedback. Second, the information is not preserved otherwise.

The scale $c_s$ allows us to set an upper bound $e^+$ for the applied corrections. From the perspective of using human feedback as exploration, let us consider the case where the policy suggests some negative action and receives $h = 1$ since optimality is contained in the positive half of the action space (Fig. 2). Although we cannot make any general statements about the reachability of the state-space, it is clear that feedback can only have the intended effect when $\hat{e}$ is large, else there is no escape from the wrong half of the action space.

### B. Integrating the Selector with Autonomous Learning

The ideas established in the previous paragraph raise requirements for the eventual algorithm. The probabilistic combination of the policy and the feedback results in off-policy data in a continuous action space. As critic-only methods are suitable for a discrete action space whilst actor-only methods are on-policy, an off-policy actor-critic scheme remains as the evident choice.

Fig. 1 illustrates how the system is interconnected and the actions selected. It is assumed that the human provides binary feedback signals $h$ occasionally and bases this on the observed state sequence (and possibly the actions). Delays between human perception and feedback are not taken into account. In order to memorise and generalise the advised corrected samples, those corrected actions are estimated in the *predictor*, a supervised learner further discussed in Sec. III-D. First, the $Q$-filter (critic) decides whether the policy's action $a_p$ or the estimated corrected action $\hat{a}_c$ is preferred as the suggested action $a_Q$ (line 7). Then, in accordance with the description in the previous paragraph, the selector module adjusts $a_Q$ with respect to $h$ and we arrive at the actually executed action $a$ (line 11). In case feedback is not provided the algorithm relies on its own policy, including exploration noise. Autonomy is hereby preserved.

**Algorithm 1** Predictive Probabilistic Merging of Policies

1: **Initialize:**
   Neural network parameters $\theta, \theta', \psi, \psi', \phi$
   Replay buffers $B$ and $B_c$
   Feedback covariance $\Sigma_{hh}$
   Scale $c_s$, and offset $c_o$
2: **for** episode $e = 1$ to $M$ **do**
3: **Initialize:**
   Ornstein-Uhlenbeck process $\nu$
   Randomly set active head $j_e$
4: **for** timestep $k$ = 1 to $T$ **do**
5: $a_p \leftarrow \pi_j(s_k \mid \psi) + \nu_k$
6: $\hat{a}_c \leftarrow P(s \mid \phi) + \mathcal{N}(0, \sigma_a)$
7: $a_Q \leftarrow \arg\max_a Q(s, a)|_{s=s_k, a=a_p \vee a=\hat{a}_c}$
8: $\Sigma_{a_p a_p} \leftarrow \mathrm{cov}(\pi(s_k \mid \psi))$
9: $G \leftarrow \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$
10: $a_k \leftarrow a_Q + (G\,\mathrm{diag}(c_s) + \mathbf{1}c_o^T)h_k$
11: Store $(s_k, a_k)$ in $B_c$ when $h_k \neq 0$
12: Obtain $s_{k+1}$ and $r_k$ by executing $a_k$
13: Store transition $(s_k, a_k, r_k, s_{k+1}, j_e)$ in $B$
14: Sample $N$ tuples $(s_i, a_i, r_i, s_{i+1}, j_i)$ from $B$
15: Compute target $Q$-values
$y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'_{j_i}(s_{i+1} \mid \psi') \mid \theta'\right)$
16: Update $\theta$, $J^Q = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q(s_i, a_i \mid \theta)\right)^2$
17: Update $\psi$ using multihead policy gradient (1)
18: Randomly sample $N$ transitions $(s_i, a_i)$ from $B_c$
19: Update $\phi$, $J^P = \frac{1}{N}\sum_{i=1}^{N}\left(P(s_i \mid \phi) - a_i\right)\right)^2$
20: Update target network $Q$ : $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
21: Update target network $\pi$ : $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
22: **end for**
23: **end for**

### C. Multihead Actor Network

In contrast to the Deep Deterministic Policy Gradient algorithm (DDPG) [24] we need not only to estimate an action, but furthermore to estimate the covariance in this estimate. In [25] it is established that the uncertainty over a deep neural networks output may be obtained from multiple passes with dropout. However, in the context of RL, [26] reports how a multihead neural network that maintains multiple hypotheses is a more consistent approach to generate posterior samples

than dropout. Whereas in their study the eventual purpose of the multihead network is to use the posterior for exploration rather than to quantify confidence (as desired for our approach), Rupprecht et al. [27] indeed establish how the multiple hypotheses allow accurate estimation of abilities in a deep learning classification problem. As it is furthermore desired to have efficient and scalable estimation, we apply the multihead architecture as discussed in [26] to the actor network.

Effectively, the modification of a regular actor network to its multihead counterpart results in $K$ copies of the output layer that estimate the optimal action $a_j = \pi_j(s \mid \psi)$ (line 5), where $j$ indicates the head and $\psi$ is the parameter set of the network. For the training, we establish an extension to the sampled policy gradient in [24] that features individual values of $\nabla_a Q$ and $\nabla_\psi \pi$ for each head. This sampled multihead policy gradient is given by

$$\nabla_\psi J^\pi \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q(s, a|\theta)|_{a=\pi(s_i)} \; \nabla_\psi \pi(s|\psi)|_{s=s_i} \,, \quad (1)$$

with a slight abuse of notation in the row-wise expansion of $\nabla_a Q$ that contains evaluations for all $K$ policies in $\pi$. To determine the policy during action selection, we choose to randomly select a head $j_e$ per episode, preserving both temporal consistency and compliance with multimodalities (not preserved when averaging). For the training of the critic (line 13 in Algorithm 1), $\pi'_{j_i}(s_{i+1} \mid \psi')$ is evaluated for $j_i$, the same head as in $a_i$.

### D. Predictor Module

The corrected actions are estimated as $\hat{a}_c = P(s \mid \phi)$, where $P$ is the prediction network with parametrisation $\phi$, trained with human-corrected samples $(s, a)$ from buffer $B_c$. Whilst these predictions can greatly improve the performance especially during the early learning stage (where the improvements need to take place), taking the predicted actions has two important disadvantages. First, the corrections and their estimates are coarse improvements that primarily aim to explore. The eventual performance is limited and at some point the actor will perform better and the predictor's influence needs to be scheduled away.

A second problem is that the predictor generalises from few feedback samples and its policy may not be very expressive. As a corollary, the variance in the interactions is reduced and this impedes the learning from this data. As clearly demonstrated in [28], learning from data generated by a stationary policy will cause for instability, presumably because of overfitting. In addition, we suspect that the `Adam` optimiser [29] may become over-confident in its gradient estimate (which is now artificially consistent) and raises at least some of the adaptive learning rates to an unstable value. In [28] the problems are overcome by collecting data with a random or pristine policy. Accordingly, we disable the predictor during the first $N_p$ non-corrected samples. As a second countermeasure, we inject noise to the estimates with variance $\sigma_{\hat{a}_c}$ (line 6), such that the original distribution is somewhat restored and the variance problems partly alleviated. In our experience, noise injection is a necessity, but finding alternatives that address this overfitting problem would be an interesting venue for further research.

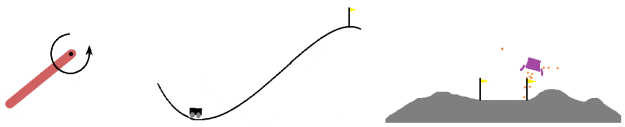

Fig. 3. From left to right: Pendulum-v0, MountaincarContinuous-v0 and LunarLanderContinuous-v2 from the OpenAI gym [31]. The respective goals in these underactuated problems is to swing up and balance, drive up the mountain and gently land between the flags.

Finally, note that in successful actor-critic learning, the critic learns faster than the actor [30]. We can therefore interleave $\hat{a}_c$ and $a_p$ using a $Q$-filter that selects the action with the greatest value (line 7). Besides the retaining of buffer variance, emphasis will now be scheduled towards the actor upon its convergence so the $Q$-filter also solves the first problem (transcending the predictor performance). Because the critic needs to be learned before it can correctly schedule, it is enabled after $N_Q$ samples. Note that, in contrast to the use of direct scheduling heuristics [19], there is a wide range in which $N_Q$ and $N_P$ are successfully set (Fig. 6).

## IV. Implementation and Evaluation

Our code is available at github.com/janscholten/ppmp. All five neural networks (2 for the actor, 2 for the critic, and one for the predictor) are of size (400, 300) and use ReLU activation (except for hyperbolic tangent output layers that delimit actions within their bounds). We train with `Adam` [29], using learning rates of 0.002 for the critic 0.005, 0.0001 for the actor and 0.0002 for the predictor. The actor has $K = 10$ heads. The soft target mixing factor $\tau = 0.003$. The initial variance in the network initialisations is 0.001. The buffers $B$ and $B_c$ have size 1M and 1600 respectively and the minibatch size is 64. The discount rate is $\gamma = 0.99$. The OU-process has volatility 0.3, damping 0.15 and timestep 0.01. The selector and predictor have (as a fraction of the action range per channel) resolution $d = 0.125$, scale $c_s = 0.5$ and variance $\sigma_{\hat{a}_c} = 0.025$. The correction variance is set to $\Sigma_{hh} = 1 \cdot 10^{-8}$ The predictor and $Q$-filter are enabled after $N_P = 1500$ and $N_Q = 4000$ samples respectively.

For benchmarking purposes we regard the problem set in Fig. 3. The continuous state space of the pendulum environment consists of x- and y-positions of the tip and the angular velocity. The control input is the applied torque. Negative reward is given both for displacement from the upright equilibrium and for applied torque. An episode lasts 200 timesteps. The mountain car's state space consists of position and speed and it is controlled by a (reverse) driving force (again, all continuous). When the car reaches the flag, the episode is over and a reward of 100 is obtained, discounted by the cumulative squared action. The state space of the lunar lander has eight dimensions, both continuous (positions/velocities) and binary (leg contact). It is controlled with two inputs, one for the main engine and a second for the steering rockets. An episode ends upon soft landing to

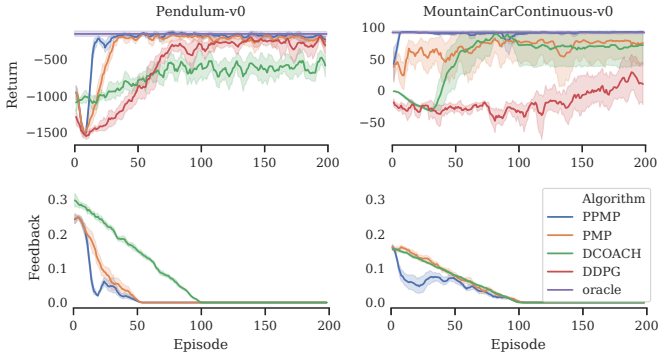


Fig. 4. Our methods PPMP and its ablation PMP (without prediction) outperform all baselines. Depicted is the moving average of ten evaluations (window size 5) along with the feedback rate.

rest (100 points) or crashing (-100), and navigation yields between 100 and 140 points. The applied actions are discounted. Unsolved episodes terminate after 1000 timesteps.

For comparability (that is, to eliminate the effect of inconsistencies in human feedback such as delays [19]), we use synthesised feedback (an oracle). To study applicability, we additionally test with human participants. The oracle compares $a$ with a converged policy. We apply the assumed distance $d$ as a threshold for the feedback. The feedback rate is controlled with a biased coin-flip and annealed over time. To infer robustness we apply erroneous feedback, implemented as in [12].

The results with human participants were obtained from three participants in the age of 20 to 30 with different backgrounds. For each algorithm, they were given a single demonstration and a test run possibility to get familiar with the interface. The subsequent four runs were recorded.

We evaluate PPMP and its ablation PMP (without the predictor) and compare with DDPG [24] and DCOACH [32] (a non-RL deep method that learns from corrective feedback only). Implementations are from P. Emami and R Pérez Dattari on Github.com. We generated ten random seeds (with another random generator) which we applied to ten runs of each algorithm respectively, such that the environments feature equal stochasticity for the different implementations. We evaluate the results on four criteria: sample efficiency, feedback efficiency, final performance, and robustness to erroneous feedback.

## V. Results

Fig. 4 shows that our methods outperform other algorithms in every respect, and whereas the baselines seem to suit one problem in particular, PPMP is consistent. DDPG agents only learn good policies within the range of 200 episodes in the Pendulum problem, for the other environments its poor sample efficiency is even more outspoken. From a comparison with PMP, we infer that the predictor module provides for better sample efficiency, more consistent performance, greater final performance and a reduction of the feedback demand. Fig. 5 shows the effect of erroneous feedback. In virtue of value-based learning, our methods prove very robust

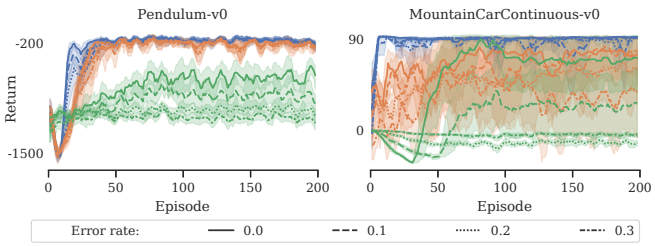


Fig. 5. In case of erroneous feedback, our method proves to be robust and the sample efficiency is hardly affected. The curves with perfect feedback are equal to those in Fig. 4 and the same legend applies.

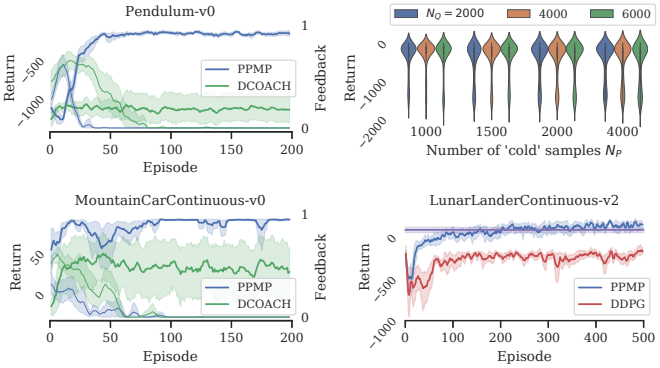


Fig. 6. **Left:** Performance with human participants, averaged over 12 experiments (feedback rate is the thinner line). **Top right:** A sensitivity analysis of the introduced hyperparameters. **Bottom right:** PPMP learns to land, and thereby outperforms the oracle that only knows how to fly.

to erroneous feedback and there is no serious impediment of final performance. In contrast, DCOACH is greatly affected and fails for error rates beyond 10%.

In Fig. 6 additional results are presented. On the left, results from human participants confirm our findings from simulated feedback. DCOACH obtains a considerable amount of feedback, but inconsistent feedback causes failure nonetheless. PPMP is more feedback efficient, learns fast, and consistently attains great final performance. There are some set-backs in performance in the Mountain Car problem, presumably a result of participants that assume the learning is finished after some early success.

Top right in Fig. 6 is a sensitivity analysis for the new hyperparameters $N_Q$ and $N_p$. We compare the distribution of the return during the first 15000 timesteps in the Pendulum domain. As stated in Sec. III-D the new parameters neither require meticulous tuning nor cause brittleness.

Next, in the bottom right, let us consider a typical use case where the feedback has limited performance and is not able to fully solve the problem itself. This scenario is different from the previous studies (with erroneous feedback), where incidental mistakes may have been overcome by low-pass dynamics and generalisation but corrections eventually extended to the end-goal. We use the Lunar Lander environment, where the oracle is now partial as it knows how to fly but can not land. The sequel of the problem is thus left to the agent. It is emphasised that the reward function of this environment stresses stable operation by assigning great negative reward to crashes. Only the last 100

reward units that our method obtains correspond to having learned to land properly. As such, our method allows to solve a problem that is otherwise not feasible.

## VI. Conclusion

This work discusses how binary corrective feedback may be used as a probabilistic exploration signal in DRL in order to improve its sample efficiency. By slight modification of an off-policy algorithm (here DDPG) the uncertainty in the policy was obtained and coupled with the magnitude of the correction induced by the feedback. To generalise corrections and improve memorisation, a predictor network provides estimates of the corrected policy, which can substitute for the actual policy when it increases the value of the state-action pair (especially during early learning). Our method, Predictive Probabilistic Merging of Policies (PPMP), is easily implemented and makes realistic assumptions about the feedback: it does not need to be a full demonstration, expertise may be limited, we do not assume feedback is abundant, neither do we require simulation or post-processing. Nevertheless, PPMP consistently improves on sample efficiency, final performance and robustness in comparison to pure RL (DDPG) and learning from corrections only (DCOACH), both for simulated and human feedback.

A first topic further research should address, is how PPMP carries over to real-world scenarios. Although the scalability and robustness are promising, the applicability is not yet proven by this work. From the possible extensions to this study, an exciting avenue would be to refine the probabilistic part. In particular, we would like to dispose of the Gaussian approximation and connect with full distributional learning [33], [34], possibly combined with uncertainty-handling estimation of human feedback [35]. This could give better estimates of the abilities and allow for more sophisticated action selection, e.g., posterior or Thompson sampling [26].

## References


[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, second ed., 2018.

[2] R. Pinsler, R. Akrour, T. Osa, J. Peters, and G. Neumann, "Sample and feedback efficient hierarchical reinforcement learning from human preferences," in *IEEE Int. Conf. Robotics & Automation (ICRA)*, 2018.

[3] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *Int. Conf. Intelligent Transp. Syst. (ITSC)*, 2018.

[4] J. Camhi, "AI in supply chain and logistics," *Business Insider Intelligence*, 2018.

[5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *AAAI Conf. Artificial Intelligence*, 2018.

[6] D. Ernst, G.-B. Stan, J. Goncalves, and L. Wehenkel, "Clinical data based optimal STI strategies for HIV: A reinforcement learning approach," in *IEEE Conf. Decision and Control (CDC)*, 2007.

[7] Y. Zhao, M. R. Kosorok, and D. Zeng, "Reinforcement learning design for cancer clinical trials," *Statistics in Medicine*, vol. 28, no. 26, pp. 3294–3315, 2009.

[8] American Diabetes Association, "Economic costs of diabetes in the U.S. in 2017," *Diabetes Care*, 2018.

[9] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *AAAI Conf. Artificial Intelligence (AAAI)*, 2018.

[10] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[11] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and MDP reward," in *Int. Conf. Autonomous Agents and Multiagent Syst. (AAMAS)*, 2012.

[12] C. Celemin, J. Ruiz-del Solar, and J. Kober, "A fast hybrid reinforcement learning framework with human corrective feedback," *Autonomous Robots*, vol. 43, no. 5, pp. 1173–1186, 2019.

[13] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *IEEE Int. Conf. Robotics & Automation (ICRA)*, 2018.

[14] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothrl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017. arXiv:1707.08817 [cs.AI].

[15] M. Monfort, M. Johnson, A. Oliva, and K. Hofmann, "Asynchronous data aggregation for training end to end visual control networks," in *Conf. Autonomous Agents and MultiAgent Syst. (AAMAS)*, 2017.

[16] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances Neural Information Processing Syst. (NIPS)*, 2017.

[17] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Int. Symp. Robot and Human Interactive Communication (RO-MAN)*, 2011.

[18] W. B. Knox and P. Stone, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *Int. Conf. Autonomous Agents and Multiagent Syst. (AAMAS)*, 2010.

[19] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Advances Neural Information Processing Syst. (NIPS)*, 2013.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Int. Conf. Machine Learning (ICML)*, 2018.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[22] D. P. Losey and M. K. O'Malley, "Including uncertainty when learning from human corrections," in *Conf. Robot Learning (CoRL)*, 2018.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Int. Conf. Learning Representations (ICLR)*, 2016.

[25] Y. Gal, *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[26] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances Neural Information Processing Syst. (NIPS)*, 2016.

[27] C. Rupprecht, I. Laina, R. DiPietro, and M. Baust, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *IEEE Int. Conf. Computer Vision (ICCV)*, 2017.

[28] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS*, 2015.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learning Representations (ICLR)*, 2015.

[30] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances Neural Information Processing Syst. (NIPS)*, 2000.

[31] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016. arXiv:1606.01540 [cs.LG].

[32] R. Pérez Dattari, C. Celemin, J. Ruiz Del Solar, and J. Kober, "Interactive learning with corrective feedback for policies based on deep neural networks," in *Int. Symp. Experim. Robotics (ISER)*, 2018.

[33] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Int. Conf. Machine Learning (ICML)*, 2017.

[34] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, "Distributional policy gradients," in *Int. Conf. Learning Representations (ICLR)*, 2018.

[35] D. Wout, J. Scholten, C. Celemin, and J. Kober, "Learning Gaussian policies from corrective human feedback," 2019. arXiv:1903.05216 [cs.LG].