Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents (Extended Abstract)*

Marlos C. Machado†1, **Marc G. Bellemare**2, **Erik Talvitie**3, **Joel Veness**4, **Matthew Hausknecht**5 and **Michael Bowling**1,4

1 University of Alberta
2 Google Brain
3 Franklin & Marshall College
4 DeepMind
5 Microsoft Research

Abstract

The Arcade Learning Environment (ALE) is an evaluation platform that poses the challenge of building AI agents with general competency across dozens of Atari 2600 games. It supports a variety of different problem settings and it has been receiving increasing attention from the scientific community. In this paper we take a big picture look at how the ALE is being used by the research community. We focus on how diverse the evaluation methodologies in the ALE have become and we highlight some key concerns when evaluating agents in this platform. We use this discussion to present what we consider to be the best practices for future evaluations in the ALE. To further the progress in the field, we also introduce a new version of the ALE that supports multiple game modes and provides a form of stochasticity we call sticky actions.

1 Introduction

The Arcade Learning Environment (ALE) is both a challenge problem and a platform for evaluating general competency in artificial intelligence (AI). Originally proposed by Bellemare *et al.* [2013], the ALE makes dozens of Atari 2600 games available for agent evaluation. The agent is expected to do well in as many games as possible without game-specific information, generally perceiving the world through a video stream. Atari 2600 games are excellent environments for evaluating AI agents for three main reasons: 1) they are varied enough to provide multiple different tasks, requiring general competence, 2) they are interesting and challenging for humans, and 3) they are free of experimenter's bias.

The usefulness of the ALE is reflected in the amount of attention it has received from the scientific community. The number of papers using the ALE as a testbed has exploded in recent years. This has resulted in some high-profile success stories, such as the much publicized Deep Q-Networks (DQN), the first algorithm to achieve human-level control in a large fraction of Atari 2600 games [Mnih *et al.*, 2015]. This interest has also led to the first dedicated workshop on the topic, the AAAI Workshop on Learning for General Competency in Video Games [Albrecht *et al.*, 2015]. Several of the ideas presented in this article were first discussed at this workshop, such as the need for standardizing evaluation and for distinguishing open-loop behaviours from closed-loop ones.

Given the ALE's increasing importance in the AI literature, this article aims to be a "check-up" for the ALE, taking a big picture look at how the ALE is being used by researchers. The primary goal is to highlight some subtle issues that are often overlooked and propose some small course corrections to maximize the scientific value of future research based on this testbed. The ALE has incentivized the AI community to build more generally competent agents. The lessons learned from that experience may help further that progress and may inform best practices as other testbeds for general competency are developed [e.g., Beattie *et al.*, 2016; Brockman *et al.*, 2016; Johnson *et al.*, 2016; Nichol *et al.*, 2018].

The main contributions of this article are: 1) To discuss the different evaluation methods present in the literature and to identify, for the typical reinforcement learning setting, some methodological best practices gleaned from experience with the ALE. 2) To address concerns regarding the deterministic dynamics of previous versions of the platform, by introducing a new version of the ALE that supports a form of stochasticity we call *sticky actions*. 3) To introduce a new feature to the platform that allows existing environments to be instantiated in multiple difficulty levels and game modes. More results, as well as a much longer discussion about all these topics can be found in the original paper [Machado *et al.*, 2018].

2 The Divergent Evaluation Methodologies

Hundreds of papers have used the ALE as a testbed since it was introduced as a platform to evaluate general competency in AI. They employ many distinct evaluation protocols and, unfortunately, these different protocols are often not carefully distinguished, making direct comparisons difficult or misleading. In this section we discuss a number of methodological differences that have emerged in the literature.

One of the reasons that authors compare results generated with differing experimental protocols is the high computational cost of evaluating algorithms in the ALE — it is difficult to re-evaluate existing approaches to ensure matching

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Machado *et al.*, 2018].

†Corresponding author: machado@ualberta.ca

methodologies. For that reason it is especially important to establish a standard methodology for the ALE in order to reduce the cost of principled comparison and analysis.

To illustrate the diversity in evaluation protocols, we discuss some methodological differences found in the literature. While these differences may be individually benign, they are frequently ignored when comparing results, which undermines the validity of direct comparisons. We also provide recommendations to what we believe to be the best methodological practices that could be used in the ALE.

Episode Termination: In the initial ALE benchmark results, episodes terminate when the game is over. However, in some games the player has a number of "lives" which are lost one at a time. Terminating only when the game is over often makes it difficult for agents to learn the significance of losing a life. Mnih *et al.* [2015] terminated training episodes when the agent lost a life (evaluation episodes still lasted for the entire game). While this approach could potentially teach an agent to avoid "death", Bellemare *et al.* [2016] noted that it can in fact be detrimental to an agent's performance. Both approaches are still common in the literature. We often see episodes terminating when the game is over, as well as when the agent loses a life. Considering the ideal of minimizing the use of game-specific information and the questionable utility of termination using the "lives" signal, *we recommend that only the game over signal be used for termination*.

Setting of Hyperparameters: One of the primary goals of the ALE is to enable the evaluation of agents' general ability to learn in complex, high-dimensional decision-making problems. Ideally agents would be evaluated in entirely novel problems to test their generality, but this is of course impractical. With only 60 available games in the standard suite there is a risk that methods could "overfit" to the finite set of problems. In analogy to typical methodology in supervised learning, Bellemare *et al.* [2013] split games into "training" and "test" sets, only using results from training games for the purpose of selecting hyperparameters, then fully evaluating the agent in the test games only once hyperparameters have been selected. This methodology has been inconsistently applied in subsequent work, but for the sake of evaluating generality, *we advocate for a train/test game split as a way to evaluate agents in problems they were not specifically tuned for*.

Measuring Training Data: The first benchmarks in the ALE trained agents for a fixed number of episodes before evaluating them. This can be misleading since episode lengths differ from game to game. Worse yet, in many games the better an agent performs the longer episodes last. Recently it has become more common to measure the amount of training data in terms of the total number of frames experienced by the agent [Mnih *et al.*, 2015], which aids reproducibility, inter-game analysis, and fair comparisons. Frame skipping is also an important aspect to be taken into consideration as it is a common practice in the ALE that is not reported consistently in the literature. *We advocate evaluating from full training episodes from a fixed number of frames*, and *we advocate taking the number of skipped frames into consideration when measuring training data*, as the time scale in which the agent operates is also an algorithmic choice.

Summarizing Learning Performance: When evaluating an agent in 60 games, it becomes necessary to compactly summarize the agent's performance in each game in order to make the results accessible and to facilitate comparisons. Authors have employed various statistics for summarizing agent performance and this diversity makes it difficult to directly compare reported results. *We recommend reporting training performance at different intervals during learning.*

Injecting Stochasticity: The original Atari 2600 console had no source of entropy for generating pseudo-random numbers. The ALE is also fully deterministic. As such, it is possible to achieve high scores by learning an open-loop policy [Bellemare *et al.*, 2015]. Various approaches have been developed to add forms of stochasticity to the ALE dynamics in order to encourage and evaluate robustness in agents [e.g., Brockman *et al.*, 2016; Hausknecht and Stone, 2015; Mnih *et al.*, 2015]. *Our recommendation is to use* sticky actions, *implemented in the latest version of the ALE.*

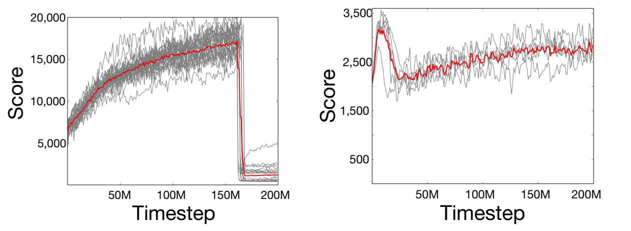
3 Summarizing Learning Performance

One traditional goal in RL is for agents to continually improve their performance as they obtain more data [Ring, 1997]. Measuring the extent to which this is the case for a given agent can be a challenge, and this challenge is exacerbated in the ALE, where the agent is evaluated across 60 games. When evaluating an agent in only a few problems, it is common practice to plot learning curves, which provide a rich description of the agent's performance: how quickly it learns, the highest performance it attains, its stability, whether it is likely to continue to improve with more data, etc.

While some have reported results in the ALE using learning curves [e.g., Schaul *et al.*, 2016], it is difficult to even effectively display, let alone comprehend and compare, 60 learning curves. For the sake of comparison and compact reporting, most researchers have applied various approaches to numerically summarize an agent's performance in each game [e.g., Bellemare *et al.*, 2013; Hausknecht *et al.*, 2014]. Unfortunately, the variety of different summary statistics in results tables makes direct comparison difficult. In this section we consider some common performance measures seen in the literature and ultimately identify one as being particularly in line with the continual learning goal and advocate for it as the standard for reporting learning results in the ALE.

Evaluation after Learning: Algorithms' performance are often summarized by the average score obtained in a number of evaluation episodes, with no learning, after the agent is trained for a fixed period [e.g., Bellemare *et al.*, 2013; Liang *et al.*, 2016]. This approach hides issues of sample efficiency because agents are not evaluated during the entire training period. Moreover, an agent can receive a high score using this metric without continually improving its performance. While the problem of developing a good policy during an unevaluated training period is an interesting one, in RL we typically expect algorithms to continually improve with experience.

Evaluation of the Best Policy: When evaluating DQN, Mnih *et al.* [2015] also trained agents for a fixed training period. Along the way, they regularly evaluated the performance of the learned policy. At the end of the training period they evaluated the *best* policy in a number of evaluation



(a) Sarsa(λ) +Blob-PROST (b) DQN

Figure 1: Comparison between learn. curves of DQN and Sarsa(λ) in CENTIPEDE. Notice the y-axes are not on the same scale. Each point corresponds to the avg. performance over the last 10 episodes. The red curve depicts the average over all trials (grey curves).

episodes with no learning. A great deal of follow-up work has replicated this methodology [e.g., Schaul *et al.*, 2016; van Hasselt *et al.*, 2016]. This protocol retains the downsides of evaluation after learning, and adds an additional one: it does not evaluate the *stability* of the agent's learning progress. Fig. 1 illustrates the importance of this issue by showing different learning curves in the game CENTIPEDE. On one hand, Sarsa(λ) with fixed linear features (Blob-PROST) [Liang *et al.*, 2016] achieves a high score early on but then becomes unstable and fails to retain this successful policy. DQN's best score is much lower but it is also more stable (though not perfectly so). Reporting the performance of the best policy fails to recognize the plummeting behavior of both algorithms. Note also that the best score achieved across training is a statistically biased estimate of an agent's best performance: to avoid this bias, one should perform a second, independent evaluation of the agent at that particular point in time.

Area Under the Learning Curve: Recently, eschewing an explicit evaluation phase, Stadie *et al.* [2015] proposed the area under the learning curve (AUC) as an evaluation metric. Intuitively, the AUC is proportional to how long a method achieves "good" performance, i.e., the average performance during training. Methods that only have performance spikes and methods that are unstable generally perform poorly under such metric. However, AUC does not capture the "plummeting" behavior illustrated in Figure 1. For example, in this case, Sarsa(λ) looks much better than DQN using this metric. Furthermore, AUC cannot distinguish a high-variance, unstable learning process from steady progress towards a good policy, even though we typically prefer the latter.

Proposal: Performance During Training

The performance metric we propose as a standard is simple and has been adopted before [Bellemare *et al.*, 2012]. At the end of training (and ideally at other points as well) report the average performance of the last k episodes. This protocol does not use the explicit evaluation phase, requiring an agent to perform well while it is learning. This better aligns the performance metric with the goal of continual learning while also simplifying evaluation methodology. Unstable methods that exhibit spiking and/or plummeting learning curves will score poorly compared to those that stably and continually improve, even if they perform well during most of training.

Additionally, this metric is well-suited for analysis of an algorithm's sample efficiency. While the agent's performance near the end of training is typically of most interest, it is also straightforward to report the same statistic at various points during training, effectively summarizing the learning curve with a few selected points along the curve. Currently, it is fairly standard to train agents for 200 million frames, in order to facilitate comparison with the DQN results reported by Mnih *et al.* [2015]. This is equivalent to approximately 38 days of real-time gameplay and even at fast frame rates this represents a significant computational expense. By reporting performance at multiple points during training, researchers can easily draw comparisons earlier in the learning process, reducing the computational burden of evaluating agents.

In accordance with this proposal, the benchmark results we present in the journal article that motivated this paper [Machado *et al.*, 2018] report the agent's average score of the last 100 episodes before the agent reaches 10, 50, 100, and 200 million frames. This evaluation protocol allows us to derive insights regarding the learning rate and stability of the algorithms and we believe it will offer flexibility to researchers wishing to use these benchmarks in the future.

4 Determinism and Stochasticity in the ALE

The dynamics within Stella itself (the Atari 2600 emulator embedded within the ALE) are deterministic given the agent's actions. The agent always starts at the same initial state and a given sequence of actions always leads to the same outcome. This determinism can be exploited by agents that simply memorize an effective sequence of actions, sometimes attaining state-of-the-art scores while ignoring the agent's perceived state altogether [Bellemare *et al.*, 2015]. Obviously, such an approach is not likely to be successful beyond the ALE – in most problems of interest it is difficult, if not impossible, to exactly reproduce a specific state-action sequence. An agent that relies upon the determinism of the ALE may achieve high scores, but may also be highly sensitive to small perturbations. As an evaluation platform, the deterministic ALE does not effectively distinguish between agents that learn robustly from brittle memorization-based agents.

Recognizing this limitation in earlier versions of the ALE, many researchers have augmented the standard behavior of the ALE to evaluate the robustness of their agents and to discourage memorization using different approaches [Brockman *et al.*, 2016; Hausknecht and Stone, 2015; Mnih *et al.*, 2015]. Again, this wide range of experimental protocols makes direct comparison of results difficult. We believe the research community would benefit from a single standard protocol to evaluate the robustness of the learned policies. In this section we introduce the *sticky actions* method for injecting stochasticity into the ALE and show that it effectively distinguishes algorithms that explicitly exploit the environment's determinism from methods that learn more robust policies.

4.1 The Brute

The Brute is an algorithm designed to exploit the determinism of the original ALE [Bellemare *et al.*, 2015]. It uses the agent's trajectory as state representation, assigning individual values to each state. Because of the ALE's determinism, a single sample from each state-action pair is sufficient for a perfect estimate of the agent's return up to that point. The

Game	The Brute		Sarsa(λ) + Blob-PROST		DQN	
	Deterministic	Stochastic	Deterministic	Stochastic	Deterministic	Stochastic
ASTERIX	6909 (1018)	308 (31)	4173 (872)	3411 (414)	3501 (420)	3123 (96)
BEAM RIDER	1132 (178)	428 (18)	2098 (508)	1851 (407)	4687 (704)	4552 (849)
FREEWAY	1.1 (0.4)	0.0 (0.0)	32.1 (0.4)	31.8 (0.3)	32.2 (0.1)	31.6 (0.7)
SEAQUEST	621 (192)	81 (7)	1340 (245)	1204 (190)	1397 (215)	1431 (162)
SPACE INVADERS	1432 (226)	148 (11)	723 (86)	583 (31)	673 (18)	687 (37)

Table 1: Performance of different algorithms in the deterministic ($\varsigma = 0.0$) and stochastic ($\varsigma = 0.25$) ALE. We report the average over 24 trials for Sarsa(λ) + Blob-PROST and the Brute, and the average over 5 trials for DQN. Standard deviation over trials is within parentheses.

Brute maintains a partial history tree that contains all visited histories. Each node, associated with a history, maintains an action-conditional transition function and a reward function. The Brute estimates the value for any history-action pair using bottom-up dynamic programming. The agent follows the best trajectory found so far, with infrequent random actions used to search for better trajectories. Further details are available in the longer version of this paper [Machado *et al.*, 2018].

4.2 Sticky Actions

Sticky actions is our approach to injecting stochasticity into the ALE. In sticky actions there is a *stickiness* parameter ς, the probability at every time step that the environment will execute the agent's previous action again, instead of the agent's new action. More specifically, at time step t if the agent decides to execute action a it will only execute a with probability $1 - \varsigma$, otherwise it will execute its previous action, a_{t-1}. In other words, if $\varsigma = 0.25$, there is 25% chance the environment will not execute the desired action right away.

Notice that sticky actions interplay well with other aspects of the ALE. Most Atari 2600 games are deterministic and it is very hard to change their dynamics. Our approach only impacts which actions are sent to be executed. Sticky actions also interacts well with frame skipping. With sticky actions, at each intermediate time step between the skipped frames there is a probability ς of executing the previous action. Obviously, this applies until the current action is executed, when the previous and current action taken become the same.

Evaluating the Effectiveness of Sticky Actions

In order to evaluate the effectiveness of sticky actions we evaluated the performance of the Brute, Sarsa(λ), and DQN on the five training games proposed by Bellemare *et al.* [2013]. The average score obtained by these methods when sticky actions were disabled (Deterministic) and enabled (Stochastic, $\varsigma = 0.25$) are presented in Table 1. Agents interacted with the environment for 50 million frames and the numbers reported are the average scores agents obtained in the last 100 episodes played while learning.

The Brute is crude but we see that it leads to competitive performance in a number of games. In fact, Bellemare *et al.* [2015], using a different evaluation protocol, report that the Brute outperformed the best learning method at the time on 45 out of 55 Atari 2600 games. However, as we will see, this performance critically depends on the environment's determinism. Sarsa(λ) and DQN, which are not built under the assumption that the environment is deterministic, are much more robust to small random perturbations. These results suggest that sticky actions enable us to empirically evaluate an agent's robustness to perturbation. In fact, such an approach is already being used in newer testbeds for general competency [Nichol *et al.*, 2018].

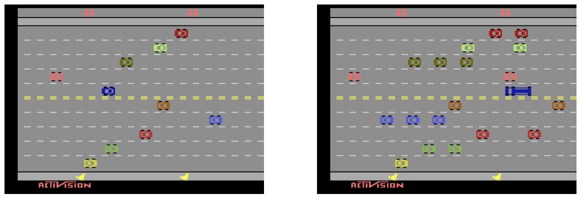

(a) Mode 1 (b) Mode 2

Figure 2: Two of the different modes of the game FREEWAY.

5 Modes and Difficulties in the ALE

Originally, many Atari 2600 games had a default game mode and difficulty level that could be changed by changing physical switches on the console. These mode/difficulty switches had different consequences such as changing the game dynamics or introducing new actions (see Figure 2). Until recently, the ALE allowed agents to play games only in their default mode and difficulty. The newest version of the ALE now also allows one to select among all different game modes and difficulties that are single player games. This new feature opens up research avenues by introducing dozens of new environments that are very similar. This new feature could potentially be used to further our understanding of different topics in RL such as generalization and transfer learning.

6 Conclusion

In this article we focused on discussing the different evaluation protocols that have been employed in the ALE and how they have been frequently conflated in the literature. To further the progress in the field, we presented some methodological best practices and a new version of the ALE that supports stochasticity and multiple game modes. We hope such methodological practices, with the new ALE, allow one to clearly distinguish between the different evaluation protocols.

In the extended version of this paper [Machado *et al.*, 2018], aside from providing a much deeper discussion on the issues aforementioned, we provide benchmark results following the methodological best practices discussed here. We evaluated RL algorithms that use linear [Liang *et al.*, 2016] and non-linear [Mnih *et al.*, 2015] function approximation, also promoting a discussion about sample efficiency by reporting algorithms' performance at different moments of the learning period. Finally, in its extended version we also revisit the challenges posed in the ALE's original article, highlighting some of the problems we consider to remain open and surveying the current state-of-the-art in each one of them.

References

[Albrecht *et al.*, 2015] Stefano V. Albrecht, J. Christopher Beck, David L. Buckeridge, Adi Botea, Cornelia Caragea, Chi-Hung Chi, Theodoros Damoulas, Bistra N. Dilkina, Eric Eaton, Pooyan Fazli, Sam Ganzfried, Marius Thomas Lindauer, Marlos C. Machado, Yuri Malitsky, Gary Marcus, Sebastiaan Meijer, Francesca Rossi, Arash Shaban-Nejad, Sylvie Thiébaux, Manuela M. Veloso, Toby Walsh, Can Wang, Jie Zhang, and Yu Zheng. Reports on the 2015 AAAI Workshop Program. *AI Magazine*, 36(2):90–101, 2015.

[Beattie *et al.*, 2016] Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. DeepMind Lab. *CoRR*, abs/1612.03801, 2016.

[Bellemare *et al.*, 2012] Marc G. Bellemare, Joel Veness, and Michael Bowling. Investigating Contingency Awareness using Atari 2600 Games. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 864–871, 2012.

[Bellemare *et al.*, 2013] 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*, 47:253–279, 2013.

[Bellemare *et al.*, 2015] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents (Extended Abstract). In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4148–4152, 2015.

[Bellemare *et al.*, 2016] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1471–1479, 2016.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *CoRR*, abs/1507.04296, 2016.

[Hausknecht and Stone, 2015] Matthew Hausknecht and Peter Stone. The Impact of Determinism on Learning Atari 2600 Games. In *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[Hausknecht *et al.*, 2014] Matthew J. Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. A Neuroevolution Approach to General Atari Game Playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):355–366, 2014.

[Johnson *et al.*, 2016] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The Malmo Platform for Artificial Intelligence Experimentation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4246–4247, 2016.

[Liang *et al.*, 2016] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael H. Bowling. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pages 485–493, 2016.

[Machado *et al.*, 2018] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[Mnih *et al.*, 2015] 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(7540):529–533, 02 2015.

[Nichol *et al.*, 2018] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta Learn Fast: A New Benchmark for Generalization in RL. *CoRR*, abs/1804.03720, 2018.

[Ring, 1997] Mark B. Ring. CHILD: A First Step Towards Continual Learning. *Machine Learning*, 28(1):77–104, 1997.

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[Stadie *et al.*, 2015] Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing Exploration in Reinforcement Learning With Deep Predictive Models. *CoRR*, abs/1507.00814, 2015.

[van Hasselt *et al.*, 2016] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 2094–2100, 2016.