Effect of a slowdown correlated to the current state of the environment on an asynchronous learning architecture

Idriss Abdallah, Laurent Ciarletta , Patrick Hénaff, Matthieu Bonavent, Jonathan Champagne

Keywords: Deep reinforcement learning, Asynchronous architecture, Environment slowdown

Summary

In an industrial context, we apply deep reinforcement learning (DRL) to a simulator of an unmanned underwater vehicle (UUV). This UUV is moving in a complex environment that needs to compute acoustic propagation in very different scenarios. Consequently, the computation time per timestep varies greatly due to the complexity of the acoustic situation and the variation in the number of elements simulated. Therefore, we use an asynchronous actor-learner parallelization scheme to avoid any loss of computational resource efficiency. However, there is a strong correlation between the current state of the environment and this variability in computation time. The classical benchmarks in the DRL are not representative of our environment slowdowns, neither in magnitude nor in their correlation with the current observation. The aim of this paper is therefore to investigate the possible existence of a bias that could be induced by an observation-correlated slowdown in the case of a DRL algorithm using an asynchronous architecture. We empirically demonstrate the existence of such a bias in a modified Cartpole environment. We then study the evolution of this bias as a function of several parameters: the number of parallel environments, the exploration, and the positioning of slowdowns. Results reveal that the bias is highly dependent on the capacity of the policy to discover trajectories that avoid the slowdown areas.

Contribution(s)

1. We show that classical reinforcement learning benchmarks are not representative of our industrial environment in terms of the effects of slowdowns correlated with observation.
   Context: We based our analysis on the two most used deep reinforcement learning benchmarks : Mujoco (Todorov et al., 2012) and Atari (Bellemare et al., 2012) using the learning framework TorchRL (Bou et al., 2024).
2. We provide empirical evidence on a modified version of the Cartpole environment that an environment with observation-correlated slowdowns can induce a bias on the data generated and the learned policy for an algorithm using an asynchronous architecture.
   Context: We used the Dueling Double Deep Q-Learning (Wang et al., 2016) with the actor-learner architecture described by Espeholt et al. (2018) which decouples threads when generating transitions to reduce inter-process synchronization to achieve greater scalability.
3. We investigated bias changes as a function of the number of parallel actors, the exploration, and the positioning of the slowdown zone. Our results show that the bias is highly dependent on the capacity of the policy to find trajectories that avoid the slowdown areas.
   Context: None

Effect of a slowdown correlated to the current state of the environment on an asynchronous learning architecture

Idriss Abdallah[1,2], Laurent Ciarletta [1], Patrick Hénaff[1], Matthieu Bonavent[2], Jonathan Champagne[2]
{idriss.abdallah, laurent.ciarletta}@loria.fr,
{jonathan.champagne, matthieu.bonavent}@naval-group.com,
patrick.henaff@enib.fr

[1]**LORIA, Université de Lorraine, Nancy, France**
[2]**Naval Group, Gassin, France**

Abstract

In an industrial context, we apply deep reinforcement learning (DRL) to a simulator of an unmanned underwater vehicle (UUV). This UUV is moving in a complex environment that needs to compute acoustic propagation in very different scenarios. Consequently, the computation time per timestep varies greatly due to the complexity of the acoustic situation and the variation in the number of elements simulated. Therefore, we use an asynchronous actor-learner parallelization scheme to avoid any loss of computational resource efficiency. However, there is a strong correlation between the current state of the environment and this variability in computation time. The classical benchmarks in the DRL are not representative of our environment slowdowns, neither in magnitude nor in their correlation with the current observation. The aim of this paper is therefore to investigate the possible existence of a bias that could be induced by an observation-correlated slowdown in the case of a DRL algorithm using an asynchronous architecture. We empirically demonstrate the existence of such a bias in a modified Cartpole environment. We then study the evolution of this bias as a function of several parameters: the number of parallel environments, the exploration, and the positioning of slowdowns. Results reveal that the bias is highly dependent on the capacity of the policy to discover trajectories that avoid the slowdown areas.

1 Introduction

Reinforcement Learning (RL) is a machine learning approach that uses the actions of an agent in an environment to learn a behavior that maximizes a reward obtained during its interactions (Sutton & Barto, 2018). This approach, coupled with the ability of neural networks to approximate functions, has led to the emergence of Deep Reinforcement Learning (DRL), which is the basis of several advances in the field of sequential control (Mnih et al., 2015) (Silver et al., 2017).

Its success has come in particular from model-free approaches, which do not require a transition model of the environment but only a reward function. Although defining a problem with a reward function is an expressive method (Silver et al., 2021), its design significantly affects the algorithms' learning, making it difficult to use in practice (Gupta et al., 2022). This freedom also comes at the price of the need for a large number of interactions to obtain optimal behavior, notably due to the exploration-exploitation dilemma (Sutton & Barto, 2018). Furthermore, some intrinsic neural

network flaws result from this coupling of RL and deep learning, including explicability (Zahavy et al., 2016), training stability, and reproducibility (Henderson et al., 2018). Therefore, a number of critical aspects still need improvements to enable the application of DRL for a wider range of situations (Dulac-Arnold et al., 2020).

One of the main concerns of the DRL scientific community is to improve the speed and convergence capacity of algorithms. Several algorithms are used to generate and learn from data to improve the learning process of the policy (Sutton & Barto, 2018). Among them, a number of studies have focused on the efficiency and scalability of these algorithms, enabling them to make the most of large-scale computing power, such as Gorila (Nair et al., 2015), A3C (Mnih et al., 2016), Ape-X (Horgan et al., 2018), IMPALA (Espeholt et al., 2018), and SEED (Espeholt et al., 2020). They take advantage of different concepts such as making the different environment threads asynchronous, multiplying the number of processes updating the policy, vectorizing inferences, and minimizing inter-process communications.

In an industrial context, we apply DRL to a simulator of an UUV. This simulator models the kinematics of several vehicles as well as the acoustic signal propagation. As a result, we observe highly variable computation times depending on the current position of the UUV in the environment and the scenario in progress. This naturally led us to use an asynchronous architecture for reasons of resource efficiency and scalability of the computing resources available. But we also questioned whether this may lead to an overrepresentation of trajectories with faster computation times. This article's contributions are:

- Evidence on the non-representativeness of the Atari and Mujoco benchmarks for problems related to a variable computation time as in the considered industrial environment.
- Empirical evidences that a lag correlated with observation can generate bias in the data generated and in the policy learned.
- The study of the influence of certain learning parameters and environmental slowdowns on this bias.

First of all, we will define the concepts required for DRL, the parallelization architecture used, and certain characteristics of our industrial environment (section 2). We then present the various tools used in this article (section 3) and the results obtained (section 4). Finally, we will discuss the limitations of our results (section 5) and conclude.

2 Context

2.1 Reinforcement learning

RL is a field of machine learning in which an agent interacts with an environment through an action, which induces a change in the state of the environment. It then receives a reward in the form of a scalar that evaluates the transition made. The aim of RL is to learn, through interactions with the environment, the behavior that maximizes the reward obtained during these interactions.

The mathematical formalism used to represent such a problem is the Markov Decision Process (MDP). It is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathbf{T}, \mathbf{R}, \rho_0 \rangle$ with $\mathcal{S}$, the set of possible states, $\mathcal{A}$, the set of actions available to the agent, $\mathbf{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, a transition function that handles the evolution of the environment, $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, the reward function, and ρ_0, which defines the distribution of the initial state over all the states. We define the sum of future discounted rewards as $\sum_k \gamma^k r_{t+k+1}$, where r_t denotes the reward received at time t. This allows us to estimate the reward acquired over a specific horizon length dependent on the discount factor $\gamma \in [0, 1[$. A policy function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ represents a decision-making function which, on the basis of an observation, makes it possible to choose an action. The problem is to find π^*, the optimal policy that maximizes the expected sum of future discounted rewards.

The Dueling DDQN method, as proposed by Wang et al. (2016), serves as the foundation algorithm for the rest of this article. We used an asynchronous architecture similar to the single-learner architecture described in Espeholt et al. (2018). The implementations were made using ray (Moritz et al., 2017), PyTorch, and more precisely, the framework TorchRL (Bou et al., 2024).

In the remainder of this study, we did not use the Prioritized Experience Replay (PER) (Schaul et al., 2015), even though it greatly increases the training performances (Hessel et al., 2018). In fact, this modification involves changes to the sampling mechanism from the replay buffer and changes the distribution of the data provided to the policy. As the effect studied in this article is expected to directly affect the distribution of the data generated, it is therefore possible that the PER has an effect on the existence and magnitude of the studied bias, although it was not initially designed to do so. Hence, we removed it to not interfere with a potential bias linked to slowdowns correlated with observation (which is the topic of this paper).

2.2 Industrial environment

As part of our research work applied to an industrial context, we use an environment that allows us to run scenarios with diversified initial conditions and numbers of vehicles. Because it involves different vehicles that are not necessarily within detection range, the complexity of the acoustic environment depends on the current chosen scenario and state of the UUV during the simulation. This leads to highly variable computation times for each timestep (depending on both the scenario and the current state of the simulation). In the rest of this article, we refer to this phenomenon as **slowdowns**.

In order to have metrics for comparison between the computation time distributions observed in the rest of the article, we defined interquantile ratios at $\alpha\%$ ($IQR_{\alpha\%}$) as the ratio between the $100 - \frac{\alpha}{2}\%$ quantile and the $\frac{\alpha}{2}\%$ quantile. These metrics depict the slowdown order of magnitude on $100 - \alpha\%$ of the timesteps of an environment. Figure 1 shows a histogram of the computation times of the considered industrial environment with the quantiles used to compute the $IQR_{\alpha\%}$ in Figure 2. We generated these time steps using multiple scenarios with variations in the number, type, position and reactions of the simulated vehicles. It shows a high variability between timesteps, with a computation time ratio up to 10^4 between fast timesteps (e.g., lack of interaction between vehicles, and therefore reliance solely on kinematic models) and slower timesteps to simulate (e.g., presence of interactions between vehicles translated by calls to complex acoustic propagation models).

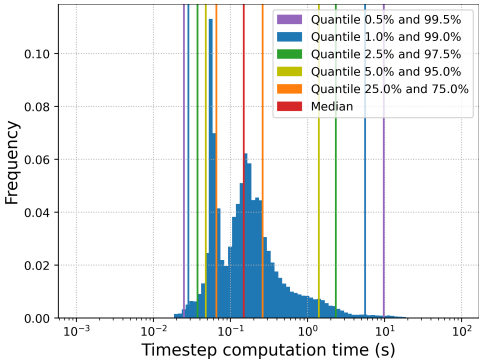


Figure 1: Histogram of 200,000 computation time for a timestep in our industrial environment.

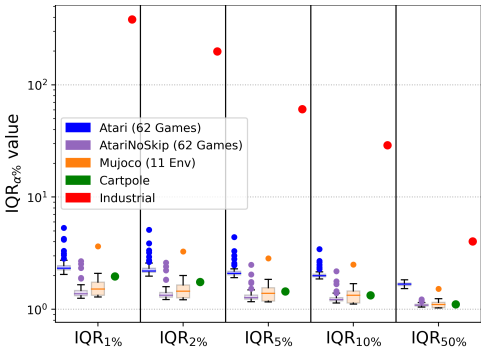


Figure 2: $IQR_{\alpha\%}$ values for Atari, Atari without frame skip, Mujoco, Cartpole and the considered industrial environment.

Slowdowns correlated to the current environment state are a well-known phenomenon. It can be seen in simulators and video games with varying numbers of graphical and physical components. We measured the computation times for the Mujoco and Atari benchmarks using their gymnasium implementation (Brockman et al., 2016). In fact, as Figure 2 illustrates, the median $IQR_{1\%}$ on Atari

and Mujoco are more than one hundred times lower than in the considered industrial environment. Moreover, the frame skip mechanism (Marlos C. Machado & Bowling, 2018) is responsible for the majority of the variability seen on Atari, but it is not linked to the current state of the environment. Therefore, even if there is a computation time correlated to the internal state of these environments, the slowdowns are several orders of magnitude lower than those we have in our industrial environment.

In addition, Table 1 shows that, unlike our industrial environment, the computation times of the Atari and Mujoco environments are actually quite low compared to the policy inference and observation processing times. Therefore, the potential effect induced by a slowdown correlated to the current state of the environment is even more negligible since this computation variability is absorbed by the other uncorrelated computation time.

Table 1: Ratios between the median policy inference and observation processing time with the median simulation time for one timestep on multiple environments.[1]

Environment	Atari (Median on 62 games)	Atari No Skip (Median on 62 games)	Mujoco (Median on 11 envs)	Cartpole	Considered industrial environment
Policy inference to environment time ratio	3.36	8.17	4.66	6.62	1.91×10^{-3}
Observation processing to environment time ratio	4.70	1.14×10^{1}	5.81	6.72×10^{-1}	2.01×10^{-2}

2.3 Asynchronous architecture

Usually, DRL algorithms employ a synchronous parallelization scheme. Figure 3 describes the threads' timeline of this architecture. It leads to waiting times between the various threads (simulation, synchronization, communication, and learning). In our industrial setting, the waiting between simulation processes (i.e. synchronization) is exacerbated because we use several hundred environments in parallel and have highly variable computation times. Under these circumstances, using a synchronous architecture reduces our computing resources' efficiency to 9%.

Hence, we use the asynchronous architecture described in Figure 4. The core idea is to decouple the threads into the actors (which perform transitions with the environment), the learner (which performs the policy update), and a buffer (which agregates transitions and the current policy). As illustrated in Figure 4, asynchronous architecture avoids mutual waiting effects since each actors and the learner communicate with the master as soon as their current task is over to give their results and get a new task. This architecture is therefore highly scalable to a large number of parallel environments (Espeholt et al., 2018).

3 Tools

3.1 Environment used

The environment used is based on Cartpole (Barto Andrew G. & W., 1983). It's a 1D toy environment where a cart has to balance a pole using two discrete actions: go left or go right. The use

[1] All these measurements were carried out using the implementation and hyperparameters proposed by the TorchRL github (Bou et al., 2024), using a single environment on a single CPU core of a 12th Generation Intel(R) Core(TM) i7-12700H. For the industrial environment, we used a neural network with two fully connected hidden layers of 256 neurons.

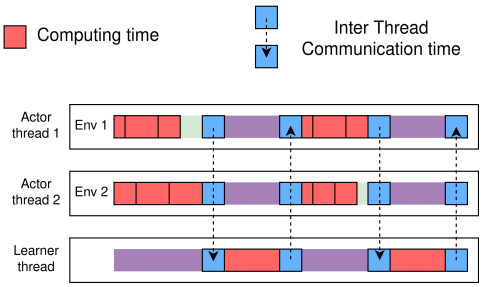


Figure 3: Threads' timeline for a DRL algorithm using a synchronous parallelization architecture.

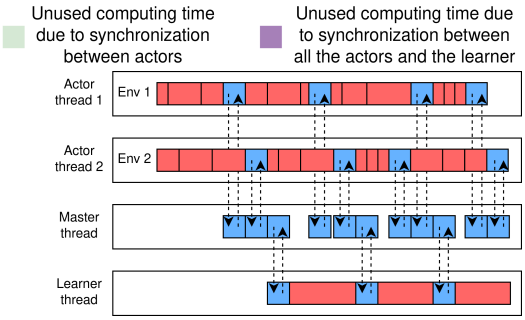


Figure 4: Threads' timeline for a DRL algorithm using an asynchronous parallelization architecture.

of a light and fast environment is motivated by the possibility of obtaining statistically significant results. Due to the extreme stochasticity of the DRL, it is challenging to discern distinct patterns among several trainings (Agarwal et al., 2021). In addition, using an environment with simple and symmetrical dynamics simplifies the study of the existence of a potential bias. The simplicity of finding an adequate policy also makes it possible to sweep across hyperparameters while still being able to converge.

The objective is to study the potential effects of a slowdown correlated to the observation. However, as shown in Figure 2, there is little slowdown in this environment. Therefore, we created two symmetrical zones defined by the position of the cart as illustrated by Figure 5. As depicted in Figure 6, we used these zones to add artificial slowdowns that stop the process for a multiple of the computation time actually taken for the whole transition (environment transition, observation processing, and inference). We define the **slowdown coefficient** of a zone as the value of this multiple when the cart is in this zone. Thus, it is feasible to observe a potential bias by comparing the policy behavior in the two different zones while maintaining different slowdown coefficients during training for both zones.

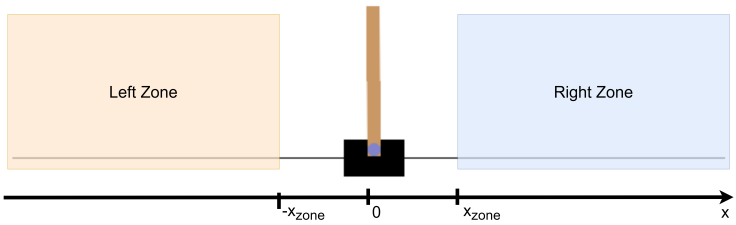


Figure 5: Cartpole environment with the two symmetric zones defined on x, the position of the cart, such as the left zone ($x \in [-\infty; -x_{zone}]$) and the right zone ($x \in [x_{zone}; +\infty]$).

3.2 Update scheduler

As illustrated in Figure 6, adding artificial slowdown to all interactions with the environment on an asynchronous architecture causes the learning process to execute more update steps for the same number of timesteps generated. As shown in Figure 7, this leads to changes in learning performances. This effect is already known and does not affect the data generation. Hence, we created an update scheduler that performs learner updates at fixed moments of the training based on the timesteps generated. This method eliminates the influence of slowdowns on the number of updates performed and allows us to investigate only the influence on the distribution of the generated data. For the rest of the paper, we scheduled the updates according to a Cartpole training without any artificial slowdowns.

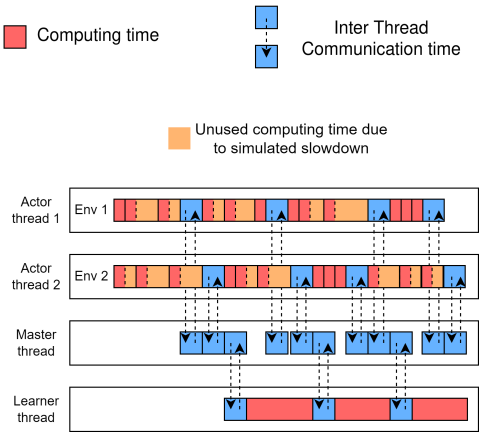


Figure 6: Threads' timeline for a DRL algorithm using an asynchronous architecture with artificial slowdowns.

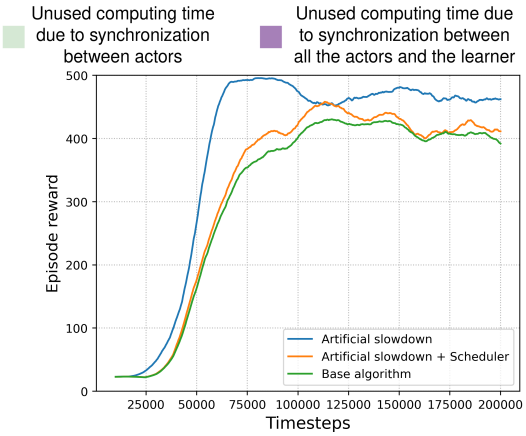


Figure 7: Learning performance with the addition of a slowdown coefficient of 10 on all transitions and the addition of an update scheduler averaged on 25 different seeds.

3.3 Simulated asynchronous architecture

The use of the environment described in section 3.1 induced a non-representativeness of the communication and learning time ratios w.r.t. to the considered industrial case and limited values for the slowdown coefficient (artificial slowdown slows the training algorithm real time).

Therefore, we created a single-process version that uses an event-driven loop in simulated time to simulate an asynchronous architecture. The core idea is to compute each task allocated to the simulated actors and the simulated learner without any artificial slowdown. Then, given the slowdown coefficient, the real computation time, and the trajectory, we evaluate the simulated time (which does take into account the artificial slowdown) at which each actor should end its task. Therefore, the actor with the lowest simulated time is added to the replay buffer, and its next task is computed to have its next simulated time. Figures 8 and 9 illustrate the task order of the main process on a case with 2 actors where each actor's task consists of 3 transitions, some timesteps have a slowdown coefficient of 10, and the learner is scheduled to make an update every 9 transitions collected. This enabled us to neglect the time for communication and learning and to use arbitrary values for slowdowns and the number of actors while maintaining a constant calculation time. Moreover, this version is highly parallelizable.

4 Results

The slowdown coefficients studied range from 10^0 to 10^4, which gives us a broad coverage of the distribution of slowdowns observed in the considered industrial environment. The hyperparameters correspond to those mentioned in the Appendix, unless their values are explicitly specified. Notably, as compared to a synchronous algorithm, the algorithm's performance in terms of total timesteps used is extremely poor. In fact, the environment is too simple and fast to take advantage of this architecture, but this section only aims to study the existence of a bias induced by a slowdown correlated to the observation.

All trainings were carried out using the simulated asynchronous algorithm described in section 3.3. As the results in this paper are solely based on empirical evidence, it was necessary to refine the confidence intervals as much as possible, given the highly random nature of the DRL (Agarwal et al., 2021). Hence, each point of each curve presented in this section is the average result obtained over 1000 separate training sessions with the same set of hyperparameters with its 95% confidence

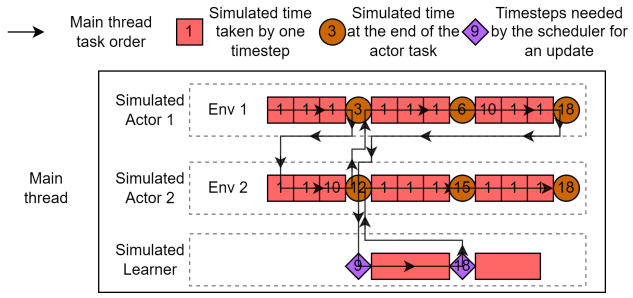


Figure 8: Threads' timeline for our simulated asynchronous architecture with artificial slowdowns.

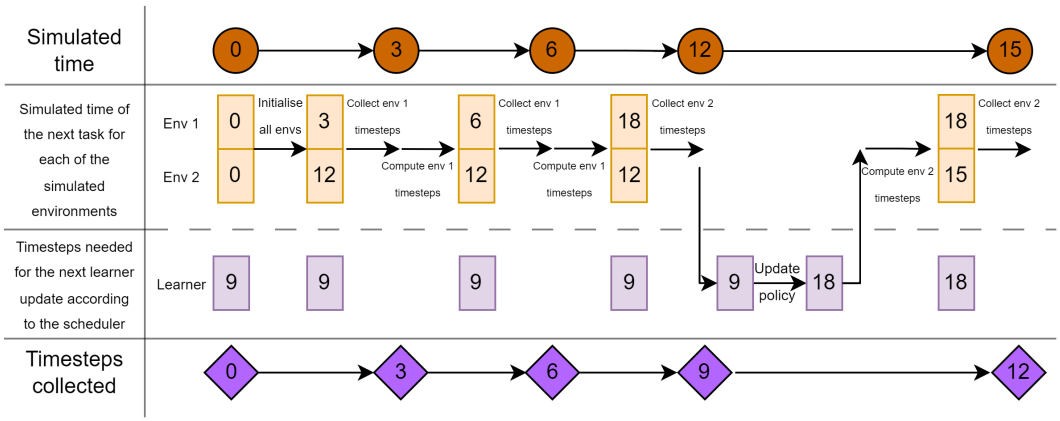


Figure 9: Task management for the simulated asynchronous architecture shown in Figure 8.

interval. This equates to 197,000 trainings for all the results in this section, which is one of the reasons why we chose to carry out this study on CartPole.

4.1 Bias induced on training data and policy by a slowdown zone

In this subsection, we study the influence of the left zone slowdown coefficient on the data generated during training and on the final policy learned. The metric used is the average presence per episode for both zones. Since the environment and the zones are symmetrical, if no bias is induced by a slowdown, then the presence must be similar between the two zones. This is what we observe when there is no slowdown, i.e., when the slowdown coefficient is 10^0 in figures 10 and 11.

In Figure 10 we observe a bias in the proportion of data generated during training, which increases presence in the right zone and decreases presence in the left zone. Additionally, we can see that the effect increases with the slowdown coefficient up to around 500 and then seems to stabilize.

Our understanding of this phenomenon is that during learning using an asynchronous architecture for interactions with the environment, a slowdown correlated with observation can lead to a bias in data generation. Actors taking trajectories involving a computational overhead will be underrepresented because they will be in 'competition' with other actors who have avoided these slowdown zones.

Since the left zone (i.e., the zone with the modified slowdown coefficient) is discrete, there comes a point at which a single timestep in this zone causes the trajectory to have a computation time so high that it becomes extremely difficult to take as much computation time in the rest of the environment (i.e., the right zone and the central zone that do not have artificial slowdown). Hence, the ordering of the actors becomes the number of timesteps spent in the left zone, regardless of the slowdown coefficient.

Figure 11 demonstrates that the final policy retains a behavior that is more present in the right zone. The learned behavior is therefore biased. As a result, we conclude that when learning with an

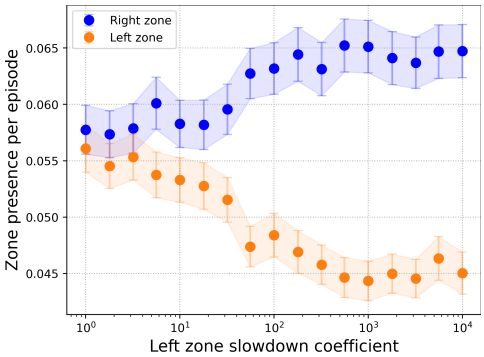


Figure 10: Average frequency of presence per episode per zone during training as a function of the slowdown coefficient in the left zone.

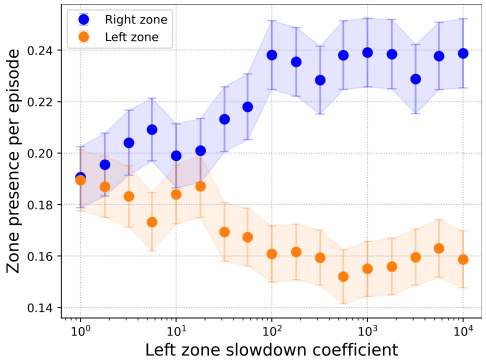


Figure 11: Average frequency of presence per episode per zone over 100 episodes with the final policy as a function of the slowdown coefficient in the left zone.

asynchronous architecture, a slowdown correlated with observation can lead to a bias in the data generated and also induce a bias in the policy learned.

4.2 Influence of parallel environments number

This subsection aims to evaluate the influence of the number of parallel environments used on the presence bias towards the right zone. The first noticeable influence is that training is impacted by the number of actors. This is shown in Figure 12, in the absence of slowdown, there is a decrease in presence in both zones (e.g., the curves with bullets). Figure 14 also illustrates an impact from this parameter, since the learning curves are not identical with the same slowdown coefficients. Our explanation is the access to more independent data from multiple distinct environments and an overall higher value of ε (i.e., the exploration parameter used for $\varepsilon - Greedy$) since the annealing mechanism does not take into account the timesteps currently computed and not returned by the actors.

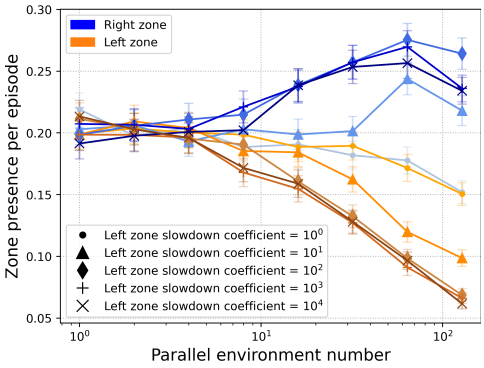


Figure 12: Average frequency of presence per episode per zone during training averaged over 1000 training sessions with a 95% confidence interval as a function of the number of parallel environments for different values of slowdown coefficients in the left zone.

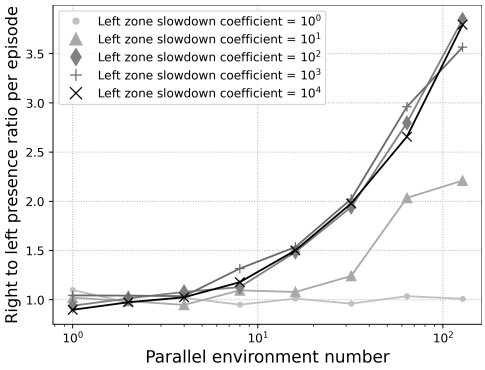
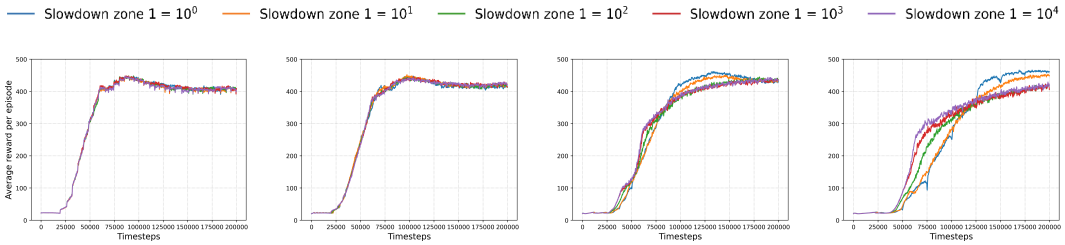


Figure 13: Average right zone to left zone presence ratio per episode during training as a function of the number of parallel environments for different values of slowdown coefficients in the left zone.

Figure 13 shows the presence ratio between the zones in Figure 12. It shows that the number of parallel environments amplifies the bias induced by the slowdown in the left zone. Actually, it appears impossible to identify any influence of the slowdowns for less than four parallel environments.



(a) 1 environment. (b) 16 environments. (c) 64 environments. (d) 128 environments.

Figure 14: Average rewards over 1000 seeds during training for different slowdown values and number of parallel environments.

Nonetheless, for all slowdown values, with more than four parallel environments, the more parallel environments there are, the more prevalent the right zone is in comparison to the left zone. In addition, we can see in figures 14c and 14d that the slowdown coefficient even seems to induce a loss of learning performance in the cases using a bigger parallel environment number, in contrast to trainings using fewer (14a and 14b).

We therefore conclude that the effect of the bias is amplified by the parallel environments number. Our explanation of this phenomenon is that it increases the number of actors in "competition" and therefore the chances that trajectories with a very low computation time will be discovered.

4.3 Influence of the exploration

This subsection studies the influence of increasing the exploratory behavior on the bias magnitude. To do this, we have modified the final timestep for annealing ε.

Similar to the section 4.2, Figure 15 demonstrates that, even in the absence of any slowdown, the parameter affects learning generally. However, figure 16 demonstrates that the bias increases when the final timestep for annealing ε is increased.

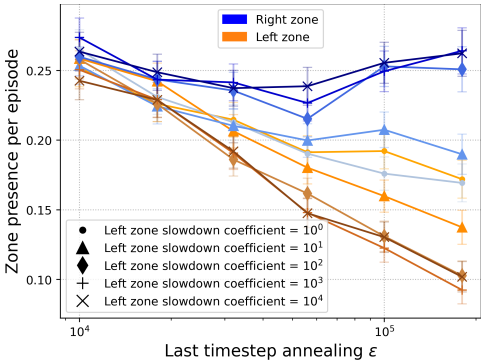


Figure 15: Average frequency of presence per episode per zone during training averaged over 1000 training sessions with a 95% confidence interval as a function of the last timestep for annealing epsilon for different values of slowdown coefficients in the left zone.

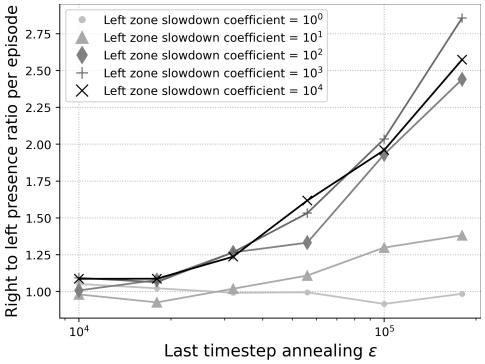


Figure 16: Average right zone to left zone presence ratio per episode during training as a function of the last timestep for annealing epsilon for different values of slowdown coefficients in left zone.

We conclude that the amount of exploration chosen also has an effect on the potential bias induced by a slowdown correlated to the observation. Our explanation for this trend is the increase in the

probability of discovering trajectories that avoid the slowdown zone and the reduction in the proportion of actions driven by the reward signal.

4.4 Influence of the zone position

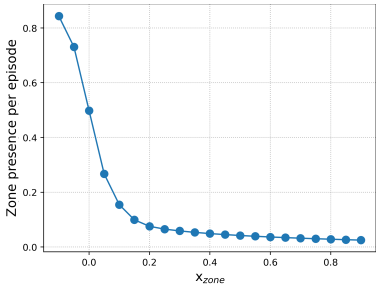


Figure 17: Average frequency of presence per episode per zone during training averaged over 1000 training sessions without slowdowns with a 95% confidence interval as a function of zone size.

This subsection aims to evaluate the influence of the slowdown zone on the bias induced on the final policy. Figure 17 shows that the presence value per zone is strongly related to the size of the zone. In order to evaluate and compare the bias while varying the x_{zone} value, we therefore varied x_{zone} but measured presence in the fixed zones corresponding to $x_{zone} = 0$.

As seen in Figures 18 and 19, the impact of slowdowns is not directly proportional to the zone's size. Indeed, we can observe that for slowdown zones representing a large part of the observation space (see Figure 17 between -0.1 and 0.1), there is in fact almost no bias on the policies learned. However, the bias worsens for values of x_{zone} between 0.2 and 0.4, even though the slowed zone represents a smaller proportion of the data encountered during training. Then, the bias magnitude decreases as x_{zone} diminishes for values greater than 0.4.

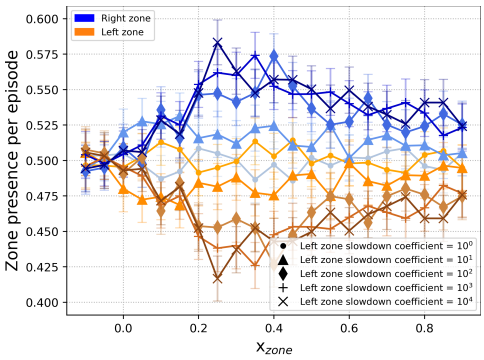


Figure 18: Average frequency of presence per episode per zone over 100 episodes with the final policy with a 95% confidence interval as a function of the size of the slowdown zone for different values of slowdown coefficients in the left zone.

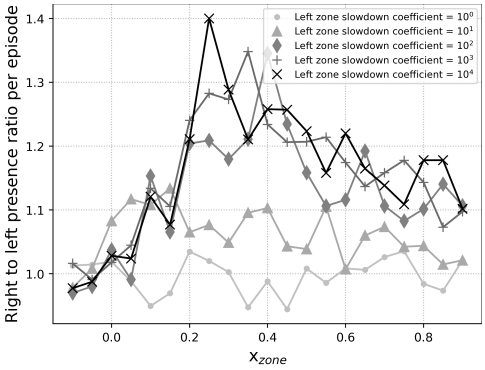


Figure 19: Average right zone to left zone presence ratio per episode during training as a function of the size of the slowdown zone for different values of slowdown coefficients in the left zone.

Since the episodes are randomly initialized so that $x \in [-0.05, 0.05]$, it is quite difficult to avoid a slowing zone that is too central from the perspective of environment dynamics. As a result, a slowdown zone that cannot be avoided has little or no influence. However, a slowdown in a lightly explored zone, where the policy is able to learn how to avoid it, can have an effect on the final policy. Finally, a slowdown in a zone that is almost never explored will not have any impact on the policy.

We therefore conclude that the effect of the bias is closely linked to the internal dynamic of the environment and the positioning of the slowdown zone in the state space.

5 Limitations

The main limitation of this study lies in the scope of validity of the environment used. Since our aim was to study the potential existence of a bias induced by slowdowns correlated with the observation, we chose a toy environment to obtain statistically significant results. Furthermore, the perfect symmetry of the dynamics and the reward function make it easier to exacerbate this phenomenon. It would therefore be interesting to study the feasibility of creating such a bias in a similar way in a more complex environment.

Furthermore, the slowdown zone considered is oversimplified, which does not accurately reflect our industrial context. Actually, the slowdown zone is discrete and purely geometrical, whereas industrially it comes from a coupling between the positions of all the vehicles in the operational theater.

Conclusion

Our industrial research work is based on an environment characterized by a high variability in computation time between each timestep. Therefore, we believed that using an asynchronous architecture was necessary to avoid a significant decrease in computational efficiency. However, the orders of magnitude of the observed slowdown and their correlation with the current state of the environment are not representative of conventional DRL benchmarks.

In the first part of this paper, we investigated the potential existence of a bias induced by such slowdowns in a modified version of the CartPole environment. Our results show that while the environment's dynamics and reward function are fully symmetrical, the final policy avoids the slowdown zone. This phenomenon is problematic because it shows that the slowdowns have an effect on the distribution of the data generated by the environment, potentially inducing a bias in the learned policy.

Next, we examined a number of algorithmic and slowdown zone parameters to determine how they affected the bias. Our results show that the bias is amplified by the number of parallel environments and by the proportion of actions dedicated to exploration. Furthermore, the zone's location is crucial because the bias only manifests when the zone is sufficiently eccentric. Our understanding is that the creation of a bias is strongly linked to the ability to find trajectories that avoid the slowdown zone.

Studying the presence of this bias in our industrial environment will be the primary focus of our upcoming work. This study shows that the extent of the bias is strongly related to the policy's ability to avoid slowdowns. If we do indeed observe a bias, we will then evaluate the pros and cons of using an asynchronous architecture and potentially study approaches to mitigate it.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, pp. 29304–29320, 2021.

Sutton Richard S. Barto Andrew G. and Anderson Charles W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. 47:253–279, 2012.

Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. Torchrl: A data-driven decision-making library for pytorch. In *ICLR*, 2024.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL http://arxiv.org/abs/1606.01540.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *CoRR*, abs/2003.11881, 2020.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018.

Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. In *ICLR*, 2020.

Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham M. Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. In *NeurIPS*, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pp. 3215–3222, 2018.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *ICLR*, 2018.

Erik Talvitie Joel Veness Matthew J. Hausknecht Marlos C. Machado, Marc G. Bellemare and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

Philipp Moritz, , Robert Nishihara, , Stephanie Wang, , Alexey Tumanov, , Richard Liaw, , Eric Liang, , William Paul, , Michael I. Jordan, , and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning. *CoRR*, abs/1507.04296, 2015.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, 2017.

David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. Reward is enough. *Artif. Intell.*, 299:103535, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, volume 48, pp. 1995–2003, 2016.

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *ICML*, volume 48, pp. 1899–1908, 2016.

Appendix

Parameter	Value
Network architecture	[64, 64, 32]
Learning rate	0.001
Optimizer	Adam
Total timestep	200000
Parallel actors	16
Parallel learners	1
Transition per task for actor	200
Update per task for learner	32
Timestep per update for learner	1024
γ	0.99
n-step	5
Initial ε value	1
End ε value	0.005
Last timestep for annealing ε	50000
Initial random step	10000
Replay buffer size	50000
x_{zone}	0.3
Right zone slowdown coefficient	1
Left zone slowdown coefficient	1

Table 2: Default hyperparameters value used