Augmented Memory Replay in Reinforcement Learning With Continuous Control

Mirza Ramičić
Artificial Intelligence Center
Faculty of Electrical Engineering
Czech Technical University in Prague
Prague, Czech Republic
Email: ramicmir@fel.cvut.cz

Andrea Bonarini
Artificial Intelligence and Robotics Lab
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
Email: andrea.bonarini@polimi.it

***Abstract*—Online reinforcement learning agents are currently able to process an increasing amount of data by converting it into a higher order value functions. This expansion of the information collected from the environment increases the agent's state space enabling it to scale up to more complex problems but also increases the risk of forgetting by learning on redundant or conflicting data. To improve the approximation of a large amount of data, a random mini-batch of the past experiences that are stored in the replay memory buffer is often replayed at each learning step. The proposed work takes inspiration from a biological mechanism which acts as a protective layer of higher cognitive functions found in mammalian brain: active memory consolidation mitigates the effect of forgetting previous memories by dynamically processing the new ones. Similar dynamics are implemented by the proposed augmented memory replay or *AMR* algorithm. The architecture of *AMR*, based on a simple artificial neural network is able to provide an *augmentation policy* which modifies each of the agents experiences by augmenting their relevance prior to storing them in the replay memory. The function approximator of *AMR* is evolved using genetic algorithm in order to obtain the specific *augmentation policy* function that yields the best performance of a learning agent in a specific environment given by its received cumulative reward. Experimental results show that an evolved *AMR* augmentation function capable of increasing the significance of the specific memories is able to further increase the stability and convergence speed of the learning algorithms dealing with the complexity of continuous action domains.**

***Index Terms*—temporal-difference learning, deterministic policy gradient, replay memory, continuous action spaces, adaptive actor-critic, active memory consolidation, computational memory architectures, deep neural networks**

I. Introduction

Due to their limited capacity it is crucial for both artificial and biological learning systems to maintain the balance between retaining previously learned information and integrating newly obtained one. This problem has been known as *stability-plasticity* dilemma ever since Carpenter and Grossberg [1] proposed a computational *Adaptive Resonance Theory*, or ART, in order to mitigate the effects of near-complete forgetting of pre-existing knowledge in neural network architectures (also known as catastrophic interference [2]).

Since the *artificial neural network* or ANN architectures take inspiration from biological learning systems, the burden of explanation naturally turned on the broad area of *neuroscience*: how can a human brain sustain its amazing adaptation property of *neuroplasticity* [3], [4] while still be able to retain the previously learned information in an effective way?

Extensive clinical evidence, supporting the fact that the damages occurred in mammalian hippocampus region only disrupts the recent memory while leaving the past ones intact, prompted for a *connectionist* approach [5] of the memory organization in the brain. The work by McLelalland et al. [5] proposes that, although adaptation of the synaptic links occurs in a wide range of areas the mammalian brain makes use of two main complementary systems for its memory retention capabilities: one system is in charge of adapting the synapses of the neurons directly responsible for the processing of the information namely in the *neocortex*. The other part adapts the synaptic connections within a specialized memory system such as *hippocampus*.

The complementary learning systems models such as [5] rely on the ubiquitous interaction between the two systems in order to explain the inherent plasticity of the mammalian brain: the *hippocampal* memory system is able to support the rapid acquisition of the new memories and play them back to the *neocortical* system in an "off-line" manner.

This reinstatement of the memories is expanded in recent literature as *active memory consolidation* [6]–[9] or AMC. Consolidation represents a buffer or a layer which facilitates a better memory integration into the higher level cortical structures and also prevents forgetting previously encoded information; it usually occurs while sleeping or resting [10], a time when the brain is not encoding or perceiving new stimuli. Before its integration in the long term cortical structures, each experience is mediated by a reactivation and replay in the hippocampal memory as a part of the reinstatement. In the process of reinstatement, or consolidation, memories are altered either by relevance or compression [11] in a way that their further integration into the existing knowledge would not induce to forget the previous ones.

More recent studies [12]–[14] suggest that hippocampal replay is not a simple function of experience and that its *content* reflects the cognitive and behavioural demands of a specific task. In other words, the content of the replay structures is

believed to be heavily modified by the goal directed behaviour. The active structural modification of AMC is selective and it will facilitate strengthening the memories that are deemed to be the more important ones, so to reach a certain retrieval threshold. However, if the memory trace is deemed as not strong enough for some memories, it will result in their loss [15], [16]. Biological architectures found in human brain and the computational reinforcement learning processes use a functionally similar mechanism of replay memory. Along the introduction of artificial neural networks, or ANN, as function approximators in temporal-difference, or TD, learning [17], the techniques that aim at their efficient training most commonly use a replay buffer of previous experiences out of which a mini-batch is sampled for re-learning at each time step. This technique has been recently revived in Deep Q-learning [18], [19]. Since in TD approaches the ANN is constantly updated to better represent the state-action value pairs $Q(s, a)$ that govern the agent's policy π, the mechanisms involved in its training such as mini-batch replay became increasingly influential to the learning process itself.

Another advantage of the replay memory structure is that, when implemented, it acts as a form of agent's cognition: depending on the way it is populated, it can alter how the agent perceives the information. In this way, a learning agent is not only concerned about the information it receives from its immediate environment, but also about the way in which this information is interpreted by this cognitive mechanism. The replay memory can now pose as an artificial mediating buffer which facilitates the integration of the acquired information into the main Q-approximating ANN.

In the proposed approach, a simple, still effective mechanism of replay memory is extended with the ability to actively and dynamically process the information during the replay, thus bringing it closer to the functional characteristics of actual biological mechanisms. The dynamic processing mechanism of *Augmented Memory Replay*, or *AMR*, presented here is inspired by human *memory consolidation* and it is capable of altering the relevance of specific memories by aggregating their externally supplied reinforcement signal with a dynamically-generated, *augmenting* reinforcement signal based on the concept of *adaptive critic elements* introduced by [20]. The dynamics of the *AMR's augmenting* secondary reinforcement signal are meta-learned in order to maximize their adaptive, consolidation-supporting effect of increasing or decreasing the importance of a specific memory.

In the experiments reported in this paper, the *augmentation* dynamics are evolved over generations of learning agents performing reinforcement learning tasks on realistic real-world problems with continuous action spaces. Their fitness function is defined in a straightforward way as their cumulative performance over a specific environment. Experimental results indicate that *AMR* type of memory buffer shows an improvement in learning performance over the standard static replay method in all the tested environments.

II. Related Work

A. Computational Approaches of Complementary Learning Systems

The increase of computational power along with the increasing complexity of ANN architectures brought forward a scientific community interested in modeling the overwhelming complexity of the processes that a mammalian brain involves in consolidating memories.

Pioneering works on the subject are largely based on O'Reillys Hebbian learning model LEABRA [21] and exploit the anatomical and physiological evidence presented in [5], [22]; [23] introduced a computational model of hippocampus able to facilitate a recollection of memories in such a way that would mitigate the high level of interference and it was followed by others [24]–[26]. Latter computational models simulated in the work by Kali et al. [27] suggest that the cortical memory replay *alone* protects episodic type of memories against representational changes.

The *oscillating inhibition* approach presented by Norman et al. [28] addresses Leabra's lacks of a mechanism that would selectively "punish" less important memories during their retrieval. In line with the hereby proposed approach of *AMR* [28] similarly introduces a cortical and hippocampal computational model capable of combining selective strengthening of the memory traces along with their targeted punishment.

One of the key points of the proposed *AMR* technique is expanding the memory protection mechanisms such as the one presented in [28] to the general RL problems which are also susceptible to the catastrophic forgetting effect [2], since they heavily rely on the ANN ability to approximate large state spaces while using the power of well known RL techniques such as replay memory [17]–[19].

The *AMR* also stems from the more recent works by Kumaran et al. [29] that are bridging the gap between the complementary learning systems such as [5], [28] and the RL ones.

B. Adaptive Critic: Generating an Intrinsic Reward Signal

The *adaptive critic* approach [20] was primarily developed to address the *problem of delayed rewards* caused by an inherit *reward sparsity* of many modern RL problems: estimating the expected returns proved to be a challenging task when faced with a reinforcement function which rarely yields non-zero rewards, or yields them only after a long sequence of actions. It is also know as *temporal credit assignment problem*, indicative of the inability of the temporal-difference RL algorithms such as bootstrapping Q-variants to effectively propagate the effects of a sparse delayed reward to the states preceding the one where reinforcement has been obtained.

Adaptive critic agent mitigates the effects of a sparse reinforcement by *predicting* its return, and in turn, acts upon that prediction: an action that improves the likelihood of a higher *predicted* return is reinforced. The predictive property of an *adaptive critic* gives rise to a concept of a *secondary reinforcer*: an intrinsic reward augmenting stimulus that has,

through a process of learning, obtained an ability to act as a external or primary reinforcer of the agent's immediate environment [30]. Building upon the concept of *adaptive critic*, *AMR* is able to alter the importance of a specific memory by generating a secondary, intrinsic reinforcement signal that is augmenting its stored primary reinforcement each time the memory is replayed.

C. Reinforcement Learning

Although the proposed *AMR* approach can be applied to any RL problem, its potential for memory augmentation can be best demonstrated over a specific class of complex RL problems that are dealing with a continuous action space. These types of problems cannot be solved using vanilla Q-learning [31] simply because approximating the Q-values for infinite amount of possible actions is not feasible. Instead of this, the agent is required to follow a deterministic actor policy and allow a critic to determine how valuable the state-action value pair is. This actor-critic architecture is referred to as *deep deterministic policy gradient*, or DDPG, and has been introduced by Lillicrap et al. [32]. Since then, the DDPG algorithm has been successful in dealing with continuous action tasks in its basic form or extensions. *AMR* takes advantage of the fact that the specific memories stored in the replay buffer are transitions that have been executed using *deterministic* actions selected by the agent's actor policy. Thus, in the process of augmenting the reward signal or importance of a memory, *AMR* also reinforces the importance of a deterministic action choice stored in that memory.

Through its meta-learning process *AMR* is able to associate the most effective deterministic action choices to specific environment states. By augmenting the importance of the memories it indirectly augments specific state-action pairs that are predicted to be more effective in exploring the infinite state/action space.

Mediated by *AMR* process, good deterministic actions choices can be selected/reinforced even when their influence on the agent's future primary reward is highly indirect.

D. Extending the DDPG

An extension of DDPG algorithm was proposed by Hausknecht and Stone [33] allowing it to deal with a low level parameterized-continuous action space. However, the evaluation of the approach was limited to a single simulated environment of RoboCup 2D Half-Field-Offense [34]. Hoothoft et. al [35] proposed a meta-learning approach capable of evolving a specialized loss function for a specific task distribution that would provide higher rewards during its minimization by stochastic gradient descent. The algorithm can produce a significant improvement of the agent's convergence to the optimal policy but, as it is the case with the *AMR* approach, the evolved improvements are task specific.

Wang et. al [36] introduced an approach that is combining the importance or prioritized sampling techniques together with stochastic dueling networks in order to improve the convergence of some continuous action tasks, such as Walker [37] and Humanoid [38].

Another improvement of a vanilla DDPG is presented by Dai et al. [39] as Dual-Critic architecture where the critic is not updated using the standard temporal-difference algorithms, but it is optimized according to the gradient of the actor.

An approach by Pacella et al. [40] evolved basic emotions such as fear, used as a kind of motivational drive that governs the agent's behavior by directly influencing action selection. As in the *AMR* approach, a population of virtual agents were tested at each generation. In this process, each of the agents evolved a specific neural network that was capable of selecting its actions based on the input; this consisted of temporal information, visual perception and good and bad sensation neurons. Over time, the selection of best performing agents gave rise to populations that adopted specific behavioral drives such as being cautious or fearful as a part of a survival strategy. Contrary to *AMR*, which evolves a cognitive mechanism that only complements the main learning process, in [40] the genetic algorithm represents the learning process itself.

Another evolutionary approach that is used to complement the main reinforcement learning algorithm was presented in [41]. Similarly to *AMR*, it uses a genetic algorithm to evolve an *optimal reward function* which builds upon the basic reward function in a way that maximizes the agent's fitness over a distribution of environments. Experimental results show the emergence of an intrinsic reward function that supports the actions that are not in line with the primary goal of the agent. Schembri et al. [42] also presented a reinforcement learning approach that relied on an evolved reinforcer in order to support learning atomic meta-skills. The reinforcement was evolved in a *childhood* phase, which equipped the agents with the meta-actions or skills for the use in the *adulthood* phase.

Persiani et al. [43] proposed a cognitive improvement through the use of the replay memory structure, like *AMR*. The algorithm makes it possible to learn which chunks of agent's experiences are most appropriate for replay based on their ability to maximize the future expected reward.

A cognitive filter structure was proposed by Ramicic and Bonarini [44] and was able to improve the convergence of temporal-difference learning implementing discrete control rather than a continuous one. It was able to evolve the ANN capable of selecting whether a specific experience will be sampled into replay memory or not. Unlike *AMR* this approach did not modify the properties of the experiences.

Similarly to the here presented *AMR* approach, a recent work by Mattar et al. [45] have also dealt with the intersection between the neuroscientific models of hippocampal memory consolidation and the artificial replay memory in reinforcement learning: they implemented a prioritization of the agents replay memory to a DYNA-based reinforcement learning algorithm [46], based on its predicted utility. Another proposal by Khamassi et al. [47] applied prioritization based on the model-based search to the value-iterated reinforcement learning problems [48].

[49], [50] present computational models of the

hippocampal-cortex interaction mostly based on the [5] with regards to the temporal-difference reinforcement learning.

III. Theoretical Background

A. Temporal-difference learning

The goal of a reinforcement learning agent is to constantly update the function which maps its state to their actions, i.e. its policy π as close as possible to the *optimal policy* π^*. The *optimal policy* is a policy that selects the actions that maximize the future expected reward of an agent in the long run [48] and it is represented by a function, possibly approximated by an *Artificial Neural Network* or ANN. The process of updating the policy is performed iteratively after each of the consecutive discrete time-steps in which the agent interacts with its environment by executing its action a_t, and gets the immediate reward scalar r_t defined by the *reinforcement function*. This iterative step is defined as a transition over a *Markov Decision Process*, and is represented it by a tuple (s_t, a_t, r_t, s_{t+1}). After each transition the agent corrects its existing policy π according to the optimal action-value function shown in Equation 1 in order to maximize its expected reward within the existing policy. In the approaches that deal with discrete action spaces, such as [31], the agent can follow the optimal policy π^* by taking an optimal action $a^*(s)$ which maximizes the optimal action-value function $Q^*(s, a)$ defined by Equation 2.

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi] \tag{1}$$

$$\mu(s) = a^*(s) = \max_{a} Q^*(s, a) \tag{2}$$

$$Q^{\pi}(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^{\pi}(s', a') | s, a\right] \tag{3}$$

The correction update to the policy π starts by determining how wrong the current policy is with respect to the expectation, or value for the current state-action pair $Q(s, a)$. In case of a discrete action space, the expected return is defined by the Bellman-optimality Equation 3 and it is basically the sum of the immediate reward r and the discounted prediction of a maximum Q-value, given the state s‘ over all of the possible actions a‘.

B. Facing continuous action spaces

Maximizing over actions in Equation 2 is not a problem when facing discrete action spaces, since the Q-values for each of the possible actions can be estimated and compared. However, when coping with continuous action values this approach is not realistic: we cannot just brute force the values of the whole action space in order to find the maximum. The more recent approach of [32] eliminates the maximization problem by approximating the optimal action $a^*(s)$ and thus creating a deterministic policy $\mu(s)$ in addition to the optimal state-value function $Q^*(s, a)$. Taking the new approximated policy into consideration, the Bellman-optimality equation takes the form of Equation 4 and avoids the inner expectation.

$$Q^{\mu}(s, a) = \mathbb{E}\left[r + \gamma Q^{\mu}(s', \mu(s')) | s, a\right] \tag{4}$$

The concept of *temporal difference*, or TD error, is common among all the before mentioned approaches; it is basically the difference between the current approximate prediction and the expectation of the Q value. The learning process performs an iterative reduction of a TD error using the Bellman-optimality equation as a target, which guarantees the convergence of the agent's policy to the optimal one given an infinite amount of steps [48].

C. Function approximation

In order to deal with the increasing dimension and continuous nature of state and action spaces imposed by the real-life applications, the aforementioned algorithms depend heavily on approximation methods usually implemented using ANN. A primary function approximation makes it possible to predict a Q value for each of the possible actions available to the agent by providing an agent's current state as input of the ANN. After each time step, the expected Q value is computed using Equation 4, and then compared to the estimate that the function approximator provides as its output $Q(s, a; \Theta) \approx Q^*(s, a)$ by forwarding the state s as its input. The difference between the previous estimate of the approximator and the expectation represents the TD error. This discrepancy is represented by the actual loss function $L_i(\Theta_i)$ that can be minimized by performing a *stochastic gradient descent* on the parameters Θ. The iterative process updates the current approximation of $Q^*(s, a)$ according to Equation 5:

$$\nabla_{\Theta_i} L_i(\Theta_i) = \left(y_i - Q(s, a; \Theta_i)\right) \nabla_{\Theta_i} Q(s, a; \Theta_i), \tag{5}$$

where $y_i = r + \gamma Q^{\mu}(s', \mu(s')); \Theta_{i-1})$ is in fact the Bellman equation defining the target value which depends on a yet another ANN that approximates the policy function $\mu(s)$ in policy-gradient approaches such as [32]. The update to the policy function approximator $\mu_{\Theta}(s)$ is more straightforward as it is possible to perform a gradient ascent on the respective network parameters Θ in order to maximize the $Q^{\mu}(s, a)$ as shown in Equation 6.

$$\max_{\theta} \mathop{\mathrm{E}}_{s \sim \mathcal{D}} \left[Q^{\mu}(s, \mu_{\theta}(s))\right] \tag{6}$$

IV. Model Architecture and Learning Algorithm

In this section we propose a new model that combines the learning approach of genetic algorithms with reinforcement learning in order to improve the convergence of the latter.

The genetic algorithm, as a meta-learning component, is used to learn a specific *augmentation policy* $A_t(s_t, s_{t+1}, r_t)$ capable of altering the experiences stored in the agent's replay memory in a way that would lead to the maximization of their cumulative reward gained in the process of *reinforcement learning*.

For clarity, the proposed model is presented as two main functional parts: the reinforcement learning *evaluation* phase

outlined in Figure 1 and *evolution* phase in which an optimal *augmentation policy* is obtained through the use *genetic algorithm* or GA techniques.

The evaluation part is defined as a temporal-difference reinforcement problem as Algorithm 1 where a reward function is dynamically modified by the proposed *AMR* block represented in the section (f) of Figure 1. Architecturally, the *AMR* block (f) consists of a function approximator implemented by an ANN, capable of outputting a single scalar value. The value of the outputted scalar A_t determines the agents augmentation policy, which, in turn is able to modify the reinforcement value of a specific transition before storing it to the replay memory sliding window structure in Figure 1 (a).

The architecture of the AMR neural network approximator (f) consists of three layers: three input nodes fully connected to a hidden layer of four nodes, in turn connected to two softmax nodes to produce the final classification. This ANN is able to approximate the four parameters of the experience, respectively given an input, as TD error, reinforcement r_t, entropy of the starting state s_t, and entropy of the next transitioning state s_{t+1}, to a regression output layer that provides a scalar *augmentation* value $A_t(TD_t, r_t, H(s_t), H(s_{t+1}); \Theta^{AMR})$ parameterized by its learnable weights Θ^{AMR}.

The *augmentation* process alters the reward value of each transition by an *augmentation rate* or β as shown in Equation 7.

$$r_t := r_t + \beta A_t^* \tag{7}$$

where

$$A_t^* \approx A_t(TD_t, r_t, H(s_t), H(s_{t+1}); \Theta^{AMR}) \tag{8}$$

In order to perform the actual temporal-difference learning the main learning loop in Figure 1 (b) samples a minibatch of K experiences augmented by (f) from the replay memory buffer (a). At each learning step, a loss function for the augmented minibatch is minimized by performing a stochastic gradient descent on the function approximator (d).[r2]

By altering the reward scalars r_t of the transitions prior to their memory storage the *AMR* block is able to dynamically modify the amount of influence each transition exerts on the learning process (b), which, in the context of Q-learning approaches, can be represented by its TD-error.[r2]

TD-error of a specific transition, defined by Bellman-optimality Equation 3 is a function of its immediate reward scalar r_t. Thus, by applying *AMR* dynamics to a specific transition r_t the Algorithm 1 is able to impose an indirect importance advantage $A_t > 0$ or importance penalty $A_t < 0$ through Equation 7. The importance-altering dynamics of *AMR* computational processes can mimic the properties of the *connectionist learning systems* approaches [5], [12]–[14] ability to *alter* the experiences stored in the memory in the process of their consolidation.

The second component of the proposed architecture constantly evolves the ANN's weights of the *AMR* block used in evaluation DDPG trials as a component of the integrated RL-GA architecture, marked as $b)$ in Figure 2, using a *genetic algorithm* in order to improve the *augmentation policy*. Therefore, learning the optimal *augmentation policy* for a specific environment through generations of trials enables the agent to increase their performance in DDPG trials by changing the way they manage or "consolidate" their experiences stored in replay memory structure.

Both the evolutionary and reinforcement learning phase are therefore implemented in the *integrated architecture* outlined in Figure 2 with the *reinforcement learning* phase consisting of n independent DDPG trials being represented as the $b)$ component of Figure 2, each DDPG trial being an independent single instance of the architecture presented in Figure 1 and defined as Algorithm 1.

At each of the k generations, the learning performance of $n = 50$ DDPG agents (in Figure 2 $b)$ component) were evaluated during a total 200 reinforcement learning episodes per trial. First, n genotypes represented by the helix shape in $c)$ section of Figure 2 were extracted from each of the n AMR phenotypes assigned as an $AMR-ANN$ in n DDPG trials. A genotype was defined as the weights and biases of the specific $AMR-ANN$. In d step of Figure 2 those genotypes were ranked based on the DDPG trial performance defined by the total cumulative reward $sumr_n$ received during the reinforcement learning. A subset of the $m = 20$ best performing genotypes were selected (in Figure 2 $e)$) in order to undergo the evolutionary processes of uniform crossover and mutation (Figure 2 $f)$) producing a new set of n genotypes (in Figure 2 $g)$) that would be used as a basis for n $AMR-ANN$ phenotypes implemented in the next generation of n DDPG trials. At time $k = 0$, the genotypes were generated randomly (Figure 2 $a)$) and mutation rate, implemented in Figure 2 $f)$, was 0.25, and included adding a random scalar between 0.1 and -0.1 to the weights.

V. Experimental Setup

A. Environment

The evaluation phase applied the proposed variations of the $DDPG$ learning algorithm to a variety of continuous control tasks running on an efficient and realistic physics simulator as a part of OpenAI Gym framework [51] and shown in Figure 3.

For the purpose of standardized benchmarking, all of the presented algortihms and their modifications were implemented using recently developed stable-baselines3 framework [52] with the corresponding optimized hyperparameters given by [53] for each of the learning environments. The standardization of the baseline frameworks allowed us to perform a benchmark the proposed *AMR* approach against state-of-the-art algorithms commonly used for tasks requiring a continuous action representation.

The considered environments range from a relatively simple 2D environments like *LunarLanderContinuous-v2*, with a humble 8-dimensional state space, to a complex four-legged 3D robot such as *AntBulletEnv-v0* [54], which boasts a total of 111-dimensional states coupled with 8 possible continuous actions. Different tasks of intermediate complexity like making

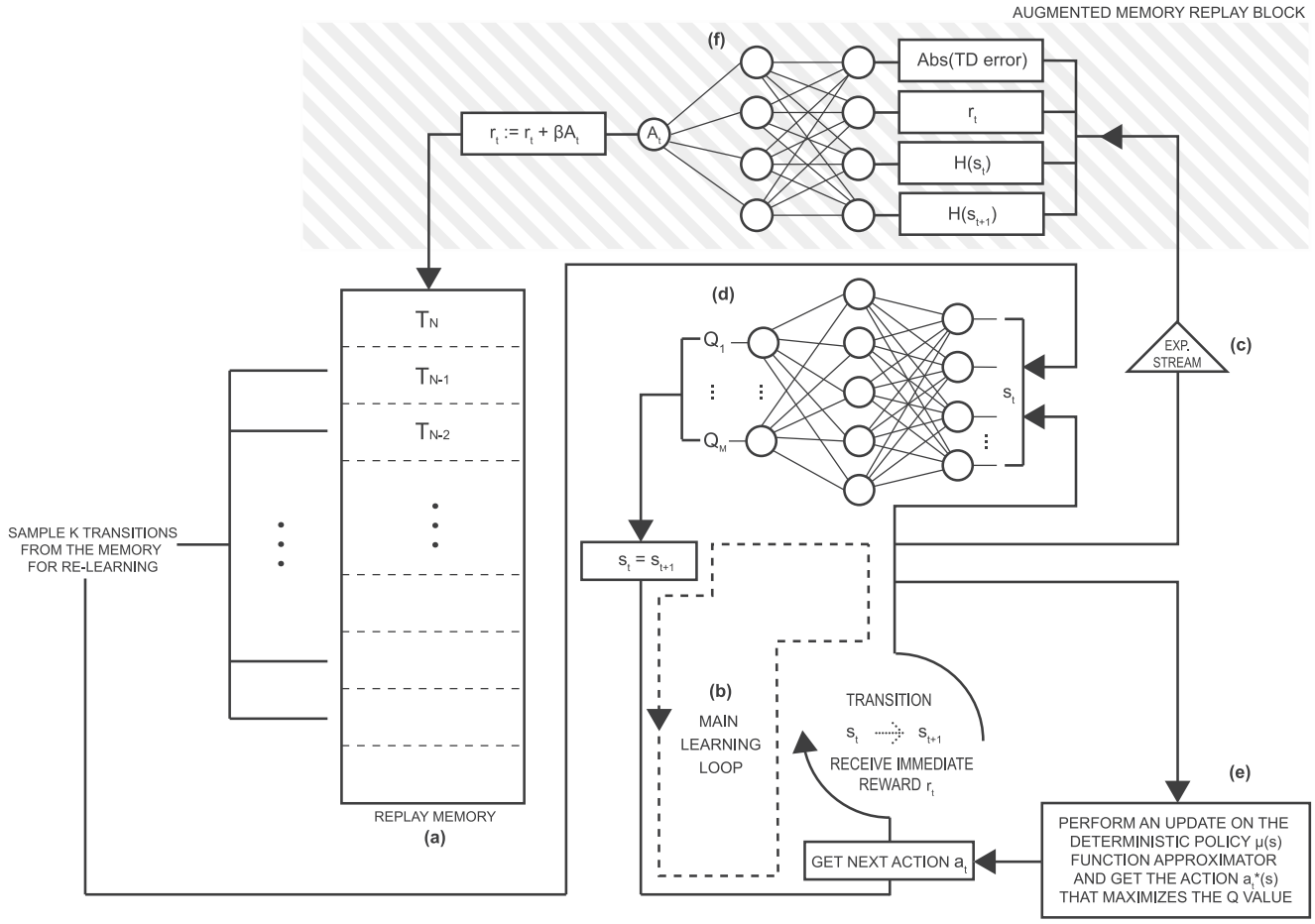


Fig. 1: Evaluation phase learning model architecture including the evolved *augmented memory replay block* or AMR: (a) Replay memory; Main learning loop (b) which performs the actual learning by updating the critic Q-function approximator ANN (d); Actor ANN approximator (e) providing a deterministic policy $\mu(s)$; Augmented Memory Replay block (f) consisting of an ANN that approximates the augmentation parameter A_t, based on the properties of the experience. The augmentation parameter A_t provides an augmentation policy which directly alters the raw experience stream of an agent (c) prior to its storage in the replay memory (a)

a 2D animal robot run in *HalfCheetahBulletEnv-v0* were evaluated.

B. Function Approximation

~~An approximation of $Q(s, a; \Theta) \approx Q^*(s, a)$ has been implemented using an ANN with one hidden fully connected layer of 50 neurons, able to take an agent's state as input and to produce as output the Q values of all the actions available to the agent. The learning rate of a *critic Q* approximator α is set to 0.002.~~

~~The *actor* function approximator of $a(s; \Theta) \approx a^*(s)$ is implemented using one hidden dense layer of 30 neurons which outputs a deterministic action policy based on the agent's current state. The *actor* ANN has been trained using a slightly higher learning rate of 0.001 compared to the critic one.~~

The architecture of the *AMR* function approximator consists of three layers: four input nodes connected to a fully connected hidden layer of four nodes, in turn connected to a single regression node able to produce an *augmentation scalar* as output. This ANN is able to approximate four parameters of the current agent's experience, respectively given in input as an absolute value of TD error, reinforcement, entropy of the starting state s_t, entropy of the transitioning state s_{t+1}, to a scalar value A_t that indicates how important the specific experience is to the learning algorithm. In the process of approximating the reward augmentation value A_t *AMR* neural network exploits the input information about the reinforcement reward that its augmenting together with an indicator of the transition's importance/potential for learning quantified by its respective TD-error.

Additionally *AMR* relies on methods for representing transition importance that go beyond TD-error by including in its input measures of transitioning states s_t, s_{t+1} energies quantified by their respective Shannon's entropy levels. Thus far, entropy of the observations has been used as an additional

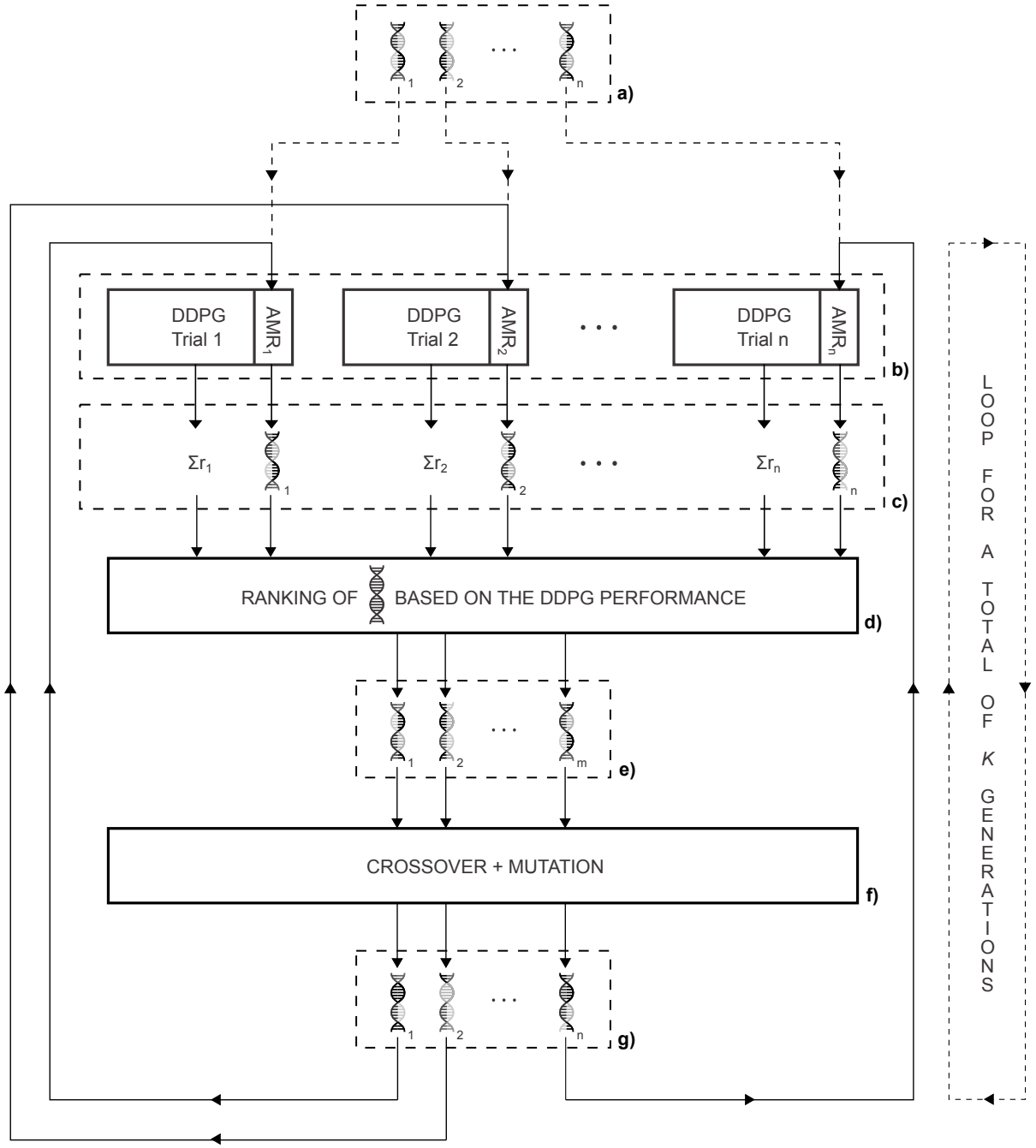


Fig. 2: Integrated architecture of *AMR* implementing both evaluation RL phase in component $b)$ and the evolution phase defined over k generations; $a)$ Random initialization of genotypes for $k = 0$; $b)$ Performing n independent instances of DDPG reinforcement learning defined by 1 and outlined in 1; $c)$ Extracting the genotypes represented by a double-helix shape from the n $AMR - ANN$ phenotypes used in n DDPG trials and their performance represented by the mean cumulative *primary* reinforcement $\sum r_n$ obtained from a total of 6 evaluation modes performed in between agents reinforcement learning steps as defined in subsection V-D; $d)$ Ranking the obtained genotypes based on the DDPG performance or $\sum r_n$ and selecting a subset of m genotypes in $e)$ that will undergo crossover and mutation in $f)$ in order to form a new set of genotypes in $g)$; The new set of genotypes $g)$ is then used to form a next population of DDPG trials forming a new generation. This process is repeated for a total of k generations.

Algorithm 1 Evaluation phase of DDPG with Augmented Memory

Initialize critic network $Q(s, a|\Theta^Q)$, actor network $\mu(s|\Theta^\mu)$ and augmentation network $A(s, r|\Theta^\beta)$ with random weights Θ^Q, Θ^μ and Θ^β
Initialize target network Q' and μ' with weights $\Theta^{Q'} \leftarrow \Theta^Q$ and $\Theta^{\mu'} \leftarrow \Theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process N for action exploration
  Observe initial state s_1
  for t=1, T **do**
    Select action $a_t = \mu(s_t|\Theta^\mu) + N_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and new state s_{t+1}
    Augment the reward $r_t \leftarrow r_t + A_t(s_t, s_{t+1}, r_t)$ according to the augmentation network parameters Θ^β
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of S transitions (s_t, a_t, r_t, s_{t+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\Theta^{\mu'})|\Theta^{Q'})$
    Update critic by minimizing the loss $L = \frac{1}{S}\sum_i (y_i - Q(s_i, a_i|\Theta^{Q'}))^2$
    Update the actor policy using the sampled policy gradient
    $\nabla_{\Theta^\mu} J \approx \frac{1}{S}\sum_i \nabla_a Q(s, a|\Theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\Theta^\mu}\mu(s|\Theta^\mu)|$
    Update the networks
    $\Theta^{Q'} \leftarrow \tau\Theta^Q + (1-\tau)\Theta^{Q'}$
    $\Theta^{\mu'} \leftarrow \tau\Theta^\mu + (1-\tau)\Theta^{\mu'}$
  end for
end for

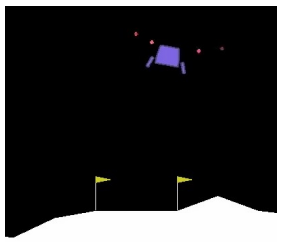
criterion in prioritized experience replay sampling in both single $H(s_t)$ form [55] and extended intrinsic-exploration one implemented in the approach of [56] and defined as a level of Kullback–Leibler divergence between the transitioning states $H(s_{t+1} \parallel s_t)$.

In the case of the *AMR* approximator a single form of state-energy criterion is provided directly by its $H(s_t)$ input while the latter is imposed implicitly through the inclusion of the $H(s_{t+1})$ as a separate input.

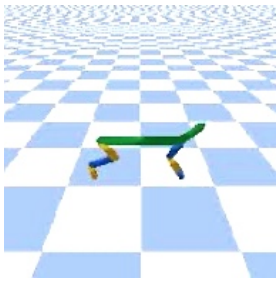
C. Learning Parameters

All of the hyperparameters related to RL algorithms were taken from environment-specific optimized values defined in [53]. ~~During the evaluation phase presented in Figure 1, at each learning step a batch of 32 experiences were replayed from the fixed capacity memory buffer of 10000.~~

~~Learning steps per episode were limited to a maximum of 1000 learning steps. Reward discount factor γ was set to a high 0.9 and soft replacement parameter τ was 0.01. In order to achieve action space exploration an artificially generated noise is added to the deterministic action policy which is approximated by the *actor* ANN. The noise is gradually decreased or adjusted linearly from an initial scalar value 3.0 to 0.0 towards the end of the learning.~~



(a) LunarLanderContinuous-v2



(b) HalfCheetahBulletEnv-v0

(c) AntBulletEnv-v0

Fig. 3: Variety of OpenAI Gym environments considered in evaluation. Ordered from low to high complexity.

D. Evaluation of the agents performance

The agent enters evaluation mode every 25000 learning steps. At each of the evaluation sequences the agent performs actual real-time steps for a total of 25 episodes in the simulated environment in order to evaluate the immediate primary reward reinforcement which, in this case is not mediated by the *AMR* process. The evaluation episodes are also characterized by a zero amount of *Gaussian* exploration-supporting action noise, which distinguishes them from exploration-greedy training episodes. A *single evaluation sequence score* is defined as a cumulative total *primary* reinforcement received over the course of its 25 evaluation episodes.

The reported final evaluation scores of agents, are formed by taking an unweighted arithmetic mean of a total of 6 *single evaluation sequence scores* performed in between the training episodes.

The agent evaluation returns are averaged after an agent reaches $6 \cdot 25000$ learning steps, at which point the agent concludes its independent, learning and evaluation trial detailed in Figure 1.

E. Experimental Results

The proposed integrated algorithm in Figure 2 evolved the *AMR's* neural network weights Θ^{AMR} trough a number of generations. The evolutionary impact on agent performance during the *evolution* phase has been characterized by the changes in approximate distributions of three variables (average cumulative return, standard deviation of that return and the average episode duration in frames) across the total of K generation populations. The evolution of population performance has been reported in Figure 4 $a)$ as the distribution graphs representing the average agent return (x axis) over K number of generations (y axis) for each of the considered environments: *AntBulletEnv-v0*, *HalfCheetahBulletEnv-v0* and *LunarLanderContinuous-v2*.[r2]

Figure 4 $a)$ also showcases how the different characteristics of the environments give rise to a different convergence dynamics of the evolutionary algorithm itself; We can notice that the *AntBulletEnv-v0* starts with a agent population yielding a rather disperse and uniform reward distribution at the initial generation 1 for $K = 0$. As the evolutionary algorithm converges, this initial dispersion is replaced by a strong central tendency visible in the intermediary generations $(7 - 9)$; The populations around this mark tend to include agent phenotypes that are consistent in their high reward yield and can be characterized by the absence of low performing outlier phenotypes. During the subsequent generations (> 9), the center of the trend is moved to a higher reward yield at the cost of a slight dispersion increase: the populations of the final generations tend to favor further exploration of the genotype space producing the best performing agent phenotypes yielding rewards well above 600, at a cost of including, to some extent, the low performing ones.[r2]

The effect of the AMR phenotype on the initial performance distribution proved to be quite different when applied to diverse environments; Contrary to the uniform nature of the *AntBulletEnv-v0* initial performance distribution, the first generations for both *HalfCheetahBulletEnv-v0* and *LunarLanderContinuous-v2* environments had a dominant central tendency towards producing low performing agents. The subsequent generations of *HalfCheetahBulletEnv-v0* and *LunarLanderContinuous-v2* show a more rapid tendency shift towards the higher return cluster. The differences in reported evolutionary dynamics show that the direct impact of the augmentation on the agents immediate performance induced by the AMR phenotype is higher for the *HalfCheetahBulletEnv-v0* and *LunarLanderContinuous-v2* environment types.

The $b)$ row of Figure 4 reports the evolutionary dynamics of the additional characterizing variable represented by the standard deviation of agent's returns. Although the diversity of the agent rewards, here represented by their STD, depends hugely on the reward function implementation for the specific environment, giving rise to different value ranges, the relative environment-specific changes in the STD variable distributions can be descriptive of the evolutionary process dynamics.

Figure 4 $b)$ shows that the evolutionary process resulted in a gradual increase of the variance of received rewards, supporting the exploration of the continuous reward space represented by the specific agent's reinforcement function.

The evolutionary disposition for exploratory behavior can also be seen along the changes in distributions of a variable representing an average episode length, measured in learning steps and outlined in the $c)$ section of Figure 4, as the preference for longer lasting episodes increments over generations. [r2]

F. Evaluation Results

Best performing evolved genotypes for each of the environments have been benchmarked against vanilla DDPG variant and other state-of-the-art RL algorithms such as *proximal policy optimization* [57] PPO and a variation of *asynchronous advantage actor-critic* [58] A2C. The evaluation benchmark was performed over a total of 1200 independent trials for each of the configurations of algorithm/environment. The distributions of the three characterizing variables, namely, average agent return, STD of the agent returns and average episode duration, are reported along the $a), b) and c)$ rows of Figure 5, respectively. The evaluation distributions outlined in Figure 5 are accompanied by the Tables I, II, and III displaying their mean, minimum, and maximum values of the score given by the corresponding average return variable. Along with the distributions visualizations in Figure 5, the summary in Tables I, II, and III provides an account of performance differences obtained by the different algorithms facing the evaluation benchmarks. [r2]

From the benchmark summary showcased along the Tables I, II, and III, we can notice that the most complex setup (i.e. *AntBulletEnv-v0*) improved its learning performance the most when using the proposed *AMR* approach when compared to the baseline vanilla DDPG approach, which did not use *memory augmentation*.

As evident from the mean scores reported in Table II the *AMR* evolutionary approach improves the *Ant's* quadlegged robot learning about walking by double compared to vanilla DDPG, significantly improves over PPO and provides a reasonable improvement over A2C algorithm. The augmented approach, also, improves the vanilla DDPG maximum reported score of a single agent by more than double. This high increase in agent's performance under augmentation is accompanied by the more extreme minimum scoring agent, also observable in Figure's 5 $a)$ distribution view and it is consistent with the *AMR's* evolved preference for exploration over exploitation. Under the exploration tendency, the augmented *AntBulletEnv-v0*) agents performed the longest learning episodes compared to their non-augmented counterparts as seen in the Figure's 5 $c)$ biased distributions.[r2]

AMR algorithm have also showed a significant improvement in *LunarLanderContinuous-v2* setup: the simplest of the benchmarked environments. During the benchmarks the proposed evolutionary approach made the 2D lunar craft improve its landing performance as it produced a significant increase in agent's total cumulative score. The augmented approach, while facing a far less demanding *LunarLanderContinuous-v2* environment outperformed all of the considered algorithms in

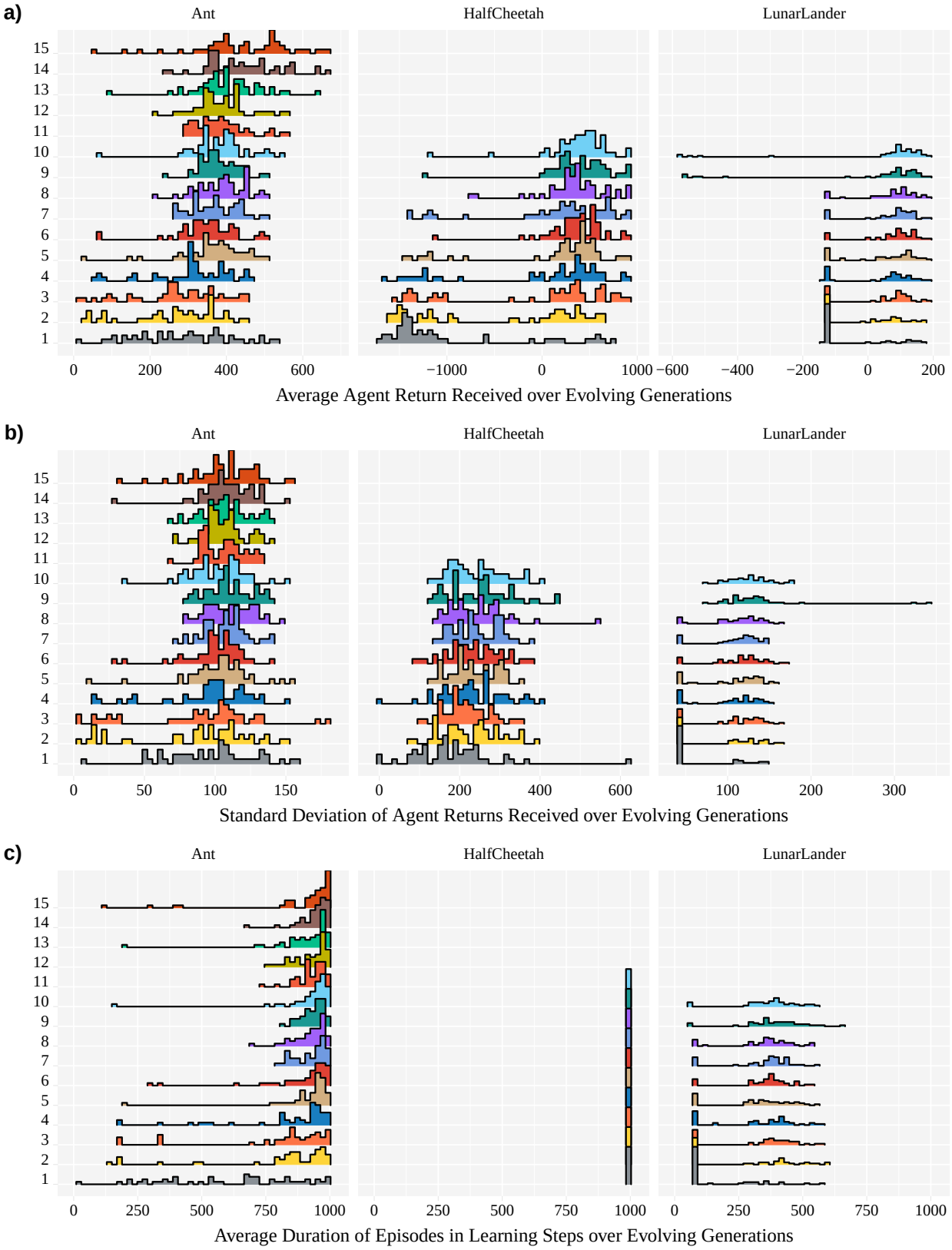


Fig. 4: Changes in the three characterizing variable distributions measured in agent populations evolved over the first K generations for each of the considered environments designated in columns (on the ordinates); a) Average return; b) Standard deviation of received returns and c) Average episode duration measured in frames.

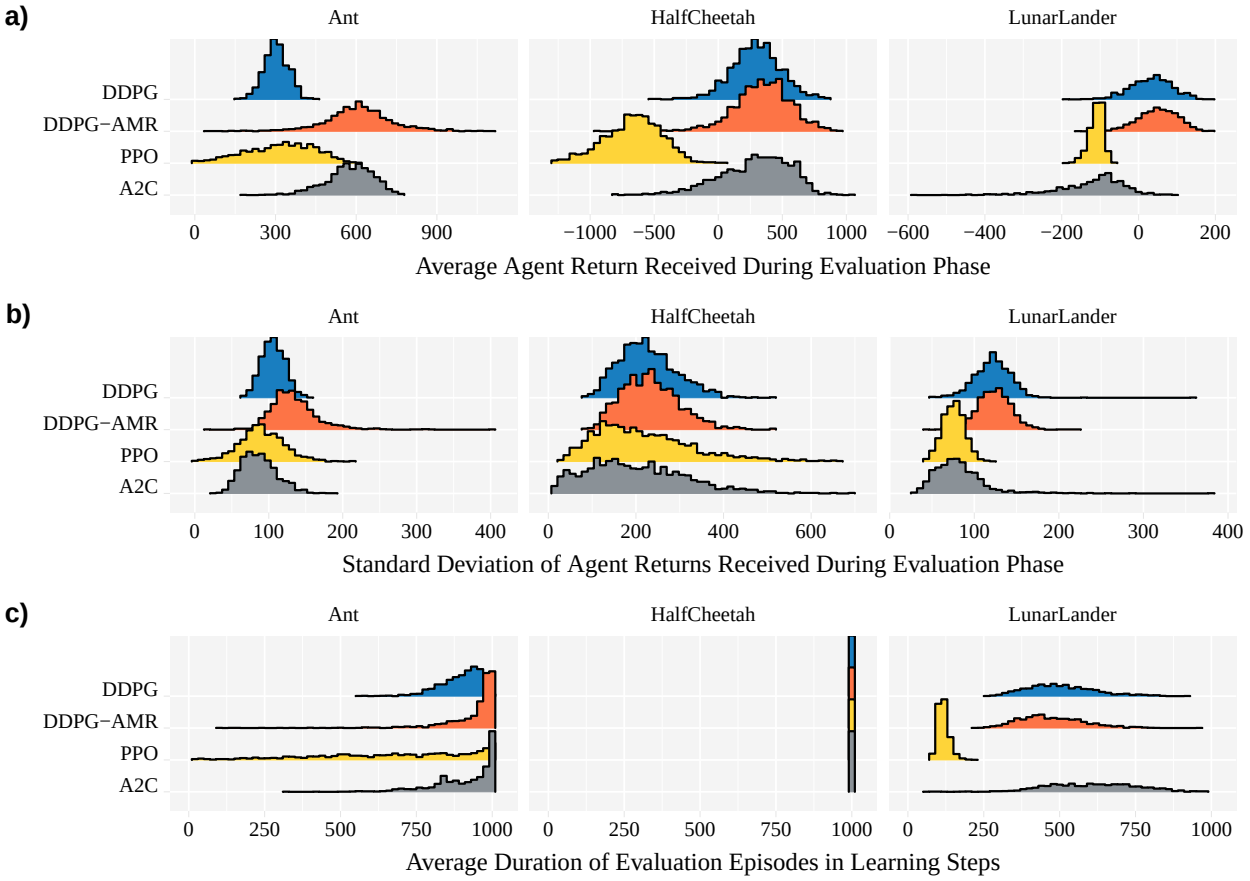


Fig. 5: Comparison of characterizing variables distributions obtained over a total of 1200 independent evaluation trials for each of the benchmarked algorithm/environment combination; a) Average return; b) Standard deviation of received returns and c) Average episode duration measured in frames.

both the mean and minimum score values while keeping the maximum score comparable with its vanilla DDPG baseline.

Although not as relevant as in *AntBulletEnv-v0* and *LunarLanderContinuous-v2* setups, the *AMR* approach was able to show some improvement over the baselines in the mean agent scores over the *HalfCheetahBulletEnv-v0* environment as summarised in Table III. While *HalfCheetahBulletEnv-v0* tends to move towards the direction of a relative increase of the reward diversity during the evolution stage as noticeable from Figure's 5 b) the evolutionary backed exploration tendency is not as prominent as in the case of *AntBulletEnv-v0* and *LunarLanderContinuous-v2* setups; The distinguishing characteristics of *HalfCheetahBulletEnv-v0* also create an environment dynamics that are unable to satisfy, in all trialed cases, its end-of-episode criterion for each of the performed episodes, up until the maximum episode length which is subject to a hard limit of 1000 steps. This characteristic, specific to *HalfCheetahBulletEnv-v0* accounts for the average episode duration variable to be constant across each of the evolved generations of agents in Figure 5 c), taking the value of the imposed hard limit for each of the episodes. ~~We can also notice the difference in score variance between the setups which can be attributed to distinctive robot/environment characteristics; while *LunarLanderContinuous-v2* shows relatively low variance in its scores, other settings like *AntBulletEnv-v0* and *HalfCheetahBulletEnv-v0* exhibit a very high variance.~~ r2r2

TABLE I: Summary of the evaluation benchmarks results reported in Figure 5 comparing agent performances of the proposed DDPG + AMR approach (using a best performing evolved *AMR* genotype) against vanilla DDPG and other state-of-the-art RL algorithms over *LunarLanderContinuous-v2* environment. The summary is based on the mean, minimum and maximum values of the performance measure distribution in a) section of Figure 5. Best performances are highlighted in bold.

RL Algo.	Mean Score	Min Score	Max Score
DDPG + AMR	**50.40**	**-160.60**	185.54
DDPG	33.07	-195.78	**192.99**
PPO	-109.97	-184.72	-63.90
A2C	-120.48	-581.19	94.51

TABLE II: Summary of the evaluation benchmarks results reported in Figure 5 comparing agent performances of the proposed DDPG + AMR approach (using a best performing evolved *AMR* genotype) against vanilla DDPG and other state-of-the-art RL algorithms over *AntBulletEnv-v0* environment. The summary is based on the mean, minimum and maximum values of the performance measure distribution in a) section of Figure 5. Best performances are highlighted in bold.

RL Algo.	Mean Score	Min Score	Max Score
DDPG + AMR	**610.02**	53.24	**1107.89**
DDPG	302.68	148.17	453.31
PPO	316.99	1.99	615.31
A2C	575.32	**185.36**	771.15

TABLE III: Summary of the evaluation benchmarks results reported in Figure 5 comparing agent performances of the proposed DDPG + AMR approach (using a best performing evolved *AMR* genotype) against vanilla DDPG and other state-of-the-art RL algorithms over *HalfCheetahBulletEnv-v0* environment. The summary is based on the mean, minimum and maximum values of the performance measure distribution in a) section of Figure 5. Best performances are highlighted in bold.

RL Algo.	Mean Score	Min Score	Max Score
DDPG + AMR	**345.52**	-970.91	964.55
DDPG	295.98	**-525.33**	841.86
PPO	-665.81	-1295.71	49.40
A2C	287.97	-807.71	**1025.74**

VI. Discussion

The presented approach implements a biologically inspired mechanism that enables artificial learning agents to better adapt to a specific environment by selectively modifying the relevance of the perceived memories used by the RL for re-learning. An agent implementing an *AMR* neural network is able to evolve in few generations its memory augmentation criteria in order to dynamically create a secondary *adaptive critic* reinforcer that is best adapted to the faced environment.

The secondary *AMR* reinforcer enables the agent to explore its action and state space more efficiently by augmenting the replayed memories that contain deterministic state-action choices that are predicted to have a greater impact on its learning performance during the training phase.

Although during its training episodes the agent is maximizing its expected return of a compound reinforcement signal represented by the sum of the primary reinforcement given by the environment and a secondary, augmenting reinforcement signal provided by *AMR*, it yields a direct improvement in performance during the evaluation episodes as well.

This relationship is possible because the secondary reinforcement signal is in fact a function of the primary external one. This creates an *adaptive* temporal relationship or feedback loop between the two reinforcers [20].

Augmenting memory with a meta-learned environment-specific augmentation policy enables the emergence of an *artificial cognition* as an intermediary dynamic filtering mechanism in learning agents, which, in all the benchmarked environments, have evolved a tendency for exploration as a reward maximization strategy. The strategy was consistent with respect to both complex hyper-realistic control problems modelled over three-dimensional physics simulator, and the game-like, highly idealized problems constrained to a two-dimensional plane.[r2]

VII. References

References

[1] G. A. Carpenter and S. Grossberg, "The art of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77–88, 1988.

[2] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[3] C. Chen and S. Tonegawa, "Molecular genetic analysis of synaptic plasticity, activity-dependent neural development, learning, and memory in the mammalian brain," *Annual review of neuroscience*, vol. 20, no. 1, pp. 157–184, 1997.

[4] P. J. Horner and F. H. Gage, "Regenerating the damaged central nervous system," *Nature*, vol. 407, no. 6807, pp. 963–970, 2000.

[5] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[6] P. Maquet, "The role of sleep in learning and memory," *science*, vol. 294, no. 5544, pp. 1048–1052, 2001.

[7] B. Rasch and J. Born, "Maintaining memories by reactivation," *Current opinion in neurobiology*, vol. 17, no. 6, pp. 698–703, 2007.

[8] R. Stickgold, "Sleep-dependent memory consolidation," *Nature*, vol. 437, no. 7063, pp. 1272–1278, 2005.

[9] S. Diekelmann and J. Born, "The memory function of sleep," *Nature Reviews Neuroscience*, vol. 11, no. 2, p. 114, 2010.

[10] D. Marr, D. Willshaw, and B. McNaughton, "Simple memory: a theory for archicortex," in *From the Retina to the Neocortex*. Springer, 1991, pp. 59–128.

[11] B. L. Mcnaughton, B. Leonard, and L. Chen, "Cortical-hippocampal interactions and cognitive mapping: A hypothesis based on reintegration of the parietal and inferotemporal pathways for visual processing," *Psychobiology*, vol. 17, no. 3, pp. 230–235, 1989.

[12] A. S. Gupta, M. A. van der Meer, D. S. Touretzky, and A. D. Redish, "Hippocampal replay is not a simple function of experience," *Neuron*, vol. 65, no. 5, pp. 695–705, 2010.

[13] A. A. Carey, Y. Tanaka, and M. A. van Der Meer, "Reward revaluation biases hippocampal replay content away from the preferred outcome," *Nature neuroscience*, pp. 1–10, 2019.

[14] R. A. Swanson, D. Levenstein, K. McClain, D. Tingley, and G. Buzsáki, "Variable specificity of memory trace reactivation during hippocampal sharp wave ripples," *Current Opinion in Behavioral Sciences*, vol. 32, pp. 126–135, 2020.

[15] N. Dumay, "Sleep not just protects memories against forgetting, it also makes them more accessible," *Cortex*, vol. 74, pp. 289–296, 2016.

[16] T. Schreiner and B. Rasch, "To gain or not to gain–the complex role of sleep for memory," *Cortex*, vol. 101, pp. 282–287, 2018.

[17] L.-J. Lin, "Reinforcement learning for robots using neural networks," DTIC Document, Tech. Rep., 1993.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[21] R. C. O'Reilly, "The leabra model of neural interactions and learning in the neocortex," Ph.D. dissertation, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1996.

[22] R. C. O'reilly and J. L. McClelland, "Hippocampal conjunctive encoding, storage, and recall: Avoiding a trade-off," *Hippocampus*, vol. 4, no. 6, pp. 661–682, 1994.

[23] R. C. O'reilly, K. A. Norman, and J. L. McClelland, "A hippocampal model of recognition memory," in *Advances in neural information processing systems*, 1998, pp. 73–79.

[24] R. C. O'reilly and Y. Munakata, *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. MIT press, 2000.

[25] J. W. Rudy and R. C. O'Reilly, "Conjunctive representations, the hippocampus, and contextual fear conditioning," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 1, no. 1, pp. 66–82, 2001.

[26] K. A. Norman and R. C. O'Reilly, "Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach." *Psychological review*, vol. 110, no. 4, p. 611, 2003.

[27] S. Káli and P. Dayan, "Off-line replay maintains declarative memories in a model of hippocampal-neocortical interactions," *Nature neuroscience*, vol. 7, no. 3, pp. 286–294, 2004.

[28] K. A. Norman, E. L. Newman, and A. J. Perotte, "Methods for reducing interference in the complementary learning systems model: oscillating inhibition and autonomous memory rehearsal," *Neural Networks*, vol. 18, no. 9, pp. 1212–1228, 2005.

[29] D. Kumaran, D. Hassabis, and J. L. McClelland, "What learning systems do intelligent agents need? complementary learning systems theory updated," *Trends in Cognitive Sciences*, vol. 20, no. 7, pp. 512–534, 2016.

[30] A. G. Barto, "Intrinsic motivation and reinforcement learning," in *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013, pp. 17–47.

[31] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[33] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.

[34] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half field offense: An environment for multiagent learning and ad hoc teamwork," in *AAMAS Adaptive Learning Agents (ALA) Workshop*, 2016.

[35] R. Houthooft, Y. Chen, P. Isola, B. Stadie, F. Wolski, O. J. Ho, and P. Abbeel, "Evolved policy gradients," in *Advances in Neural Information Processing Systems*, 2018, pp. 5400–5409.

[36] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[37] T. Erez, Y. Tassa, and E. Todorov, "Infinite horizon model predictive control for nonlinear periodic tasks," *Manuscript under review*, vol. 4, 2011.

[38] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[39] B. Dai, A. Shaw, N. He, L. Li, and L. Song, "Boosting the actor with dual critic," *arXiv preprint arXiv:1712.10282*, 2017.

[40] D. Pacella, M. Ponticorvo, O. Gigliotta, and O. Miglino, "Basic emotions and adaptation. a computational and evolutionary model," *PLoS one*, vol. 12, no. 11, p. e0187463, 2017.

[41] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically motivated reinforcement learning: An evolutionary perspective," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 70–82, 2010.

[42] M. Schembri, M. Mirolli, and G. Baldassarre, "Evolution and learning in an intrinsically motivated reinforcement learning robot," in *European Conference on Artificial Life*. Springer, 2007, pp. 294–303.

[43] M. Persiani, A. M. Franchi, and G. Gini, "A working memory model improves cognitive control in agents and robots," *Cognitive Systems Research*, vol. 51, pp. 1–13, 2018.

[44] M. Ramicic and A. Bonarini, "Selective perception as a mechanism to adapt agents to the environment: An evolutionary approach," *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

[45] M. G. Mattar and N. D. Daw, "Prioritized memory access explains planning and hippocampal replay," *Nature neuroscience*, vol. 21, no. 11, pp. 1609–1617, 2018.

[46] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[47] M. Khamassi and B. Girard, "Modeling awake hippocampal reactivations with model-based bidirectional search," *Biological Cybernetics*, pp. 1–18, 2020.

[48] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[49] D. Mareschal and S. Blakeman, "A complementary learning systems approach to temporal difference learning," *Neural Networks*, 2019.

[50] S. Blakeman and D. Mareschal, "A complementary learning systems approach to temporal difference learning," *Neural Networks*, vol. 122, pp. 218–230, 2020.

[51] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[52] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[53] A. Raffin, "Rl baselines3 zoo," https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[54] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[55] M. Ramicic and A. Bonarini, "Entropy-based prioritized sampling in deep q-learning," in *Image, Vision and Computing (ICIVC), 2017 2nd International Conference on*. IEEE, 2017, pp. 1068–1072.

[56] ——, "Towards learning agents with personality traits: Modeling openness to experience," *Cognitive Systems Research*, vol. 55, pp. 124–134, 2019.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[58] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.