Deep Reinforcement Learning from Variance-Reduced Policy-Dependant Human Feedback

Amath SOW*
African Master in Machine Intelligence
Accra, Ghana
asow@aimsammi.org

Pr. Matthew E. Taylor†
director, Intelligent Robot Learning Lab
Alberta University, Canada
matthew.e.taylor@ualberta.ca

ABSTRACT

Human-Centered Reinforcement Learning (HCRL) aims to integrate human guidance with Reinforcement Learning (RL) algorithm in order to improve performance. An example of human guidance is real time binary feedback ('Good' or 'Bad') based on agent's states and actions. One of the famous HCRL algorithms is COACH [10] which is an actor-critic algorithm where the *advantage* function is a good model of human feedback that helps an agent to cope with human behaviors. In this paper, we consider a slight modification of COACH such that the human feedback can be interpreted as a *reward* signal typically present in RL. Therefore , we propose a new HCLR algorithm, Variance-Reduce COACH(VR-COACH) which interprets the human feedback as reward and apply variance reduction technique on policy gradient commonly used in RL. We experiments VR-COACH in the classic MountainCar environment and demonstrate that it learn faster that COACH and TAMER [8]. Moreover, In order to learn complex task in a reasonable time, we upgrade our original VR-COACH to his DEEP version (DEEP VR-COACH) where agent's policy is represented as a deep neural network and apply Convolution Auto Encoder strategy, a Feedback Replay Buffer and entropy regularization. We, then demonstrate the effectiveness of DEEP VR-COACH in the rich Malmo Mine-craft environment while comparing with Deep COACH [3] and DEEP TAMER [18].

KEYWORDS

Interactive learning; Reinforcement learning; Human-centered reinforcement learning, robotics

1 INTRODUCTION

Nowadays, *human-centered reinforcement-learning* (HCRL) draw attention of many Reinforcement Learning researchers. HCRL refers to a problem that arises when an artificial agent interacts with an environment that can be described as a modified Markov decision process (MDP), and the agent's goal is to adapt its behavior to match the desires of a human trainer. More specifically, during the agent-environment interaction, the human trainer can communicates with the agent through *human feedback* (e.g., numerical values) in an attempt to teach the agent how to perform the desired behavior. The fundamental objective of HCRL is for the agent to utilize the human's feedback to learn its goal. The benefits of successful HCRL are numerous, including enabling the real-time learning of unique, trainer-specific agent behaviors, learning from non-expert trainers, and, even for tasks for which we already have predefined reward functions, providing a mechanism by which we can increase the speed of agent learning. [8, 10, 18] compared to traditional reinforcement learning (RL) [17].

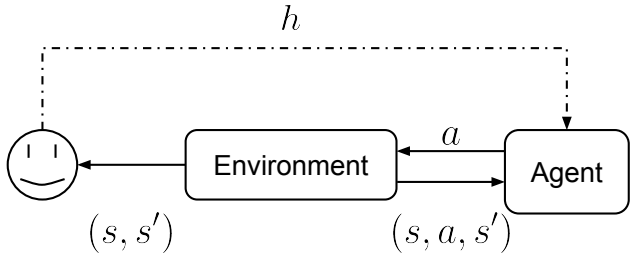


Figure 1: Generic HCRL framework. The tuple (s, a, s') denotes a transition of the environment from state s to s' when the agent takes an action a. A human observes these transitions (s, s') and gives feedback h. The solid lines denote interactions at every timestep while the dotted line activates only when the human gives feedback.

As the problem name suggests, most work for algorithmic approaches capable of performing HCRL has focused on applying and adapting techniques from classical RL. In doing so, algorithm designers must decide what, exactly, the human feedback signal means within the RL context. Thus far, this has been handled by equating the feedback with specific RL quantities. To illustrate this point, consider two popular examples of HCRL techniques: training an agent manually via evaluative reinforcement (TAMER) [8] and convergent actor-critic by humans (COACH) [10]. In TAMER, agents employ a supervised value-function learning technique. In this context, the human feedback is interpreted to be the *value* of recent state-action pairs under the policy that the human desires (agent behavior is based upon the learned value function). Agents that use COACH, on the other hand, take a policy-gradient approach to behavior learning. In that context, the human feedback is interpreted to be the *advantage* (i.e., the difference between the expected return for taking a particular action versus the expected return when taking the optimal action) under the agent's *current* policy.

*African Master in Machine Intelligence
†Intelligent Robot Learning Lab

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021),May 3–7, 2021,London, UK,U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), 2021

It is important to note that, even though the discussion surrounding HCRL techniques uses terms such as *reward*, *value*, *advantage*, and so on, these terms are used only to convey the interpretation of the human feedback in the context of classical RL techniques. In particular, because human feedback is typically non-stationary and inconsistent, the usual ways in which the above terms are understood do not apply in the HCRL setting. For example, since a stationary underlying reward function does not exist in HCRL, there are no notions of optimality with respect to reward, value, policy, and so on.

Both HCRL techniques discussed above (TAMER and COACH) use different learning algorithms and different interpretations of human feedback, yet both have been used to successfully solve HCRL problems in certain domains. Therefore, it would seem that both the question of what learning algorithm to use and the question of how to interpret human feedback are still far from settled.

In this paper, we propose a new algorithm called *variance-reduced convergent actor-critic by humans* (VR-COACH), which includes a critic in the learning framework and exhibits increased learning speed and stability compared to other HCRL approaches. Our work is divided into three distinct parts. First, we analyze the real-time variant of COACH and derive an alternative interpretation of that algorithm in which the agent can be understood to be executing the classical REINFORCE algorithm with the human feedback being interpreted as the *reward* rather than advantage. Second, motivated by this interpretation, we add to COACH a commonly-used variance-reduction technique from the RL literature and study whether or not it can improve agent performance in the HCRL setting. Third, we upgrade our original VR-COACH to its DEEP version in order to learn complex tasks in a reasonable time.

The primary result of the paper is theoretical in nature and provides a basis for using variance-reduction techniques in HCRL. As a proof-of-concept study, we experimentally evaluate our technique in the context of the classical Mountain Car environment while comparing to COACH and (TAMER. We conduct also an evaluation of our deep version, DEEP VR-COACH against DEEP-COACH and DEEP TAMER on Malmo Minecraft Environment and Bowling Atari game using respectively simulated human trainer and real human trainer.

2 RELATED WORK

Our approach is directly inspired by COACH algorithm. which is an actor-critic based model where the advantage function is considered as a good model of human feedback. In fact, with a series of demonstration , the authors proof that human feedback has some properties that are inconsistent with traditional reward signal. For example, a reward function will continuously give positive feedback if the agent demonstrate good behaviors whereas human trainer will reduce feedback when the agent start following good behaviors. Therefore human trainer is less likely to give redundant feedback. So the human feedback ca be considered as an evaluation of the agent's action choice in the context of his current behavior. That's why in COACH, interaction between learning agent and human trainer is considered as an actor-critic algorithm where the human trainer represents the critic that evaluate actor's policy. The authors introduce real-time COACH to address issues about sparseness of human feedback and where the empirical results use hand coded images features detectors using linear function approximation to learn policy.

The TAMER framework [8] is another HCLR algorithm which allows an agent to learn from a human trainer through a series of critiques. It use a regression function to represent the reward function consistent with the feedback signals provided by a human trainer. TAMER framework has proven efficient on several limited tasks. Recently, [18] propose DEEP TAMER, which use deep neural network on top of TAMER framework. It integrate many Reinforcement Learning techniques which permit the algorithm to achieve satisfactory performance on a chosen ATARI game task of Bowling. In fact, beyond the usage of deep neural network function approximation, DEEP TAMER differs from the original TAMER in several ways. First, in the loss function, DEEP TAMER minimizes a weighted difference between the human reward and the predicted value for each state-action pair. Second, the author integrate a deep auto-encoder that reduce the number of parameters and learning time. The lastly difference is the frequency of learning. TAMER learns one from each state-action pair, while DEEP TAMER can learn from each multiple thanks to the feedback replay buffer.

There is another HCLR algorithm that was inspired to TAMER framework proposed by [1]. Specifically, they collect data from human observing and noting preferences between agent trajectories and use these collected data to asynchronously learn a reward model. Indeed , even with thousands of samples collected from human feedback, the algorithm requires millions of steps to converge to a satisfactory policy for a certain task.Our approach differs in that we apply reinforcement learning (respectively deep reinforcement learning) based on direct human feedback considered as reward instead of attempting to learn human preferences.

Others related works [13] and [14] consider human feedback as a label of actions optimality. These policy shaping approaches integrate information about the human trainer based on observed feedback to improve learning.

Lastly, we make a distinction between the HCLR setting presented in this work and the learning from demonstration paradigm [12] where the agent is provided a dataset of demonstration which capture a desired behavior. Based on the dataset, the agent is meant to solve the policy that best describe the observed data.

The work we present here studies first, whether or not using an RL variance-reduction strategy in the policy update step of COACH results in better HCRL performance. In the context of variance reduction in policy gradient techniques, there have been several works since REINFORCE [19], such as introducing a control variate [4], learning a Q-function (deterministic policy gradients) [9] and expected policy gradients [2]. While any of these directions provide an opportunity to reduce variance in HCRL especially in COACH, in this paper, we will study the effect of utilizing the control variate approach. Second, we add deep neural network on top of VR-COACH that we call DEEP VR-COACH and we do our experiment using Malmo project platform [2] which allows for the creation and deployment of AI experiments within Minecraft and Bolwing atari game using respectively simulated human feedback and real human feedback.

3 PROBLEM STATEMENT

In this paper, we are concerned with answering the 2 following questions: what is the impact of considering the human feedback as reward in HCLR while integrating variance reduced technique on agent's policy gradient? what is the effectiveness of using deep Reinforcement Learning techniques on top of the VR-COACH algorithm in order to learn in high dimensional observations.

Let's consider an MDP(Marcov Decision Process)[11] for representing the underlying sequential decision-making problem. In more detail, an MDP is a tuple (S, A, P, F, γ), where S is a set of states, A is a set of actions, $P : S \times A \rightarrow \mathbb{D}(S)$ (where $\mathbb{D}(S)$ is the space of probability distributions over S) is a state transition distribution, F is the feedback that replace the reward function R ,γ is a discount factor. The agent selects an action at each timestep t following a policy π in order to maximize a total discounted sum of reward. In this context, we consider π as a stochastic policy parameterized by θ_t, denoted π_{θ_t} , which defines a probability distribution over all actions given the current state. In VR-COACH algorithm, even if it exists the environment reward, we use the feedback signal of a human trainer at each timestep, $f_t \in \{-1, 0, 1\}$.

In MDP, there are two fundamental concepts that are widely used in RL setting which are, the state value function V^π and state-action value function Q^π. The value function defines the expected future discounted reward from each state when following some policy and the state-action value function refers to the expected future discounted reward when an agent takes some action in some state and then follows some policy thereafter. We recursively define through the Bellman Equations : $V_\pi(s) = \sum_a \pi(s, a)Q^\pi(s, a)$ and $Q^\pi(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma V^\pi(s')]$

One of the most famous policy-based algorithm is actor-critic algorithm [16] where the actor is a model parameterized by π_{θ_t} for action selection whereas the critic parameterized either by $V^{\pi_{\theta t}}(s, a)$ or $Q^{\pi_{\theta t}}(s, a)$ is another model that estimates the value function at each timestep for the actor and provides critiques that are used to update the policy parameters at the end of each episode. The critic is the Temporal difference(TD) error which is defined by the bellow equation : $\delta_t = f_t + \gamma V(s_t) - V(s_{t-1})$. In VR-COACH we interpret f_t as a reward function.

4 VR-COACH

In this section, we first analyze an existing HCRL framework, COACH, an actor-critic based algorithm from policy dependant feedback. We also present Episodic COACH which is a minor modification of COACH where we consider gradient update at the end of each episode instead of each time step. Based on the similarity of this variant of COACH and the classical REINFORCE algorithm from the RL literature, we end up with a new algorithm Variance-Reduced COACH in which we consider the human feedback as reward and applying at the same time technique that reduce variance on the policy gradient.

4.1 COACH

COACH [10] is an actor-critic algorithm when the Advantage function is considered as a good model of human feedback.

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \ , \tag{1}$$

where π is the agent's current policy, and a_t and s_t, are the action and state observed at time t, respectively.

Let consider the policy gradient theorem [17], the gradient for a single timestep is given by:

$$\Delta\theta_t = \alpha\nabla_{\theta_t} \ln \pi_\theta(a_t|s_t)A^\pi(s_t, a_t) \ , \tag{2}$$

In COACH, The agent's policy is directly modified by the human feedback without using any critic component. Then the gradient update become:

$$\Delta\theta_t = \nabla_{\theta_t} \ln \pi_\theta(a_t|s_t)h_t \ , \tag{3}$$

where h_t is the human feedback observed at time t that replace the advantage function.

Indeed, the implementation of the algorithm above for real time use is problematic due to the sparsity of human feedback. This issue is addressing through the development of Real time COACH by using an eligibility trace to help apply feedback to the relevant transitions. An eligibility trace is a vector that keeps track of the policy gradient and decays exponentially with a parameter λ and is updated for each timestep according to :

$$e_t = \lambda e_{t-1} + \nabla_{\theta_t} \ln \pi_\theta(a_t|s_t) \ , \tag{4}$$

Policy parameters are then updated in the direction of the trace , giving more consideration on recents actions than the older ones. The policy gradient for Real time COACH become :

$$\Delta\theta_t = e_t h_t \ , \tag{5}$$

4.2 Episodic COACH

Let now consider a slight modification of real-time coach and assume that $\Delta\theta_t$ is computed at each timestep but the gradient update is done at the end of the episode (i.e., T timesteps have elapsed).

$$\Delta\theta_{episode} = \sum_{t=0}^{T} e_t h_t \ , \tag{6}$$

where $e_t = \lambda e_{t-1} + \nabla_\theta \ln \pi_\theta(a_t|s_t)$ (i.e., we now differentiate with respect to the policy parameters θ at the start of the episode rather than θ_t).

Examining the eligibility trace itself, we see that it can be written as

$$e_t = \sum_{t'=0}^{t} \lambda^{t-t'} \nabla_\theta \ln \pi_\theta(a_{t'}|s_{t'}) \ , \tag{7}$$

and the episodic update can be rewritten as,

$$\Delta\theta_{episode} = \sum_{t=0}^{T} e_t h_t \tag{8}$$

$$= \sum_{t=0}^{T} h_t \sum_{t'=0}^{t} \lambda^{t-t'} \nabla_\theta \ln \pi_\theta(a_{t'}|s_{t'}) \tag{9}$$

$$= \Big[h_0 \big(\nabla_\theta \ln \pi_\theta(a_0|s_0)\big) + h_1 \big(\lambda \nabla_\theta \ln \pi_\theta(a_0|s_0) + \nabla_\theta \ln \pi_\theta(a_1|s_1)\big) + \cdots \Big] \tag{10}$$

$$= \Big[\nabla_\theta \ln \pi_\theta(a_0|s_0) \big(h_0 + \lambda h_1 + \cdots\big) + \nabla_\theta \ln \pi_\theta(a_1|s_1) \big(h_1 + \lambda h_2 + \cdots\big) + \cdots \Big] \tag{11}$$

$$= \sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t) \big[h_t + \lambda h_{t+1} + \lambda^2 h_{t+2} + \cdots \big] , \tag{12}$$

In (12) equation , we observe that it similar to the gradient update in REINFORCE [19] except that we have the human feedback feedback in place of the reward and the eligibility decay λ in place of the discount factor

We, therefore, arrive at a new interpretation of this episodic variant of real-time COACH: that it is the classical REINFORCE algorithm where the human feedback signal is interpreted as the reward.

4.3 VR-COACH

REINFORCE belongs to a special class of Reinforcement Learning algorithms called Policy Gradient algorithms. Thank to his similarity to Episodic COACH, we think that it may provide benefit in the HCLR setting. Particularly, we are interested in whether reducing the policy-gradient variance based on control variate are able to improve the learning speed and stability of COACH algorithm. We thereby build another HCLR algorithm called VR-COACH.

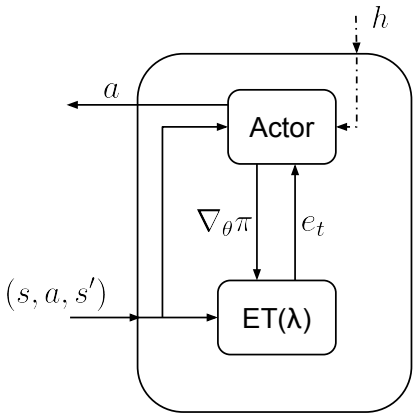
VR-COACH is an actor-critic based algorithm in which the human feedback is interpreted as reward. Compared to COACH, VR-COACH reintegrates the critic part that maintains an approximation of the value function V_w, where w represents the function parameters, based on observations of the human's feedback. As in [4], VR-COACH, uses this value function output as a control variate baseline in the computation of return samples which results in a modified estimate of the policy gradient.

The actor and critic update for VR-COACH are given by the following equations:

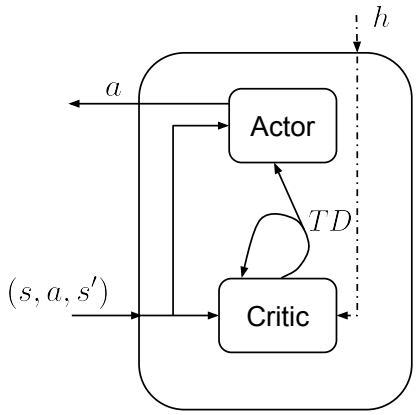
$$\Delta\theta_t = \alpha \nabla_\theta \ln \pi_\theta(a_t|s_t) \big[h_t + \gamma V_w(s_{t+1}) - V_w(s_t) \big] , \tag{13}$$

$$\Delta w_t = \big[h_t + \gamma V_w(s_{t+1}) - V_w(s_t) \big] \nabla_w V_w(s_t) \tag{14}$$

The Fig. 2 illustrate the difference between COACH and VR-COACH. So in COACH, when no feedback is given , the actor doesn't learn anything while in VR-COACH, the actor still learn from the critic even if the human doesn't provide any feedback, i.e., the critic learns to predict the discounted sum of feedbacks given by the human and the actor learns from this critic.



(a) COACH



(b) VR-COACH

Figure 2: Illustration of the agents used in the HCRL framework (Fig. 1) for COACH and VR-COACH. The tuple (s, a, s') denotes a transition from state s due to action a and that lands in the new state s'. The human feedback is denoted by h. The solid lines denote interactions at every timestep while the dotted lines occur intermittently whenever the human gives feedback.

5 DEEP VR-COACH

In order to be able to learn agents in environments with complex state spaces, we upgrade VR-COACH to its deep version by applying a series of modifications on the top of the algorithm. Therefore, the actor's and critic's policies are represented by deep neural network.

To reduce the number of parameter and training time, we use a Convolutional Auto Encoder where the output is used as input of the two policies networks(actor and critic networks). We add also a feedback replay buffer that store experience as dataset that can help the agent to learn the policy in a limited amount of feedbacks. We also integrate an entropy regularization strategy that increase

Algorithm 1 VR-COACH(π_{θ_0}, V_{w_0},γ, α, β, T)

Input:
$\pi_{\theta_0} \rightarrow$ The initial policy with parameters θ_0
$V_{w_0} \rightarrow$ The initial value function with parameters w_0
$\gamma \rightarrow$ Discount factor
$\alpha \rightarrow$ Actor learning rate
$\beta \rightarrow$ Critic learning rate
$T \rightarrow$ Maximum timesteps
Output:
$\pi_{\theta_T} \rightarrow$ The learned policy with parameters θ_T, available after T timesteps
Observe initial state s_0
for $t = 0 : T$ **do**
  Sample and execute action $a_t \sim \pi_{\theta_t}(\cdot \mid s_t)$
  Observe next state, s_{t+1} and human feedback, $h_t \triangleright h_t = 0$ if no feedback
  $\theta_{t+1} \leftarrow \theta_t + \alpha\big[h_t + \gamma V(s_{t+1}) - V(s_t)\big]\nabla_{\theta_t} \ln \pi_{\theta_t}(a_t|s_t)$
  $w_{t+1} \leftarrow w_t + \beta\big[h_t + \gamma V(s_{t+1}) - V(s_t)\big]\nabla_{w_t} V(s_t)$

the uncertainty(more exploration) and avoid the agent to stuck in local minima.

5.1 Convolutional Auto Encoder

Given the desires for learning quickly over high dimensional observations, we use a Convolutional Auto Encoder(CAE) [5] for our policies networks. By definition a CAE is a pair of functions $(f\theta_m, g\theta_n)$ where $f\theta_m$ is an encoder mapping a raw observation x to low dimensional observation and $g\theta_n$ is the decoder of $f\theta_m$ in order to reconstruct the original observation x.

The parameters of the encoder θ_m and the decoder θ_n are found by minimising the reconstruction loss given a minibatch of k samples:

$$L(x) = 1/n \sum_{i=0}^{k} (g\theta_n(f\theta_m(x_i) - x_i^2) \tag{15}$$

In practice, the encoder permit to pass from high dimensional observations to low dimensional observations while preserving relevant features of the input that are important for the reconstruction with high fidelity.

5.2 Feedback Replay Buffer

Even if we initialize our policy with encoded observations, due to the fully connected layers that comprise the second half of $f\theta_m$ we still have an important numbers of parameters to learn. In order to learn quickly with a minimum number of feedbacks, we construct a dataset of experience : D = (s,a,f,s',p(a/s)) where s is the actual state, a is the action and the agent transit to the next state s', f is the human feedback, p(a/s) give the probability of taking an action on a given state.

In DEEP VR-COACH, we use experience window which is a transition between 2 non zeros feedbacks i.e each time a human give feedback to the agent, he complete an entire window that is then stored to the Replay Buffer for futures training update. For each uniformly sampled window from the buffer, we apply gradient update and perform the training.

Algorithm 2 DEEP VR-COACH algorithm

Input: Pretrained Convolutional encoder parameters θ and w,
$\pi_{\theta_0} \rightarrow$ The initial policy with parameters θ_0
$V_{w_0} \rightarrow$ The initial value function with parameters w_0
$d \rightarrow$ human delay
$L \rightarrow$ window size
$m \rightarrow$ minibatch size
$\gamma \rightarrow$ Discount factor
$\alpha \rightarrow$ Actor learning rate
$\beta \rightarrow$ Critic learning rate
$T \rightarrow$ Maximum timesteps
$\rho \rightarrow$ entropy regularisation coefficient
$W : \{\} \rightarrow$ initialize window
$B : \emptyset \rightarrow$ initialize replay buffer
Output:
$\pi_{\theta_T} \rightarrow$ The learned policy with parameters θ_T, available after T timesteps
Observe initial state s_0
for $t = 0 : T$ **do**
  Sample and execute action $a_t \sim \pi_{\theta_t}(\cdot \mid s_t)$
  Record $p_t \rightarrow \pi_{\theta_t}(a_t \mid s_t)$
  Observe next state, s_{t+1} and human feedback, $h_t \triangleright h_t = 0$ if no feedback
  Append $(s_{t-d}, a_{t-d}, p_{t-d}, s_{t+1-d}, h_t)$ to the end of W
  if h_t is not Null **then**
    take L most recent entries of W and append to B
    $W \leftarrow \{\}$
  Randomly sample a minibatch N of m windows from B
  for $n \in N$ **do**
    $g \leftarrow 0$
    for $s, a, p, s', h \in n$ **do**
      compute V(s') and V(s)
      $\delta = h + \gamma V(s') - V(s)$
      $g \leftarrow g + \delta \frac{\pi_{\theta_t}(a \mid s)}{d} \nabla_{\theta_t} \ln \pi_{\theta_t}(a|s)$
      $w_{t+1} \leftarrow w_t + \beta\delta\nabla_{w_t} V(s_t)$
    $g \leftarrow \frac{1}{m} g + \rho\nabla_{\theta_t} H(\pi_{\theta_t}(.|s_t))$
    $\theta_{t+1} \leftarrow \theta_t + \alpha g$

5.3 Entropy Regularization

Here, our agent greedly selects the action that have the highest probability under the current policy at each time step instead of randomly sampling from the distribution. To do so, it may happen that our agent still choosing the same action because it produce some positive reward. It could exist another action that can give a higher reward but the agent will never try it because it will just exploit what it has already learn. In this situation, the agent get stuck to local optimum and will never reach the global one. To avoid that, we use entropy regularization to encourage exploration and avoid getting stack to local optima. We employ entropy regularization[6] of the form $\beta\Delta_{\theta_t} H(\pi_{\theta_t}(.|s_t))$ where β is the regularization coefficient that is used to maintain a high entropy policy.

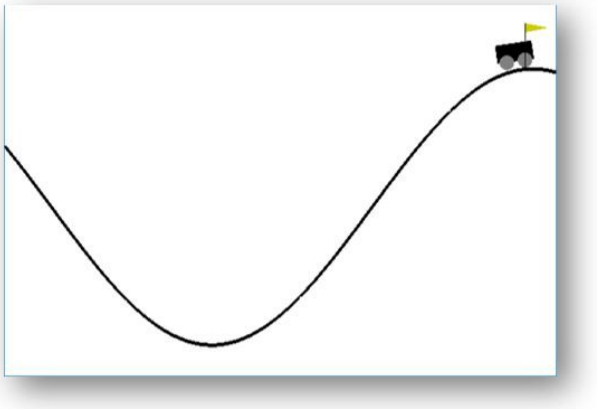


Figure 3: Classic MountainCar

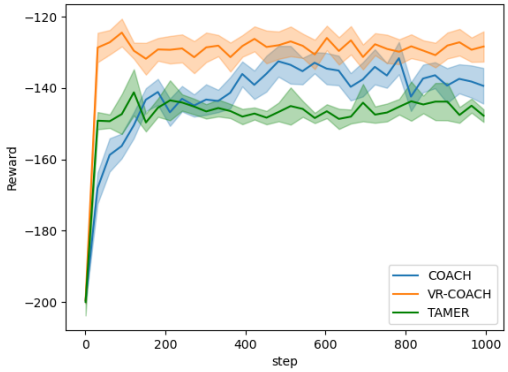


Figure 4: Comparison between VR-COACH COACH and TAMER with simulated human using mean and standard error over 30 runs of each algorithm

6 EXPERIMENTS

In this section, we are going to demonstrate the effectiveness of VR-COACH in HCLR setting. To do so, we first perform a small proof of concept on VR-COACH on classical MountainCar domain and see whether adding variance reduction strategy on policy gradient presents any advantage with regard to others HCLR algorithms(COACH, TAMER [8]).

We also perform a test of DEEP VR-COACH on a rich 3D Mincraft Malmo environment and Bowling atari game using respectively simulated human trainer and real human trainer and compare against DEEP COACH[3] and DEEP TAMER [18] .

6.1 VR-COACH Experiment

6.1.1 ***Classical MountainCar:*** It's a classic RL problem when the objective is to build an algorithm that learns to climb a steep hill to reach the goal marked by yellow flag (Fig. 3). This is not an easy task because the car's engine is not powerful enough to drive up the hill without a head start so the car must drive up the left hill to obtain enough momentum to scale the steeper hill to the right and reach the goal.

6.1.2 ***Proxy Human Feedback Strategy:*** We build a program that can replace the human (called proxy human) and give feedback based on agent's behavior. In the context of MountainCar, it consist of giving positive feedback(+1) if the action taken is along the current velocity and negative feedback(-1) otherwise.

6.1.3 ***Experiment details and results:*** The inputs to both the policy and value networks are normalized over a window of 200 previous data points. Both the eligibility trace decay term (λ) and the discount factor (γ) were set to 0.95.

For the experiment, both COACH and VR-COACH in this domain use the same artificial neural network to represent the policy. We use a critic network in the case of VR-COACH.Our policy networks are fully-connected, and use 16 hidden units and *relu* activation functions.We take a soft-max on this 3-dimensional output to get the probability of picking each action. The weights of the last layer of the policy are initialized with small values in order to get an initial policy with high entropy.. We use a learning rate of $\alpha_{actor} = 0.0025$ and $\alpha_{critic} = 0.01$, a discount factor of $\gamma = 0.99$ and the eligibility trace decay is = 0.95. For TAMER's reward function approximation, we use the same parameters in COACH.

The Fig. 4 presents comparison between all three HCRL algorithms over 30 trials. We can observe that VR-COACH solve the MountainCar problem faster than COACH and TAMER while being able to get a mean reward around -120. In fact, from the beginning VR-COACH's agent is able to reach the goal within around 120 steps whereas COACH's agent is trying to get the good behavior and TAMER's agent solve the problem within 150 steps. After around 15mn of training, all three algorithm manage to reach the goal within a mean reward of -120, -140 and -150 for respectively VR-COACH. COACH and TAMER. We can see that, in the case of VR-COACH how variance reduction techniques in Reinforcement Learning setting can help to adjust the policy gradient and accelerate the learning.

6.2 DEEP VR-COACH Experiment

6.2.1 ***Goal Navigation Task on Malmo/Minecraft:*** The agent here is randomly placed in a 10 × 10 grid facing a single gold block in the center of the room (Fig. 5) and should navigate from its start location to the gold block. The environment reward structure for the task silently provides a reward of +200 to the agent for reaching the gold block while each step taken by the agent has a cost of 1. Each episode runs until the agent finds the goal or until the agent reaches a quota of 200 steps.

A. Proxy Human Feedback Strategy: For the goal navigation task, we build a simulated human (a program) that give feedback based on the behavior of the agent. To do so, we compare the the distance from the goal of the actual agent state against the previous one.

B. Experiment details and results: For the experiment, we first train a Convolutional Auto Encoder(CAE) from a dataset of 20 000 images collected by executing a random policy and then use that CAE to initialize our actor and critic networks. All observations were represented by 84x84 images. For policy optimization,

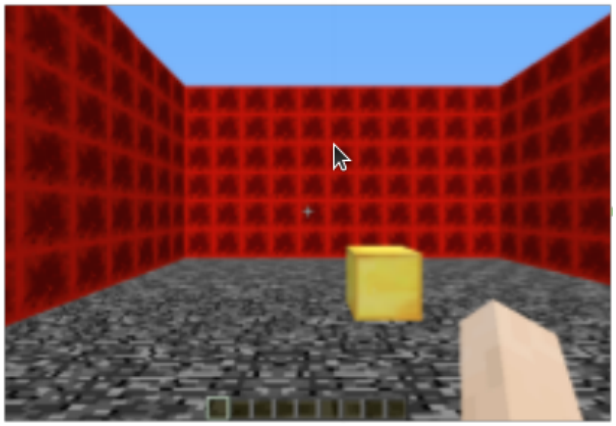


Figure 5: Malmo: Goal Navigation Task

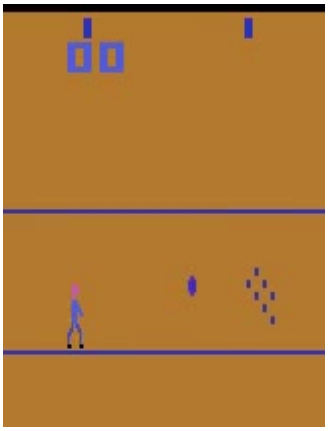


Figure 7: Bowling Atari game

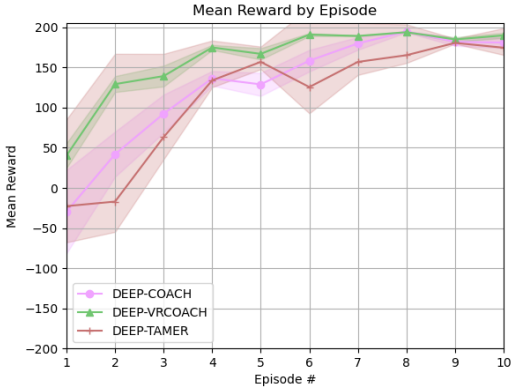


Figure 6: Mean episodic reward for the goal navigation task

we use RMSProp[15] for the actor policy and Adam[7] for the critic policy network. We use a learning rate $\alpha_{actor} = 0.00025$ and $\alpha_{critic} = 0.001$, a mini batch size of 16, a discount factor $\gamma = 0.99$, an entropy regularization coefficient $\beta = 1.5$ and window size L=10. We applied gradient clipping to avoid exploding gradient and then share some training variables between actor and critic networks. For DEEP COACH, we use the same CAE to initialize the policy network and the same hyperparameters as the DEEP VR-COACH's actor network. The DEEP TAMER reward network is identical to DEEP VR-COACH policy network with the only difference is the utilization of a final softmax activation function. Adam optimizer is used to optimize the network with the same learning rate as DEEP COACH, with a buffer update interval of 10 and a credit assignment interval of [0.2, 4.0].

The Fig. 6 presents the mean episodic environment reward obtained by all algorithms over ten trials of goal navigation task. We observe that in the early episodes, both DEEP TAMER and DEEP COACH has a large degree of variance when acquiring the simulated human feedback whereas DEEP VR-COACH seem to start accommodating to the good behavior. After few episodes of learning, all algorithm agents arrive at a policy that consistently reach the target(gold). Furthermore, we observe that DEEP VR-COACH always reach the goal within a minimum cost compare to DEEP TAMER and DEEP COACH. So, it turn out, in DEEP VR-C0ACH since the agent can learn through the critic even if any feedback is given , the agent's policy converge faster to the good behavior.

6.2.2 ***Atari Environment: Bowling game.*** It's an Atari game (7) which use pixel level state where the bowler can take action like (throw, No-op, Move up and Move down) and the goal is to knock down as much pins as possible. The bowler has two throws to knock down all pins. Possibles results could be strike(the bowler knock down all pins on the first throw), sparse(the bowler knock down all pins on the second throw) and open frame(the bowler knock down less than 10 pins).

A. Real human feedback strategy on Bowling: We assume that the human trainer is familiar with the game of Bowling. Therefore the human can train the agent to play the game and give binary feedback(+1 -> "good" , -1 -> "bad") based on the behavior of the agent.

B. Experiment details and results on Bowling game: We use the same hyperparameters as we have in the Goal navigation task on Malmo.

The Fig 8 presents mean episodic reward on the game of Bowling obtained by all algorithms over 10 humans trainers. At the beginning, all algorithms have high variance when receiving human feedback. After few episodes of training, DEEP TAMER is able to have a score greater than 100 whereas DEEP VR-COACH and DEEP COACH still improving slowly. This behavior is due to the sparsity of the reward on Bowling game that avoid the agent to cope with the good policy. After 15mn of training, we observe that DEEP TAMER perform better than DEEP VR-COACH and DEEP COACH. The sparsity and delayed reward on Bowling make somehow difficult for DEEP VR-COACH and DEEP COACH, actor-critic algorithms in general to learn quickly. To improve result, we need to add some RL techniques(reward clipping/scaling, more exploration, etc) to compensate theses two aspects.

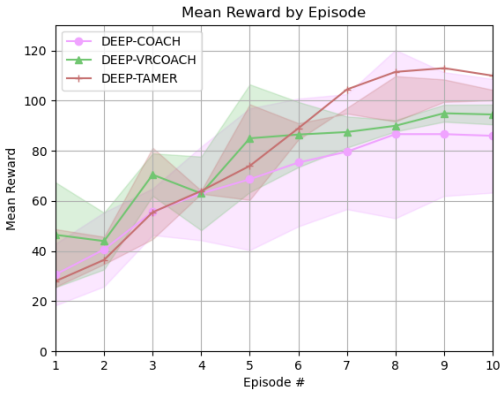


Figure 8: Mean episodic reward on Bowling Atari Game

7 CONCLUSION AND FUTURE WORK

In this paper, we present a new HCRL algorithm, an extension of episodic COACH that interprets human feedback as reward and incorporate variance reduction policy gradient techniques commonly used in RL. We then experimentally validated whether VR-COACH would lead to performance improvements in the HCRL setting by performing a proof-of-concept study on Classic MountainCar environment with simulated and human trainer. We found that , it lead to an improvement of the learning speed compare to COACH and TAMER. In order to learn complex task, we implement some deep reinforcement learning techniques on top of VR-COACH and demonstrate the effectiveness of DEEP VR-CAOCH compare to DEEP TAMER and DEEP COACH using simulated human feedback. Indeed , we finish our experiment on Atari Bowling game using real human feedback. In this case, DEEP TAMER outperforms DEEP VR-COACH and DEEP COACH.

One possible direction of future work could be to study deep reinforcement learning techniques that can help to deal with sparsity and delayed reward, specifically on atari games and compare against DEEP TAMER which has proven to achieve height score after 15mn of training.

REFERENCES

[1] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Neural Information Processing Systems*.
[2] Kamil Ciosek and Shimon Whiteson. 2018. Expected policy gradients. *AAAI Conference on Artificial Intelligence*.
[3] Sophie Saskin Michael Littman Dilip Arumugam, Jun Ki Lee. 2018. Deep RL from Policy Dependant Human Feedback. In *Human-Centered Reinforcement Learning: A Survey*.
[4] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5, Nov (2004), 1471–1530.
[5] Salakhutdinov Hinton, G E. 2006. Reducing the dimensionality of data with neural networks. In *Human-Centered Reinforcement Learning: A Survey*.
[6] Ronald J. illiams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. In *Connection Science*.
[7] Diederik P. Kingma and Jimmy. Adam Ba. 2014. A method for stochastic optimization.. In *CoRR, abs/1412.6980*.
[8] W Bradley Knox and Peter Stone. 2008. TAMER: Training an agent manually via evaluative reinforcement. In *IEEE International Conference on Development and Learning*.
[9] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.
[10] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*.
[11] Martin L. Puterman. 1994. Markov Decision Processes—Discrete Stochastic Dynamic Programming. In *John Wiley Sons, Inc., New York, NY, 1994.*
[12] Chernova Sonia Veloso Manuela M. rgall, Brenna and Brett Browning. 2009. survey of robot learning from demonstration. Robotics and Autonomous Systems. In *Aobotics and Autonomous Systems*.
[13] Subramanian Kaushik Scholz Jonathan Isbell Charles Lee riffith, Shane and Andrea Lockerd Thomaz. 2013. Integrating human feedback with reinforcement learning.. In *Human-Centered Reinforcement Learning: Policy shaping*.
[14] Subramanian Kaushik Scholz Jonathan Isbell Charles Lee riffith, Shane and Andrea Lockerd Thomaz. 2015. Learning behaviors via humandelivered discrete feedback. In *Autonomous Agents and Multi-Agent Systems*.
[15] Wolski Filip Dhariwal Prafulla Radford Alec Schulman, John and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *CoRR, abs/1707.06347, 2017.*
[16] McAllester David A. Singh Satinder P. Sutton, Richard S. and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.
[17] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement Learning: An Introduction*. Vol. 1.
[18] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. 2018. Deep TAMER: Interactive agent shaping in high-dimensional state spaces. *AAAI Conference on Artificial Intelligence*.
[19] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 5–32.