Neutrons Sensitivity of Deep Reinforcement Learning Policies on EdgeAI Accelerators

Pablo R. Bodmann, Matteo Saveriano, Angeliki Kritikakou, and Paolo Rech

***Abstract*—Autonomous robots and their application are becoming popular in several different fields, including tasks where robots closely interact with humans. Therefore, the reliability of computation must be paramount. In this work, we measure the reliability of Google's Coral Edge TPU executing three Deep Reinforcement Learning (DRL) models through an accelerated neutrons beam. We experimentally collect data that, when scaled to the natural neutron flux, accounts for more than 5 million years. Based on our extensive evaluation, we quantify and qualify the radiation-induced corruption on the correctness of DRL. Crucially, our data shows that the Edge TPU executing DRL has an error rate that is up to 18 times higher the limit imposed by international reliability standards. We found that, despite the feedback and intrinsic redundancy of DRL, the propagation of the fault induces the model to fail in the vast majority of cases or the model manages to finish but reports wrong metrics (i.e. speed, final position, reward). We provide insights on how radiation corrupts the model, on how the fault propagates in the computation, and about the failure characteristic of the controlled robot.**

I. Introduction

Autonomous robots and their application are becoming popular in different fields, including smart manufacturing, agriculture, surgery, and space [1], [2]. Robots are asked to solve a variety of tasks of increasing complexity, which demands advanced learning and adaptation capabilities. Recent advances in Deep Reinforcement Learning (DRL) [3] have shown an exciting potential to address these demands. However, DRL algorithms are extremely resource-greedy and run on power-hungry Graphics Processing Units (GPUs). This limits the deployment of DRL solutions on resource-constrained devices like autonomous robots. Tiny Machine Learning (TinyML) is a promising strategy to reduce the computational resources required for the deployment of robot learning approaches [4]. A key component of TinyML solutions is the availability of low-power and low-cost Commercial-Off-The-Shelf (COTS) devices like the Tensor Processing Unit (TPU) developed by Google and named Coral Edge TPU [5]. Such EdgeAI accelerators allow the processing of neural networks on a small, cost-effective, and energy-efficient device. Therefore, EdgeAI accelerators are rapidly becoming the enabling technology for the deployment of AI-based solutions on resource-constrained devices like robots. As robot capabilities increase, the number and the complexity of interactions with other agents, including human beings also increases. When humans and robots share the same workspace, reliability must be paramount to prevent injuries [6]. Existing approaches for safe human-robot interaction focus on developing software solutions to ensure safe operations [7]–[9] and on measuring how humans perceive the interaction with a robotic co-worker [10], [11]. However, little research has been done on characterizing the reliability of hardware devices where the control policy of the robot is computed. A lack of reliable computation can cause significant faults and cause dangerous interactions with the robot's surroundings, including human beings. This can negatively affect both the training phase [12] and the deployment of the RL-based solution.

During normal operation in a real environment, there are several sources of transient errors that can lower EdgeAI accelerator's reliability, including environmental perturbations, software errors, and process/temperature/voltage variations [13], [14]. Radiation-induced soft errors are particularly critical, as they have been found to dominate error rates in commercial devices [15]. Crucially, advanced technology reduced power consumption, and increased operation frequency, despite bringing unquestionable benefits, increased the radiation sensitivity of computing devices, even for terrestrial applications [15]. Figure 1 teases the motivation of our evaluation, showing an example of an observed radiation-induced error on Hopper V3 (Figure 1b) and Humanoid V3 (Figure 1d). Because of a neutron impact, the EdgeAI accelerator wrongly computes the model, which is not able to recover and falls. In this paper, we investigate the sensitivity of DRL policies used to control autonomous robots as executed on Edge TPU. To gather a statistically significant amount of data, we expose the Edge TPU to an accelerated neutron beam. Overall, the Edge TPU has been irradiated with a total fluence of more than $6.5 \times 10^{11} n/cm^2$. With the proposed extensive experimental evaluation we aim at understanding if COTS EdgeAI accelerators are able to digest, during the computation, the radiation-induced fault or if they generate a measurable misbehavior. We also investigate the effects of this misbehavior on the robot's

This project has received funding from the Italian Ministry for University and Research (MUR) through the "Departments of Excellence 2023-27" program (L.232/2016) awarded to the Department of Industrial Engineering, the European Union's Horizon 2020 research and innovation programme under grant agreement No 101008126, from the SMART-ER project, funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement #101016888, and from the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. Neutron beam time was provided by ChipIR (DOI: 10.5286/ISIS.E.RB2000161) thanks to C. Cazzaniga, M. Kastriotou, and C. Frost.

Pablo R. Bodmann with the Informatics Institute on the Federal University of Rio Grande do Sul, Porto Alegre, Brazil (e-mail: prjbodmann@inf.ufrgs.br).

Matteo Saverino and Paolo Rech are with the Department of Industrial Engineering, University of Trento, Italy (email: matteo.saveriano@unitn.it and paolo.rech@unitn.it)

Angeliki Kritikakou is with INRIA, Rennes, France (email: Angeliki.Kritikakou@irisa.fr)

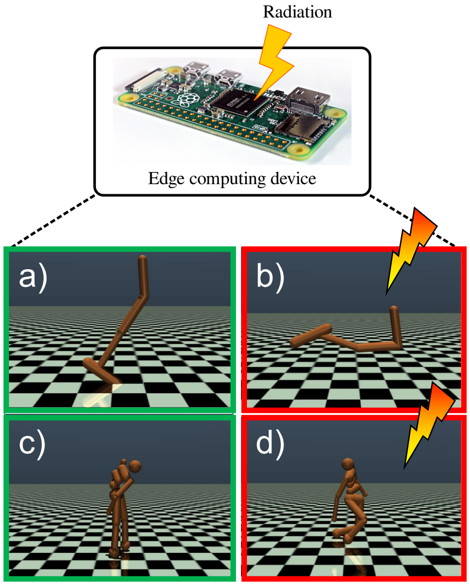

Fig. 1. Example of radiation effects on deep reinforcement learning policies, observed with accelerated neutrons experiments. Figures a) and c) show a correct execution of the Hopper V3 and Humanoid V3, respectively. Figure b) shows the effect of a radiation-induced fault that makes Hopper V3's leg to be in a position that will make the next jump impossible and Picture d) shows Humanoid V3 tripping which will cause a fall.

actuation and quantify how these erroneous actions negatively impact the execution of the robotic task. To the best of our knowledge, this is the first study that addresses the reliability of DRL policies.

The rest of the paper is structured as follows. Section II presents the background and this paper's contribution. Section III shows the experiment setup, device used, and the models chosen. In Section IV, the experimental results are presented and finally, in Section V the conclusions are drawn.

II. Background and Contribution

In this Section, we introduce the reliability issue, discuss the related works in the area, and highlight our contribution.

A. Reinforcement Learning

In a typical Reinforcement Learning (RL) setting, an agent takes actions in a (partially) unknown environment and it gets rewarded based on its state. More in details, at time t the agent is in state $\boldsymbol{s}_t$ and takes an action $\boldsymbol{a}_t$ according to a stochastic policy:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \Pr(\boldsymbol{a} = \boldsymbol{a}_t \mid \boldsymbol{s} = \boldsymbol{s}_t), \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^n$ are the parameters of the policy and π is the probability distribution of sampling action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$ at time t. Performing action $\boldsymbol{a}_t$ changes the state to $\boldsymbol{s}_{t+1}$ and the agent receives a reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Roughly speaking, the reward is a performance indicator that guides the agent to improve the learned policy. To this end, the agent explores the environment trying to maximize the expected return of the policy [16], i.e.,

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \tag{2}$$

In deep RL, the policy is usually represented by a Deep Neural Network (DNN). During the training, an additive (Gaussian) noise is added to the sampled action in order to favor the exploration and increase the robustness.

In this paper, we exploit the Soft-Actor Critic (SAC) [17] which is an example of an entropy-regularized deep RL algorithm. SAC is designed to handle continuous RL problems, i.e., tasks with continuous state and action spaces, and, therefore, it is well-suited for robotic applications. Instead of only seeking to maximize the expected return, SAC seeks to also maximize the entropy of the policy $\mathcal{H}(\cdot)$, i.e.,

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)\right]. \tag{3}$$

A high entropy in the policy is beneficial to explicitly encourage exploration. The scalar $\alpha > 0$ is used to balance between the two objectives, i.e., maximize the return and maximize the entropy.

B. Radiation Effects in Electronic Devices and DNNs

The Earth is constantly bombarded by high-energy particles coming from space. These particles interact with Earth's atmosphere, producing a flux of several different particles, mainly neutrons. About 13 $neutrons/((cm^2) \times hour)$ reach the ground [18]. When a neutron hits a transistor, the strike may perturb its state which in turn generates bit-flips in memory (the stored values are corrupted), or produces current spikes in logic circuits that, if latched, lead to an error (the operation output is wrong) [19]. Radiation-induced errors are *transients*, i.e. they do not permanently damage the hardware circuit: they corrupt the memory value or operation output. Neutron-induced errors are *not accumulative*, i.e. the probability for one neutron to induce a failure does not depend on how long the device has been exposed to neutrons nor to the number of errors that happened in the past [15].

When a code is being executed over the corrupted hardware, a transient fault may not affect the program output (i.e., the fault is masked, or the corrupted data is not used) or may be propagated through the stack of system layers leading to a Silent Data Corruption (SDC - the output is corrupted without any indication), or Detected Unrecoverable Errors (DUEs - a program or system crash). To exemplify the concept of SDCs, the computing device will output a value that is different from the expected one, without any flag or warning that the value is wrong. SDCs, being silent, are much more critical than DUEs that, by definition, are detectable and can trigger safety actions [15], [20]. When a DNN is executed on a device experiencing a transient fault, its output corruption can be tolerable, if the system behavior is maintained sufficiently correct, or critical, when it causes a system failure [21], [22]. What we show in this paper is that, despite the intrinsic redundancy and feedback effect of DRL models, the probability for critical errors to occur is far from being negligible.

C. Coral Edge TPU

Vendors have developed low-cost accelerators for machine learning execution, named *EdgeAI* devices that execute elementary operations (i.e., convolutions and some other matrices

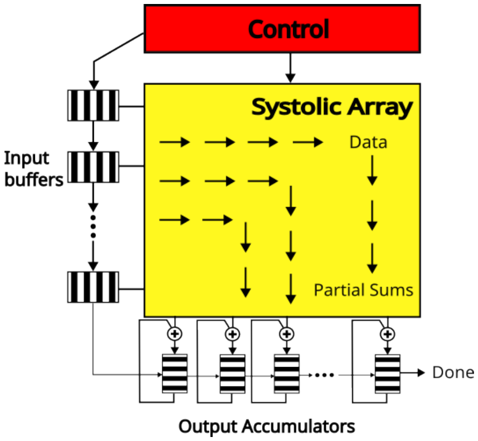

Fig. 2. Google's Coral Edge TPU architecture. Adapted from [5].

operations) in low precision (16-bit floating point or even 8-bit integer). Coupled with a good software framework (e.g., Tensorflow) that runs on a host device, EdgeAI devices significantly reduce the time and power consumption of DNNs execution.

The Edge TPU developed by Google is a coprocessor specialized in efficiently computing convolutions. Convolutions are the building blocks of CNNs, making the Edge TPU efficient in accelerating CNNs. The Edge TPU we use as the case study device is capable of computing 4 Tera operations per second with a maximum consumption of 2 watts. In order to maintain a small size and keep a low power profile, in the Edge TPU all internal operations are in unsigned 8-bit integers. The use of unsigned 8-bit integers also helps to diminish the latency of data transfer between the host and Edge TPU. Since the host uses neural networks in floating-point, the input and output need to be converted to unsigned integers before and after the model using a layer of quantization before the input and a layer of de-quantization after the output of the Edge TPU. This process is performed by the host device. Figure 2 shows an overview of the architecture of the Edge TPU, which is composed of a main systolic array fed by a large set of input buffers. The systolic array applies the model's weights on each layer's input and passes them into the activation unit, where the partial sums are accumulated and the activation function is applied. It is worth noting that these buffers and the main systolic array do not have any error correction capability.

Some previous work has been done on the reliability of the Edge TPU. In [23], researchers tested several different TPUs, including the Coral. However, in their work, they irradiate the Edge TPU with heavy ions and high-energy protons, while in this work we irradiate the Edge TPU with high-energy neutrons. Also, we tested only object detection neural networks, while we explored the reliability of reinforcement learning models. In [24], the authors report the Edge TPU reliability under high energy and thermal neutron radiation. The authors tested several different types of classification and object detection neural networks, while in this work we use the same device and the same radiation type but report the reinforcement learning reliability of 4 different models.

We use a Raspberry Pi 4 as the host device in order to control the Edge TPU. As discussed in Section III-A, we place the host CPU meters from the neutron beam, that is, we only irradiate the Edge TPU and consider the host procedures unaffected by radiation. The framework used to control the Edge TPU is based on the TensorFlow Lite, a light version, optimized for embedded devices, of the TensorFlow. To execute a CNN model on the Edge TPU, the model needs to be prepared first. Starting as a normal TensorFlow model using a 32-bit floating point in the model layers, the first thing is to convert from a floating-point number to an 8-bit unsigned integer and convert it to TensorFlow Lite. Lastly, convert quantized TensorFlow Lite to TensorFlow Lite compatible with the Edge TPU. All of these steps are already provided by Google as either Python or C++ libraries.

D. Contributions

We provide the first experimental evaluation of the impact of radiation in the execution correctness of Deep Reinforcement Learning (DLR) on Edge TPUs. Besides estimating the error rate on a realistic application, we also distinguish between critical and tolerable errors. For the former, we provide an in-depth analysis of the causes of the system failure. Our study is the first to target the reliability of reinforcement learning and the fault outcome in robotics applications. With our data, we provide useful information to improve the reliability of embedded reinforcement learning.

III. Methodology

In this section, we describe the radiation experiment setup, the tested application, and the devices.

A. Neutron Beam Experiments

We performed the radiation experiments at the ChipIR facility of the Rutherford Appleton Laboratory (RAL) in Didcot, UK. ChipIR delivers a neutron beam that mimics the atmospheric neutron one [25]. The average neutron flux at ChipIR was about $3.5 \times 10^6 n/(cm^2/s)$, which is about 2×10^9 higher than the natural one ($13\ \mathrm{neutrons}/(\mathrm{cm}^2 \times \mathrm{hour})$). We test the Edge TPU for about 62 effective hours (i.e., without considering the setup and result checking time) at ChipIR. Figure 3 shows part of our setup at ChipIR.

We run the Deep Reinforcement Learning (DRL) models on the Edge TPU accelerator. We irradiate the Edge TPU with a 3 × 3 cm beam spot, which is sufficient to irradiate the chip uniformly. A host Raspberry Pi, connected with a USB extension, was placed 2 meters from the beam spot, where the number of neutrons is negligible.

During irradiation, we execute continuously the models (discussed in the following subsection) with known inputs and compare the experimental output with pre-computed fault-free output. Once a mismatch is detected we count a Silent Data Corruption (SDC) and download data for post-processing. A watchdog detects application or system crashes, to count Detected Unrecoverable Errors (DUEs, i.e., crashes or hangs).

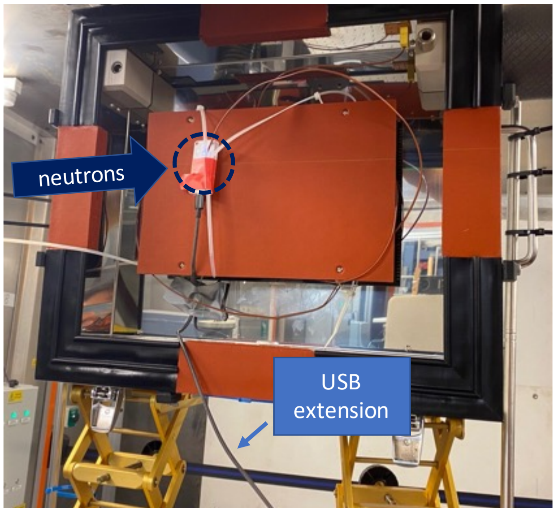

Fig. 3. The neutron beam test setup at ChipIR. the Edge TPU is aligned with the neutron beam, while the host Raspberry Pi is placed 2 meters away.

B. Tested Deep Reinforcement Learning Models

For our experiments, we consider Soft-Actor Critic (SAC) [17], a widely adopted DRL algorithm. We have chosen four different SAC models: Walker2D V3, Half-Cheetah V3, Hopper V3, and Humanoid V3. The models were chosen due to their increasing complexity: Walker2D V3 models only the lower part of a bipedal body, Half-Cheetah V3 models only the horizontal half of a quadruped animal (a single front leg and a single hind leg), Hopper V3 is composed of a single leg with a foot, and Humanoid V3 is a simplified complete humanoid body, with legs, torso, arms, and head. Considered models and trained SAC policies are publicly available [26].

We have designed two types of experiments, one with and one without environment simulation: we fed the Edge TPU with pre-computed data (without simulation) or with real-time processing based on the (possibly corrupted) feedback from the Edge TPU (with simulation). The experiment without simulation is useful to measure the overall FIT rate of the device, since the execution time on the host (not irradiated) is short, reducing the wasted beam time (if the Edge TPU is idle neutrons have no effect on the experiment). Then, to track the neutron effect on the model's behavior, we perform the experiment with full environment simulation. In the experiments with simulation, we log several different parameters including the model's velocity and position, if they differ from the expected values. The comparison is made in each step. It is important to notice that all models run entirely on the TPU, so the communication between TPU and host happens only at the beginning and at the end of each step. During execution, the TPU uses its own internal memory, so the distance does not influence the execution time. Lastly, the model's parameters are kept in the TPU, since the model does not change with the steps.

IV. Experimental Results

In this section, we present and discuss the experimental results obtained when exposing the Edge TPU executing DRL models to an accelerated neutron beam.

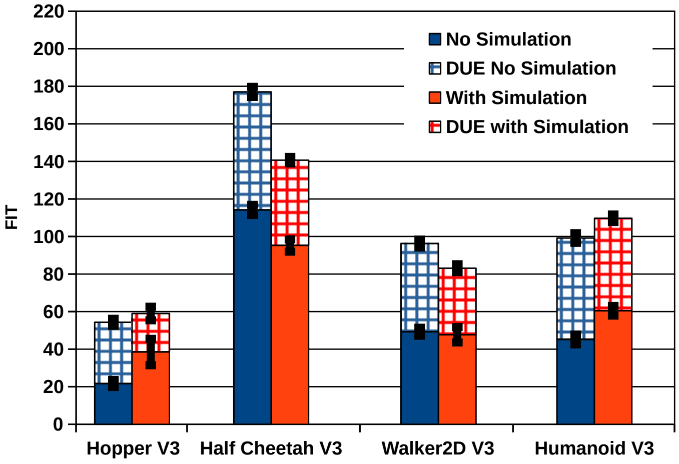

Fig. 4. SDC FIT rate and DUE FIT rate of all models with and without simulation. Data is shown with 95% confidence intervals considering a Poisson distribution.

A. FIT rate

Figure 4 shows the Failure In Time (FIT) rate of the Edge TPU executing the four models, with (red) and without environment simulation (blue). We separate the contribution of Silent Data Corruptions (SDCs, i.e., output error, in the filled bars) and of Detected Unrecoverable Error (DUE, i.e., crashes, patterned bars). We recall that the FIT is the number of errors expected in 10^9 hours of operation. The FIT values might seem low, in the sense that one TPU executing the tested models continuously for 10^9 hours ($\sim$ 114,000 years) will experience up to 176 errors (Half Cheetah V3 without simulation) 114 SDCs, and 62 DUEs. This might seem a reasonable error rate for a stand-alone chip employed in user applications. However, the expected large-scale adoption of EdgeAI devices exacerbates the probability of having several errors per hour, urging accurate evaluations. Indeed, according to the ISO2626-2 international reliability standard, a device employed in a safety-critical system, such as an autonomous vehicle, must have a FIT rate lower than 10 [27]. Our data shows that the Edge TPU can have an error rate that is 10 times higher. For instance, if Half-Cheetah V3, the model with the highest FIT rate, is employed in 1 billion devices, the overall expected error rate will be approximately *176 errors per hour*. From the observed FIT rate, we draw a first conclusion that the reliability of the DRL policies running on the Coral Edge TPU needs to be further investigated and characterized.

As mentioned in Section III-B, in the experiment without simulation we fed the Edge TPU with pre-computed steps (thus reducing the host calculation overhead), while in the experiment with simulation, the results of the computation are transferred in real-time to the robot simulator and the updated state is re-fed to the Edge TPU. We made the experiments without simulation in order to get the FIT rate more efficiently since the simulation takes a considerable amount of time to run (up to 80%). However, the experiment with the simulation is necessary to track the fault propagation through the model's behavior. The results for the effect of faults are shown and discussed in the next subsection. It's important to note that the simulation is not affected by neutrons since it is executed on the Raspberry Pi's CPU. The two configurations (with and without simulation) have similar FIT rates, for all four tested models. The FIT rate of the executions with and without simulation are similar, the difference being less than 43 % This should not be a surprise, since in both cases, the Edge TPU executes the same operations, while the simulation runs on the Raspberry Pi which is out of the beam. Neutron effects are transient and not cumulative, thus in the FIT calculation we only consider the fluence in the irradiated TPU during the execution of the steps, not during simulation.

We observe that DUEs are less likely to occur than SDCs in Hopper (only on the experiments with simulation), Half Cheetah, and Walker, while they are comparable with SDCs in Humanoid. We focus our discussion on the experiments with simulation, which is the realistic configuration for Reinforcement Learning. Hopper V3 has different trends in the ratio SDC/DUE, which is due to the lower SDC FIT rate compared to other models, which makes the DUE component relatively more important. DUEs are detectable (the application crashes) and, thus, are to be considered less critical than SDCs [15]. Nonetheless, given the high FIT rate of DUEs we have observed, it is necessary to guarantee that, in the event of an application crash, the robot will maintain a safe position while the computing system is rebooted. DUEs can be easily detected with a watchdog that, when triggered, makes the robot enter a safe position which depends on the environment and application. Since DUEs do not provide further information, besides the observed application crash, in the following discussion we will focus on SDCs, to understand how the operation of the robot is influenced by the wrong computation.

Figure 4 shows that the more complex models (Walker2D V3, Half Cheetah v3, and Humanoid V3), i.e., those with more joints, are the ones with higher FIT rates. This difference is up to $5.26\times$ between the lowest (Hopper V3) and the highest (Half Cheetah V3) on the experiment without simulation (blue bars). Of all models, Humanoid V3 is the most complex due to the fact that it has an upper body section with a sectioned torso, arms, and a head. In contrast, Half Cheetah V3 has a pair of legs (front and hind legs) and the Walker2D V3 only models a pair of human legs. The higher FIT rate for Half Cheetah, Walker, and Humanoid comes from the higher number of operations executed on the Edge TPU. Indeed, DRL control policies take as input the robot state (joint positions and velocities) and output control actions, leading to networks of different sizes. For comparison, the Hopper V3 (the simplest model) model has 15 states and 3 actions, while Humanoid V3 (the more complex model) has 44 states and 17 actions. A higher number of operations increases the probability for a neutron to hit the Edge TPU while processing, thus increasing the error rate. But this also may not be the only contributing factor. When comparing Humanoid with Half Cheetah, we can see that Half Cheetah has a higher FIT rate than Humanoid, despite Humanoid being more complex. This may indicate that the model structure may also contribute to the overall FIT rate since Half Cheetah has fewer states and actions(16 states and 6 actions) than Humanoid. Despite not being the most complex model, Half-Cheetah has the highest FIT rate. Half-Cheetah, despite having just two legs as all the models we test, is based on a four-legged being. The four-legged structure is shown to be more stable than a two-legged one in the occurrence of a fault. In fact, as our results show, it is unlikely for a fault in the Half-Cheetah model execution to induce the robot to fall. Also, as will be discussed further the type of predominant critical SDC also changes.

TABLE I
PERCENTAGE OF THE OUTCOME OF THE ERRONEOUS EXECUTIONS.

	FINISHED		NOT FINISHED
	W/ ERROR	W/O ERROR	
HOPPER V3	33.33%	33.33%	33.33%
WALKER2D V3	9.09%	9.09%	81.82%
HUMANOID V3	8.57%	8.57%	82.86%
HALF CHEETAH V3	88.24%	11.76%	0%

B. Radiation-induced effect on the model's execution

To have a further fine-grain analysis of the outcome of radiation-induced faults, we list, in Table I, the observed outcome of the SDC during the execution of the robotic task. For the four considered models, the goal is to walk straight (x direction) as much as possible within a maximum number of simulation steps. An execution without failures is considered as the expected outcome of each model. We count the number of detected SDCs on the experiment with simulation and classify all erroneous executions into three categories: (i) *Finished Without Error*: the model shows only errors on the intermediary steps, before the final one, but can still reach the correct final step. (ii) *Finished with Error*: the model completes the expected number of steps, but the final position has at least one wrong parameter (for example speed, position, and reward); (iii) *Not Finished:* the model stops before the expected number of steps. We consider both Finished with Error and Not Finished as critical, while Finished without Error indicates that an error occurred in one of the steps but, in the following steps, the model was able to recover.

It is interesting to see that, for the majority of the models, being Half Cheetah the exception, the fault makes the robot to stop *before* completing the expected number of steps. This means that it is highly probable that a radiation-induced fault will lead the robot to completely fail the task on these models. In the worst case, more than 82.86% of the erroneous runs make the model stop before expected (Humanoid V3). Walker2D V3 has a similar behavior, with 81.82% of the errors forcing the model not to reach the end. In the considered models, the stop condition is triggered if the maximum number of steps is reached, but also when the robot falls (this is measured by a health function that considers the height of the model and the position of the torso). Therefore, if the model does not arrive at the expected number of steps, we can assume that the robot has fallen due to the radiation-induced fault. Hopper V3 is the only model that has equal probability in the three categories. However, this might be due to poor

This article has been accepted for publication in IEEE Transactions on Nuclear Science. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNS.2024.3387087

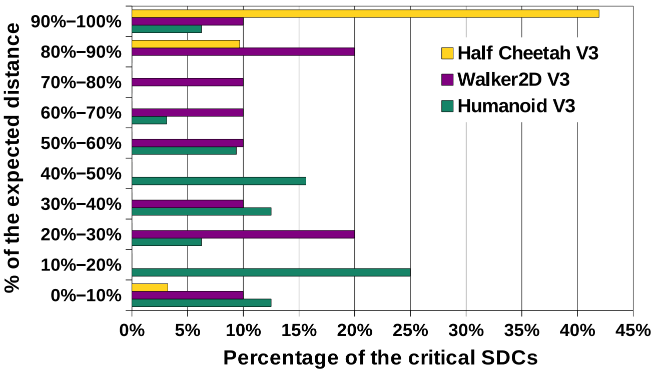

Fig. 5. Histogram of the distance traveled by the model. The data is normalized by the expected distance.

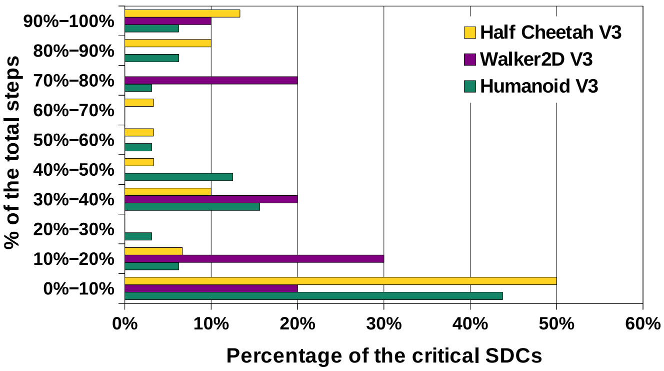

Fig. 6. Histogram of the first erroneous step of the model. The data is normalized by the expected number of steps.

statistics, since for Hopper V3, due to beam time constraints and the lower error rate, we have observed only a few errors.

However, when looking at Half Cheetah we can see that it has the highest FIT rate but it's not the most complex model. Also, it is interesting to see that the model manages to always finish even in the presence of errors, with the majority of times finishing with error. The fact that it manages to finish all the time, can be explained by the fact that Half Cheetah is more stable due to the disposition of legs (like a quadruped animal) and the lateral stability is not modeled (the model cannot fall sideways). Also, when comparing to the other models we can assume that due to the structure of the model of Half Cheetah, a fault is more likely to make the model misbehave but they are less likely to make the model fall, while in Humanoid, Walker 2D, and Hopper the neutron is less probable to create an error but they are more probable to make the model fall.

We can further analyze the observed critical SDCs considering the distance covered by the model before falling. Figure 5 shows a histogram of the distance where the model has stopped normalized by the expected distance (from a successful execution). The critical errors are grouped in ranges of 10%, from 0% to 100%. Humanoid V3, in 84.38% of the critical SDCs, has traveled up to 70% of the expected distance. Walker2D V3 traveled, on average, a little further, with 60% of the critical errors happening on up to 60% of the expected distance. This means that, despite Walker2D V3 having a similar FIT rate and a similar rate of critical errors to Humanoid V3, the former is able to reach a closer position to the final one. This can be justified considering that Humanoid V3 has an upper body, which is easier to destabilize than Walker 2D V3. The outlier is Half Cheetah where in 96% of the critical SDCs, the model has traveled more than 80% of the expected distance while finishing the expected number of steps, and in 80% of the critical SDCs, Half Cheetah has traveled more than 95% of the expected distance. This shows that a fault in this specific model is more likely to disturb the speed and, therefore the position than make it fall. One main factor to this is the quadruped arrangement of the legs (lateral stability is not modeled, i.e., the model cannot fall on its side) which grants more stability for the model. These characteristics influence the fault impact on the model behavior and how far the model will go after experiencing a fault. Even if the model arrives at the destination, the wrong speed may cause injury or equipment damage. Therefore, a developer must employ good hardening solutions when developing a robotic system with reliability in mind.

To understand the origin of the (critical) failures, besides how far the model can travel, it is also important to see in which step the faults are more likely to be generated. Our setup allows us to track the fault from the first step in which a mismatch was observed till the final step. As previously done for the traveled distance, we take the critical errors and see in which step the fault was first observed. We also normalized these results with the expected number of steps for each model. Figure 6 shows the histogram of probabilities of the step in which the fault was observed. It is interesting to notice that most of the errors happen in the first steps. For Humanoid V3, 84.38% of the SDCs begins in the first half of the expected steps. For Walker2D V3, 70% of critical errors start in the first half. For Half Cheetah, 70% of the errors begin in the first half. These observations, when correlated with Figure 5, indicate that a fault does not make the robot fall immediately. On the contrary, the radiation-induced corruption triggers a disturbance to the equilibrium that the model is unable to handle, in the case of Walker 2D and Humanoid. In the case of Half Cheetah, this disturbance seems to affect the speed of the model and not its stability resulting in the model finishing the expected number of steps but a little behind or further than expected. Such a result is crucial since it demystifies the false myth that reinforcement learning models are intrinsically reliable thanks to their feedback. Also, the model's architecture may also dictate the consequence of an error on its behavior.

To further quantify and qualify how critical a (small) radiation-induced corruption can be for the model equilibrium, we measure the difference between the corrupted and the expected status before the failure. Figure 7 shows the difference between the forward position at the last step before the model stopped and the expected position at that step. The data is plotted by the normalized expected final distance. The single negative Walker 2D V3 SDC is a failure where the model went backward and fell, and the lowest one of Half Cheetah is where the model stayed still. On Humanoid V3, there are two SDCs where the model finished beyond the expected distance. The

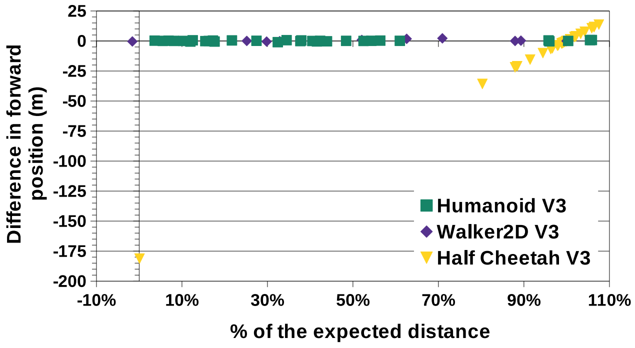

Fig. 7. Difference between the position where the model stopped and where it should be per normalized expected distance for the Humanoid V3 and Walker V3.

first thing interesting to notice is that even a small deviation may cause the model to fall. There are several SDCs with very little difference in the stopping position. Most of the SDCs are between 0.4 and -0.4 meters for Walker and Humanoid. Also, faults that either delay the model or make it arrive early, have similar probabilities of making the model fall. Finally, in most cases, the model manages to go further (but still falls) only if the failure happens later in the execution. For Half Cheetah we report that the model always finished the simulation but either a little behind than the expected distance or a little after. When combining this information with Table I, we can see that a fault does not make the model fall but affects its speed. Despite managing to arrive at the goal and finish the expected number of steps, a wrong speed may cause personal injury or equipment damage due to arriving at a position earlier or later than expected resulting in a collision with other equipment or personnel. Our results clearly show that feedback controllers learned by SAC are not able to reject disturbances, losing one of the important properties of feedback control.

V. Conclusion

In this paper, we have evaluated the neutron reliability of deep reinforcement learning algorithms when the learned control policies are executed on the Coral Edge TPU. In our analysis, we considered four robot models of increasing complexity: Hopper V3, Walker V3, Half Cheetah V3, and Humanoid V3. Our experimental results indicate that using the Edge TPU device to control an autonomous robot is an unreliable solution, despite being cost-effective and energy-efficient. Moreover, we show that most of the SDC errors are critical. This means that, when a robot suffers a failure, the probability of it not finishing its mission as expected is high. In our experiments, we observed that Half Cheetah, Humanoid V3, and Walker V3 fall in more than 80% of the cases. This is despite the fact that SAC policies are feedback controllers, commonly considered robust to external disturbances.

Our evaluation opens new lines of research that will address the reliability problem, like designing hardening solutions during the training to increase the capabilities of DRL policies to reject disturbances.

References

[1] Z. Liu *et al.*, “Robot learning towards smart robotic manufacturing: A review,” *Robotics and Computer-Integrated Manufacturing*, vol. 77, Art. No. 102360, Oct. 2022.

[2] Y. Gao and S. Chien, “Review on space robotics: Toward top-level science through space exploration,” *Science Robotics*, vol. 2, no. 7, Art. No. 5074, Jun. 2017.

[3] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, “How to train your robot with deep reinforcement learning: Lessons we have learned,” *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, Apr. 2021.

[4] S. M. Neuman, B. Plancher, B. P. Duisterhof, *et al.*, “Tiny robot learning: Challenges and directions for machine learning in resource-constrained robots,” in *IEEE International Conference on Artificial Intelligence Circuits and Systems*, 2022, pp. 296–299.

[5] Q-Engineering. “Google coral edge tpu explained in depth.” Accessed: 2023-02-01. (), [Online]. Available: https://qengineering.eu/google-corals-tpu-explained.html.

[6] S. Haddadin, A. De Luca, and A. Albu-Schäffer, “Robot collisions: A survey on detection, isolation, and identification,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, Dec. 2017.

[7] S. Haddadin, S. Haddadin, A. Khoury, *et al.*, “On making robots understand safety: Embedding injury knowledge into control,” *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1578–1602, Nov. 2012.

[8] M. Saveriano, F. Hirt, and D. Lee, “Human-aware motion reshaping using dynamical systems,” *Pattern Recognition Letters*, vol. 99, pp. 96–104, Nov. 2017.

[9] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, and F. Leali, “Human-robot coexistence and interaction in open industrial cells,” *Robotics and Computer-Integrated Manufacturing*, vol. 61, Art. No. 101846, Feb. 2020.

[10] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, “Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots,” *International journal of social robotics*, vol. 1, no. 1, pp. 71–81, Jan. 2009.

[11] P. A. Lasota and J. A. Shah, “Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration,” *Human factors*, vol. 57, no. 1, pp. 21–33, Feb. 2015.

[12] P. Falco, A. Attawia, M. Saveriano, and D. Lee, “On policy learning robust to irreversible events: An application to robotic in-hand manipulation,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1482–1489, Jul. 2018.

[13] J. C. Laprie, “Dependable computing and fault tolerance : Concepts and terminology,” in *Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years., Twenty-Fifth International Symposium on*, Jun. 1995, pp. 2–11. DOI: 10.1109/FTCSH.1995.532603.

[14] M. Nicolaidis, “Time redundancy based soft-error tolerance to rescue nanometer technologies,” in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 86–94. DOI: 10.1109/VTEST.1999.766651.

[15] R. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies,” *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Jul. 2005, ISSN: 1530-4388. DOI: 10.1109/TDMR.2005.853449.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] 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*, 2018, pp. 1861–1870.

[18] C. Slayman, “Jedec standards on measurement and reporting of alpha particle and terrestrial cosmic ray induced soft errors,” in *Soft Errors in Modern Electronic Systems*, vol. 41, Boston,

MA, USA: Springer US, 2011, ch. 3, pp. 55–76, ISBN: 978-1-4419-6993-4. DOI: 10.1007/978-1-4419-6993-4_3. [Online]. Available: https://doi.org/10.1007/978-1-4419-6993-4%5C_3.

[19] N. Mahatme, T. Jagannathan, L. Massengill, B. Bhuva, S.-J. Wen, and R. Wong, “Comparison of Combinational and Sequential Error Rates for a Deep Submicron Process,” *IEEE Transactions on Nuclear Science*, vol. 58, no. 6, pp. 2719–2725, Dec. 2011.

[20] V. Sridharan and D. R. Kaeli, “Using hardware vulnerability factors to enhance AVF analysis,” in *Proceedings of the 37th annual international symposium on Computer architecture*, Saint-Malo, France, 2010, pp. 461–472, ISBN: 978-1-4503-0053-7. DOI: 10.1145/1815961.1816023.

[21] F. F. d. Santos, P. F. Pimenta, C. Lunardi, *et al.*, “Analyzing and increasing the reliability of convolutional neural networks on gpus,” *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, Jun. 2019, ISSN: 1558-1721. DOI: 10.1109/TR.2018.2878387.

[22] D. Oliveira, L. Pilla, N. DeBardeleben, *et al.*, “Experimental and analytical study of xeon phi reliability,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, Colorado, 2017, pp. 333–344, ISBN: 978-1-4503-5114-0. DOI: 10.1145/3126908.3126960.

[23] M. C. Casey, J. S. Goodwill, E. J. Wyrwas, *et al.*, “Single-event effects on commercial-off-the-shelf edge-processing artificial intelligence asics,” *IEEE Transactions on Nuclear Science*, vol. 70, no. 8, pp. 1716–1723, Aug. 2023. DOI: 10.1109/TNS.2023.3286728.

[24] R. L. Rech Junior, S. Malde, C. Cazzaniga, *et al.*, “High energy and thermal neutron sensitivity of google tensor processing units,” *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 567–575, Mar. 2022. DOI: 10.1109/TNS.2022.3142092.

[25] C. Cazzaniga and C. D. Frost, “Progress of the scientific commissioning of a fast neutron beamline for chip irradiation,” *Journal of Physics*, vol. 1021, pp. 12 037–12 041, May 2018. DOI: 10.1088/1742-6596/1021/1/012037.

[26] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, vol. 22, Art. No. 268, Jan. 2021.

[27] *Road vehicles — functional safety*, ISO, 2015.