Neural Network Compatible Off-Policy Natural Actor-Critic Algorithm

Raghuram Bharadwaj Diddigi*
Department of Computer Science and Automation
Indian Institute of Science
Bengaluru, India
raghub@iisc.ac.in

Prateek Jain*
Department of Computer Science and Engineering
Indian Institute of Technology
Dharwad, India
170010007@iitdh.ac.in

Prabuchandran K.J.
Department of Computer Science and Engineering
Indian Institute of Technology
Dharwad, India
prabukj@iitdh.ac.in

Shalabh Bhatnagar
Department of Computer Science and Automation
Indian Institute of Science
Bengaluru, India
shalabh@iisc.ac.in

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892303

***Abstract*—Learning optimal behavior from existing data is one of the most important problems in Reinforcement Learning (RL). This is known as "off-policy control" in RL where an agent's objective is to compute an optimal policy based on the data obtained from the given policy (known as the behavior policy). As the optimal policy can be very different from the behavior policy, learning optimal behavior is very hard in the "off-policy" setting compared to the "on-policy" setting where new data from the policy updates is typically utilized in learning. This work proposes an off-policy natural actor-critic algorithm that utilizes state-action distribution correction for handling the off-policy behavior and the natural policy gradient for sample efficiency. The existing natural gradient-based actor-critic algorithms with convergence guarantees require fixed features for approximating both policy and value functions. This often leads to sub-optimal learning in many RL applications. On the other hand, our proposed algorithm utilizes compatible features that enable one to use arbitrary neural networks to approximate the policy and the value function and yet guarantee convergence to a locally optimal policy. We illustrate the benefit of the proposed off-policy natural gradient algorithm by comparing it with the vanilla gradient actor-critic algorithm on benchmark RL tasks.**

***Index Terms*—Reinforcement Learning, Off-policy control, Natural Actor-Critic**

I. Introduction

Reinforcement Learning (RL) [2], [28] deals with learning algorithms for an agent to learn optimal behavior based on the feedback received from an unknown environment. In a standard RL setting, the agent learns from the feedback it receives from the environment for playing out various actions, that helps improve its ability to pick better actions in future plays. This cycle continues until a best sequence of actions corresponding to the various states of the environment is learnt. Note that, in each cycle, the agent gets a feedback from the environment for its current choice of actions. In many real-life scenarios such as medical research [18], this paradigm is not desired. For example, consider training an RL model to learn the best course of treatment for combating a disease. In this application, it is dangerous to try exploratory actions as done in the standard RL paradigm. The most appropriate paradigm here would be to train the RL algorithm from the past patient records with certain treatments successfully implemented. Of course, the efficacy of the trained model would depend on the exhaustiveness of the data, but this approach is safer for learning a basic model that can be used as a guide for a doctor. Such a setting is referred to as the off-policy setting in the RL literature.

Two sub-problems often studied under the off-policy setting in RL are the problems of prediction and control. The off-policy prediction problem [7], [17], [19], [20], [29], [30] refers to estimating the value function of a given policy [28] (the target policy) from data obtained from another policy (the behavior policy). On the other hand, the goal in the off-policy control problem is to find the optimal policy from data obtained using the behavior policy. In this work, we consider the off-policy control problem. The off-policy RL algorithm finds its applications in many important settings like the design of H_∞ tracking controller for non-linear continuous settings [22], multiplayer non-zero sum games [26], optimal control of unknown system with disturbances [27] etc. We first discuss the on-policy setting before we present the off-policy problem.

Standard actor-critic algorithms [15], [21], [31] are a popular class of on-policy RL algorithms for estimating an optimal policy. In this paradigm, the actor improves the policy based on the feedback from the critic using the policy gradient rule. The gradient updates in these algorithms are highly sensitive to the learning rates, and it is often difficult to *a priori* know the right learning rate schedule for an efficient update of

* Equal Contribution.

Raghuram Bharadwaj was supported by a fellowship grant from the Centre for Networked Intelligence (a Cisco CSR initiative) of the Indian Institute of Science, Bangalore. Prabuchandran K.J. was supported by the Science and Engineering Board (SERB), Department of Science and Technology, Government of India for the startup research grant 'SRG/2021/000048'. Shalabh Bhatnagar was supported by the J.C.Bose Fellowship, a project from DST under the ICPS Program and the RBCCPS, IISc.

parameters. This forces one to perform an extensive hyper-parameter search to arrive at a good solution. To alleviate this problem, algorithms based on natural gradients have been studied in the literature [13], [23].

The idea in natural gradient algorithms is to move in the right direction in the distribution space induced by the function parameters rather than in the parameter space. As a result, the natural gradient algorithms are not sensitive to model re-parameterization. Moreover, these algorithms are shown to be more efficient and faster compared to standard gradient algorithms [23], [35]. The actor-critic algorithms that make use of natural gradients in their actor updates as opposed to regular policy gradients are commonly referred to as natural actor-critic algorithms [3], [13], [23], [24]. In [11], an incremental on-policy natural actor-critic algorithm has been proposed and is shown to improve the stability of the iterates. The trust region optimization techniques like the TRPO [25] and the ACKTR [33] use natural policy gradients in their updates. However, they explicitly compute/approximate a distance metric using the Fisher information matrix (making use of curvature information) giving rise to complex algorithms. This estimation/approximation of the Fisher information matrix can be entirely avoided (see equation (11) of Section III) if the critic employs a linear function approximator with well-defined features known as compatible features [15].

The idea of using compatible features in the critic updates has been first proposed in [15], [31] in the context of standard policy gradient algorithms. Actor-critic algorithms using compatible features in their critic updates enjoy nice convergence properties. However, the use of compatible features forces the critic to employ only linear function approximation architecture. This constrains the computational capability of the critic to accurately estimate the feedback needed for the actor which becomes a bottleneck, especially in environments with large/continuous state spaces. Further, utilising compatible features together with nonlinear function approximators based on say neural networks for actor-critic algorithms is a challenging task. Therefore, typically in practical implementations, the actor and critic use different neural networks to approximate the policy and the value function that can result in undesired behavior. The aforementioned limitations have restricted the development of successful neural network compatible natural actor-critic algorithms. For ease of exposition, we refer to neural network compatible natural actor-critic algorithms as deep natural actor-critic algorithms.

In our work, we tackle the aforementioned problems by constructing our algorithm with three components - (1) an actor that deploys a deep neural network for approximating the policy, (2) an advantage critic that uses linear function approximation with compatible features for estimating the advantage value function [28] (that we refer to as neural network compatible features), and (3) a value function critic that deploys deep neural network for approximating the value function. To the best of our knowledge, ours is the first work that allows compatible features to be used in conjunction with the neural network approximations of policy and value function. Our first contribution in this work is a "Deep Natural Actor-Critic" algorithm with compatible features and its comparative study with the standard deep actor-critic algorithm. Note that our algorithm implements natural policy gradients in a simple manner while utilizing the power of neural networks.

We now describe the "off-policy" problem. As mentioned earlier, this refers to the setting where the training data comes from the behavior policy that can be very different from the target policy. As a result, the state-action distribution of the Markov Chain induced by the behavior policy and the one induced by the learning policy will be different. This mismatch in distributions is what makes the off-policy training unstable. In [6], the first off-policy actor-critic algorithm was proposed and shown to converge under the tabular setting. Since then, many off-policy actor-critic algorithms have been proposed mainly under two objective criteria: Excursion and Alternate Life [36]. Under the excursion objective criterion, the performance of a policy is evaluated for the expectation with respect to the stationary distribution of the behavior policy. The algorithms proposed in [6], [10], [37] fall under this category. However, it is shown in [36] that this setting can be misleading regarding the performance of the optimal policy. Under alternate life objective criterion, the performance of a policy is evaluated with respect to the stationary distribution of the target policy itself. This seems to be the most natural setting for off-policy control. The algorithms proposed in [7], [18], [34] fall within this category.

In [36], a unified criterion combining both excursion and after-life objectives has been considered and is shown to provide better performance. Recently, in [14], an off-policy version of natural actor-critic has been proposed under the tabular setting and a finite sample analysis is provided. In [18], a standard actor-critic algorithm has been proposed for solving the off-policy control problem. The actor in the algorithm of [18] corrects for the discrepancy in state and action distribution between target and behavior policy while updating the policy parameters. However, a similar state distribution correction is not performed for the value function update, which might lead to divergence [28]. Our second contribution is an off-policy natural actor-critic algorithm under the alternate life objective criterion which corrects the state-action mismatch in actor and critic updates. To the best of our knowledge, ours is the first off-policy natural actor-critic algorithm that utilizes non-linear or deep function approximation. We now summarize the main contributions of our work.

1) We propose a new deep actor-critic algorithm under the on-policy setting with two critics and an actor. Here, the actor employs a natural policy gradient, the advantage critic uses compatible features, and the value critic uses non-linear function approximation.
2) We then extend this algorithm to the off-policy setting by appropriately correcting the state-action distribution.
3) We present a theoretical analysis of convergence of the proposed scheme.
4) We successfully integrate compatible features in the implementation of our proposed algorithms and demon-

strate their advantages through comparisons with standard actor-critic algorithm on benchmark RL tasks.

Finally, it is important to observe the distinction between the off-policy setting that we consider in this paper, and the actor-critic algorithms like Soft Actor-Critic (SAC) [9] and Actor-Critic with Experience Replay (ACER) [32] that make use of experience replay. Although the data from the replay buffer is technically off-policy, the data available to these algorithms is online and comes from previous iterations [18, Section E, Appendix]. In contrast, the data available to our algorithm comes from an entirely different (behavior) policy. This is a more challenging setting due to the instability issues that arise in this case.

II. Problem Formulation

We consider an infinite-horizon MDP defined by the tuple $(S, A, P, r, \gamma, d_0)$, where S denotes the state space, A denotes the action space (here we assume that $|A| < \infty$), P is the transition probability rule where $P(s'|s,a)$ denotes the probability of transitioning to state s' when action a is taken in state s, r is the single-stage reward function where $r(s,a)$ denotes the reward obtained by the agent from the environment $\mathcal{E}$ when action a is taken in state s, $0 < \gamma < 1$ denotes the discount factor and d_0 denotes the initial distribution over states. We define a policy $\pi : S \to \Delta(A)$[1] as a mapping from state space to the set of distributions over actions. For a given policy π, we define $J(\pi)$, the (normalized) discounted long-run reward as follows:

$$J(\pi) = (1-\gamma)\ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big| s_0 \sim d_0\right], \tag{1}$$

where the expectation is taken over the states and actions encountered at times $t = 0, 1, 2, \ldots, \infty$, with $a_t \sim \pi(s_t, \cdot)$. The objective of an RL agent is to compute a policy π^* such that,

$$\pi^* = \arg\max_{\pi \in \Pi} J(\pi), \tag{2}$$

where Π is the set of all polices.

To obtain π^*, the RL agent typically starts with an initial policy and iteratively finds improved policies. If the agent executes the current learned policy at every stage, we refer to such a setting as "on-policy control". As discussed earlier, this may not be a favorable setting in many realistic scenarios. Instead, the agent would have to rely on the data generated by a different policy (known as the behavior policy) at every stage of policy improvement. This setting is referred to as "off-policy control". Let μ denote the behavior policy using which the data (trajectories/samples) gets generated. An example of such a policy would be the uniform policy, i.e., the one that takes all actions with equal probability in all the states.

Note that the set of all stochastic policies Π is uncountably infinite which renders the maximization in (2) intractable. This forces one to resort to function approximation techniques [28]. One popular way of doing this is by parameterizing the policy π. Let $\tilde{\Pi} = \{\pi_\theta \mid \theta \in \mathbb{R}^k,\ k \geq 1\}$ be a family of policies parameterized by θ. For example, a soft-max policy parameterization $\forall s, a$ is described as:

$$\pi_\theta(s,a) = \frac{\exp\left(f(\theta, s, a)\right)}{\sum_{b \in A} \exp\left(f(\theta, s, b)\right)}, \tag{3}$$

where $f(\theta, s, a)$ is a linear or non-linear function over parameters θ.

The objective of an agent in such a case is to compute an optimal $\theta^* = \arg\max_\theta J(\pi_\theta)$. Note that unlike (2) where the optimization is over all stochastic policies, the optimization here is only over the parameterized policy space $\tilde{\Pi}$. This can be done by performing gradient ascent as follows: consider an initial θ_0 and update the parameters as:

$$\theta_{t+1} = \theta_t + \beta_t \nabla_\theta J(\theta_t),\ \forall t \geq 0, \tag{4}$$

where $\beta_t,\ t \geq 0$ is the step-size sequence. Therefore, the problem of computing an optimal θ^*[2] requires efficiently estimating the gradient $\nabla_\theta J(\pi_\theta)$ from the available data. We first discuss how this is done in the on-policy setting where the data is generated from the same policy π_θ itself.

III. On-Policy Actor-Critic Algorithms

A significant result in RL is the policy gradient theorem [28] that gives the following closed form expression for $\nabla_\theta J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \tilde{d}_{\pi_\theta}, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)], \tag{5}$$

where $\tilde{d}_\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\{s_t = s \mid s_0 \sim d_0\}$ is the discounted visitation distribution, $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$ is the advantage function, $Q^{\pi_\theta}(s,a)$ [28] is the Q-value corresponding to state $s \in S$ and action $a \in A$, and $V^{\pi_\theta}(s)$ [28] is the value function of policy π_θ corresponding to state $s \in S$. Note that in (5), we only require the projection of advantage function onto $\nabla_{\theta_i} \log \pi_\theta(s,a), \forall i \in \{1, 2, \ldots, k\}$ rather than the advantage function itself for computing the gradient. Therefore, in [15], [31], a linear function approximation for $A^{\pi_\theta}(s,a)$ has been considered, i.e.,

$$A^{\pi_\theta}(s,a) \approx x^T \nabla_\theta \log \pi_\theta(s,a), \tag{6}$$

where $\nabla_\theta \log \pi_\theta(s,a)$ is the compatible feature [3] corresponding to state s and action a. If x^* satisfies the optimality condition for the projection (see [15], [31]):

$$\mathbb{E}_{s \sim \tilde{d}_{\pi_\theta}, a \sim \pi_\theta}[(A^{\pi_\theta}(s,a) - x^{*T} \nabla_\theta \log \pi_\theta(s,a)) \nabla_\theta \log \pi_\theta(s,a)] = 0, \tag{7}$$

then the policy gradient (5) can be rewritten as:

$$\nabla_\theta J(\theta) = F(\theta) x^*, \tag{8}$$

[1]Here Δ denotes the simplex whose dimension is the cardinality of actions.

[2]Note that the gradient ascent algorithm guarantees convergence only to the locally optimal solution.

where

$$F(\theta) = \mathbb{E}_{s\sim\tilde{d}_{\pi_\theta}, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T]. \quad (9)$$

Actor-critic algorithms [12], [28] are a popular class of RL algorithms that utilise the policy gradient theorem to compute the optimal policy π_{θ^*}. Here, the critic estimates the advantage value function (x) parameters and the actor uses it to improve the policy parameters (θ).

The policy gradient update rule in (4) does not take into account the curvature information of the objective function. This makes the policy parameter updates sensitive to the learning rates. To tackle this problem, natural policy gradients have been considered in the literature [3], [23], [24]. Here, the standard gradient $\nabla_\theta J(\theta)$ is replaced by the natural gradient:

$$G^{-1}(\theta)\nabla_\theta J(\theta), \quad (10)$$

where $G(\theta)$ is the Fisher information matrix that encodes curvature information [1].

An important result pertaining to natural policy gradients comes from [24, Section 3], where it is shown that $G(\theta) = F(\theta)$ for the infinite horizon discounted problem. Therefore, from (8) and (10), the natural gradient turns out to be the optimal critic parameters x^*, i.e.,

$$G^{-1}(\theta)\nabla_\theta J(\theta) = G^{-1}(\theta)F(\theta)x^* = x^*. \quad (11)$$

We are now ready to propose our on-policy algorithm "Deep Natural Actor-Critic" (cf. Algorithm 1), that utilizes natural policy gradients. The actor update in the proposed algorithm is performed as:

$$\theta_{t+1} = \theta_t + \beta_t G^{-1}(\theta_t)\nabla_\theta J(\theta_t) = \theta_t + \alpha_t x_t, \ \forall t \geq 0, \quad (12)$$

where x_t satisfies:

$$\mathbb{E}_{s\sim\tilde{d}_{\pi_{\theta_t}}, a\sim\pi_{\theta_t}}[(A^{\pi_{\theta_t}}(s,a) - x_t^T \nabla_\theta \log \pi_{\theta_t}(s,a)) \nabla_\theta \log \pi_{\theta_t}(s,a)] = 0. \quad (13)$$

In order to estimate x_t satisfying (13), the Algorithm 1 employs two critics - an advantage critic and a value function critic. The advantage critic solves (13) iteratively (step 12 of Algorithm 1) by approximating[3] $A^{\pi_{\theta_t}}(s,a)$ as $r(s,a) + \gamma V^{\pi_{\theta_t}}(s') - V^{\pi_{\theta_t}}(s)$, where s' is the next state after s. The value function critic then employs deep neural network parameterised by ψ to estimate the value function $V^{\pi_{\theta_t}}$ of the policy π_{θ_t}. In this work, we utilize Temporal Difference (TD(0)) prediction[4] [28], a simple and effective technique to learn the value function (step 10 of Algorithm 1). Finally, using the advantage critic weight x, the policy parameter θ is updated according to (12) (step 14 of Algorithm 1).

Note that $\nabla_\theta \log \pi_\theta(s,a)$ is a matrix of neural network weights (whose size depends on the number of layers and neurons in each layer). Let k be the total number of neural network weights in the policy network. In step 11 of Algorithm 1, we flatten this matrix into a vector that becomes feature vector for the advantage value function estimation (step 12). Finally, we reshape the advantage value function parameters (step 13) to the matrix form for updating the policy network parameters (step 14).

[3]Note that the exact computation of $A^{\pi_{\theta_t}}$ requires computing the expectation over all possible next states. Further, the $V^{\pi_{\theta_t}}(.)$ obtained from the critic is only an approximation to the true value function.

[4]In general, one could also use TD with eligibility traces, i.e., TD(λ), in prediction for trading bias with variance. For ease of exposition, we consider TD(0) prediction here.

Algorithm 1 On-Policy Deep Natural Actor-Critic (Deep NAC)

1: Initialize the policy network parameter θ.
2: Initialize the value Function network parameter ψ.
3: Initialize the advantage value function parameter $x \in \mathbb{R}^k$.
4: Select step-size sequences $\alpha_n, \beta_n, \ \forall n$.
5: **for** $n = 0, \ldots, \infty$ **do**
6: Initialise $s \sim d_0(.)$
7: **while** the trajectory has not terminated **do**
8: Obtain an action $a \sim \pi_\theta(s,.)$.
9: Obtain next state s' and reward r from the environment.
10: $\psi \leftarrow \psi + \alpha_n\big(r + \gamma V_\psi(s') - V_\psi(s)\big)\nabla_\psi V_\psi(s)$
11: $f \leftarrow$ Flatten matrix $\nabla_\theta \log \pi_\theta(s,a)$ into a $k \times 1$ vector.
12: $x \leftarrow x + \alpha_n\big(r + \gamma V_\psi(s') - V_\psi(s) - x^T f\big)f$
13: $M \leftarrow$ Reshape x to the neural network weight matrix.
14: $\theta \leftarrow \theta + \beta_n M$
15: $s \leftarrow s'$

IV. Extension to the Off-Policy Setting

Unlike the on-policy setting, the data in "off-policy control" is obtained from a given behavior policy μ. Note that the data in this setting is obtained in the form of samples (s,a,s'), where $s \sim \tilde{d}_\mu(\cdot)$, $a \sim \mu(s,\cdot)$ and $s' \sim p(\cdot|s,a)$.

Now the policy gradient (5) can be rewritten as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\tilde{d}_{\pi_\theta}, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)] \quad (14)$$

$$= \sum_{s,a} \tilde{d}_{\pi_\theta}(s)\pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)$$

$$= \sum_{s,a} \tilde{d}_\mu(s) \frac{\tilde{d}_{\pi_\theta}(s)}{\tilde{d}_\mu(s)} \mu(s,a) \frac{\pi_\theta(s,a)}{\mu(s,a)} \nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)$$

$$= \mathbb{E}_{s\sim\tilde{d}_\mu, a\sim\mu}[w_\theta(s)\rho_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)], \quad (15)$$

where $w_\theta(s) = \frac{\tilde{d}_{\pi_\theta}(s)}{\tilde{d}_\mu(s)}$ and $\rho_\theta(s,a) = \frac{\pi_\theta(s,a)}{\mu(s,a)}$ are the importance sampling ratios. It is important to note the crucial difference between (14) and (15). While the expectation in (14) is over the state, action samples from the policy π_θ itself, the expectation in (15) is over the behaviour policy μ. This modification allows one to use the state, action samples from the behavior policy during the training of an off-policy actor-critic algorithm.

While estimating $\rho_\theta(s,a)$ is straightforward, estimating the ratio $w_{\theta_t}(s_t)$ from data samples is non-trivial and we employ the iterative algorithm proposed in [17] for this purpose. We

now briefly discuss the off-policy actor-critic algorithm proposed in [18]. Utilising (15), the policy parameters $\theta_t,\ t \geq 1$ are updated using batch stochastic gradient ascent as:

$$\theta_{t+1} = \theta_t + \beta_t \sum_{s \in \mathcal{B}} \frac{w_{\theta_t}(s)}{z} \rho_{\theta_t}(s,a) \nabla_\theta \log \pi_{\theta_t}(s,a) Q^{\pi_{\theta_t}}(s,a), \tag{16}$$

where $\mathcal{B}$ is the batch of state-action samples derived from policy μ at time t, and $z = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} w_{\theta_t}(s)$. They estimate $Q^{\pi_{\theta_t}}$ using non-linear TD(λ) update. The stationary distribution of the Markov chain $(s_t, a_t),\ t \geq 0$ under the behavior policy μ is different from the stationary distribution of the Markov chain under the current policy π_{θ_t}. This difference needs to be corrected while estimating the value function of the current policy. However, their estimation procedure does not correct for the discrepancy in the state distribution (step 14 of Algorithm 1 of [18]).

In our work, we make this important correction in the value function estimation procedure and utilize it to estimate the advantage value function that uses compatible features. We finally use the advantage critic weights for computing the natural policy gradient.

V. Proposed Algorithm

Algorithm 2 Off-Policy Deep Natural Actor-Critic (Deep OffNAC)

1: μ : Behavior policy.
2: Initialize the policy network parameter θ.
3: State Initialize the value function network parameter ψ.
4: Initialize the advantage value function parameter $x \in \mathbb{R}^k$.
5: **for** $n = 0, \ldots, \infty$ **do**
6: Initialize $s \sim d_0(\cdot)$
7: **while** the trajectory has not terminated **do**
8: Estimate $\hat{w}_\theta$ and w_θ using Algorithms 1 and 2, respectively, of [17].
9: Obtain an action $a \sim \mu(s, \cdot)$.
10: Obtain next state s' and reward r from the environment.
11: Set $\rho(s,a) = \frac{\pi_\theta(s,a)}{\mu(s,a)}$.
12: $\psi \leftarrow \psi + \alpha_n \Big(\hat{w}_\theta(s) \rho(s,a) \big(r + \gamma V_\psi(s') - V_\psi(s) \big) \nabla_\psi V_\psi(s) \Big)$
13: $f \leftarrow$ Flatten matrix $\nabla_\theta \log \pi_\theta(s,a)$ into a $k \times 1$ vector.
14: $x \leftarrow x + \alpha_n \Big(w_\theta(s) \rho(s,a) \big(r + \gamma V_\psi(s') - V_\psi(s) - x^T f \big) f \Big)$
15: $M \leftarrow$ Reshape x to the neural network weight matrix.
16: $\theta \leftarrow \theta + \beta_n M$
17: $s \leftarrow s'$

In this section, we alleviate the issue discussed above (see section IV) and propose our on-line convergent off-policy natural actor-critic algorithm. In our algorithm, there are two neural networks and one linear network: a policy neural network that updates the policy parameters θ, a value function neural network parameterized by ψ that estimates the value function corresponding to the policy parameters θ and a linear advantage value critic that uses compatible features $\nabla_\theta \log \pi_\theta(s,a)$ from the actor to compute the natural policy gradient. Recall that $w_\theta(s) = \frac{\tilde{d}_{\pi_\theta}(s)}{\tilde{d}_\mu(s)}$ and $\rho_\theta(s,a) = \frac{\pi_\theta(s,a)}{\mu(s,a)}$. Let $\hat{w}_\theta(s) = \frac{d_{\pi_\theta}(s)}{d_\mu(s)}$ denote the ratio of stationary distributions of the Markov chain $s_t,\ t \geq 0$ following policies π_θ and μ respectively. We assume here that the Markov Chains under μ and $\pi_\theta,\ \forall \theta$, have unique stationary distributions d_μ and d_{π_θ}, respectively. The pseudo-code of our proposed algorithm is given in Algorithm 2. The input to our algorithm is the data samples from the behavior policy. Let the current policy parameters be θ. There are three steps involved in the update of policy parameters at each iteration. These steps are described below:

1. *Estimating the Ratio of Distributions:* We now briefly discuss the procedure for estimating $\hat{w}_\theta$ and w_θ (proposed in [17]) that we utilize in our proposed "Deep OffNAC" Algorithm 2 for correcting the state-action distribution.

For any function $f : S \to \mathbb{R}$ and $w : S \to \mathbb{R}$, define two functions L_1 and L_2 as follows:

$$L_1(w, f) = \mathbb{E}_{(s,a,s')}[\Delta(w; s, a, s') f(s')], \tag{17}$$

where the sample (s, a, s') is drawn according to the distribution $d_\mu(s) \times \mu(s,a) \times P(s'|s,a)$ and $\Delta(w; s, a, s') = w(s)\rho_\theta(s,a) - w(s)$.

$$\begin{aligned} L_2(w, f) =& \gamma \mathbb{E}_{(s,a,s')}[\Delta(w; s, a, s') f(s')] + \\ & (1-\gamma) \mathbb{E}_{s \sim d_0}[(1 - w(s)) f(s)], \end{aligned} \tag{18}$$

where the tuple (s, a, s') is sampled from the distribution $\tilde{d}_\mu(s) \times \mu(s,a) \times P(s'|s,a)$ and $\Delta(w; s, a, s') = w(s)\rho_\theta(s,a) - w(s)$. Note that in (17), the state is sampled from the stationary distribution d_μ and in (18), it is sampled according to the state visitation distribution $\tilde{d}_\mu$.

It is shown in [17, Theorem 1] and [17, Theorem 4] that:

$$L_1(w, f) = 0, \forall f, \text{ iff } w(s) \propto \hat{w}_\theta(s),\ \forall s \in S, \tag{19}$$

$$L_2(w, f) = 0, \forall f, \text{ iff } w(s) = w_\theta(s),\ \forall s \in S, \tag{20}$$

respectively.

Utilising (19), $\hat{w}_\theta$ could be estimated by solving the following min-max optimization problem:

$$\min_w \Big\{ D(w) \coloneqq \max_{f \in \mathcal{F}} L_1(w, f)^2 \Big\}, \tag{21}$$

where $\mathcal{F} = \{ f : S \to \mathbb{R} \}$ is the set of all functions. It is not feasible to directly optimize over all functions $f \in \mathcal{F}$ in (21). Hence, [17] proposes to optimize over a restricted family of functions like Reproducing Kernel Hilbert space (RKHS) with a suitable kernel. This has an additional advantage of obtaining the closed form expression for $D(w)$ as follows:

$$D(w) = \mathbb{E}[\Delta(w; s, a, s') \Delta(w; \bar{s}, \bar{a}, \bar{s}') k(s, s')], \tag{22}$$

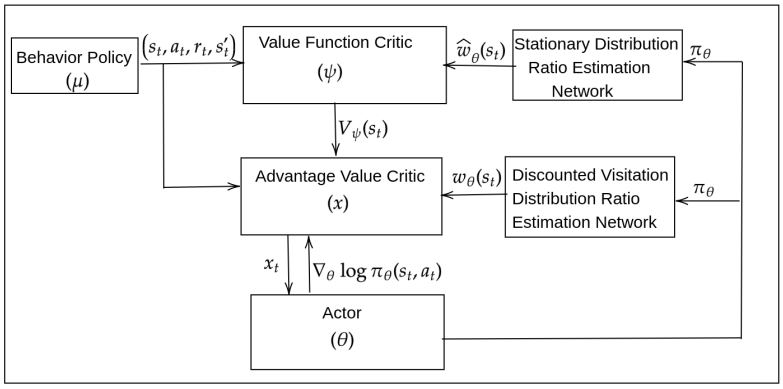


Fig. 1: Flowchart of our proposed off-Policy NAC Algorithm

where (s, a, s') and $(\bar{s}, \bar{a}, \bar{s}')$ are two independent transition pairs obtained from policy μ and k is the positive definite kernel. If $|S| < \infty$ then using (22), $\hat{w}_\theta$ can be estimated by performing gradient descent over w. However, when the number of states is large or infinite, it is not possible to estimate $\hat{w}_\theta(s),\ \forall s$. Hence the function $w : S \to \mathbb{R}$ can be parameterized (by a neural network) and the gradient descent can be performed over the parameters of the network. By replacing L_1 with L_2 in (21) and obtaining a closed form expression similar to (22) (see [17, equation 28, Appendix D]), w_θ could also be estimated in a similar manner.

The ratio of distributions $\hat{w}_\theta$ and w_θ are estimated respectively using Algorithm 1 and Algorithm 2 of [17]. These algorithms take the behavior policy μ and the current policy π_θ as inputs and estimate the ratios $\hat{w}_\theta(s)$ and $w_\theta(s)$ for all $s \in S$. The ratio $\hat{w}_\theta(s)$ is used by the value function critic and $w_\theta(s)$ is used by the advantage value function critic for correcting the state-action distribution mismatch. This will be further explained below.

2. *Estimating the value function:* Next, the value function corresponding to the current policy is estimated. Under function approximation, this corresponds to computing optimal parameter ψ^*. In on-policy setting, the value function is estimated via the update [28]:

$$\psi_{t+1} = \psi_t + \alpha_t(r_t + \gamma V_{\psi_t}(s_t') - V_{\psi_t}(s_t))\nabla_\psi V_{\psi_t}(s_t), \quad \forall t \geq 0, \tag{23}$$

where α_t is the step-size and (s_t, a_t, r_t, s_t') is the data sample derived from policy π_{θ_t}.

However, as the data sample in the off-policy problem is obtained from the behavior policy, i.e., s_t is sampled from $d_\mu(\cdot)$, a_t is sampled from $\mu(s_t, \cdot)$, the update (23) has to be corrected for state and action distribution mismatch. Therefore, we correct the update (23) as follows:

$$\psi_{t+1} = \psi_t + \alpha_t \hat{w}_\theta(s_t)\rho(s_t, a_t)(r_t + \gamma V_{\psi_t}(s_t') - V_{\psi_t}(s_t)) \nabla_\psi V_{\psi_t}(s_t),\ \forall t \geq 0. \tag{24}$$

As $t \to \infty$, $\psi_t \to \psi^*$, almost surely[5]. The correction in (24) is very important as off-policy value function computation without state and action distribution correction can diverge [28]. This crucial value function correction was not done in the previous off-policy control algorithms like [18].

3. *Estimating x corresponding to policy π_θ:* As discussed earlier, the parameter x for the policy π_θ should satisfy (13). This can be computed through an iterative update as follows:

$$x_{t+1} = x_t + \alpha_t(r_t + \gamma V_{\psi^*}(s_t') - V_{\psi^*}(s_t) - x^T \nabla_\theta \log \pi_\theta(s_t, a_t))\nabla_\theta \log \pi_\theta(s_t, a_t),\ \forall t \geq 0, \tag{25}$$

where $r_t + \gamma V_\psi(s_t') - V_\psi(s_t)$ is used as a sample estimate for $A^{\pi_\theta}(s, a)$.

However, the data samples above are obtained from π_θ. In the off-policy problem, we correct the update (25) as follows:

$$x_{t+1} = x_t + \alpha_t w_\theta(s_t)\rho(s_t, a_t)(r_t + \gamma V_{\psi^*}(s_t') - V_{\psi^*}(s_t) - x^T \nabla_\theta \log \pi_\theta(s, a))\nabla_\theta \log \pi_\theta(s, a),\ \forall t \geq 0. \tag{26}$$

We now state a result on the convergence of $\{x_t\},\ t \geq 0$.

Theorem 1. *For a given policy parameter θ, the advantage critic updates $\{x_t,\ t \geq 0\}$ given by* (26)*, where x_0 is a randomly chosen initial point, converge to x^*, where x^* satisfies:*

$$\mathbb{E}_{s \sim \tilde{d}_{\pi_\theta}, a \sim \pi_\theta}[(A^{\pi_\theta}(s, a) - (x^*)^T \nabla_\theta \log \pi_\theta(s, a)) \nabla_\theta \log \pi_\theta(s, a)] = 0. \tag{27}$$

Proof. Please refer Section VI for the proof of Theorem 1. □

4. *Updating the policy parameters:* Finally, as derived in (12), the current policy parameter θ is updated using the natural policy gradient x^* as follows:

$$\theta \leftarrow \theta + \beta x^*, \tag{28}$$

where β is the step-size. As x^* satisfies (27), it is the natural gradient direction corresponding to policy π_θ and hence the policy parameters converge to a local optimal solution.

In the above discussion, each step needs to be carried out till convergence before updating the next step. That is, value

[5]We assume here that the neural network is rich enough to estimate the actual value function.

function parameters ψ_t in (24) have to be iteratively run until they converge to obtain ψ^*. Subsequently, this ψ^* is used in the update of x_t parameters, which must also be run until convergence to compute x^*. Finally, the policy parameter can be updated using (28). However, this procedure is not feasible for implementation and hence we utilize the concepts of two-timescale stochastic approximation [4], [16] to carry out all the steps sequentially with step-size schedules $\alpha_n, \beta_n,\ n \geq 0$ in steps 12-16 of the Algorithm 2 satisfying

$$\sum_{n=0}^{\infty} \alpha_n = \sum_{n=0}^{\infty} \beta_n = \infty,\ \sum_{n=0}^{\infty} (\alpha_n + \beta_n)^2 < \infty,\ \frac{\beta_n}{\alpha_n} \to 0.$$

The flow chart of an iteration of our algorithm is illustrated in Figure 1. Note that the value function ψ_t and natural gradient x_t updates are carried out on a faster time-scale and the policy update θ_t is carried out on a slower time-scale to ensure convergence.

VI. Theoretical Result

In this section, we present the proof of Theorem 1. To prove Theorem 1, we cast the x_t updates (25) in the framework of stochastic approximation. In section VI-A, we briefly describe the standard stochastic approximation scheme [4, Chapters 2,3] and assumptions required for convergence. We then give the convergence proof of iterates x_t followed by the proof of Theorem 1 in section VI-B.

A. Stochastic Approximation scheme

Consider an incremental update stochastic recursion of the following form:

$$x_{n+1} = x_n + a(n)(h(x_n) + M_{n+1}), \tag{29}$$

with a prescribed initial $x_0 \in \mathbb{R}^d$, where

(C1) The map $h : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous.

(C2) $\{a(n)\}$ satisfies $a(n) > 0$, $\forall n$, Also,

$$\sum_{n=0}^{\infty} a(n) = \infty,\ \sum_{n=0}^{\infty} a(n)^2 < \infty.$$

(C3) $\{M_n\}$ is a martingale difference sequence w.r.t increasing sequence of sigma-fields:

$$\mathcal{F}_n = \sigma\{x_m, M_m,\ m \leq n\},\ n \geq 0.$$

(C4) $\mathbb{E}[\|M_{n+1}\|^2 \mid \mathcal{F}_n] \leq K(1 + \|x_n\|^2),\ \forall n,$ for some $0 < K < \infty,$ almost surely.

(C5) The functions $h_c(x) \doteq \frac{h(cx)}{c},\ c \geq 1$ satisfy $h_c(x) \to h_\infty(x)$ as $c \to \infty$, uniformly on compacts, for some $h_\infty : \mathbb{R}^d \to \mathbb{R}^d$. Furthermore, the ODE $\dot{x}(t) = h_\infty(x(t))$ has the origin in $\mathbb{R}^d$ as its unique globally asymptotic stable equilibrium.

Under **(C1-C5)**, we have the following result from [4, Theorem 2, Chapter 2]:

Theorem 2. *Almost surely, the sequence $\{x_n\}$ generated by (29) converges to a compact connected internally chain transitive invariant set of the ODE $\dot{x}(t) = h(x(t))$.*

B. Proof of Theorem 1

For simplicity, we assume that the state and action spaces are finite. The proof can be extended for continuous state spaces as well.

Suppose $s_t \sim \tilde{d_\mu}(.), a_t \sim \mu(s,.), s'_t \sim P(\cdot|s_t, a_t)$ be the sample of state, action and next state obtained at time t. Let us re-write x_{t+1} in (26) in the form of a stochastic approximation scheme defined in (29), i.e.,

$$x_{t+1} = x_t + \alpha_t(h(x_t) + M_{t+1}), \tag{30}$$

where $h(x_t) = -\frac{1}{2}\nabla_x \mathbb{E}_{s\sim\tilde{d}_{\pi_\theta}, a\sim\pi_\theta}[(A^{\pi_\theta}(s,a) - x_t^T \nabla_\theta \log \pi_\theta(s,a))^2]$, and $M_{t+1} = w_\theta(s_t)\rho_\theta(s_t,a_t)(r_t + \gamma V_{\psi^*}(s'_t) - V_{\psi^*}(s_t) - x_t^T \nabla_\theta \log \pi_\theta(s_t,a_t))\nabla_\theta \log \pi_\theta(s_t,a_t) - h(x_t)$.

Let $\mathcal{F}_t = \sigma\{\theta_m, x_m, M_m,\ m \leq t\},\ t \geq 0$, be the increasing family of sigma-fields. We will now verify conditions **(C1-C5)** and invoke Theorem 2 to show the convergence. Define

$$F(\theta) = \mathbb{E}_{s\sim\tilde{d}_{\pi_\theta}, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T]. \tag{31}$$

Then, $h(x_t) = \mathbb{E}[A^{\pi_\theta}(s,a)\nabla \log \pi_\theta(s,a)] - F(\theta)x_t$. For any $x_1, x_2 \in \mathbb{R}^d$, we have

$$\|h(x_1) - h(x_2)\| \leq \|F(\theta)\|\|x_1 - x_2\|. \tag{32}$$

Therefore h is a Lipschitz function with $\|F(\theta)\|$ as the Lipschitz constant and hence **(C1)** is true. Moreover, **(C2)** is satisfied, for example, if the step-size sequence α_t is chosen to be $\alpha_t = \frac{1}{t+1},\ t \geq 0$.

Let us define $\delta_t = r_t + \gamma V_{\psi^*}(s'_t) - V_{\psi^*}(s_t)$. Then, we have

$$\begin{aligned}
\mathbb{E}[\delta_t \mid s_t, a_t, \theta] &= r(s_t,a_t) + \sum_{s'} p(s'|s_t,a_t) V_{\psi^*}(s') - V_{\psi^*}(s_t) \\
&= Q^{\pi_\theta}(s_t,a_t) - V_{\psi^*}(s_t) \\
&= A^{\pi_\theta}(s_t,a_t).
\end{aligned}$$

Therefore, $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t]$

$$\begin{aligned}
&= \mathbb{E}[w_\theta(s_t)\rho_\theta(s_t,a_t)(\delta_t - x_t^T \nabla_\theta \log \pi_\theta(s_t,a_t)) \\
&\qquad \nabla_\theta \log \pi_\theta(s_t,a_t) - h(x_t) \Big| \mathcal{F}_t] \\
&= \mathbb{E}[w_\theta(s_t)\rho_\theta(s_t,a_t)(\mathbb{E}[\delta_t \mid s_t,a_t,\theta,\mathcal{F}_t] - x_t^T \nabla_\theta \log \pi_\theta(s_t,a_t)) \\
&\qquad \nabla_\theta \log \pi_\theta(s_t,a_t) - h(x_t) \Big| \mathcal{F}_t] \\
&= \mathbb{E}[w_\theta(s_t)\rho_\theta(s_t,a_t)(A^{\pi_\theta}(s_t,a_t) - x_t^T \nabla_\theta \log \pi_\theta(s_t,a_t)) \\
&\qquad \nabla_\theta \log \pi_\theta(s_t,a_t) - h(x_t) \Big| \mathcal{F}_t] \\
&= \mathbb{E}_{s\sim\tilde{d}_\mu, a\sim\mu}[w_\theta(s)\rho_\theta(s,a)(A^{\pi_\theta}(s,a) - x_t^T \nabla_\theta \log \pi_\theta(s,a)) \\
&\qquad \nabla_\theta \log \pi_\theta(s,a)] - h(x_t) \\
&= \mathbb{E}_{s\sim\tilde{d}_{\pi_\theta}, a\sim\pi_\theta}[(A^{\pi_\theta}(s,a) - x_t^T \nabla_\theta \log \pi_\theta(s,a)) \\
&\qquad \nabla_\theta \log \pi_\theta(s,a)] - h(x_t) \\
&= 0.
\end{aligned}$$

Therefore, $\{M_t,\ t \geq 0\}$ is a Martingale difference sequence. Hence **(C3)** is proved.

For $0 < K_2, K_3 < \infty$, let

$$\max_{s,a} |r(s,a)| \leq K_2, \tag{33}$$

$$\max_{s,a} \|\nabla_\theta \log \pi_\theta(s,a)\| \leq K_3. \tag{34}$$

Then, we have,

$$|\delta_t| \leq \frac{2K_2}{1-\gamma} = K_4 \text{ (say)}, \ \forall t \geq 0. \tag{35}$$

Next, using Cauchy–Schwarz inequality, we have,

$$\begin{aligned} |x_t^T \nabla_\theta \log \pi_\theta(s,a)| &\leq \|x_t\| \|\nabla_\theta \log \pi_\theta(s,a)\| \\ &\leq K_3 \|x_t\|, \ \forall s,a,t. \end{aligned} \tag{36}$$

The behavior policy μ is selected such that there is positive probability of selecting all actions in each state. For example, the behavior policy chosen in the experiments is the one that selects all actions with equal probability in every state. Therefore, for some $0 < K_5, K_6 < \infty$, and $\forall s, a, \theta$, we have

$$\rho_\theta(s,a) = \frac{\pi_\theta(s,a)}{\mu(s,a)} \leq \frac{1}{\mu(s,a)} \leq \frac{1}{\min_{s,a} \mu(s,a)} \triangleq K_5. \tag{37}$$

$$w_\theta(s) = \frac{\tilde{d}_{\pi_\theta}(s)}{\tilde{d}_\mu(s)} \leq \frac{1}{\tilde{d}_\mu(s)} \leq \frac{1}{\min_s \tilde{d}_\mu(s)} \triangleq K_6. \tag{38}$$

From (34)-(38), it is easy to see that

$$\mathbb{E}[\|M_{t+1}\|^2 \mid \mathcal{F}_t] \leq K(1 + \|x_t\|^2), \text{ almost surely}, \tag{39}$$

where $K = (K_5 K_6)^2 \max\{2(K_3 K_4)^2, 2K_3^4\}$. Hence **(C4)** is true.

Now, let us construct the functions $h_c(x) = \frac{h(cx)}{c},\ c \geq 1$.

$$h_c(x) = \frac{\mathbb{E}[A^{\pi_\theta}(s,a) \nabla \log \pi_\theta(s,a)]}{c} - F(\theta)x. \tag{40}$$

It is easy to see that

$$h_c(x) \to -F(\theta)x, \text{ as } c \to \infty. \tag{41}$$

Note that $F(\theta)$ is the Fisher information matrix, which is known to be positive definite [23]. Therefore, the origin is the unique globally asymptotic stable equilibrium of the ODE $\dot{x}(t) = -F(\theta)x(t)$, with $V_1(x) = \frac{1}{2}\|x\|^2$ serving as the corresponding Liapunov function. Hence, **(C5)** is satisfied.

Finally, let us consider the ODE

$$\begin{aligned} &\dot{x}(t) = h(x(t)) \\ &= -\frac{1}{2} \nabla_x \mathbb{E}_{s \sim \tilde{d}_{\pi_\theta}, a \sim \pi_\theta} [(A^{\pi_\theta}(s,a) - x(t)^T \nabla_\theta \log \pi_\theta(s,a))^2]. \end{aligned} \tag{42}$$

Then, $\{x^*\}$ is a connected internally chain transitive invariant set of the ODE (42), with $V_2(x) = \mathbb{E}_{s \sim \tilde{d}_{\pi_\theta}, a \sim \pi_\theta} [(A^{\pi_\theta}(s,a) - x^T \nabla_\theta \log \pi_\theta(s,a))^2]$ serving as the corresponding Liapunov function. Therefore, from Theorem 2, we have $x_t \to x^*$, almost surely.

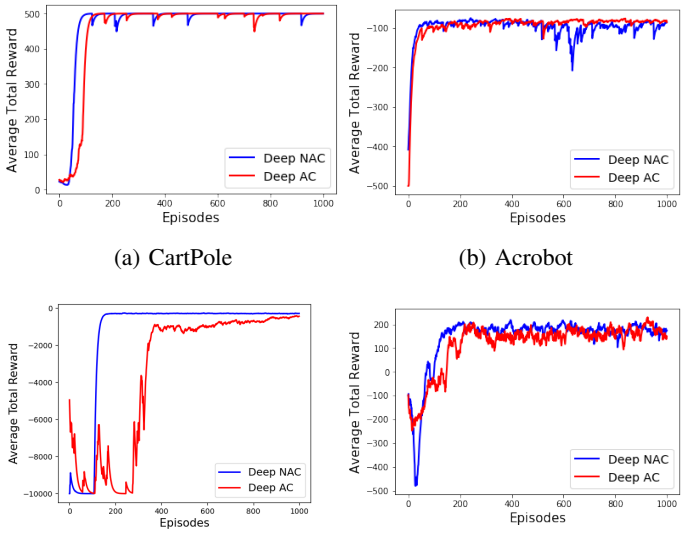
VII. Experiments and Results

In this section, we discuss the performance of the proposed on-policy and off-policy algorithms on four RL tasks available in OpenAI gym [5]. We compare our algorithms "Deep NAC" (Algorithm 1) and "Deep OffNAC" (Algorithm 2) with their standard actor-critic counterparts (which we refer to as) "Deep AC" in the on-policy setting and "Deep OffAC" in the off-policy setting.

A. Experimental Setup

We study the performance of the algorithms in the training and testing phases. The objective in the training phase is to measure how fast an algorithm reaches the maximum total reward. This is depicted through plots where the x-axis denotes the total number of episodes utilised for the training and the y-axis denotes the average total reward achieved by the algorithm. The average total reward at an episode i is calculated as: Average total reward (i) = 0.9 $\times$ total reward obtained in episode i + 0.1 $\times$ Average total reward $(i-1)$. This metric gives higher weight to the recent episodes and hence is a better indicator of the performance of the policy. Subsequently in the testing phase, the best performing policy (the policy parameters corresponding to the highest average total reward) during the training is chosen and its performance (mean and standard deviation of the total reward) over $1,000$ independent episodes is tabulated. The code for our experiments, detailed description of neural network architectures and hyper-parameters for all the algorithms are available at [8]. We discuss our numerical results in the on-policy setting followed by results in the off-policy setting.

B. On-Policy Results



(a) CartPole (b) Acrobot

(c) Mountain Car (d) Lunar Lander

Fig. 2: Performance of Deep NAC and Deep AC algorithms during the training phase

The performance of algorithms during the training and testing phases is shown in Figure 2 and Table I respectively.

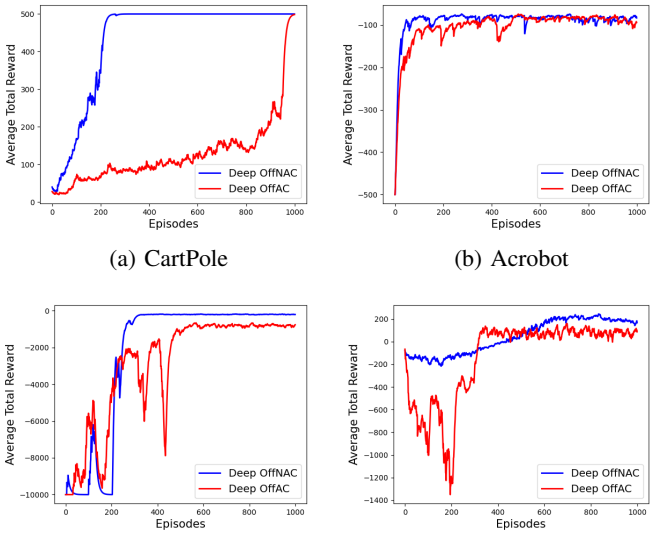
Environmment	Deep AC	Deep NAC
CartPole	497.20±6.29	498.60±4.9
Acrobot	-83.91±4.92	-84.83±6.31
Mountain Car	-448.95±29.10	-291.13±8.38
Lunar Lander	171.51±24.14	179.93±15.90

TABLE I: Comparison between Deep NAC and Deep AC during the testing phase

From Figure 2, we can observe that our proposed "Deep NAC" algorithm reaches the maximum total reward faster compared to the "Deep AC" algorithm in all the four tasks.

The natural gradients employed by the actor in the "Deep NAC" algorithm make use of the curvature information and hence learn faster than the "Deep AC"algorithm. Moreover, as the natural policy gradients are less sensitive to model reparameterization, we observe that hyper-parameter tuning for "Deep NAC" is easy compared to the "Deep AC" algorithm. Table I illustrates the performance comparison in the test phase, i.e., the mean and variance of the best learnt policy on all the four tasks.

C. *Off-Policy Results*



(a) CartPole (b) Acrobot

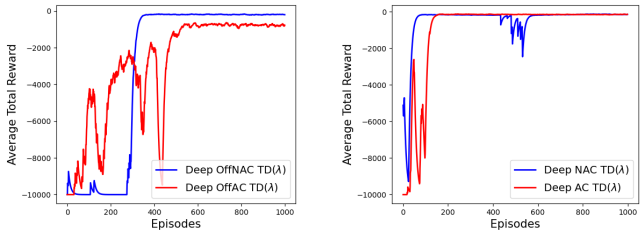
(c) Mountain Car (d) Lunar Lander

Fig. 3: Performance of Deep OffNAC and Deep OffAC algorithms during the training phase

Environmment	Deep OffAC	Deep OffNAC
CartPole	499.62±0.95	500±0
Acrobot	-85.33±5.36	-84.28±4.78
Mountain Car	-874.62±76.17	-202.62±8.85
Lunar Lander	87.05±27.55	215.14±12.97

TABLE II: Comparison between Deep OffNAC and Deep OffAC during the testing phase

Here, the data during the training is collected from a behavior policy μ where all actions are uniformly chosen in every state. In order to evaluate the performance during the training, we run an episode with the current learned parameters after every training episode. Similar to the on-policy results we see from Figure 3 and Table II that our proposed "Deep OffNAC" algorithm reaches the maximum total reward faster in the training phase. Moreover, the best policy obtained from the "Deep OffNAC" gives higher reward during the testing compared to the "Deep OffAC" algorithm.



(a) On-Policy (b) Off-Policy

Fig. 4: Comparison between Deep AC and Deep NAC with TD(λ) critic during the training phase

Mountain Car	Deep AC with TD(λ) Critic	Deep NAC with TD(λ) Critic
On-Policy	-147.70±7.39	-143.29±6.02
Off-Policy	-771.47±56.07	-192.66±9.00

TABLE III: Comparison between Deep NAC and Deep AC algorithms with TD (λ) critic on Mountain Car

From Tables I and II, we see that the average rewards achieved by both the algorithms for the Mountain Car task are not satisfactorily high. As this task suffers from delayed reward problem (it receives a reward of 0 only when it reaches the top and -1 at intermediate steps), TD(0), because of it's high bias is unable to estimate the value function optimally. Therefore, we consider TD(λ) critic in place of TD(0) for optimally estimating the value function [28]. In Figure 4 and Table III, we show the results of Mountain Car trained using TD(λ) critic in both on-policy and off-policy settings during training and testing, respectively. The parameter λ is treated as a hyperparameter and the best value is selected in each setting. We observe that the average rewards have increased in both on-policy and off-policy settings. Further, the performance of both Deep NAC and Deep OffNAC is better than their standard actor-critic counterparts.

VIII. Conclusions and Future Work

In this work, we have proposed the first on-policy deep actor-critic algorithm that utilizes natural gradients in the policy update and compatible features in the advantage critic update for faster convergence. We then extended this algorithm to the off-policy control and proposed a convergent algorithm.

We illustrated the advantages of the proposed algorithms by comparing them with the standard actor-critic algorithms.

In the future, we would like to perform ablation studies to better understand the role of the ratio of state-action distributions in the performance of off-policy algorithms. Moreover, it would be interesting to integrate algorithms like Gradient TD [29], Emphatic TD [30] (for value function estimation) with the proposed algorithm. Another important future work would be to test the efficacy of our algorithms on the Atari games test bed. As these games require deeper neural networks along with convolutional neural networks extracting compatible features and implementing natural gradients in these settings is a challenging task.

IX. Acknowledgements

The authors would like to thank Prof. Koteswararao Kondepu at IIT Dharwad for generously providing GPU support to run our experiments. We also thank High Performance Computing (HPC), IIT Dharwad for computational resources.

References

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*, volume 5. Athena Scientific Belmont, MA, 1996.

[3] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[4] Vivek S Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge Univ. Press, 2008.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[7] Carles Gelada and Marc G Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3647–3655, 2019.

[8] Github. Experimental codes. https://github.com/prateekjain2606/NeuralNetworkCompatibleOffPolicyNaturalActorCriticAlgorithm, 2022.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[10] Ehsan Imani, Eric Graves, and Martha White. An off-policy policy gradient theorem using emphatic weightings. *arXiv preprint arXiv:1811.09013*, 2018.

[11] Ryo Iwaki and Minoru Asada. Implicit incremental natural actor critic algorithm. *Neural Networks*, 109:103–112, 2019.

[12] Prabuchandran K J, Shalabh Bhatnagar, and Vivek S Borkar. Actor-critic algorithms with online feature adaptation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(4):1–26, 2016.

[13] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

[14] Sajad Khodadadian, Zaiwei Chen, and Siva Theja Maguluri. Finite-sample analysis of off-policy natural actor-critic algorithm. *arXiv preprint arXiv:2102.09318*, 2021.

[15] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014. Citeseer, 2000.

[16] Harold Joseph Kushner and Dean S Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26. Springer Science & Business Media, 2012.

[17] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *arXiv preprint arXiv:1810.12429*, 2018.

[18] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with stationary distribution correction. In *Uncertainty in Artificial Intelligence*, pages 1180–1190. PMLR, 2020.

[19] Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, Doina Precup, David Silver, and Richard S Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pages 1204–1212, 2009.

[20] Hamid Reza Maei and Richard S Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *3d Conference on Artificial General Intelligence (AGI-2010)*. Atlantis Press, 2010.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[22] Hamidreza Modares, Frank L Lewis, and Zhong-Ping Jiang. H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE transactions on neural networks and learning systems*, 26(10):2550–2562, 2015.

[23] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[24] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 2003.

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[26] Ruizhuo Song, Frank L Lewis, and Qinglai Wei. Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games. *IEEE transactions on neural networks and learning systems*, 28(3):704–713, 2016.

[27] Ruizhuo Song, Frank L Lewis, Qinglai Wei, and Huaguang Zhang. Off-policy actor-critic structure for optimal control of unknown systems with disturbances. *IEEE Transactions on Cybernetics*, 46(5):1041–1050, 2015.

[28] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT press Cambridge, 2018.

[29] Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000. ACM, 2009.

[30] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016.

[31] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063. Citeseer, 1999.

[32] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[33] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *arXiv preprint arXiv:1708.05144*, 2017.

[34] Tengyu Xu, Zhuoran Yang, Zhaoran Wang, and Yingbin Liang. Doubly robust off-policy actor-critic: Convergence and optimality. *arXiv preprint arXiv:2102.11866*, 2021.

[35] Guodong Zhang, James Martens, and Roger Grosse. Fast convergence of natural gradient descent for overparameterized neural networks. *arXiv preprint arXiv:1905.10961*, 2019.

[36] Shangtong Zhang, Wendelin Boehmer, and Shimon Whiteson. Generalized off-policy actor-critic. *arXiv preprint arXiv:1903.11329*, 2019.

[37] Shangtong Zhang, Bo Liu, Hengshuai Yao, and Shimon Whiteson. Provably convergent two-timescale off-policy actor-critic with function approximation. In *International Conference on Machine Learning*, pages 11204–11213. PMLR, 2020.