CS 394R: Final Project - Playing SpaceInvaders

Chloe Chen
EID: ctc2688

Logan Persyn
EID: LMP3328

Abstract

Atari 2600 games have been extensively studied in reinforcement learning. In this project, we seek to explore the effects of state space on agent's performance in the SpaceInvaders game. Agents are trained and evaluated on the RAM state space, the original pixel state space, and noisy augmented state space. Soft Actor-Critic (SAC) and Rainbow algorithm are used to train the agents. Our results show that agents trained in noisy environments are able to achieve comparable performances as and in some cases outperformed their counterparts that are trained in normal environments, demonstrating robustness and effectiveness in handling environmental noise. In addition, the agent is able to learn to play the game better than a random agent using the RAM state space, showing that minimal information can be enough in order to achieve decent performance. However, the agents struggle to generalize to different random seeds in noisy environment experiments, and further research is needed to understand the effects of different noise settings on generalization.

1 Introduction

Reinforcement learning excels at mapping situations to actions to maximize a numerical reward signal and has been successfully applied to various domains that require machine learning agents to explore and discover the optimal actions given the current state. Many reinforcement learning methods have been applied to Atari 2600 games. It is a great setting for testing reinforcement learning agents due to accessibility, wide range of difficulty levels, and the environments' sequential nature.

We plan to implement an agent to play the SpaceInvaders Atari game included in Gymnasium API (Towers et al., 2023). SpaceInvaders is an arcade game where the player wards off alien invaders by shooting a laser cannon that moves horizontally across the screen. The player's actions directly affect the next state, making the game a sequential environment. The API provides two options for state space representation out of the box: the pixel values of each frame of the game and the current state of the Atari's RAM. There are 4 legal actions that the player can take: do nothing, move left, move right, and fire. The goal of the game is to achieve the highest possible score by destroying the invaders. Each variant of invaders has a different score value; for example, the invaders in the farther rows are worth more points than those in the closer rows. In our experimental design, the scores obtained from hitting the invaders are the reward signals. The game is lost when your spaceship is destroyed and you have run out of lives or when the invaders have reached Earth.

Given that the original state space is comprised of screen pixels and thus quite large, our intuition is that learning a policy through policy gradient methods will be quicker. Both a soft actor-critic and Rainbow model are used. The goal of this project is to evaluate the performance an agent can achieve on three different state space representations: RAM-based, the original screen, and a noisy screen. Our code is available on GitHub: https://github.com/lpersyn/CS394R_Final. A demo video can be found here: https://youtu.be/ae6hiMKkm5w

2 Related Work

This project was inspired by the work of Bellemare et al. (2013), which introduced a framework for developing agents that play Atari 2600 games and what the Gymnasium API we will be using is based on (Towers et al., 2023). Bellemare et al. (2013) provided benchmark results of an agent trained on a SARSA(λ) algorithm and evaluated its performance on SpaceInvaders, on five sets of features with linear function approximation. Of the five sets of features used, the RAM method observed Atari's 1024 bits of memory, which we seek to replicate in our first goal. In addition, (Bellemare et al., 2013) laid the foundation for many other research projects that we took inspiration from.

To construct the function approximators that represent the policy and both Q-functions in Soft Actor-Critic and the DQN in Rainbow, we will use a similar architecture to Mnih et al. (2013). Mnih et al. (2013) used convolutional neural networks trained using "Deep Q-Learning with Experience Replay". They chose this method of offline learning with a replay buffer in contrast to regular online Q-learning for two reasons: the replay buffer allows data to be used multiple times for more efficiency and learning from non-sequential samples reduces the variance of model updates. Mnih et al. (2015) is a continuation of the work done in Mnih et al. (2013). To expand the uses of reinforcement learning agents to domains, a deep neural network was used to develop a deep Q-network that could learn directly from high-dimensional inputs. The deep Q-network agents were tested on Atari 2600 games. Receiving only the raw pixels and current game scores, the agents were able to achieve better performance than human experts on the majority of the 49 games of Atari 2600. In a comparison of learning algorithms on the ALE, Defazio and Graepel (2014), an actor-critic method was among the top models which guided our choice in selecting Soft Actor-Critic for one of our learning algorithms. We also chose to test a Rainbow method based on the results from Hessel et al. (2017).

A main challenge of learning to play Atari 2600 games is the high-dimensional state space. Simplifying the state space is a common way to combat this problem. Young and Tian (2019) presented MinAtar, a set of simplifications of some of the environments in ALE (Bellemare et al., 2013). Extracting useful features from the raw pixel inputs is a computationally expensive process, which Young and Tian (2019) sought to address. MinAtar featured a reduced spatial dimension and action space as well as simplified rewards and game mechanics and more. To evaluate Young and Tian (2019)'s simplification of the Atari 2600 environments, an agent trained with an online actor-critic with eligibility traces algorithm was tested. The results showed that the simplified environment was able to capture key features of the original state space. Although MinAtar simplified more than just the state space and did not utilize the RAM-based environment, it supports our idea that an agent can learn with minimal information and gives a strong baseline for what our agent might be able to achieve. Similarly, Buesing et al. (2018) designed state-space models, a compact state representation, for model-based learning. It established that state-space models accurately represented the dynamics of Atari 2600 games. While we do not plan to use model-based methods, the results supported that a simplified state space could be both accurate and computationally efficient.

While deep reinforcement learning has seen great success in classic video games, modern video games with higher visual complexity remains a challenge. Kich et al. (2022) presented an image preprocessing pipeline that included grayscaling, cropping, and binary filtering, which allowed them to extract important information from a high-dimensional space to a lower-dimensional space. It is generally difficult to train agents using high-dimensional input space efficiently. This approach enabled the agent to perform well in visually complex games using images only as inputs. Laskin et al. (2020) introduced RAD

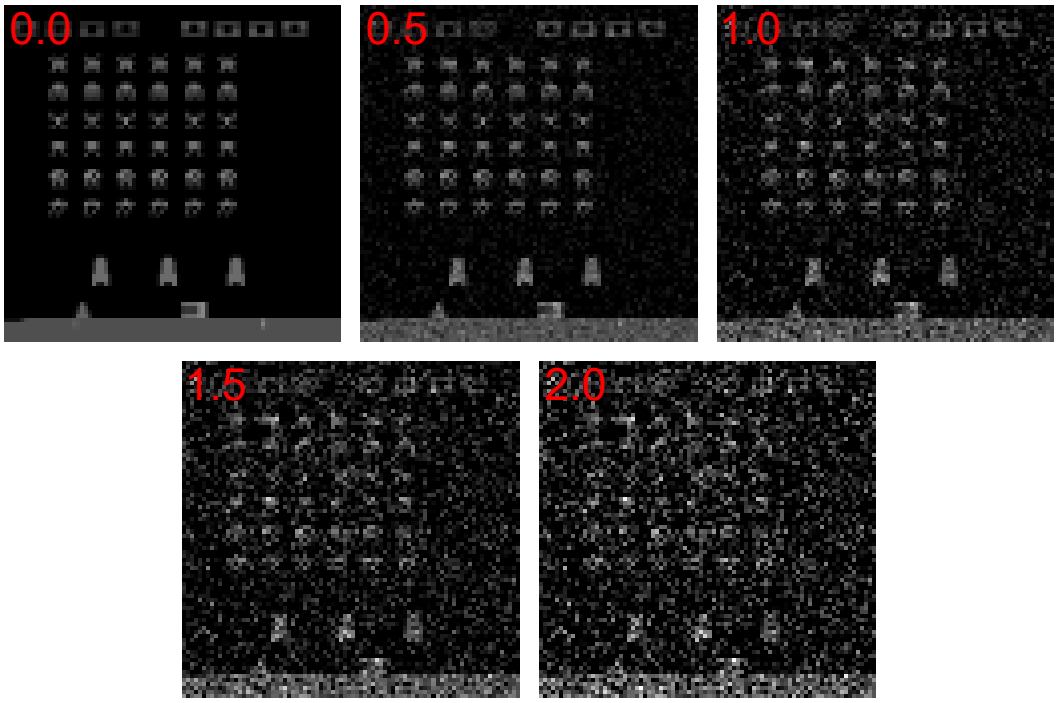

Figure 1: Examples of input images to the CNN. The respective "noise weight" of added white noise is in red.

(Reinforcement Learning with Data Augmentation). RAD includes many image augmentation techniques such as crop, rotate, translate, grayscale and color jitter, as well as random amplitude scaling and added Gaussian noise. Augmenting the input images allows for better generalization, for example the color of an obstacle is less important than its shape. Additionally, augmenting the same image multiple times allows for better data efficiency. Both of these data augmentation techniques showed improved performance and data efficiency over no image preprocessing. Instead of training agents directly on augmented data, we plan to test how augmentation affects the performance of our agents that are trained on the original state space. We hypothesize that the agent will perform worse initially but eventually achieve better performance.

Lastly, Machado et al. (2018) presented the challenge of transfer learning. Many Atari games have similar dynamics. If we could leverage knowledge transfer between similar games, it could reduce the number of samples needed to learn additional games. One candidate for transfer learning is Demon Attack which shares similar game mechanics with SpaceInvaders but with a wider range of rewards as well as a larger action space.

3 Methods

3.1 RAM State Space

First, we trained and evaluated an agent's performance on a simple state space: a RAM-based environment that fully encodes each state provided by the Gymnasium API. The size of the state space is then reduced from (210, 160, 3) to (128,). Learning from this state representation should be an easier goal than learning from pixels. This experiment evaluates how well the agent performs on minimal information. In addition, the reduced complexity of the state space allows us to use simpler methods when constructing the agent. The results should give us a strong baseline to compare our more complex agents and problem settings to. quantumiracle (2024)'s code was modified to train the agent on this task for 500 episodes.

3.2 Original Pixel State Space

Then, we trained an agent on the original screen state space, which in raw form is an

RGB image of dimension (210, 160, 3). As in Mnih et al. (2013) the environment is altered in the following ways. The image is first processed by gray-scaling, down sampling, and cropping to the size (84, 84, 1). Additionally, the past four frames are stacked for a final size of (84, 84, 4), all rewards clipped to the range [-1,1], and transitions that cause a life to be lost are recorded as the end of an episode. Tianshou's implementation of Soft Actor-Critic (SAC) and Rainbow (based on Pytorch (Paszke et al., 2019)) are used to construct and train the agents (Weng et al., 2022). We used the default model architectures and hyperparameters provided by Tianshou which are based on the algorithm's respective paper, Christodoulou (2019) for SAC and Hessel et al. (2017) for Rainbow. All training was done on TACC.

3.3 Augmented State Space

Finally, to test the agent's robustness to variations in the screen, we evaluated a trained agent on a noisy environment. White noise is added to the image after the gray-scaling, down sampling, and cropping step. The amount of noise added can be varied by a "noise weight" hyperparameter (see Figure 1). We then trained a new agent on the noisy environment to test if the agent can learn to overcome the added challenge of the noise and if it can generalize back to a normal (non-noisy) environment. The new noisy-trained agent used the same hyperparameters and architecture as the non-noisy-trained model. All training was also done on TACC.

4 Results

4.1 RAM model

The agent was trained using the RAM state space provided by Gymnasium for 500 episodes and a maximum of 10,000 steps per episode. We were able to obtain an average reward of 272.98, which was significantly better than the results of an agent following a random policy on the original state space (achieved an average reward of 158.75). The performance of the agent fluctuated but mostly plateaued at around 270 about 10 episodes in (through 500 episodes, the agent achieved a maximum score of 800 and a minimum score of 70). This suggests that the agent was able to learn to play the game fairly quickly compared to image-based models albeit at the cost of performance (see Section 4.2).

4.2 Image-based models

Both SAC and Rainbow were trained for 5 hours of compute time on TACC. Rainbow was able to train quicker, finishing 8.2 million training steps while SAC was only able to complete 5.1 million in the same time. Due to time constraints and availability of compute resources, we were unable to train SAC more. See Figure 2 for mean rewards of test episodes during training. Despite being trained on a noisy environment, both SAC and Rainbow were able to achieve comparable scores when evaluated on the same noisy environment to their counterparts trained and evaluated on a normal environment.

Additionally, we evaluated both noisy- and normal-trained versions of SAC and Rainbow on a range of noise weights (See Figure 3). Each noise weight was evaluated over 100 episodes. For lower noise weights, the noisy versions of SAC and Rainbow performed slightly better. However, for higher noise weights noisy SAC performed better than normal SAC, while conversely, normal Rainbow performed better than noisy Rainbow. Also note that the noisy-trained models have roughly the same performance as the normal models on a noise weight of 1.0 despite being trained on that noise weight. This is somewhat unexpected. Although we believe it ultimately came down to a bug in the setting of the random seed, different seeds were used during training than used during the different noise weight evaluations. In both cases, the noisy models were unable to generalize to other random seeds. Unfortunately, more testing was unable to be done but is a direction of future work.

Figure 2: Solid line is reward averaged over 10 episodes, tested every 100,000 training steps. Shaded area represents 1 standard deviation over rewards of 10 test episodes. Models are trained on an environment using the noise weight in the legend.

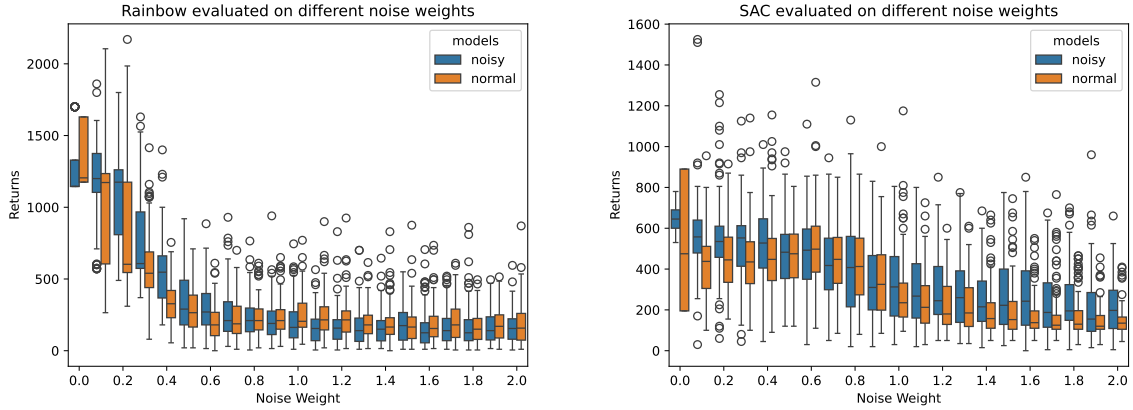

Figure 3: Two Rainbow models and two SAC models, "noisy" trained on an environment with a noise weight of 1.0 and "normal" trained with a noise weight of 0.0. They are evaluated for 100 episodes over a range of noise weights.

5 Future Directions

With more time we would have liked to implement more of the data augmentation techniques as seen in Laskin et al. (2020). Our current implementation simply adds noise to observed images and offers no method to increase sample efficiency by reusing observations with different augmentations. We show the noisy models were able to preform as well as the normal models and they may be able to train faster using the data-efficiency methods in Laskin et al. (2020). Similarly, adding "natural" noise, i.e. video frames, to the background of the images, as seen in Zhang et al. (2018), may have the same effect. Combining both methods would also be an interesting topic to explore.

Additionally, we would like to test the agent on Demon Attack, a similar Atari game to SpaceInvaders also included in the Gymnasium API. Transfer learning between SpaceInvaders and Demon Attack was explored in Machado et al. (2018). The two games are quite similar, however in Demon Attack, the action space is larger with more

movement options. There is also a wider range of rewards depending on the type of demons and which wave the agent in. Some topics we would like to explore are: Would beginning with an agent trained on SpaceInvaders give a "jump start" to an agent trained on Demon Attack, and vice versa? Could a combined agent learn to play both games? And would the data augmentation techniques mentioned above improve generalization and sample efficiency?

Acknowledgments

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. URL: http://www.tacc.utexas.edu

References

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research, vol. 47, pp. 253-279*.

Lars Buesing, Theophane Weber, Sebastien Racaniere, S. M. Ali Eslami, Danilo Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. 2018. Learning and querying fast generative models for reinforcement learning.

Petros Christodoulou. 2019. Soft actor-critic for discrete action settings.

Aaron Defazio and Thore Graepel. 2014. A comparison of learning algorithms on the arcade learning environment.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2017. Rainbow: Combining improvements in deep reinforcement learning.

Victor Augusto Kich, Junior Costa de Jesus, Ricardo Bedin Grando, Alisson Henrique Kolling, Gabriel Vinícius Heisler, and Rodrigo da Silva Guerra. 2022. Deep reinforcement learning using a low-dimensional observation filter for visual complex video game playing.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. 2018. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning.

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. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

quantumiracle. 2024. Popular-rl-algorithms. https://github.com/quantumiracle/Popular-RL-Algorithms.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. 2023. Gymnasium.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. 2022. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6.

Kenny Young and Tian Tian. 2019. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments.

Amy Zhang, Yuxin Wu, and Joelle Pineau. 2018. Natural environment benchmarks for reinforcement learning. *CoRR*, abs/1811.06032.