



**HAL**  
open science

# An Intentional Forgetting-Driven Self-Healing Method For Deep Reinforcement Learning Systems

Ahmed Haj Yahmed, Rached Bouchoucha, Housseem Ben Braiek, Foutse  
Khomh

► **To cite this version:**

Ahmed Haj Yahmed, Rached Bouchoucha, Housseem Ben Braiek, Foutse Khomh. An Intentional Forgetting-Driven Self-Healing Method For Deep Reinforcement Learning Systems. 2023. hal-04255459

**HAL Id: hal-04255459**

**<https://hal.science/hal-04255459>**

Preprint submitted on 24 Oct 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# An Intentional Forgetting-Driven Self-Healing Method For Deep Reinforcement Learning Systems

Ahmed Haj Yahmed\*, Rached Bouchoucha\*, Housseem Ben Braiek, Foutse Khomh  
{ahmed.haj-yahmed,rached.bouchoucha,housseem.ben-braiek,foutse.khomh}@polymtl.ca  
SWAT Lab., Polytechnique Montréal, Canada

**Abstract**—Deep reinforcement learning (DRL) is increasingly applied in large-scale productions like Netflix and Facebook. As with most data-driven systems, DRL systems can exhibit undesirable behaviors due to environmental drifts, which often occur in constantly-changing production settings. Continual Learning (CL) is the inherent self-healing approach for adapting the DRL agent in response to the environment’s conditions shifts. However, successive shifts of considerable magnitude may cause the production environment to drift from its original state. Recent studies have shown that these environmental drifts tend to drive CL into long, or even unsuccessful, healing cycles, which arise from inefficiencies such as catastrophic forgetting, warm-starting failure, and slow convergence. In this paper, we propose *Dr. DRL*, an effective self-healing approach for DRL systems that integrates a novel mechanism of intentional forgetting into vanilla CL (i.e., standard CL) to overcome its main issues. *Dr. DRL* deliberately erases the DRL system’s minor behaviors to systematically prioritize the adaptation of the key problem-solving skills. Using well-established DRL algorithms, *Dr. DRL* is compared with vanilla CL on various drifted environments. *Dr. DRL* is able to reduce, on average, the healing time and fine-tuning episodes by, respectively, 18.74% and 17.72%. *Dr. DRL* successfully helps agents to adapt to 19.63% of drifted environments left unsolved by vanilla CL while maintaining and even enhancing by up to 45% the obtained rewards for drifted environments that are resolved by both approaches.

**Index Terms**—Deep Reinforcement Learning, Software Healing, Intentional Forgetting, Continual Learning.

## I. INTRODUCTION

Deep reinforcement learning (DRL), the blend of deep learning (DL) and reinforcement learning (RL), has shown promising achievements in recent years, such as AlphaStar [1], Netflix’s customized recommendation system [2], and Facebook ReAgent [3]. Nevertheless, DRL-based systems, as with all data-driven systems, are likely to generate incorrect or undesirable behaviors when operating in constantly ever-changing production settings. This phenomenon is known as environmental drift [4], which is the manifestation of concept drift [5, 6] in DRL systems. The burgeoning field of ML (and RL) operations (MLOps) (and RLOps [7]) is establishing the best practices and methods for implementing efficient, trustworthy, and maintainable self-learning systems. Among the most demanding approaches are healing strategies [8, 9, 10] that enable the ML system to recognize its malfunction after a concept shift and to adjust, accordingly, its behaviors in order to regain its performance. Conventional software healing

solutions are used to identify and respond to software failures in production by making the appropriate tweaks, at runtime, to restore the system’s normal behavior [11]. The healing workflow begins with a failure detection step. The healing step follows, which involves executing a healing operation. Finally, a verification step ensures that the software is restored properly. Otherwise, the healing operation is repeated until the failure is resolved or no further action is possible [11]. Self-healing solutions [8, 11] describe techniques that fully automate all steps of healing without any human involvement. During a self-healing operation, a software system should adjust its own structure and behavior in production based on its interpretation of the context, the occurred failure, and its requirements [8].

Indeed, a DRL system learns its optimal policy (i.e., best actions w.r.t perceived states) by interacting with its environment [12] and by performing reward-driven learning without supervision. Hence, it can be argued that the DRL agent can adapt its policy to handle a newly-shifted environment by engaging in continual learning (CL) cycles for a sufficient amount of time. In practice, various inefficiencies can adversely affect the CL ability of DRL agents, including catastrophic forgetting [13], failure of warm-starting [14], Primacy Bias [15], and Capacity Loss [16]. Thus, as the environment in production drifts from its original state, CL is likely to result in slow healing cycles [17, 18] associated with prolonged system dysfunction and downtime. Even worse, a recent study [19] reported that CL is confined within environmental parameters beyond which it is unable to heal the DRL agent. Therefore, it is of paramount importance that alternatives and improvements to vanilla CL (i.e., standard CL) be explored, in order to enlarge the adaptation frontiers and accelerate the adaptation curve for low-cost, fast, and effective self-healing DRL systems.

In this paper, we propose *Dr. DRL*, a self-healing approach for DRL that leverages intentional forgetting [20, 21] combined with continual learning in order to optimize the agent’s plasticity and accelerate its adaptation to drifted environments. Intentional forgetting is an interdisciplinary concept that has gained traction in the software engineering community, with applications in cybersecurity [22], machine learning [20, 23, 24], and logistics supply networks [21]. Its goal is to deliberately delete unnecessary knowledge in order to focus attention on the system’s key components. To apply the intentional forgetting mechanism in DRL self-healing, we adapt the top-k neuron coverage (TKNC) [25, 26], which is a fine-grained

\*Both authors contributed equally to this research.

layer coverage criterion proposed by DeepGauge [25], in order to identify the major and minor DRL agent behaviors at the level of its neural network units (i.e., neurons). Accordingly, our approach associates the DRL agent’s major behavior with hyperactive neurons (i.e., neurons with high levels of activation) and minor behavior with hypoactive neurons (i.e., neurons with low activation levels). Then, our approach intentionally erases the minor behaviors by assigning low-scaled random weights to their associated hypoactive neurons. As a first benefit, forgetting the minor behaviors that are likely to be non-transferable to the drifted environments, increases both the speed and the effectiveness of major behavior adaptation. As a second gain, under-scaling weight initialization as a minor behavior forgetting mechanism enables the self-healing at dual speeds, i.e., major behavior neurons (unchanged) would receive significant and more frequent updates than minor behavior neurons (under-scaled).

To demonstrate the effectiveness of *Dr. DRL*, we evaluate it on purposefully drifted gym [27] environments with different drifting intensities. Results show that *Dr. DRL* succeeds in speeding up the self-healing process in terms of runtime and fine-tuning episodes, compared to vanilla CL. *Dr. DRL* extends the self-healing frontiers that CL provides by 20% on average. Finally, *Dr. DRL* is able to maintain and, in some cases, to increase the average reward of CL by up to 45%.

Below, we summarize the contributions of this paper:

- We propose the first application of the intentional forgetting concept for the DRL system’s self-healing problem.
- *Dr. DRL* exploits the intricacies of DNN structure to separate the DNN’s computation units responsible for major and minor behaviors as seen in the interaction with the original environment.
- *Dr. DRL* leverages the gradient-based optimization principles to smoothly achieve dual-speed self-healing, in which major behaviors are prioritized over minor behaviors.
- We conduct the evaluation of *Dr. DRL* using (i) three popular gym environments that we deliberately drift with varying parameters shifts, namely CartPole [28], MountainCar [29], and Acrobot [30]; (ii) three well-established DRL algorithms, namely Deep Q-Learning (DQN) [31], Soft Actor-Critic (SAC) [32], and Proximal Policy Optimization (PPO) [33].
- In order to stimulate more studies on this topic and promote the undertaking of replication studies, we provide our replication package [34].

The remainder of this paper is structured as follows. Section II introduces the fundamental concepts that will be used throughout our work. Section III provides a concrete illustration of the problem we are addressing and how we are tackling it. Section IV describes each step of our approach. Section V reports the evaluation outcomes. Section VI analyzes the threats to validity. Section VII presents the related works. Finally, Section VIII concludes the paper.

## II. PRELIMINARIES

This section briefly introduces DRL, CL, environmental drifts, and self-healing with intentional forgetting.

### A. Deep Reinforcement Learning

Reinforcement Learning (RL) [12] is a mathematical framework for experience-driven autonomous learning. RL aims to create fully autonomous agents that interact with their environments to learn optimum behaviors and improve over time, through a trial-and-error paradigm [12]. Formally, the agent is given a state  $s$  at each time-step  $t$  of its interaction with the environment, and it decides which action to take based on that state [35]. Those actions are determined by its on-learning policy,  $\pi$ , encoded as state-to-action mappings. Once an action is taken, the agent receives a reward and the environment moves to the next state.

Deep reinforcement learning (DRL), leverages DL to scale RL to complex, high-dimensional state and action spaces [35, 36]. Neural Networks (NN) are non-linear function approximators that are trained to estimate quantities that depend on state and/or actions, e.g., a policy  $\pi$  or an action-value function  $Q$ . DRL algorithms are either model-free or model-based [36]. To avoid modeling biases and solve environments with complex dynamics, we decided to focus our study on model-free DRL algorithms. These algorithms can be divided into three categories [36]. First, **Policy gradients algorithms** represent the policy with a NN whose weights are updated by maximizing the expected return. This optimization is performed on-policy, meaning that each update is carried out with data coming exclusively from the actual policy. **Value-based algorithms** represent the action-value function,  $Q$ , with a NN whose weights are updated based on the Bellman equation [37]. This optimization is performed off-policy, meaning that it leverages data coming from any policy, not only from the actual policy. Finally, **Hybrid algorithms** use both policy gradients and value-based methods. One notable class of algorithms from this category is the Actor-Critic algorithm which has two networks: Actor and Critic. The actor decides which action should be taken and critics inform the actor how good the action was and how it should adjust.

### B. Environment Drift and Continual Learning

Standard RL environment settings are often not representative of all possible contexts that agents will encounter for task fulfillment [38, 39]. The behavior of these environments is subject to changes, which can manifest as alterations in real deployment environments or shifts in the parameters of virtual environments [4, 40]. These challenges, known as partial observability and non-stationarity [38] in DRL literature, hinder productionizing DRL to real-world problems [38, 39]. For instance, recommendation systems have no observations of the ever-changing mental state of the users and need to always heal their behavior accordingly. These changes in the production RL environment, called environment drift, may affect the performance of trained agents. The environment’s behavioral changes become classified as environmental drifts when their

severity/intensity is substantial, preventing the trained agent from achieving its task. Although a drifted environment yields different rewards for a subset of state-action pairs, it should not be considered a completely new environment. In fact, the original environment and its drifted counterpart share some regions of state-action space. Therefore, the agent in a drift environment only requires healing by fusing new information with existing knowledge (i.e., fine-tuning) rather than learning from scratch. This required fine-tuning is called the continual learning (CL) problem in DL [41], which investigates whether NNs can acquire new knowledge incrementally. In spite of its success in several use cases, it has been shown that new knowledge may interfere with the existing one, which results in replacing it entirely. This failure of stability in CL is called catastrophic forgetting [13], in which new experience overwrites previous experience. Thus, the fundamental objective is to design an advanced self-healing approach to improve the adaptability of DRL systems effectively and efficiently.

### III. MOTIVATING EXAMPLE

Figure 1 shows the MountainCar environment [29], where a car starts on a one-dimensional track situated between two mountains. The objective is to reach the goal position up the right mountain. The car starts at the bottom of the valley, and receives negative rewards, as each timestep passes, to encourage it to reach the goal as soon as possible. MountainCar’s environment is episodic, where episodes end when 200 timesteps have passed or when the car reaches the flag. In order to solve the MountainCar, the agent must achieve an average reward of at least  $-110$  over 100 consecutive evaluation episodes. After drift, we assign 20% of tolerance in the average reward. So the agent must achieve at least  $-132$ .

Drifts in the MountainCar environment can affect three parameters, namely, force, gravity, and goal velocity. Originally, the initial settings of a MountainCar environment,  $E$ , are set as follows: force= $1e^{-3}$ , gravity= $2.5e^{-3}$ , and goal\_velocity= $0$ . A drift  $d$  on  $E$ , for instance, can shift these parameters to force= $1.2e^{-3}$ , gravity= $4e^{-3}$ , and goal\_velocity= $0$ , leaving the agent unable to solve the drifted environment  $E_d$ . CL would be used to fine-tune the trained agent on  $E_d$  for a set number of episodes. Nevertheless, CL is prone to inefficiencies such as catastrophic forgetting [13] and slow adaptation, resulting in inferior DRL system performance and higher repair costs. A trained DQN agent, for instance, took 236 training episodes and 436 seconds to adapt to  $E_d$  leveraging CL (Figure 1). Our proposed approach, *Dr. DRL*, optimizes cost-effectiveness, thereby reducing adaptation expenses. Considering the same  $E_d$ , *Dr. DRL* adapted the same DQN agent in 185 episodes and 273 seconds. The DQN agent also achieved an average reward of  $-122.3$  when adapted with *Dr. DRL*, compared to  $-123.7$  with CL. Finally, as evidenced by Figure 1, *Dr. DRL* improved the stability and monotonicity of the adaptation process by minimizing the fluctuation breadth of the average reward.

## IV. APPROACH

In this section, we first describe the problem statement. Then, we introduce our intentional forgetting mechanism for an improved DRL healing. Last, we detail the different phases and steps involved in *Dr. DRL*’s workflow.

### A. Problem formulation

Let  $E$  be a development RL environment, defined as a set of  $n$  parameters  $\{p_1, \dots, p_n\}$ , and let  $A_E$  be an agent trained on  $E$  until it solves  $E$  or reaches the desired performance level. Next, let  $E'$  represent a new RL environment with parameters  $\{p'_1, \dots, p'_n\}$ , denoting the production RL environment. Although the  $A_E$  is trained on  $E$ , the production RL environment represents the same problem as  $E$  and the  $A_E$  is expected to be able to solve  $E'$  or maintain its performance level as long as the parameters of  $E'$  have not drifted substantially from  $E$ . If an important environmental drift occurs, the performance of  $A_E$  decreases proportionally to the deviations, i.e., the greater the drift, the lower the performance. In this case, the RL agent,  $A_E$ , requires a self-healing mechanism to adapt to the drifted production environment,  $E'$ , and the healing success depends on its ability to solve  $E'$  or regain its optimal level of performance. Conventionally, Continual Learning (CL) is the mainstream approach for self-healing against environmental drifts. Nevertheless, this approach can be inefficient due to several issues the community encounters, such as catastrophic forgetting [13] and slow adaptation [17]. In our approach, we assume that a trained agent  $A_E$  in the environment  $E$  learns both major and minor behaviors, where major behaviors encode critical problem-solving abilities and minor behaviors have no direct impact on problem-solving, but were necessary to achieve the desired performance on the development environment. Structurally, the RL agent encapsulates a neural network, thus, we propose a neuronal categorization method to depict the hyperactive neurons (i.e., contribute almost to all of the predictions) and the hypoactive neurons (i.e., contribute intermittently to the predictions). The proposed neuronal categorization provides a structural representation of the problem-solving skills acquired by the optimal RL agent as a result of interaction with the development environment. Indeed, we believe that the inefficiency of CL may be partially due to the equal importance assigned to both major and minor behaviors, resulting in slow adaptation (all neurons are considered to be adapted) and catastrophic forgetting (CL may reverse the neurons’ roles from major to minor, or vice versa). Our proposed self-healing mechanism is an improvement over vanilla CL by (i) prioritizing adjusting major behaviors to maintain the structural representation of problem-solving skills, and (ii) accelerating the behavioral adaptation using dual-speed continuous learning, in which minor behaviors are updated less frequently than major behaviors. These improvements can enhance both the speed and success of self-healing against environmental drifts, which remain unsolved by vanilla CL. In the following, we detail the main components of our approach, along with the self-healing workflow that we have adopted.

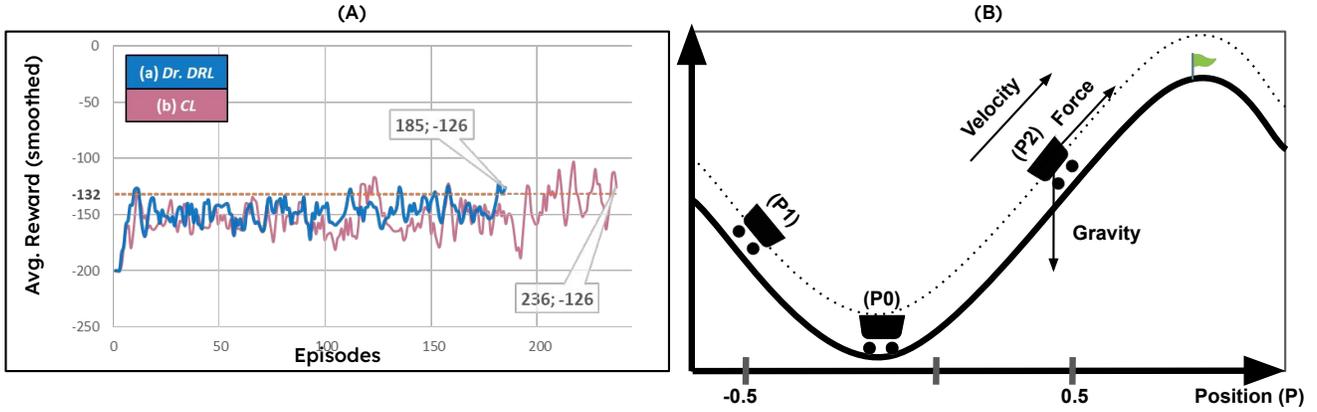


Fig. 1. (A) The adaptation curves for the DQN agent’s average reward per episode on MountainCar: (a) depicts the curve of the average reward when Dr. DRL is used, whereas (b) represents the curve of average reward when CL is used (B) Illustration of the MountainCar environment as described in Moore’s Moore’s work [29]

### B. Intentional Forgetting Mechanism

The phenomenon of winning lottery tickets has been shown [42, 43] to exist in deep neural networks, i.e., there are sub-networks that can be further fine-tuned to achieve as good performance as their source neural network. An example of its implications is the growing trend of reducing the size and resource consumption of DNNs by pruning irrelevant neurons after training [44]. In line with the lottery ticket hypothesis, the major DRL agent behaviors are likely to be yielded by a sub-network within the agent’s DNN. This sub-network encodes the critical problem-solving abilities responsible for achieving the major task. More precisely, this sub-network encapsulates the set of hyperactive neurons (i.e., major-behavior-responsible units) that are expected to represent the essential agent skills to resolve most of the occurring environment states and predict the optimal action. We hypothesize that the DRL agent’s major behaviors (i.e., set of hyperactive neural network units) should be primarily adjusted by the DRL self-healing process against a drifted environment, while the minor behaviors are more likely to be non-transferable and should be learned again. Indeed, minor behaviors are likely to have no direct impact on problem-solving but are necessary to achieve the desired performance in a specific environment. Therefore, we adopt the concept of intentional forgetting [20, 24] that removes irrelevant knowledge to improve the cognitive capabilities of intelligent systems by allowing them to focus solely on pertinent aspects of the given situation. In the following, we describe the two main steps of our intentional forgetting mechanism for the enhancement of the DRL agent’s healing ability and its associated continual learning stability.

1) *Detection of Minor Behavior Regions:* The first step of our intentional forgetting mechanism is to localize the regions of the DRL system responsible for the major behavior (i.e., contributing almost to all of the predictions) and minor

behavior (i.e., contributing intermittently to the predictions). Structurally, the RL agent encapsulates a neural network, thus, we propose a neuronal categorization method to depict the hyperactive neurons (i.e., major-behavior-responsible units) and the hypoactive neurons (i.e., minor-behavior-responsible units). This neuronal categorization method provides a structural representation of the problem-solving skills acquired by the RL agent.

A neural network’s activations represent the intermediate computations that pass essential information between layers. The activations are computed using a nonlinear function attached to each neuron, which determines whether it should be activated (“fired”) or not, based on the relevance of neuron output for the final prediction. As illustrated in Equation 1 of the forward pass [45], all the activations contribute to the last layer activation,  $a^{[L]}$ .

$$a^{[L]} = \sigma \left( W^{[L]} g \left( \dots g \left( W^{[1]} a^{[0]} + b^{[1]} \right) \dots \right) + b^{[L]} \right) \quad (1)$$

Where  $W = \{W^{[l]}, \forall l \in [1, L]\}$  and  $b = \{b^{[l]}, \forall l \in [1, L]\}$  are the weights and biases of a DNN.

Hence, the neurons with higher activation levels, called hyperactive neurons, encode the representation of relevant features associated with major behaviors at that processing layer and thus play a significant role in the overall predictions. In contrast, hypoactive neurons, characterized by lower activation levels, encode minor behavior features, but they still contribute to the model’s generalizability and confidence. Hyperactive neurons (i.e. neurons with high levels of activation) have been found to provide beneficial learning patterns for DNNs in several empirical studies[25, 26]. The concept of identifying hyper-hypoactive neurons was inspired by the Top-K-Neuron-Coverage (TKNC) technique [25]. The TKNC was developed to assess the diversity of activation patterns during DNN testing. When computing, TKNC, neurons with higher activation levels are deemed to be more significant in processing the input and are ranked higher,

whereas neurons with lower activation are ranked lower. Therefore in *Dr. DRL*, to categorize the hypoactive and hyperactive neurons, we select the  $N_H$  hypoactive neurons per layer, i.e., those triggering low activation scores on a regular basis. According to the following formula,  $N_H$  is a relative number derived from a prefixed forget rate,  $F_r$ :  $N_H^l = \text{Round}((F_r/100) * N_l)$ , where  $N_H^l$  represents the number of hypoactive neurons in layer  $l$  and  $N_l$  is the number of neurons in layer  $l$ .

2) *Forgetting & Dual Speed Gradient-based Healing*: After detecting the major and minor behaviors of the DRL system, *Dr. DRL* intentionally erases its minor behaviors and proceeds to dual-speed gradient-based healing. We assume that the inefficiency of CL may be partially due to the equal importance assigned to both major and minor behaviors, resulting in slow adaptation (all neurons are equally updated) and catastrophic forgetting (CL may reverse the neurons’ roles from major to minor, or vice versa). Hence, our approach offers two improvements to CL. As a first benefit, forgetting the minor behaviors that are likely to be non-transferable to the drifted environments, is expected to increase both the speed and effectiveness of major behavior adaptation. Second, the minor behavior forgetting mechanism enables self-healing at dual speeds, i.e., major behavior units would receive significant and more frequent updates than minor behavior units. Nevertheless, nullifying them would negatively affect the layer’s weight asymmetry, and the loss gradients w.r.t null weights would become zero as well; so their corresponding neurons freeze and no longer receive updates. Hence, our proposed method to properly forget the minor behaviors involves the re-initialization of their associated neurons using the original weight initializer. Indeed, random weights are sampled to maintain the asymmetry between neurons and calibrate the values distribution variance proportional to the size of the layer’s input. Furthermore, we under-scale these random weights generated for hypoactive neurons using a scale rate,  $S_r < 1$ .

$$\frac{\partial \mathcal{L}}{\partial z^{[1]}} = [W^{[2]}]^T \dots [W^{[L]}]^T \cdot \frac{\partial \mathcal{L}}{\partial z^{[L]}} * q^{[L-1]} * \dots * q^{[1]} \quad (2)$$

$$W^{(i+1)} = W^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial W^{[l]}} \quad ; \quad b^{(i+1)} = b^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial b^{[l]}} \quad (3)$$

Equations 2 and 3 show how the parameters of the model are updated at the iteration  $i + 1$ , where  $\mathcal{L}$  is the loss function;  $z^{[l]}$ ,  $W^{[l]}$ ,  $b^{[l]}$  are respectively the weighted-sum outputs, weights and biases of a layer  $l$  and  $q^{[l]} = g^{[l]'}(z^{[l]})$  with  $g^{[l]}'$  referring to the computed gradients in the layer  $l$ .

Equation 2 shows that the gradient-based updates include multiplication by weights, which makes their initial magnitude scale affect their growth and decay over iterations, thus controlling their learning speed. If  $S_r$  tends to 0, new weights would be very small, almost zeros, so their associated neurons would experience low update rates. If  $S_r$  tends to 1, minor behavior neurons will have original scale weights and learn at the same rate as major behavior neurons.

Hence, the reduction of weights magnitude affects directly the scale of gradient updates derived by backpropagation through layers. Lowering the magnitude of the reinitialized neurons’ weights ensures the slow re-learning of the DRL agent’s minor behavior, whereas maintaining the original magnitude of the retained neurons’ weights guarantees substantial updates of the RL agent’s major behaviors for stable and fast adaptation. Therefore, our proposed intentional forgetting mechanism leverages the weight-gradient scaling relationship to enable dual-speed gradient-based healing of the DRL agent’s behaviors.

### C. Self-healing Workflow

The overview of our proposed self-healing workflow is illustrated in Figure 2. Below are details of the workflow’s steps.

**Pre-deployment minor behavior identification.** *Dr. DRL* requires a pre-deployment step in which it collects the activation traces of the agent’s DNN at the end of development. During the training process, the agent should learn how to map between states and actions properly to maximize the accumulated rewards received in the original environment [46]. Once an optimal DRL agent has been identified, we run it on samples of observations taken from the original training episodes and save the activation traces.

**Failure detection.** In *Dr. DRL*, we develop an event handler component that monitors the production environment and triggers the healing mechanism when an environmental drift occurs. Upon a certain level of parameter shifts, the trained agent loses its ability to predict optimal actions, which means the agent cannot reach the required minimum reward. In that case, the actual agent fails to solve the newly-drifted environment, and consequently, our proposed self-healing mechanism is triggered.

**Self-Healing mechanism.** *Dr. DRL* first identifies the trained agent’s minor behavior regions (i.e., hypoactive neurons) using the collected activation traces. In each layer, we rank the neurons based on their activations, and then we select the hypoactive neurons, i.e., triggering lower activations over most of the observations (lines 2 to 6). The hypoactive neurons per layer constitute the minor behavior neurons, whose counts are derived according to the prefixed  $F_r$  (See Section IV-B1). *Dr. DRL* then intentionally erases the DRL agent’s minor behaviors and proceeds to dual-speed gradient-based healing. Algorithm 1 illustrates the steps of *Dr. DRL*’s healing mechanism. First, we reinitialize the weights associated with minor behavior neurons using the same weight initializer but under-scaled by  $S_r$  in order to prioritize the updates of major over minor behavior neurons (lines 7 to 11) (See Section IV-B2). Second, we adjust the hyperparameters of the DRL algorithm, especially those that control how much exploration the DRL agent does over the training. We reload the replay memory obtained from previous training episodes on the original environment in order to preserve the initial performance (only for off-policy RL algorithms like DQN and SAC). As

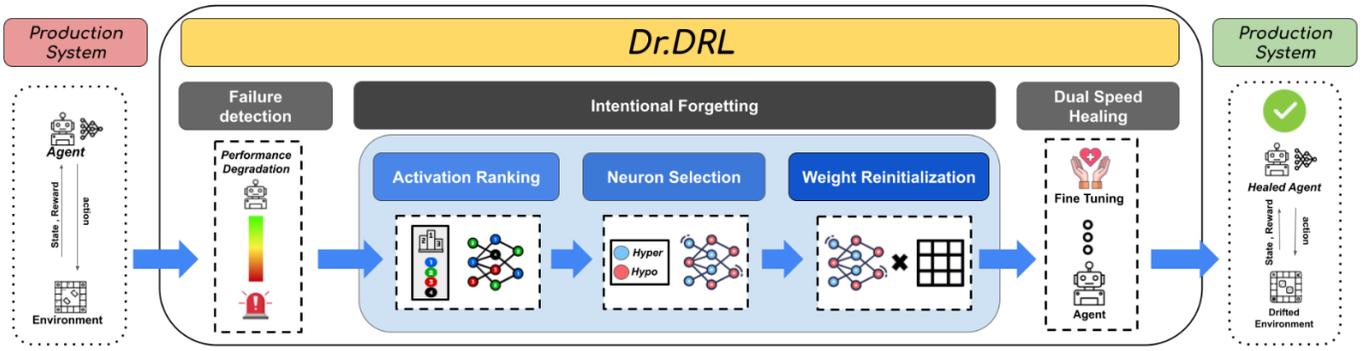


Fig. 2. Illustration of *Dr. DRL* Pipeline

soon as the setup is complete, we start fine-tuning the partially-reinitialized DRL agent on the drifted environment (lines 12 to 14) (See Section IV-B2). The objective is to heal the agent from the involved environmental drifts, i.e., achieve the same level of performance as the original environment.

---

#### Algorithm 1 *Dr. DRL's* healing mechanism

---

**Input:**  $E' = \{p'_1, \dots, p'_n\}$  drifted environment,  $A_E$ : trained agent that solves  $E = \{p_1, \dots, p_n\}$ ,  $n_t$ : training episodes,  $obs$ : set of training observations,  $F_r$ : Forget rate,  $s_r$ : Scale rate,  $init$ : weights initializer,  $hyp$ : hyperparameters of the agent.

**Output:**  $A'_{E'}$ : agent adapted to the drifted environment  $E'$

```

1:  $mask := []$ ;
   /* Detection of Minor Behavior : /*
2:  $acts := compute\_activation\_trace(A_E, obs)$ 
3: for  $i$  in  $A_E.hidden\_layers$  do
4:    $hypo\_n := detect\_minor\_regions(A_E, obs, F_r)$ ;
5:    $masks.append(generate\_mask(A_E, i, hyp, hypo\_n))$ ;
6: end for
   /* Intentional Forgetting of Minor Behavior : /*
7: for  $i$  in  $A_E.hidden\_layers$  do
8:    $w := apply\_mask(weights\_mask, A_E)$ ;
9:    $new\_w := reinitialize(w, masks[i], s_r, init)$ ;
10:   $A_E.set\_weights(new\_w)$ 
11: end for
   /* Dual Speed Gradient-based Healing : /*
12: for  $eps$  in  $n_t$  do
13:   $A'_{E'} := dual\_speed\_cl(A_E, E', hyp)$ ;
14: end for
15: return  $A'_{E'}$ 

```

---

## V. EVALUATION

In this section, we first introduce the DRL case studies (environments and algorithms), as well as our experimental settings, metrics, and procedure. Next, we evaluate *Dr. DRL* against vanilla CL in terms of cost reduction (RQ1), adaptability enhancement (RQ2), and reward improvement (RQ3).

### A. Experimental Setup

1) *Environments*: We leverage the following 3 different environments in our empirical evaluation.

**The MountainCar environment** [29] was described in Section III.

**The CartPole environment** [28] consists of an inverted pendulum attached to a cart that moves on a track controlled by a force. The agent's goal is to keep the pole upright, receiving a reward of +1 for each successful time step. The task is episodic, and the episode ends if the pole falls  $> 15^\circ$ , the cart moves  $> 2.4$  units from the center, or 200 time steps pass. To solve the task, an agent must achieve an average reward  $> 195$  over 100 testing episodes. The CartPole environment is parameterized by four parameters: masspole, length pole, mass cart, and friction.

**The Acrobot environment** [30, 47] has two joints and two links, and the goal is to raise the lower link to a specified height. If the agent achieves the goal, it gets a reward of 0, otherwise, it receives a reward of  $-1$  at each time step. Episodes end when the goal or 500 time steps are reached, and the reward threshold for the environment is  $-100$  over 100 evaluation episodes. The environment parameters are 'link length 1', 'link compos 1', 'link mass 1', and 'link mass 2'.

2) *DRL Algorithms*: In the following, we present the selected DRL algorithms for our evaluation. They are widely used and most RL libraries provide a stable implementation of them.

**Deep Q-Learning (DQN)** [31, 48] is a popular value-based algorithm that leverages neural network to approximate the action-value function,  $Q$ .

**Proximal Policy Optimization (PPO)** [33] is a state-of-the-art policy-based algorithm whose goal is to maximize policy optimization without compromising performance.

**Soft Actor Critic (SAC)** [32] is a hybrid DRL algorithm. It focuses on maximizing returns and policy entropy (i.e., degree of its "randomness"), simultaneously.

3) *Software and Hardware Configuration*: We implemented our approach as an open-source tool using Python 3.7 [49] and it supports Tensorflow (version 2.4.4) [50]. We evaluated environments from the Gym library (version 0.23.1) [27] and

DRL algorithms adapted from popular GitHub repositories [51, 52]. We used two GPU-enabled servers. The first server runs Ubuntu 20.04 and features a 24-core AMD EPYC 7413 CPU, an ASPEED AST2500 GPU, and 500 GB of RAM. The second runs CentOS 7 and has a 12-core 1.70 GHz Intel Xeon Bronze CPU, an NVIDIA GeForce RTX 2080 Ti GPU.

#### 4) Evaluation Procedure and Metrics:

**Procedure:** The DRL algorithms’ hyperparameters were tuned using the open-source repository rl-baselines-zoo [53]. As minor implementation changes may substantially affect a DRL algorithm performance [54], we conducted ten distinct training runs with various random seeds. The final hyperparameters for each DRL algorithm were chosen based on the highest average reward. For each DRL algorithm, we sampled six realistic environmental drift settings. To simulate different shifts in production, we intentionally induce drifts of varying magnitudes. For distinguishing between intensities of simulated drifts, we rely on the environment’s parameter ranges described in [19]. Then, we use either our approach, *Dr: DRL*, or vanilla CL to heal the DRL agent to the six newly-drifted environments. To have a fair comparison, we used the same hyperparameters and max number of episodes of the DRL agent beforehand. Nevertheless, we tuned the specific parameters,  $F_r$  and  $S_r$ , of *Dr: DRL* for each environment using, respectively, the ranges of [50, 40, 30, 20, 10]% and [0.00001, 0.0001, 0.001, 0.01, 0.1]. We found that the best parameters are  $F_r=50\%$  and  $S_r=0.1$  for the CartPole environment. For both Acrobot and MountainCar environments,  $F_r=10\%$  and  $S_r=0.0001$  work better. Each time,  $F_r$  and  $S_r$  values were kept the same across all neural network layers. We hypothesize that the depth and type of layer may have an effect on the values of  $F_r$  and  $S_r$ . We did not thoroughly examine this topic in our study, and it could serve as the focus of future studies.

The difference in the parameters  $F_r$  and  $S_r$  of *Dr: DRL* across CartPole, Acrobot and MountainCar can be explained by the level of difficulty. The easiest environment here is CartPole. MountainCar and Acrobot pose more challenges due to their deceptive natures. In fact, these deceptive reward environments discourage agents from exploring (by offering a negative reward), causing them to fall into local optimality. Following the guidance of the prior study [19], we assign an adaptability tolerance ratio of 20% to these two environments. As a result, we accept a performance degradation of 20% during the adaptation. For instance, in the MountainCar environment, the agent adapts to the drifted environment if it gets an average reward of  $-132$ , corresponding to  $-110$  (solved environments) minus 20% (see Figure 1(A)).

Finally, all experiments are run 10 times to overcome the stochastic nature of these DRL algorithms. We use statistical significance testing, i.e., the non-parametric Wilcoxon [55] test and the Vargha-Delaney A12 effect size [56], to compare the results obtained by *Dr: DRL* and vanilla CL.

**Environmental Drifts:** On the basis of the study [19] on the plasticity of DRL agents, two types of drifted environments are identified: 1) an adaptable environment to which the agent will be able to adapt, and 2) a non-adaptable environment to

which the agent cannot adapt. Hence, we consider both types of drifted environments. For the adaptable type, our goal is to evaluate the stability of our self-healing approach, as well as its ability to heal the behavior of the RL agent faster and more economically. Regarding the non-adaptable type, we aim to investigate how well our self-healing approach improves the DRL agent’s adaptability (healing ability) and enables its adaptation despite the failure of CL.

**Metrics:** Below, we introduce the different evaluation metrics that have been used in the empirical evaluations.

*Increase Ratio (%IR(X)) / Decrease Ratio (%DR(X)).* It consists of the percentage increase/decrease of the quantity X from applying *Dr: DRL* rather than CL, as formulated in Eq. 4/Eq. 5. Thereby, positive values indicate the on-watch quantity, X, has known an improvement, whereas negative values indicate a degradation.

$$\%IR(X) = \frac{Dr. DRL\_X - CL\_X}{CL\_X} \times 100 \quad (4)$$

$$\%DR(X) = \frac{CL\_X - Dr. DRL\_X}{CL\_X} \times 100 \quad (5)$$

*Adaptability Ratio (%AR).* It consists of the percentage of pairs (a trained DRL agent, a drifted environment) that a given approach Y is able to adapt, as formulated in Eq. 6.

$$\%AR(Y) = \frac{(Env, Agent)_{adpt(Y)}}{(Env, Agent)_{All}} \times 100 \quad (6)$$

## B. Research Questions and Answers

1) *RQ1 (Cost-effectiveness): How cost-effective Dr: DRL is in adapting DRL systems compared to continual learning?:*

**Motivation:** The purpose of this RQ is to compare *Dr: DRL* with the vanilla CL technique in terms of the cost-effectiveness of healing the DRL systems that failed to solve drifted environments. As the healing strategies operate on the DRL agent at runtime, delays in healing cycles can be resource-intensive and costly, and consequently, such operations are constrained by a maximum number of steps allowed in solving episodic environments. Thus, the more effective a healing strategy is, the shorter the time it takes.

**Method:** We measure the time and episode count necessary to heal a DRL system in response to a drifted environment. Both metrics are important to compare the cost-effectiveness of *Dr: DRL* and CL because one healing strategy can solve the drifted environment in fewer episodes while it reaches the maximum number of steps allowed in each episode. Hence, the time reflects the total number of steps done by the healing strategy. To consider a healed DRL agent as having successfully solved a drifted environment, it must achieve an average reward over 100 consecutive episodes greater or equal to the defined threshold (i.e., same as the original environment). To estimate the magnitude of the speed-up obtained when using *Dr: DRL* versus vanilla CL, we compute the decrease ratio, %DR (Eq. 5), of the fine-tuning time and episodes reached by *Dr: DRL* over the ones obtained by vanilla CL.

TABLE I  
COMPARISON OF HEALING TIME AND FINE-TUNING EPISODES  
DECREASES USING *Dr. DRL* OVER VANILLA CL.

		DQN	PPO	SAC
Episodes DR (%)	CartePole	<b>21.2</b>	<u>45.9</u>	0.9
	MountainCar	17.7	<u>2.9</u>	<b>20.1</b>
	Acrobot	3.2	9.3	<b>38.3</b>
Time DR (%)	CartePole	<b>30.3</b>	-20.5	1
	MountainCar	<b>20.3</b>	<u>24.4</u>	<b>18.7</b>
	Acrobot	5.3	16.2	<u>73</u>

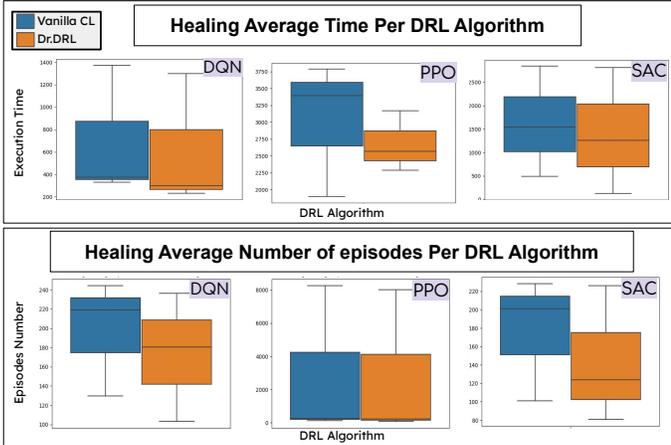


Fig. 3. Comparison of Healing Time and Fine-Tuning Episodes Box Plots Using *Dr. DRL* over vanilla CL.

**Results:** Table I shows that *Dr. DRL* often requires fewer healing cycles as measured by both fine-tuning time and episode count than its conventional alternative, vanilla CL. Indeed, the bold values indicate that the difference between the two compared healing approaches is statistically significant (i.e., the  $p$ -value  $< 0.05$ ). The underlined values indicate that their associated magnitude of the effect size  $A12$  is medium or large. Furthermore, Figure 3 demonstrates the juxtaposition of box plots drawn from the distributions of time and episode count obtained by *Dr. DRL* versus vanilla CL when tried with different pairs of DRL algorithms and environments. In line with statistical tests, the median values of both time and episode count achieved by *Dr. DRL* are consistently lower than their counterparts yielded by vanilla CL.

To be more specific, *Dr. DRL* outperforms vanilla CL in terms of fine-tuning episode count across all the pairs of agents and environments, where the gap between them were statistically significant in 44% of experiments with either a medium or large effect size. In regards to healing time, *Dr. DRL* outperforms vanilla CL across all the pairs of agents and environments, except for the pair of "PPO" algorithm and "CartePole" environment. In fact, *Dr. DRL* succeeds in healing PPO-based agents against drifted CartePole environments using longer steps by episode than its conventional alternative, vanilla CL, which delayed the healing time by *Dr. DRL* even with very few fine-tuning episodes.

**Finding 1.** *Dr. DRL* enables a cost-efficient healing of DRL systems in drifted environments, through fewer fine-tuning episodes and shorter healing time.

2) *RQ2 (adaptability frontiers): Can Dr. DRL enhance the adaptability of DRL systems to drifted environments?:*

**Motivation:** Biagiola and Tonella [19] demonstrated that vanilla CL has an environmental drift boundary beyond which it fails to heal the DRL system. *Dr. DRL* is designed with the objective of expanding the constrained scope of adaptable drifted environments. Hence, we assess the ratio of drifted environments that cannot be resolved through CL, but are instead resolved by *Dr. DRL*, and vice versa.

**Method:** We compute the adaptability ratios (%AR) (Eq. 6) obtained by the two compared healing approaches, *Dr. DRL* and vanilla CL, for all the studied agent-environment pairs. We expect these ratios can be divided into four major groups, representing four possible scenarios: (1) agent-environment pairs fitted by both healing approaches (i.e., *Dr. DRL* and vanilla CL); (2) pairs fitted exclusively by *Dr. DRL*; (3) pairs fitted exclusively by vanilla CL; and (4) pairs not fitted by both approaches.

**Results:** Figure 4 shows (a) the Pie chart demonstrating the global adaptability ratios (%AR) achieved by either *Dr. DRL*, vanilla CL, both, or none of them; and (b) Pie charts depicting the same results of adaptability ratios (%AR) broken down by environments.

As can be seen in Figure 4-a, our approach, *Dr. DRL*, succeeds in extending the boundary of adaptable environment ratio obtained by vanilla CL. More specifically, *Dr. DRL* was able to heal about 20% of agent-environment pairs on which vanilla CL fails. This shows the effectiveness of intentional forgetting of minor behaviors in the enhancement of the DRL system self-healing ability, leading to a reduced number of non-adaptable environmental drifts. Nevertheless, this expansion in the adaptability frontiers of vanilla CL using *Dr. DRL* was achieved at the cost of a few drifted environments on which *Dr. DRL* fails despite they can be resolved by vanilla CL. The reason is that any intentional forgetting, no matter how carefully screened or guided, induces a loss of information (i.e., the reset hypoactive neurons). The latter may cause inefficiencies in the DRL self-healing ability. However, the results of *RQ1* shows the cost-effectiveness resulting from our intentional forgetting mechanism in terms of the sample complexity. Hence, these 4% of non-adaptable drifted environments can be down to zero by reducing the ratio of erased minor behavior neurons over the healing trials by *Dr. DRL* in its future versions (i.e., the less minor behavior neurons are reset, the more *Dr. DRL* behaves as a vanilla CL).

In line with the previous study on DRL system plasticity [19], there are environmental drift frontiers of the DRL system adaptation in response to drifted environments that are beyond the capabilities of both vanilla CL and our novel healing approach, *Dr. DRL*. This reinforces the assertion that the continuous self-healing of in-production DRL agents to

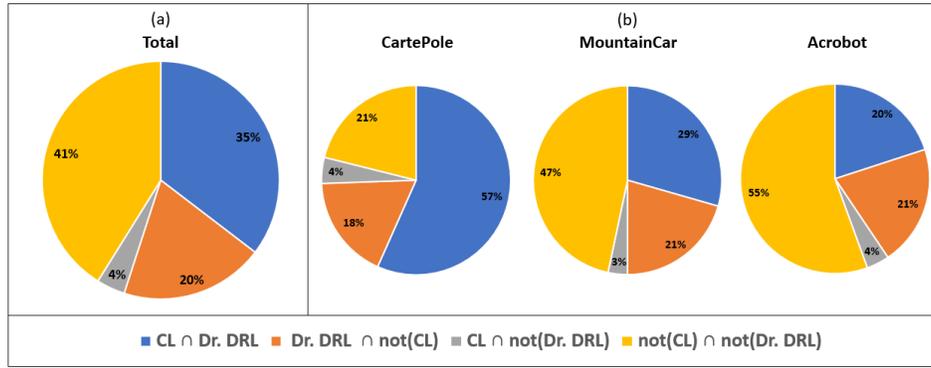


Fig. 4. Distribution of %AR over the 4 identified groups Per healing Method and RL Environment

constantly-shifting environments has limits. There are still instances of drifted environmental that require the re-training of the DRL agent from scratch or the redesign of the DRL algorithm beforehand in order to be able to resolve them.

According to Figure 4-b, the proportions of agent-environment pairs that are healed exclusively by one of the two healing approaches remain almost constant across all the studied RL environments. Meanwhile, the proportions of pairs that are either adapted or not equally by both approaches differ substantially depending on the studied environment. This can be explained by the differences in the parameterization complexity of these environments, i.e., involving high dimensional and sensitive parameters. Drifted instances derived from complex parametric environments are more challenging for self-healing approaches.

Research in healing software systems is very active and continues to advance towards equipping modern systems with self-healing capabilities in order to cope with ever-changing requirements throughout operation [9]. For DRL systems, our proposed approach, *Dr. DRL*, is in line with these current efforts, and by increasing the healing success ratio from 39% (i.e., 35% + 4%) to 55% (i.e., 35% + 20%), it represents a step further in assuring the DRL system’s self-healing ability in response to environmental drifts often encountered in production settings.

**Finding 2.** *Dr. DRL* outperforms the vanilla CL in healing a higher proportion of DRL systems to drifted environments, expanding the vanilla CL adaptability frontiers.

3) *RQ3 (Performance): Are Dr. DRL’s healed systems more performant in terms of average reward than their vanilla CL counterparts?:*

**Motivation:** Predictive performance remains the main selection criterion for any DRL system. Thus, restoring or even exceeding the original performance in drifted environment after healing is an important factor to evaluate a DRL system’s self-healing approach. Hence, we conduct a comparison between

TABLE II  
COMPARISON OF HEALED DRL AGENT REWARD INCREASES USING DR.DRL OVER CL.

		DQN	PPO	SAC
Reward IR (%)	CartePole	-0.1	-0.06	<u>-0.2</u>
	MountainCar	0.8	-0.5	<u>10.4</u>
	Acrobot	-0.5	-1.2	<u>13.1</u>
Reward IR (%)	CartePole	<u>-34.9</u>	<u>45.7</u>	2.2
	MountainCar	<u>3.5</u>	<u>25.8</u>	-15.4
	Acrobot	<u>14.2</u>	13.5	<u>21.2</u>

*Dr. DRL* and vanilla CL in terms of the resulting healed DRL system’s predictive performance in drifted environments.

**Method:** We compute the average of rewards obtained by the healed agent over the course of 100 consecutive episodes, using the two compared self-healing approaches, *Dr. DRL* and vanilla CL. Then, we calculate *Dr. DRL*’s average reward increase ratio, %IR (Eq. 4), w.r.t the vanilla CL’s average reward. Using the above-mentioned metrics, we solely consider the two following scenarios: (i) jointly adaptable environments, i.e., the agent-environment pairs on which vanilla CL and *Dr. DRL* both succeeded; (ii) jointly non-adaptable environments, i.e., the agent-environment pairs on which CL and *Dr. DRL* both failed. The remaining scenarios, where one approach, either vanilla CL or *Dr. DRL*, succeeds and the other fails, are irrelevant for predictive performance comparison because the gap between them will be wide due to the failure of one of them in healing the DRL system.

**Results:** The first row of Table II shows the reward increase ratio of *Dr. DRL* compared to vanilla CL in jointly adaptable environments. *Dr. DRL* increases the average reward obtained in 22% of experiments by at least 10%, while maintaining an almost equal average reward (no less than -1.2%) in the remaining 78% of the experiments. The second row of Table II shows the reward increase ratio of *Dr. DRL* compared to vanilla CL in jointly non-adaptable environments. *Dr. DRL* was able to improve the average reward in 78% of the configurations by up to 45% of the increase ratio over vanilla CL’s average reward. Table II demonstrates that *Dr. DRL* first

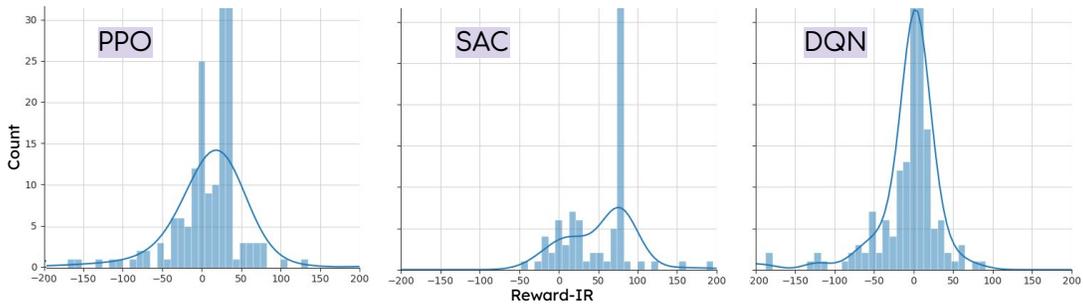


Fig. 5. Comparison of non-adaptable environments average Reward Increase Using Dr.DRL over CL.

succeeds in improving or at least maintaining the baseline average reward achieved by CL in jointly adaptable environments. Second, *Dr. DRL* meets average rewards better than vanilla CL in drifted environments which both self-healing approaches cannot resolve. For a more detailed view, we show the Figure 5, containing the distributions of *Dr. DRL*'s reward increase ratio (averaged by DRL algorithms) over vanilla CL in jointly non-adaptable environments. Overall, distributions follow a curve that tends to skew towards the right (increase ratio  $\geq 0$ ). These left-skewed distributions (also called right-leaning curves) show that our DRL system's self-healing approach, *Dr. DRL*, is able to exceed vanilla CL's predictive performance in non-adaptable environments.

**Finding 3:** *Dr. DRL* was capable of achieving similar or even higher performances in adaptable environments, and it also earned higher performances in non-adaptable ones, showing how much closer *Dr. DRL* was to meet the requirements than vanilla CL.

## VI. THREATS TO VALIDITY

In the following, we discuss the threats to validity.

**Threats to internal validity.** They may result from how the empirical study was conducted. To mitigate these issues, we have configured both *Dr. DRL* and CL with identical parameters (e.g., number of training episodes, maximum allowed steps per episode, etc.) across all studied pairs of RL environments and DRL algorithms.

**Threats to conclusion validity.** They are often related to random variations and inappropriate use of statistical tests. To mitigate these threats, we ran each experiment using either *Dr. DRL* and CL,  $6 \times 10$  times (6 environments with varying parameter shifts, and 10 different optimized DRL agents). Specifically, the training of 10 agents of each DRL algorithm using different random seeds, was intended to overcome the DRL algorithm's randomness. To confirm the statistical significance of our findings, we performed statistical hypothesis testing and effect size assessments using the non-parametric Wilcoxon test [55] and the Vargha–Delaney effect size [56].

**Threats to external validity.** They may limit the generalizability of our findings to other RL-based systems. Although

more subjects would be needed to fully assess the generalizability of our results, we have chosen 3 classic and diverse environments that are widely used in the DRL community, as well as, their implementations are included in gym library [27]. In addition, we have assessed our method on three distinct DRL algorithms from different types and having distinct learning strategies.

**Threats to reproducibility.** In order to guarantee the reproducibility of our evaluation, we offer a comprehensive replication package [34]. This package encompasses the complete source code of Dr.DRL, along with all necessary information and hyperparameters essential for replicating our assessment. This includes explicit details regarding the architecture of the tested agents and their corresponding neural networks. Moreover, in section V-A provide a thorough explanation of the fine-tuning procedures undertaken and the precise values of critical hyperparameters, notably  $S_r$  and  $F_r$ , essential for accurately reproducing our evaluation process.

## VII. RELATED WORK

This section summarizes related studies to our work.

Software healing approaches [11] have been explored to fix software system issues “in the field”, as opposed to software repair solutions [57], which are used “in-house” for bug-fixing operations. Indeed, healing approaches respond to software failures in production by making the appropriate adjustments, at runtime, to restore the system's normal behavior [11]. When these adjustments are fully automated without human intervention, the approaches are called software self-healing [8, 9, 10].

Due to their data-driven nature and unique architecture, specific healing, and repairing approaches have been proposed for ML systems. Rusu et al. [58] suggest freezing changes to previously trained networks and introducing additional sub-networks to learn new jobs. Shin et al. [59] train a generative model to create data with the same original data distribution. Thus, original data may be mixed with new data to preserve current knowledge and develop new skills. Next, the learning without forgetting (LwF) [60] approach limits network predictions on past tasks from changing while optimizing new task parameters. The above approaches focused on fine-tuning the model using more data. Yet, the fine-tuned model may not remove undesired behavior or efficiently learn new behaviors

without catastrophic forgetting [13, 61]. Other approaches tried to address the problem without model retraining. Apricot [62] is one attempt to fix DNN iteratively through a weight-adaptation method. Additionally, Arachne [63], a search-based DNN repair method, modified weights directly instead of retraining. Finally, Stocco and Tonella [64] proposed a novel CL method for misbehavior DNN predictors in self-driving cars. When data distribution shifts, this technique updates predictors using in-field confidence metric selection and error-based weighted retraining.

Despite all these attempts, DRL-specific healing still unstudied. Instead, few SE research such as Biagiola and Tonella [19] examined the confidence boundary of CL. Biagiola and Tonella [19] proposed AlphaTest, an approach for characterizing the DRL’s adaptability in its environment. AlphaTest takes a DRL agent that has been trained in a parameterized environment as input and samples the environment parameter space. It then trains the agent in CL mode on the sampled environment configurations, with the goal of characterizing the agent’s adaptability frontier. Leveraging AlphaTest, developers can learn about the healing boundaries of CL.

Unlike prior research, We propose a DRL-specific self-healing approach that uses intentional forgetting [22] and CL to optimize the agent’s adaptability. The proposed *Dr: DRL* could benefit the DRL community since no advanced self-healing approaches have yet been proposed. Instead, traditional healing alternatives are used like training a new policy from scratch or using vanilla CL.

### VIII. CONCLUSION

In this paper, we propose, *Dr: DRL*, a novel DRL self-healing approach, which involves an intentional forgetting step to systematically prioritize the adaptation of a DRL agent’s problem-solving skills when an environmental drift occurs in production settings. Indeed, *Dr: DRL* identifies the agent’s minor behaviors by watching the neurons with low-ranked activations over training episodes, called hypoactive neurons. If a drift in the environment happens and the agent becomes unable to solve it, *Dr: DRL* heals the DRL agent using two steps: (i) re-initializes the weights associated with hypoactive neurons with low-scaled values; (ii) performs continual learning updates of the patched DRL agent on the newly-drifted environment. By doing that, *Dr: DRL* enables the adaptation of DRL agent’s behaviors at dual speeds, i.e., untouched major behavior neurons maintain larger updates compared to under-scaled minor behavior neurons. In order to evaluate our proposed improvement, we conduct self-healing experiments of different DRL agents on drifted environments with varying parameter shifts using *Dr: DRL* and vanilla continual learning. The results demonstrate that *Dr: DRL* achieves (i) faster healing of the DRL agent, (ii) a higher number of adaptable environments, and (iii) equal or more elevated average rewards than vanilla CL. As part of our future work, we intend to explore other neuron categorization strategies besides activation-based ranking.

### ACKNOWLEDGMENT

This work is funded by the Fonds de Recherche du Quebec (FRQ), the Canadian Institute for Advanced Research (CIFAR), and the National Science and Engineering Research Council of Canada (NSERC). However, the findings and opinions expressed in this paper are those of the authors and do not necessarily represent or reflect those organizations/companies.

### REFERENCES

- [1] K. Arulkumaran, A. Cully, and J. Togelius, “Alphastar: An evolutionary computation perspective,” in *Proceedings of the genetic and evolutionary computation conference companion*, 2019, pp. 314–315.
- [2] F. Amat, A. Chandrashekar, T. Jebara, and J. Basilico, “Artwork personalization at netflix,” in *Proceedings of the 12th ACM conference on recommender systems*, 2018, pp. 487–488.
- [3] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, X. Ye, Z. Chen, and S. Fujimoto, “Horizon: Facebook’s open source applied reinforcement learning platform,” *arXiv preprint arXiv:1811.00260*, 2018.
- [4] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, “Incremental reinforcement learning with prioritized sweeping for dynamic environments,” *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 621–632, 2019.
- [5] S. Padakandla, P. KJ, and S. Bhatnagar, “Reinforcement learning algorithm for non-stationary environments,” *Applied Intelligence*, vol. 50, pp. 3590–3606, 2020.
- [6] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, “Characterizing concept drift,” *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.
- [7] P. Li, J. Thomas, X. Wang, A. Khalil, A. Ahmad, R. Inacio, S. Kapoor, A. Parekh, A. Doufexi, A. Shojaeifard et al., “Rlops: Development life-cycle of reinforcement learning aided open ran,” *IEEE Access*, vol. 10, pp. 113 808–113 826, 2022.
- [8] H. Psailer and S. Dustdar, “A survey on self-healing systems: approaches and systems,” *Computing*, vol. 91, pp. 43–73, 2011.
- [9] N. Perino, “A framework for self-healing software systems,” in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 1397–1400.
- [10] D. Ghosh, R. Sharman, H. R. Rao, and S. Upadhyaya, “Self-healing systems—survey and synthesis,” *Decision support systems*, vol. 42, no. 4, pp. 2164–2185, 2007.
- [11] L. Gazzola, D. Micucci, and L. Mariani, “Automatic software repair: A survey,” in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 1219–1219.
- [12] R. S. Sutton and A. G. Barto, “Reinforcement learning: an introduction mit press,” *Cambridge, MA*, vol. 22447, 1998.
- [13] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [14] B. Wexler, E. Sarafian, and S. Kraus, “Analyzing and overcoming degradation in warm-start reinforcement learning,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4048–4055.
- [15] E. Nikishin, M. Schwarzer, P. D’Oro, P.-L. Bacon, and A. Courville, “The primacy bias in deep reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 828–16 847.
- [16] C. Lyle, M. Rowland, and W. Dabney, “Understanding and preventing capacity loss in reinforcement learning,” *arXiv preprint arXiv:2204.09560*, 2022.
- [17] S. I. Mirzadeh, A. Chaudhry, D. Yin, T. Nguyen, R. Pascanu, D. Gorur, and M. Farajtabar, “Architecture matters in continual learning,” *arXiv preprint arXiv:2202.00275*, 2022.
- [18] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, “Towards continual reinforcement learning: A review and perspectives,” *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022.
- [19] M. Biagiola and P. Tonella, “Testing the plasticity of reinforcement learning based systems,” *ACM Transactions on Software Engineering and Methodology*, 2022.
- [20] C. Beierle and I. J. Timm, “Intentional forgetting: An emerging field in ai and beyond,” pp. 5–8, 2019.
- [21] I. J. Timm, O. Herzog, J. O. Berndt, and C. Beierle, “Intentional forgetting must be part of the functionality: Interview with prof. oththier herzog, jacobs university bremen, university of bremen, and tongji

- university, shanghai,” *KI-Künstliche Intelligenz*, vol. 33, pp. 89–91, 2019.
- [22] D. Shands and C. Talcott, “Intentional forgetting,” *arXiv preprint arXiv:2106.09802*, 2021.
- [23] L. Reuter, J. O. Berndt, A.-S. Ulfert, C. H. Antoni, T. Ellwart, and I. J. Timm, “Intentional forgetting in distributed artificial intelligence,” *KI-Künstliche Intelligenz*, vol. 33, pp. 69–77, 2019.
- [24] I. J. Timm, S. Staab, M. Siebers, C. Schon, U. Schmid, K. Sauerwald, L. Reuter, M. Ragni, C. Niederée, H. Maus et al., “Intentional forgetting in artificial intelligence systems: Perspectives and challenges,” in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 2018, pp. 357–365.
- [25] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu et al., “Deepgauge: Multi-granularity testing criteria for deep learning systems,” in *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, 2018, pp. 120–131.
- [26] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, “Deephunter: a coverage-guided fuzz testing framework for deep neural networks,” in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 146–157.
- [27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [28] 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.
- [29] A. W. Moore, “Efficient memory-based learning for robot control,” 1990.
- [30] A. Geramifard, C. Dann, R. H. Klein, W. Dabney, and J. P. How, “Rlpy: a value-function-based reinforcement learning framework for education and research,” *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1573–1578, 2015.
- [31] 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.
- [32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [34] “An intentional forgetting-driven self-healing method for deep reinforcement learning systems,” 2023. [Online]. Available: <https://github.com/ahmedhajyahmed/DrDRL>
- [35] Y. Li, “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*, 2017.
- [36] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [37] R. Bellman, “On the theory of dynamic programming,” *Proceedings of the national Academy of Sciences*, vol. 38, no. 8, pp. 716–719, 1952.
- [38] G. Dulac-Arnold, D. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” *arXiv preprint arXiv:1904.12901*, 2019.
- [39] M. Panzer and B. Bender, “Deep reinforcement learning in production systems: a systematic literature review,” *International Journal of Production Research*, vol. 60, no. 13, pp. 4316–4341, 2022.
- [40] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, “A novel incremental learning scheme for reinforcement learning in dynamic environments,” in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*. IEEE, 2016, pp. 2426–2431.
- [41] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, “Embracing change: Continual learning in deep neural networks,” *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.
- [42] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.
- [43] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, “Proving the lottery ticket hypothesis: Pruning is all you need,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 6682–6691.
- [44] A. E. Siswanto, “Block sparsity and weight initialization in neural network pruning,” Ph.D. dissertation, Massachusetts Institute of Technology, 2021.
- [45] C. M. Bishop, “Neural networks and their applications,” *Review of scientific instruments*, vol. 65, no. 6, pp. 1803–1832, 1994.
- [46] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [47] R. S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” *Advances in neural information processing systems*, vol. 8, 1995.
- [48] E. Urtans and A. Nikitenko, “Survey of deep q-network variants in pygame learning environment,” in *Proceedings of the 2018 2nd International Conference on Deep Learning Technologies*, 2018, pp. 27–36.
- [49] G. Van Rossum and F. L. Drake, *Python 3 reference manual*. CreateSpace, 2009.
- [50] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., “{TensorFlow}: a system for {Large-Scale} machine learning,” in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.
- [51] “Welcome to RLHive’s documentation! — RLHive 1.0.1 documentation.” [Online]. Available: <https://rlhive.readthedocs.io/en/stable/>
- [52] A. Hu, “Tf2-rl: Reinforcement learning algorithms implemented for tensorflow 2.0+,” 2020. [Online]. Available: <https://github.com/anita-hu/TF2-RL>
- [53] “Welcome to stable baselines docs! - rl baselines made easy — stable baselines 2. 10. 2 documentation.” [Online]. Available: <https://stable-baselines.readthedocs.io/en/master/>
- [54] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, “Implementation matters in deep policy gradients: A case study on ppo and trpo,” *arXiv preprint arXiv:2005.12729*, 2020.
- [55] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, p. 80, Dec. 1945. [Online]. Available: <https://www.jstor.org/stable/10.2307/3001968?origin=crossref>
- [56] A. Vargha and H. D. Delaney, “A critique and improvement of the cl common language effect size statistics of mcgraw and wong,” *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.
- [57] M. Monperrus, “Automatic software repair: a bibliography,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–24, 2018.
- [58] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.
- [59] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” *Advances in neural information processing systems*, vol. 30, 2017.
- [60] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.
- [61] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, “Experience replay for continual learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [62] H. Zhang and W. Chan, “Apricot: A weight-adaptation approach to fixing deep learning models,” in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 376–387.
- [63] J. Sohn, S. Kang, and S. Yoo, “Search based repair of deep neural networks,” *arXiv preprint arXiv:1912.12463*, 2019.
- [64] A. Stocco and P. Tonella, “Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems,” *Journal of Software: Evolution and Process*, vol. 34, no. 10, p. e2386, 2022.