Reproducibility Report

Lara Ozcelik, Axel Reumann, Caiya Zhang

December 8, 2023

Reproducibility Summary

1 Introduction

The groundbreaking study "Playing Atari with Deep Reinforcement Learning" by Mnih et. al. stands as an important contribution in the domain of deep reinforcement learning, due to it's innovative approach of implementing complex control policies such as Multi-Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN) to handle high-dimensional input data [3]. In this case, leveraging a CNN trained with a deep Q-learning variant i.e. Deep Q-Network (DQN), traversing the Atari 2600 gaming environment purely from pixel data from RGB input images. Moreover, due to the outstanding performance of the model detailed by Mnih et. al., able to outperform human experts in three of the seven games tested, it is clear the immense potential that deep reinforcement learning algorithms like DQN has in handling complex sensory-rich tasks. As a result, our research aims to replicate the results of this study on the Atari 2600 game, Breakout, to validate the performance and explore the intricacies, applicability, and limitations of the DQN model.

2 Scope of reproducibility

Introducing an innovative approach to deep reinforcEment learning, Mnih et. al leveraged Convolutional Neural Networks (CNN) and Q-learning policies to play Atari games based on real-time pixel inputs. Our research aims to explore the impact of ϵ variations in the ϵ-greedy approach to the DQN within the context of the Atari 2600 game Breakout (truncated to one environment due to computational constraints). Designed to delve deeper into some of the key aspects of the DQN, our experiments attempt to evaluate and compare the results of the ϵ-greedy strategy by adjusting the linear step size for exploration rate, varying the final ϵ value, varying the *gamma* values and finally analyzing the Breakout DQN model's performance on other Atari 2600 game environments [3]. Our hypotheses are centered on the premise that these modifications in the epsilon parameters and gaming environments will yield insightful findings regarding the generalizability and robustness of the DQN model.

- Variations in the rate at which ϵ is annealed greatly impact performance and rate of convergence.
- Adjusting final ϵ in ϵ-greedy approach significantly impacts the nature of the DQN model by changing its stochasticity.
- A DQN model trained on Atari 2600 Breakout environment will yield varying degrees of transferability to other Atari 2600 environments such as Pong and Space Invaders.

3 Methodology

In our study, we were not provided directly with the original code produced by Mnih et. al. and instead implemented the DQN using OpenAI's Stable Baselines 3 (SB3) library (which implements based on the specifications of the original code along with a later improvement by Mnih et. al. in 2015), utilizing a CNN policy [3][2]. As such, since Stable Baselines 3 DQN is compatible with PyTorch tensors, we utilized Google Colab's T4 GPU for accessible and efficient computational power, crucial for the extensive training period[4].

3.1 Model descriptions

The foundation of the DQN model lies in the Q-learning algorithm, a form of model-free, off-policy reinforcement learning which seeks to learn a policy which can tell the DQN agent what action to take based on input data. Doing so by optimizing $Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$. It states that the value of taking action a in state s and thereafter following the optimal policy is equal to the expected value of the rewards r plus the discounted value of future rewards, where γ is the discount factor, s' is the next state, and a' are possible actions in the next state. The expectation $\mathbb{E}$ is taken over the state distribution s' that results from the environment $\mathcal{E}$ [3]. Using this as the motivation for the construction of the DQN, we follow the structure outlined by Mnih et. al. such that we initialize the CNN policy for the DQN to process 4 frame input stacks representing the current state of the environment. The structure of such a CNN policy includes three convolutional layers with varying numbers of filters (16, 32, and 64 respectively) and two fully connected layers of 512 units each such that the output layer yields the computed Q-values for each possible action in the game's current state (using rectified linear units-ReLU-as activation functions to reduce the possibility of vanish and exploding gradients during training (spanning 20 epoch of training over 500,000 time steps, saving the model at the end of each epoch).

3.2 Datasets

In the "Playing Atari with Reinforcement Learning" paper, the authors strategically choose the Atari 2600 games implemented in The Arcade Learning Environment (ALE) as their testbed for evaluating the efficacy of the reinforcement learning algorithm that the paper introduces. Atari 2600 is a challenging environment for reinforcement learning due to the high dimensional visual input it provides to the agent. The visual input consists of 210 x 160 RGB video frames at a rate of 60Hz. The Atari 2600 environment is an ideal RL testbed since it offers diverse and complex game tasks intentionally designed for human-level difficulty. Authors of the paper employed a preprocessing step of the raw visual input due to its computational demands. This involved converting the RGB representation to gray-scale, down-sampling it to a 110×84 image, and obtaining the final input by cropping an 84 x 84 region. The neural network agent then learnt directly from the preproccessed visual input, showcasing the ability of the DQN model to extract meaningful representations directly from the visual input. [2]

In our replication, we leveraged the Atari 2600 environments provided by the stable baselines library [4]. We imported our environments directly from there. The original paper conducted experiments on seven Atari games. However, computational constraints led us to reproduce the results specifically on the Breakout environment. To assess the transferability of the DQN model to other environments, we further tested its performance on additional Atari 2600 environments: Pong and Space Invaders. This approach aims to understand the robustness and generalization capabilities of the DQN model.[3]

3.3 Hyperparameters

In order to replicate the outcomes of the original paper, our aim was to adhere closely to the hyperparameters specified in the original work. It's worth noting that the original paper lacks complete details on certain hyperparameters such as the specific learning rate and discount factor γ utilized during training. The authors used a consistent set of hyperparameters and network architecture for training across the different Atari games investigated in their experiments. In our reproduction efforts, we utilized a set of hyperparameters that can be observed in the provided Table 1 in order to reproduce the model introduced by the study.

Replay Buffer Size	10^6
Batch Size	32
Discount Factor γ	0.99
Learning Rate	10^{-4}
Initial ϵ	1
ϵ Fraction	0.1
Final ϵ	0.05

Table 1: Hyperparameters of the DQN Model

In order to assess the model's reproducibility and the importance of different model hyperparameters, we performed different experiments by modifying the hyperparameters of the model. One specific set of hyperparameters that we found to be particularly crucial for the DQN model is related to the ϵ-greedy policy. This policy controls

the balance between exploration and exploitation which is a crucial part of the learning process of reinforcement learning algorithms. In the original paper, the DQN model uses an initial ϵ of 1 which is then linearly annealed to a final ϵ value of 0.05 [3]. As such our hyperparameter finetuning experiments are focused on evaluating the results of varying the rate at which ϵ is linearly annealed and varying the final ϵ value. Additionally, we experimented with the γ hyperparameter which represents the discount factor that determines the importance of immediate and future rewards in Q learning. We experimented with three discount factor values: 0.9, 0.95, and 0.99.

3.4 Experimental setup and code

Initially, we conducted training and testing with the model's original hyperparameters as outlined in Section 3.3. Following this, we performed a series of experiments designed to assess the model's robustness, reproducibility, and applicability, along with testing the significance of various model components. The implementation of the Deep Q-Network (DQN) was done through the use of OpenAI's Stable Baselines 3 library [4]. The first experiment we performed involved experimenting with the ϵ-greedy policy. The next one entailed experimenting with different gamma values, and the final entailed changing the Atari 2600 gaming environment on a DQN trained on Breakout. To evaluate the models in each of the experiments, involved first, training each of the the models on the Atari 2600 Breakout environment, and then running each over a set of 10 episodes/instances of the game environment, storing the rewards earned in each episode to compute maximum reward/in-game score and the average reward over all 10 episodes.

3.4.1 Experiment 1

In this experiment, we aimed to analyze the importance of the annealing rate applied to epsilon. This is a critical hyperparameter influencing the balance between exploration and exploitation in the DQN algorithm which shapes the agent's behavior over the course of training. As such, we investigated the effects of varying the annealing rate in the Atari 2600 Breakout environment by initializing three DQN models such that their annealing rate was set to 0.5, 0.2, and 0.1 respectively such that the default value under the specifications of the paper was 0.1. Additionally, we investigated the effect of varying the final ϵ value in the ϵ-greedy Q-learning algorithm for the DQN model. To do so, we analyzed the results of individual DQN models on the Atari 2600 Breakout environment such that their final ϵ values are set to 0.5, 0.2, and 0.05 respectively, such that the default value for ϵ is set to 0.05 in the original paper by Mnih et. al. [3].

3.4.2 Experiment 2

For this experiment, we investigated the effect of using different discount factors γ for training the DQN model. In order to do so, we analyzed the results of individually trained DQN models with discount factors of 0.9, 0.95, and 0.99 on the Atari 2600 Breakout environment respectively. The discount factor is an important hyperparameter for Q learning based models as it determines how much attention the agent gives to immediate rewards compared to future rewards. The agent gives more attention to future reward values if the discount factor value is close to 1. On the other hand, the agent considers immediate rewards when the discount factor is close to 0.

3.4.3 Experiment 3

In the final experiment, we decided to extend the results of the original paper by evaluating the optimized DQN model, trained on the Atari 2600 Breakout environment, on different Atari 2600 gaming environments. In doing so, we analyzed the performance of the DQN model on three separate Atari 2600 environments: Breakout, Pong, and Space Invaders, chosen to evaluate the generalizability of the model to unseen environments with both similar (Space Invaders) and starkly different (Pong) action spaces to that of the training environment (Breakout).

3.5 Computational requirements

We conducted the experiments on Google Colab utilizing the computational power of T4 GPU. Training a DQN model on the Atari 2600 environment proved to be computationally demanding due to the high-dimensional inputs, model architecture, and long training iterations. Therefore, the DQN model was found to require high computational power for training. The first two experiments involved training the model after modifying the model structure. This increased the computational times. It is important to note that the free access to the T4

GPU on Google Colab posed limitations to the experiments as the available memory was exceeded. Using the T4 GPU provided by Google Colab, each episode of evaluation took between 20 and 25 minutes. Each episode of training took

4 Results

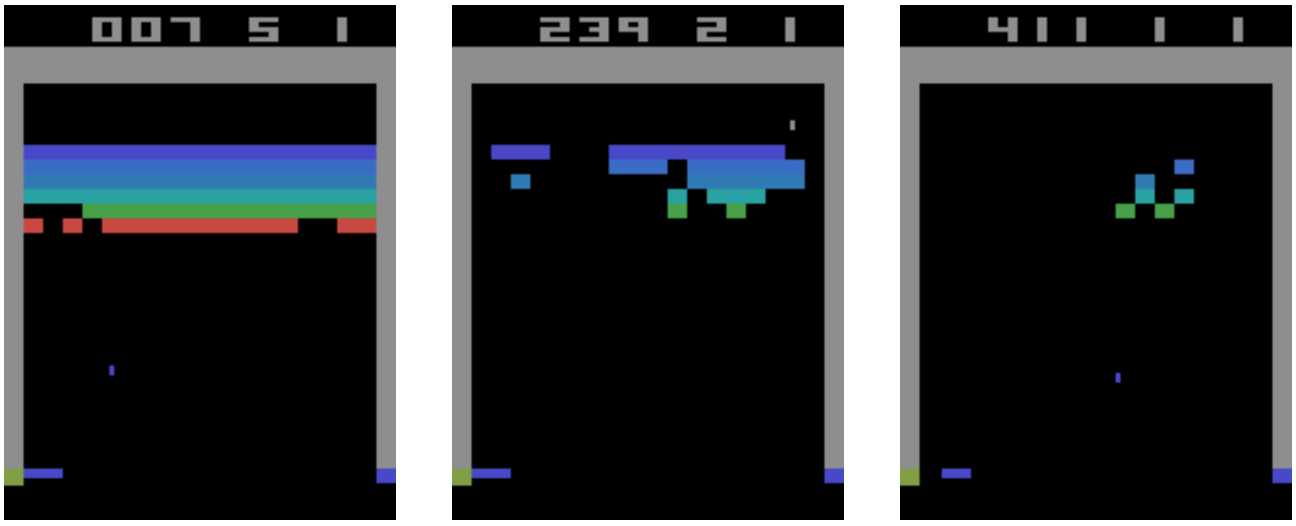
In general, our experimental results have yielded performance consistent with the theoretical expectations of the ϵ-greedy approach in Deep Q-Networks (DQN). As such, the variations in both annealing rate for ϵ and the final ϵ value yielded both runtimes/convergence speeds and maximal rewards indirectly proportional to these increases and decreases in values i.e. increasing annealing rate or ϵ yields shorter convergence speeds, and lower reward values. In contrast, in testing the results of varied Atari 2600 gaming environments on a DQN trained on the Breakout environment, we saw results which were not in line with that of our expectations prior to testing, having reward values much lower than expected for the Space Invaders environment.

4.1 Results reproducing original paper

Considering the results from the paper by Mnih et. al., our reproduction of the DQN on the Atari 2600 Breakout environment yielded results which surpassed the performance presented in Figure 1. Evaluating our model over a total of 10 evaluation episodes, we had mean rewards of 352.3 along with a maximum episode reward being 411 as shown in the Figure 2, underscoring the viability of the DQN algorithm produced by Mnih et. al.

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Figure 1: Results from paper by Mnih et. al. [3]



(a)
b

(b)
b

(c)
b

Figure 2: In-game screenshots of DQN model on Breakout.

4.2 Results beyond original paper

4.2.1 Result 1

Due to the limited computational resources required from evaluating and training DQN models with different ϵ values, we couldn't complete the experiments related to ϵ-greedy policy. However, we found the ϵ values used to highly affect the DQN model's performance by doing further research. In reinforcement learning, the ϵ-greedy policy governs the agent's ability to explore new possibilities and exploit known ones. Exploration involves trying out new actions and dicovering their effects, while exploitation involves using the known information to choose actions in order to maximize rewards. Higher ϵ values utilize more exploration, while the lower ϵ values promote exploitation.[1]

4.2.2 Result 2

Considering that the γ value affects the convergence of the model to an optimal value, we analyzed the effect of different gamma values on the DQN model's training, with all other parameters fixed. We specifically assessed the model's convergence rate with different γ values. For gamma values of 0.9, 0.95, and 0.99, the models demonstrated effective convergence. Specifically, at a gamma value of 0.9, the model converged within approximately 1500 episodes. With a gamma value of 0.95, the convergence occurred around 2500 episodes. However, at a gamma value of 0.99, the model took about 6500 episodes to converge, exhibiting a less stable convergence process. With respect to the observations, it appears that a higher gamma value suggests an emphasis on the agent's focus towards long-term outcomes, as opposed to the immediate future. This shift in focus towards distant rewards tends to make the training process slower and more challenging, as it requires the agent to integrate a broader spectrum of future considerations.

4.2.3 Result 3

Finally, shifting our focus on the influence the environment in which the DQN model is evaluated on has on performance, we tested the results of our optimized DQN model, trained on the Atari 2600 Breakout environment, over 10 evaluation episodes on the environments: Breakout i.e. training environment, Pong, and Space Invaders. As a result of doing so, the DQN achieved for Pong, a maximal and average reward of -21 i.e. losing every single point in every episode, while achieving for Space Invaders a maximum reward of 5.0 and average reward of 3.5. As such, we can conclude that the DQN is extremely biased towards it's training environment, given that it can vastly outperform human metrics shown in Figure 1 within it's training environment (Breakout), but actually performs worse than random play when testing on an unseen environment.

5 Discussion

In general, considering the results from "Playing Atari with Reinforcement Learning", our experiments are certainly consistent with Mnih et. al.[3]. Although it is the true that our results are consistent with those of Mnih et. al., we were limited by time and computational power to only analyze and evaluate a single environment out of the seven tested originally. Instead, we shifted our focus on expanding their experiments to further strengthen their claims which is the case with Experiment 1 and 2 (theoretically) yielded much better performance when abiding by the specifications of parameters and hyperparameters of the DQN model, specifically discount factor γ and exploration rate ϵ. Furthermore, in showing that the DQN model is not generalizable to other environments in Experiment 3, we further strengthened the general notion in reinforcement learning that algorithms/models, as expected, are extremely environment reliant. In summary, while our experiments were constrained, they nevertheless contribute to the body of evidence supporting the DQN model's capabilities and limitations. Future research might expand on this work by applying the model across multiple environments and utilizing more computational power, which could offer more insights into the model's adaptability and potential areas for improvement. The dependence of reinforcement learning models on their environment, as evidenced by our Experiment 3, remains a significant hurdle for the field and an avenue for continued exploration and innovation.

5.1 What was easy

During our research reproducing the results of the study "Playing Atari with Deep Reinforcement Learning" by Mnih et. al., we found the DQN model introduced by the authors to be a commonly used reinforcement learning

model. Therefore, there was an abundance of implementations of the model by different researchers and developers in the field.

5.2 What was difficult

Analyzing the reproducibility of the study "Playing Atari with Deep Reinforcement Learning" by Mnih et. al. presented challenges. Firstly, a knowledge of reinforcement learning, specifically Q Learning, was essential. While the paper successfully introduced key Q Learning concepts, reproducing and understanding its results along with conducting experiments on the model demanded a robust background in RL.

Secondly, we weren't provided with the original code produced by the researchers of the study. Thus, we had to look for reproductions of the model, perform experiments on them to see which were compatible with latest versions and performed the best and use the best one. While this helped us get more engaged with the model as we looked through its implementations, it can be time consuming.

Moreover, due to the absence of the original code developed by the researchers of the study, we were required to search for reproductions of the model. Therefore, we conducted experiments on different reproductions of the DQN model in order to identify their compatibility with the latest software versions and their performance. Although this process improved our familiarity with the model through analyzing its implementations, it did introduce a time-consuming aspect to our research.

Additionally, the computational demands of the DQN model was a significant obstacle. As outlined in Section 3.5, the training of the different DQN models we explored required a high computational power and significantly long durations. This could be a roadblock for the ability to conduct a substantial number of experiments for evaluating the model effectively.

References

[1] Alexandre dos Santos Mignon and Ricardo Luis de Azevedo da Rocha. "An Adaptive Implementation of -Greedy in Reinforcement Learning". In: *Procedia Computer Science* 109 (2017). 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal, pp. 1146–1151. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.431. URL: https://www.sciencedirect.com/science/article/pii/S1877050917311134.

[2] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533. ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[3] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[4] Antonin Raffin et al. *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3. 2020.