MinAtar: An Atari-inspired Testbed for More Efficient Reinforcement Learning Experiments

Kenny Young
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
kjyoung@ualberta.ca

Tian Tian
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
ttian@ualberta.ca

Abstract

The Arcade Learning Environment (ALE) is a popular platform for evaluating reinforcement learning agents. Much of the appeal comes from the fact that Atari games are varied, showcase aspects of competency we expect from an intelligent agent, and are not biased towards any particular solution approach. The challenge of the ALE includes 1) the representation learning problem of extracting pertinent information from the raw pixels, and 2) the behavioural learning problem of leveraging complex, delayed associations between actions and rewards. Often, in reinforcement learning research, we care more about the latter, but the representation learning problem adds significant computational expense. In response, we introduce MinAtar, short for miniature Atari, a new evaluation platform that captures the general mechanics of specific Atari games, while simplifying certain aspects. In particular, we reduce the representational complexity to focus more on behavioural challenges. MinAtar consists of analogues to five Atari games which play out on a 10x10 grid. MinAtar provides a 10x10xn state representation. The n channels correspond to game-specific objects, such as ball, paddle and brick in the game Breakout. While significantly simplified, these domains are still rich enough to allow for interesting behaviours. To demonstrate the challenges posed by these domains, we evaluated a smaller version of the DQN architecture. We also tried variants of DQN without experience replay, and without a target network, to assess the impact of those two prominent components in the MinAtar environments. In addition, we evaluated a simpler agent that used actor-critic with eligibility traces, online updating, and no experience replay. We hope that by introducing a set of simplified, Atari-like games we can allow researchers to more efficiently investigate the unique behavioural challenges provided by the ALE.

Keywords: Reinforcement Learning, Evaluation Environment

Acknowledgements

The authors would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Alberta Innovates. We would also like to thank Martha White, Andy Patterson and the rest of the University Alberta RLAI group for helpful comments and feedback.

1 Motivation

The arcade learning environment (Bellemare, Naddaf, Veness, & Bowling, 2013) (ALE) has become widely popular as a testbed for reinforcement learning (RL), and other AI algorithms. An important aspect of the ALE's appeal is that the environments are designed to be interesting for human players, and not to be amenable to any particular approach to AI. Because of this design, the platform is largely free of experimenter bias and provides diverse challenges which we associated with the kind of general intelligence seen in humans.

The challenges provided by the ALE can be broadly divided into two aspects: 1) the representation learning problem of extracting pertinent information from the raw pixels, and 2) the behavioural learning problem of leveraging complex, delayed associations between actions and rewards.

While it is important to have testbeds that provide this kind of broad-spectrum challenge, it is not always what we want as experimenters. Often, the work flow when evaluating a new RL idea is to first experiment with very simple domains, such as *Mountain Car* or a tabular MDP, then jump to complex domains, like those provided by the ALE, to validate the intuition. We believe that this jump tends to leave a wide gap in understanding that would be best filled by domains of intermediate complexity.

MinAtar is intended to bridge this gap by providing environments designed to capture the spirit of specific Atari 2600 games, while simplifying certain aspects. One aspect of the ALE that makes it difficult to use as an RL testbed is that much of the computing power expended to train a deep RL agent goes toward learning a semantically meaningful representation from the raw pixel input. When the first deep RL agents were shown to succeed in the ALE, this was an interesting challenge. Presently, however, this challenge is usually addressed with some variant of convolutional neural network and often the interesting research questions come not from this visual representation learning problem, but instead from the higher level behavioural challenges involved in the various games. A major goal of MinAtar is, thus, to reduce the complexity of this representation learning problem while maintaining the mechanics of the original games as much as possible. While our simplification also reduces the behavioural complexity of the games, the MinAtar environments are still rich enough to showcase interesting behaviours, similar to those observed in the ALE.

We emphasize that MinAtar is not a challenge problem, like Go, StarCraft (Vinyals et al., 2017) or the ALE when it was first introduced. The purpose is to serve as a more efficient way to validate intuition, and provide proof of concept for artificial intelligence ideas, which is closer to how the ALE is often used today.

2 The MinAtar Platform

Aside from replicating the spirit of a set of Atari 2600 games, the design goals of MinAtar can be broken down as follows:

- **Reduce spatial dimension:** In MinAtar, each game takes place on a 10x10 grid. This is a significant reduction from the Atari 2600 screen size of 160x210. Often, in the ALE, the input to the learning agent is down-sampled. For example, Mnih et al. (2015) downsample to 64x64. MinAtar provides a much smaller input without the need for this step.
- **Reduce action space:** In MinAtar, the action space consists of moving in one of the 4 cardinal directions, firing, or no-op. This makes for a total of just 6 actions. On the other hand, in the ALE, it is possible to move in 8 directions or stand still. For each of these choices, the player can also either fire or not fire. This makes for a total of 18 actions.
- **Provide semantically meaningful input:** Instead of raw color channels, each MinAtar environment provides a number of semantically meaningful channels. For example, for the game *Breakout*, MinAtar provides channels for *ball*, *paddle* and *brick*. The total number of such channels is game-dependent. The state provided to the agent consists of a stack of 10x10 grids, one for each channel, giving a total dimensionality of 10x10xn where n is the number of channels.
- **Reduce partial observability:** Many games in the ALE involve some benign form of partial observability. For example, the motion direction of objects is often not discernible from a single frame. Techniques like frame stacking (Mnih et al., 2015) reduce such partial observability. Minitar mitigates the need for such techniques by making the motion direction of objects discernible within a single frame. Depending on the situation, we convey motion direction either by providing a *trail* channel indicating the last location of certain objects, or by explicitly providing a channel for each possible direction of motion. We do not aim to eliminate partial observability, but merely mitigate the more trivial instances.
- **Simplify certain game mechanics:** Reduction to a 10x10 grid means that some of the more nuanced mechanics of certain Atari 2600 games are difficult or impossible to replicate. Other mechanics we left out for simplicity. For example, in *Space Invaders* we do not include the destructible defence bunkers or the mystery ship which

periodically crosses the top of the screen. We also limit each game to one life, terminating as soon as the agent dies.

- **Add stochasticity:** The Atari 2600 is deterministic. Each game begins in a unique start state and the outcome is uniquely determined by the action sequence that follows. This deterministic behaviour can be exploited by simply repeating specific sequences of actions, rather than learning policies that generalize. Machado et al. (2017) address this by adding *sticky-actions*, where the environment repeats the last action with probability 0.25 instead of executing the agent's current action. We incorporate sticky-actions in MinAtar, but with a smaller probability of 0.1. This is based on the assumption that individual actions have a larger impact in MinAtar than in the ALE due to the larger granularity of the movement discritization, thus each sticky-action can have a potentially larger negative impact. In addition, we make the spawn location of certain entities random. For example, in *Seaquest* the enemy fish, enemy submarines and divers emerge from random locations on the side of the screen.

So far, we have implemented five games for the MinAtar platform. Visualizations of each of these games are shown in Figure 1. MinAtar is available as an open-source python library under the terms of the GNU General Public License. The source code is available at:

`https://github.com/kenjyoung/MinAtar`

You can find links to videos of trained DQN agents playing the MinAtar games in the README.

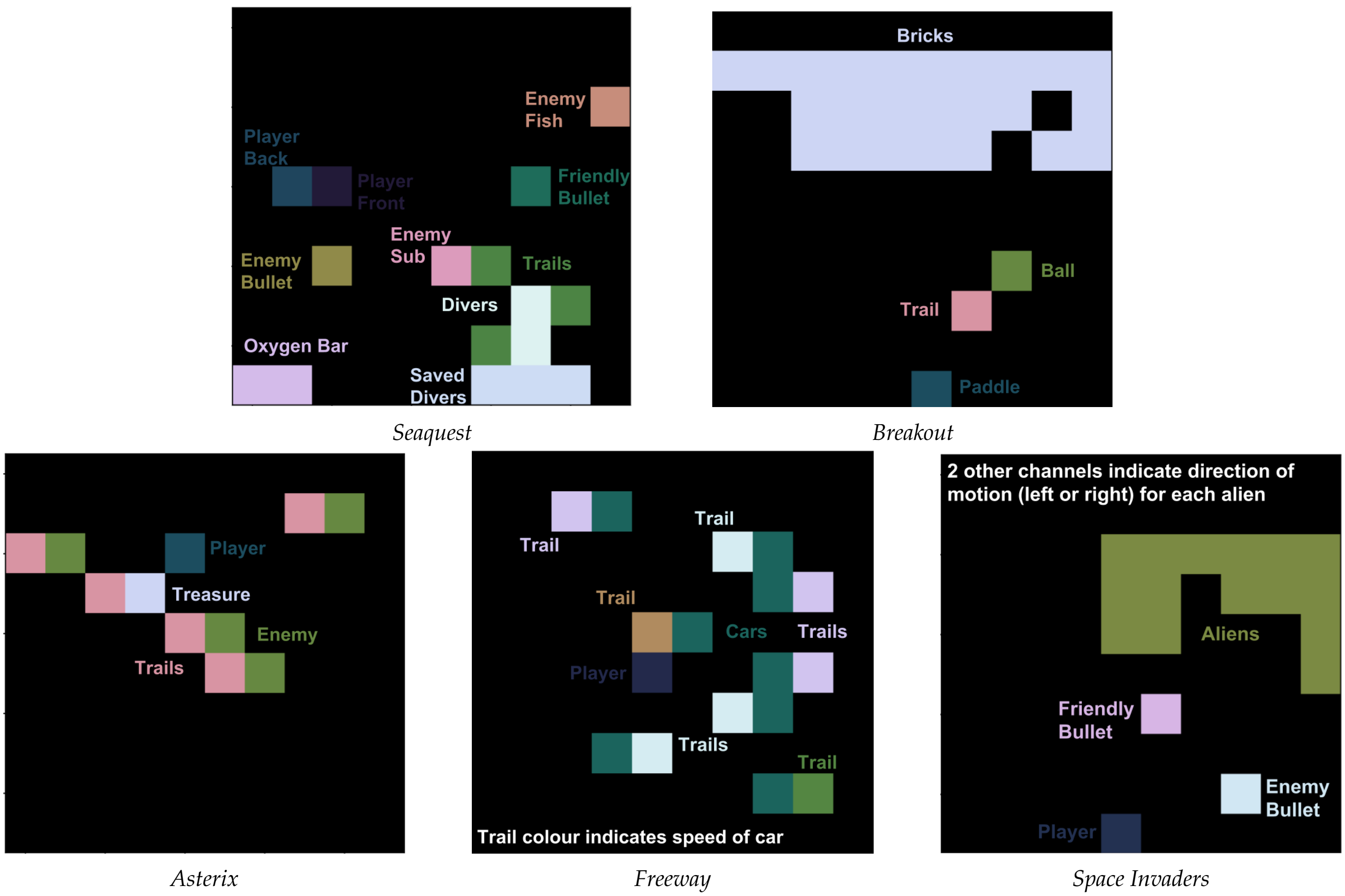

Figure 1: Visualization of each MinAtar game. Colour indicates active channel at each spatial location, but note that the representation provided by the environment consists of binary values for each channel and not RGB values.

3 Experiments

We provide results for several variants of two main algorithms on each of the five MinAtar environments. We trained each agent for a total of 5 million frames, compared to 50 million for Mnih et al. (2015) (200 million if you count in terms of emulator frames since they use frame-skipping). The reduction in the number of frames allowed us to run more repeats without inordinate expense. We were able to train 30 different random-seeds per agent-environment combination to obtain results with tighter confidence intervals. These results serve to provide a baseline for future work, as well as to illustrate the challenge posed by these new environments.

Deep Q-Network

Our DQN architecture consisted of a single convolutional layer, followed by a fully connected hidden layer. Our convolutional layer used 16 3x3 convolutions with stride 1, while our fully connected layer had 128 units. 16 and 128 were chosen as one quarter of the final convolutional layer and fully connected hidden layer respectively of Mnih et al. (2015). We also reduced the replay buffer size, target network update frequency, epsilon annealing time and replay buffer fill time, each by a factor of ten relative to Mnih et al. (2015) based on the reasoning that our environments take fewer frames to master than the original Atari games. We trained on every frame and did not employ frame skipping. The reasoning behind this decision is that each frame of our environments is more information rich. Other hyperparameters, including the step-size parameter, were set to match Mnih et al. (2015). We also tested variants of the DQN architecture without experience replay and without a target network to assess the usefulness of these components in the simpler environments.

The smaller architecture and input size means that running on CPU instead of GPU was feasible. For DQN with experience replay running on *Seaquest*, the total wall-clock time per frame was around 8 milliseconds when running on a single CPU, compared to 5 milliseconds when running on GPU. We report these times for *Seaquest* because it has the largest number of input channels and thus the largest number of network parameters.

Actor-Critic with Eligibility Traces

We also experimented with an online actor-critic with eligibility traces (AC(λ)) agent (Degris, Pilarski, & Sutton, 2012; Sutton & Barto, 2018). This agent used no experience replay or multiple parallel actors. We used a similar architecture to the one used in our DQN experiments, except that we replaced the relu activation functions with the SiLU and dSiLU activation functions introduced by Elfwing, Uchibe, and Doya (2018). Their work showed these activations to be helpful when using online eligibility traces with nonlinear function approximation. Specifically, we applied SiLU in the convolutional layer and dSiLU in the fully connected hidden layer. We set the step-size to 2^{-8}, the largest value that was found to yield stable learning across games by Young, Wang, and Taylor (2018), who used online AC(λ) to train a similar architecture for several ALE games. We tested AC(λ) with trace decay parameter of 0.8 and 0, where the latter corresponds to one-step actor-critic with no eligibility trace.

Discussion

The results of our experiments are shown in Figure 2. The first thing to note is that the MinAtar environments clearly show the advantage of using experience replay in DQN. On the other hand, the target network appeared to have little performance impact. To verify that the poor performance of DQN without experience replay was not due to a poorly tuned step-size parameter, we tried running DQN without experience replay with various values of the step-size on *Seaquest*. We choose *Seaquest* for the step-size sweep because it showed the largest discrepancy in results with and without experience replay. Specifically, we tried 30 random-seeds with each value of the step-size from the set $\{\alpha_0 \cdot 2^i | i \in \{1, 0, ..., -4\}\}$, where the original value was $\alpha_0 = 0.00025$. We swept primarily lower values, reasoning that the lack of batching would lead to higher variance, potentially requiring a lower step-size relative to DQN with experience replay. For all the step-size choices, none of the average returns over the final 100 training episodes were above 1.0.

DQN significantly outperformed the simpler AC(λ) agent in 3 of the games. In each of these 3 games, AC(λ) barely learned at all. On the other hand, perhaps surprisingly, AC(λ) significantly outperformed DQN at *Space Invaders*. The two performed similarly at *Breakout*, though DQN converged faster to it's final performance. To verify that the AC(λ)'s poor performance was not due to a poorly tuned step-size, we performed a step-size sweep on *Seaquest* running AC(0.8). Specifically, we tried 30 random-seeds with each value of the step-size from the set $\{\alpha_0 \cdot 2^i | i \in \{-1, 0, ..., 4\}\}$, where $\alpha_0 = 2^{-8}$ is the original AC(λ) step-size. We swept primarily higher step-size values, reasoning that the initial step-size, chosen for stability on ALE games, was potentially unnecessarily low for some MinAtar games. With the best step-size of $\alpha_0 \cdot 2^3$, we observed performance improvement of AC(λ), achieving a final average return of 4.4 ± 0.6 over the final 100 episodes in 5 million training frames. However, this was still far below the performance of DQN with experience replay.

Taken together, these results suggest that the MinAtar environments are effective at highlighting the strengths and weaknesses of different approaches.

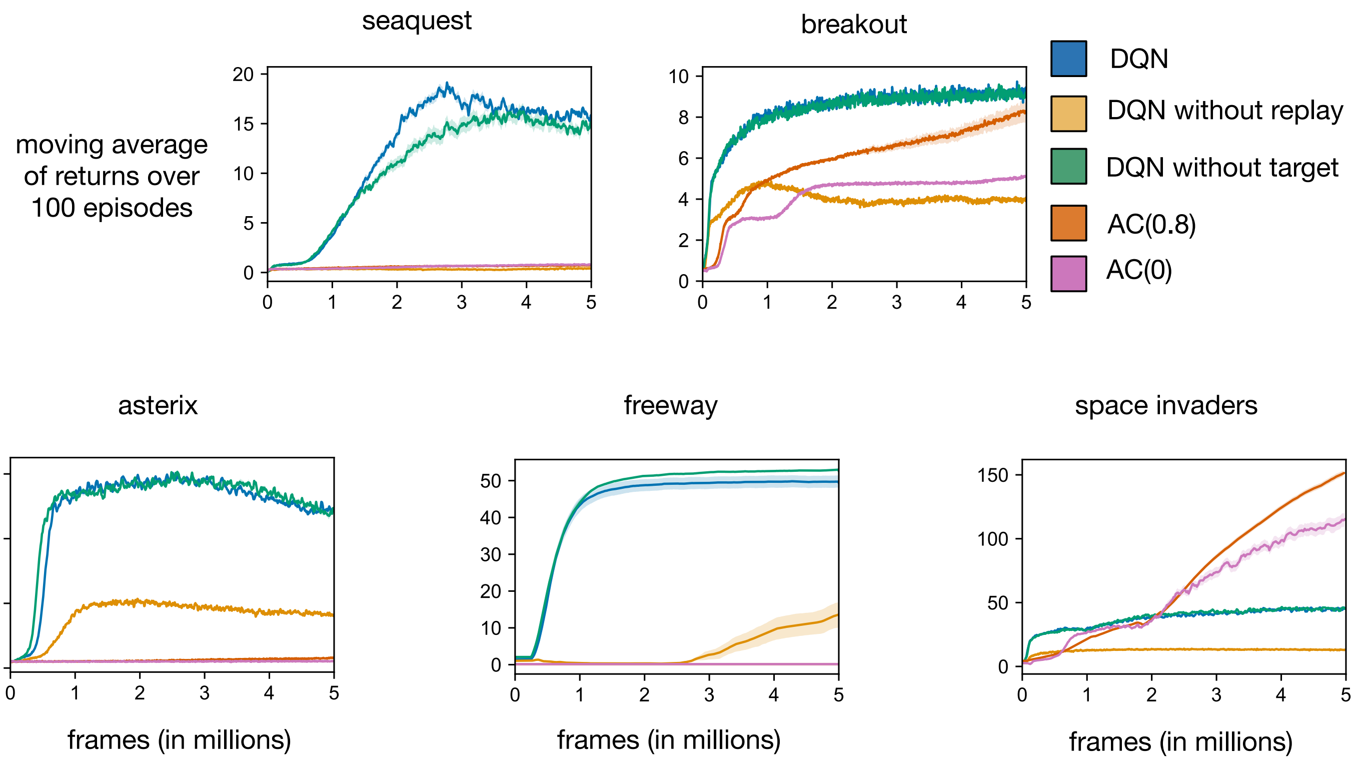

Figure 2: Average return v.s. training frames for all games and agents.

4 Conclusion

We introduce MinAtar, a new evaluation platform for reinforcement learning designed to allow for more efficient experiments by providing simpler versions of Atari 2600 games. These environments aim to reduce the representation learning burden to focus more on interesting behavioral aspects of the games, which are often of greater interest to RL experimenters. Currently the platform consists of five environments. In the future, we plan to add more. Of particular interest are the games typically considered hard exploration problems, such as *Montezuma's Revenge* and *Pitfall*. It would also be interesting to explore how other DQN additions, such as double DQN, perform in the MinAtar environments.

References

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Degris, T., Pilarski, P. M., & Sutton, R. S. (2012). Model-free reinforcement learning with continuous action in practice. In *American control conference (acc), 2012* (pp. 2177–2182).

Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2017). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv preprint arXiv:1709.06009*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... others (2017). StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Young, K., Wang, B., & Taylor, M. E. (2018). Metatrace: Online step-size tuning by meta-gradient descent for reinforcement learning control. *arXiv preprint arXiv:1805.04514*.