SCIENTIA PLENA VOL. 21, NUM. 01 2025
www.scientiaplena.org.br doi: 10.14808/sci.plena.2025.XXXXXX

Evaluating the Generalization Capability of a DQN Agent Across Two Atari Games

Avaliação da Capacidade de Generalização de Um Agente DQN Entre Dois Jogos de Atari

E. S. Dias; J. E. M. Apóstolo; J. M. R. dos Santos; M. E. P. P. dos Santos; U. J. Cavalcante

Department of Computing, Federal University of Sergipe, 49100-000, São Cristóvão-Sergipe, Brazil

edgarsz@academico.ufs.br
joao.apostolo@dcomp.ufs.br
matheus.ribeiro@dcomp.ufs.br
eduardapossari@academico.ufs.br
ulissesc2811@academico.ufs.br

Deep Reinforcement Learning (DRL) is a Machine Learning technique that combines Reinforcement Learning (RL) with Deep Learning. Given the growth of this tool, it has stood out in solving complex problems, including in video games. Thus, the objective of this work is to evaluate the generalization capability of a DRL model based on the Deep Q-Network (DQN) architecture, together with the Monte Carlo Tree Search (MCTS) algorithm, to be trained in the game Space Invaders and subsequently evaluated in the game Demon Attack, both from the Atari 2600 console. To achieve this goal, experiments will be conducted following common performance metrics, allowing the assessment of the agent's ability to adapt to new scenarios. Therefore, the obtained results contribute to understanding the potential and limitations of the DQN approach in diverse contexts, which may serve as a basis for developing new research within the scope of this work.
Keywords: Deep Reinforcement Learning, Deep Q-Network, Atari.

O Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* - DRL) é uma técnica de Aprendizagem de Máquina (*machine learning*) que combina o Aprendizado por Reforço (*Reinforcement Learning* - RL) com o Aprendizado Profundo (*deep learning*). Diante do crescimento desta ferramenta, ela tem se destacado na resolução de problemas complexos, inclusive em jogos eletrônicos. Dessa maneira, o objetivo deste trabalho é avaliar a capacidade de generalização de um modelo de DRL baseado na arquitetura *Deep Q-Network* (*DQN*), juntamente do algoritmo de Pesquisa em Árvore de Monte Carlo (*Monte Carlo Tree Search* - MCTS), para ser treinado no jogo *Space Invaders* e subsequentemente avaliado no jogo *Demon Attack*, ambos do console Atari 2600. Para alcançar este propósito, serão feitos experimentos que seguirão métricas de desempenho comuns, permitindo avaliar a competência do agente construído de se adaptar a novos cenários. Portanto, os resultados obtidos contribuem para a percepção das potencialidades e limitações da abordagem DQN em contextos divergentes, que podem servir de base para o desenvolvimento de novas pesquisas dentro do escopo deste trabalho.
Palavras-chave: Aprendizado por Reforço Profundo, Deep Q-Network, Atari.

1. INTRODUCTION

In recent years, Reinforcement Learning has stood out for its efficiency in training machines to solve problems. This technique is a machine learning paradigm in which an agent learns to make decisions through interaction with the environment in which it is embedded, that is, through experience. In this way, the artificial intelligence system receives rewards for actions that lead it toward achieving its goal and penalties for activities that move it away from completing this objective. Consequently, through trial and error, the agent learns to maximize rewards and minimize penalties, thus developing an optimal policy for manipulating that environment.

Given this context, it is well known that Q-learning is a powerful Reinforcement Learning (RL) algorithm, so much so that it was used in Mnih's (2013) [1] work, together with Convolutional Neural Networks (CNNs), to develop the first deep learning model capable of successfully learning control policies directly from high-dimensional sensory inputs using Deep Reinforcement Learning (DRL). This system was responsible for successfully playing seven Atari 2600 games.

In this context, it is undeniable that DRL has emerged as a powerful and relevant alternative for solving complex problems. In particular, the application of Deep Reinforcement Learning to video games has been extensively studied, especially motivated by the work of Mnih et al. (2015) [2], which improved the Deep Q-Network architecture developed in Mnih's previous work (2013) [1]. This study demonstrated the network's capability to surpass the performance of professional human players in Atari 2600 games. This approach also utilizes CNNs to extract useful features from the game's pixels and then passes these attributes to a Q-learning algorithm so it can learn an optimal policy.

In this work, we explore the construction of a DQN-based agent to play Space Invaders, a world-famous classic game from the Atari console, despite knowing that it is a challenging game for training. After properly training and evaluating this algorithm, we investigate its generalization ability by transferring the same actuator to another game on the same console, Demon Attack, which bears some resemblance to the previous game but has significant differences in movement dynamics and enemy behavior. These divergences will be essential for the experiments.

To ensure learning consistency in the architecture, the DRL-based model follows the traditional principles of experience replay and target networks. First, the agent's experiences are stored in a memory buffer, which, during training, randomly samples small data batches (mini-batches) from this buffer to update the neural network. Subsequently, two neural networks are updated, one at each step and another every N steps, using the weights of the main network. Additionally, the network will also use the MCTS algorithm to assist in training the agent. Monte Carlo Tree Search (MCTS) performs heuristic expansion by simulating new states. It operates based on four sequential stages: selecting a random state, expanding it, simulating a game randomly, and backpropagating the results.

Thus, the objective of this article is to analyze the system's generalization capability. Once the model is trained on Space Invaders and then applied to Demon Attack, it will be possible to compare and evaluate its performance in both games, using predefined and described metrics. This approach will allow us to understand to what extent the developed DQN system can adapt to new gaming scenarios on the classic Atari 2600 console, highlighting its limitations and potential. The factors demonstrated in this work may also be valuable for future research seeking to further explore the scope of this study.

2. METHODOLOGY

The agent used for the experiments described in this work was developed using a mixed approach, incorporating DQN, a convolutional neural network, and, as a way to accelerate the model's learning, we leveraged the MCTS algorithm to generate an initial set of experiences. In this context, after preprocessing the observations and aiming to capture temporality through frame stacking, we used a tensor of shape (88, 80, 4) as input to the CNN, corresponding to 4 frames of 88x80 pixels. Additionally, we selected three hidden convolutional layers with ReLU activation, where the first has 32 filters of size 8x8 with a stride of 4, the second has 64 filters of size 4x4 with a stride of 2, and the third also has 64 filters but with a size of 3x3 and a stride of 1. Moreover, a final fully connected hidden layer with 512 neurons and ReLU activation was included, followed by a fully connected output layer with one neuron for each possible action, representing the Q-values.

Other key factors used for training our agent include a second network (target network), which is a copy of the main network, with weights updated periodically based on the main network, allowing for more stable training; an epsilon-greedy (ε-greedy) policy with epsilon (ε)

decay over time, enabling better balancing between exploration and exploitation; and the experience replay technique, which utilizes experiences stored in a buffer to perform random sampling during training, helping to break temporal correlations in learning.

To train the agent while leveraging the previously mentioned resources, the Deep Q-Learning algorithm represented in the image below was implemented.

Algorithm 1 deep Q-Learning (DQN) with experience replay

Initialize replay memory D to capacity N
Initialize action-value function Q with random weight θ
nitialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode $= 1$ to M **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $w_1 = w(s_1)$
    for $t = 1$ to T **do**
        With probability ε, select a random action a_t
        otherwise select $a_t = \arg\max_a Q(w(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $w_{t+1} = w(s_{t+1})$
        Store transition (w_t, a_t, r_t, w_{t+1}) in D
        Sample random minibatch of transitions (w_j, a_j, r_j, w_{j+1}) from D
        Set y_j:
        if episode terminates at step $j + 1$ **then**
            $y_j = r_j$
        else
            $y_j = r_j + \gamma \cdot \max_{a'} \hat{Q}(w_{j+1}, a'; \theta^-)$
        end if
        Perform a gradient descent step on $(y_j - Q(w_j, a_j; \theta))^2$ with respect to the network parameters θ
        Every C steps reset $\hat{Q} = Q$
    end for
end for

Figure 1: Deep Q-Learning algorithm with experience replay. Source: Mnih et al. (2015) [2].

Regarding the main libraries and frameworks used in the implementations, the Stable-Baselines3 (SB3) can be mentioned, which provides stable implementations of reinforcement learning algorithms, as well as the TensorBoard tool, which allows the visualization of training metrics, model graphs, and other statistics. Additionally, Gymnasium provides access to a set of environments based on Atari games for agent training, and PyTorch offers a dynamic approach to building neural networks. As for the dataset used, the model's learning process was carried out exclusively through interaction with the environment, meaning no experience was obtained from an external dataset. Furthermore, the development and testing environment was Google Colab, using a machine with an NVIDIA T4 GPU and 16 GB of RAM. Although the Google Colab machine has good performance, we faced limitations on its usage time, making experimentation difficult.

3. EXPERIMENTS

In our experiments, we implemented a Reinforcement Learning agent using a Deep Q-Network-based scheme with a Convolutional Neural Network for visual data processing, following the Nature CNN architecture, which is more suitable for processing the input images of Atari games. To improve the quality of stored experiences and, consequently, the efficiency

of reinforcement learning, we used the Monte Carlo Tree Search algorithm to assist in generating the initial data for the replay buffer.

During testing, we started by training the agent in the Space Invaders environment, adjusting the settings until we reached a state that indicated a clear projection of performance improvement within our constraints. Once we obtained this satisfactory result, we applied the same configurations to train the agent in Demon Attack, maintaining the previously defined agent and hyperparameters. At the end of the process, we conducted a comparative analysis of the results obtained in both games, evaluating the model's effectiveness and its ability to adapt to an environment with similar yet distinct mechanics.

3.1. Test Setup

The agent was initially trained in the Space Invaders game environment, using MCTS to generate high-value initial samples for the replay buffer, accelerating learning. The agent was trained for a fixed number of iterations, with adjustments to the hyperparameters to stabilize the convergence of the optimal policy. Training was conducted for 100,000 timesteps, with a learning rate (α) of 0.0003, a discount factor (γ) of 0.99, and a replay buffer size of 50,000 transitions.

In training the agent for Demon Attack, all the test configurations from Space Invaders were kept to understand the agent's behavior in a very similar game.

3.2. Performance metric

During the tests, the average Q-value per iteration was used as the learning metric. This choice was based on the paper Playing Atari with Deep Reinforcement Learning by Mnih (2013) [1], which highlights this metric as being more stable compared to the average reward per iteration. While the average reward may show significant variations due to the stochastic nature of the environment and abrupt changes in the agent's policy, the Q-value average graph tends to grow much more smoothly, consistently reflecting the progress of learning over time.

3.3. Offline training

Due to compatibility issues between dependencies, we did not use offline training with the pre-generated dataset from the DQN Replay Dataset, created by Agarwal et al. (2020) [4], which could have accelerated and optimized the learning process. As a result, we had to work with computational resources limited by usage time, which restricted our number of iterations per test to 100,000 iterations and 50,000 transitions in the replay buffer.

3.3. Comparison of approaches

During the tests, several approaches were tested, but the ones that had the most impact on the agent's learning were the prioritization of attack and movement actions in the data generated by MCTS for the buffer, increasing the DQN learning rate, and adding a pre-training phase for the data, which improved the quality of the data in the replay buffer.

Before these changes, the agent often got stuck in suboptimal states, characterized by limited movement and, in some cases, an inability to learn to attack. These behaviors showed unsatisfactory performance, with low average rewards and little progress in the game. After the modifications, we observed a significant improvement: the agent started moving more and shooting enemies more frequently, resulting in a noticeable increase in overall performance. These adjustments not only allowed the agent to escape suboptimal states but also accelerated the model's convergence, demonstrating the effectiveness of the approaches adopted.

4. RESULTS AND DISCUSSION

It is undeniable that the experiments conducted were really important, as they allowed for the analysis of the results, thus serving as inspiration for the discussion. In this context, after the initial training of the agent playing Space Invaders, the following results were obtained:

Figure 2: Diagram of the average reward of the architecture while playing Space Invaders. Source: Prepared by the authors.

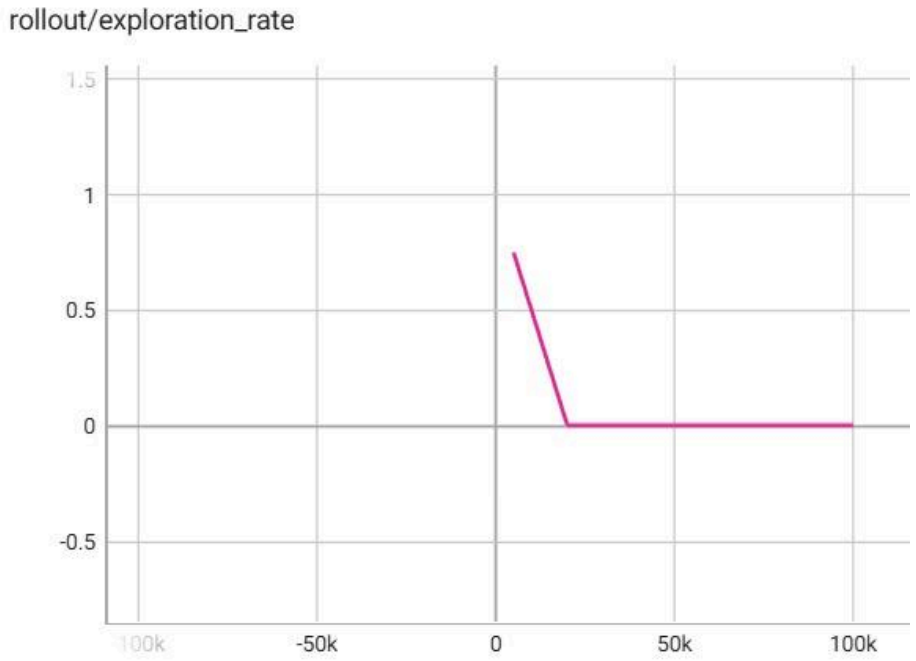

Figure 3: The epsilon/e-greedy graph represents the agent's exploration time while playing Space Invaders and Demon Attack. Source: Prepared by the authors.

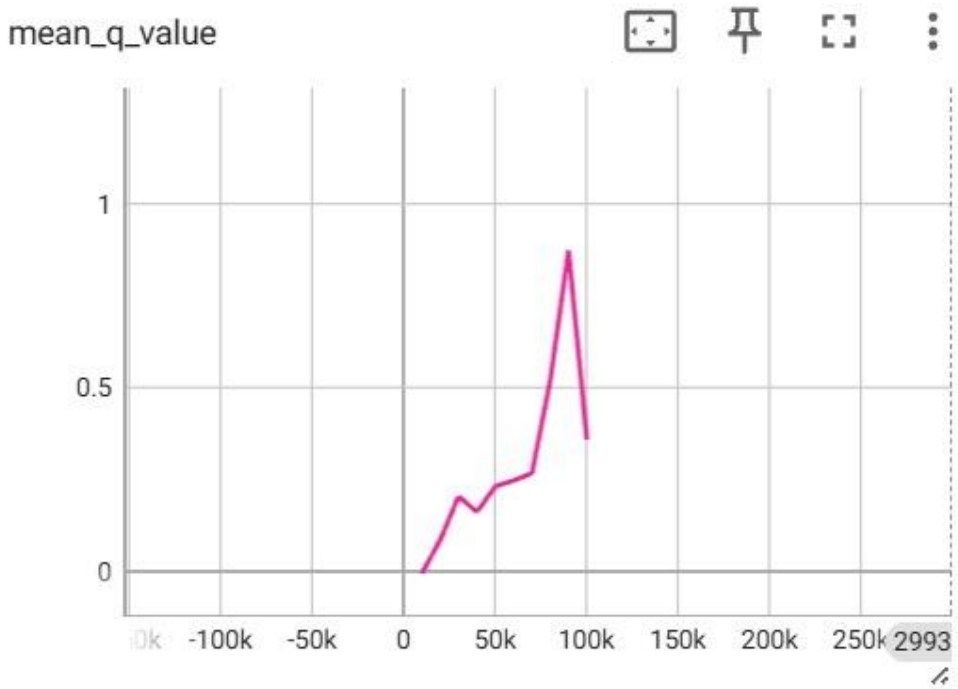

Figure 4: Diagram that characterizes the average Q-value of the model while playing Space Invaders.
Source: Prepared by the authors.

Firstly, it is noticeable that according to Figure 2, the agent shows significant fluctuations in performance, indicating that it is learning, but with inconsistencies. Additionally, it is known that this initial drop indicates that the system is exploring different strategies and adjusting its behavior, a fact that can be observed in Figure 3. The model spends 20% of the training time exploring new scenarios. After this drop, there is a considerable improvement in the rewards that the DQN-based system receives, thus indicating that it is learning to play efficiently over time. The peak, above 300 in reward, indicates that an effective strategy was found for this scenario. However, the subsequent drop after the peak shows the model's difficulty in maintaining consistent performance. Therefore, it is undeniable that to adjust this factor, the architecture may need modifications, either in the learning rate or in the exploration method.

Next, when analyzing the graph shown in Figure 4, it is noticeable that the average Q-values over the training of the Space Invaders agent also show drops. The Q-value starts near 0 and increases gradually, signaling that the setup is learning to associate actions with high rewards. However, around 100,000 steps, there is a peak followed by a drop, which demonstrates problems in the agent's learning. Finally, comparing the diagram with Figure 2, both show growth followed by a drop. This truthfulness might indicate that when the agent was learning and earning more rewards, the Q-values increased. However, if the agent began exploring less or learning suboptimal policies, the Q-values and average rewards decrease.

After conducting the agent’s experiments in the Space Invaders game, the executor – to evaluate its generalization capability – was tasked with trying to play the Demon Attack game, which is simpler to learn than the previous one. Thus, the experiences conducted resulted in the following:

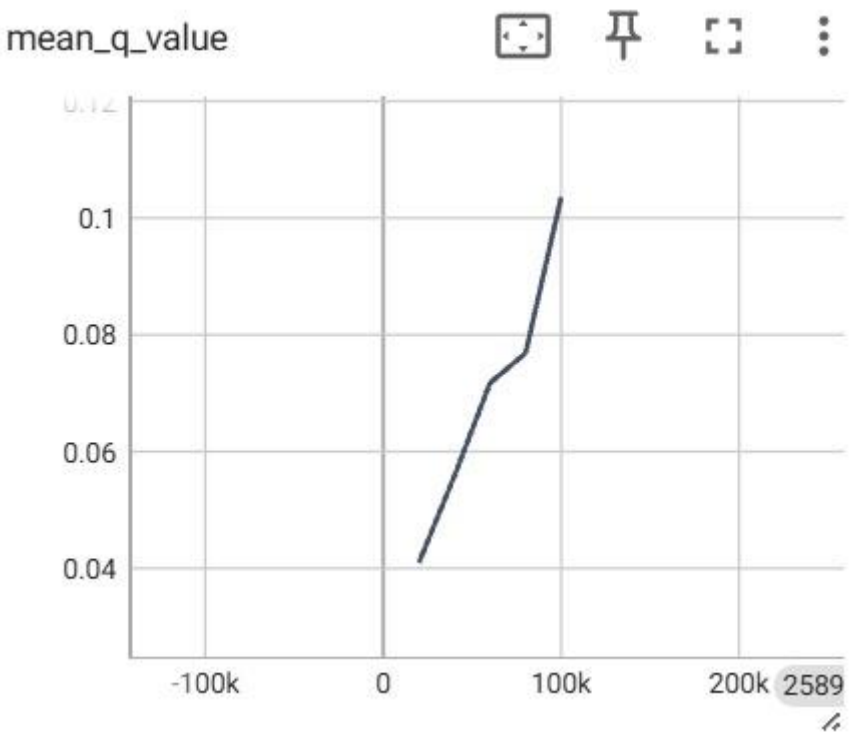

Figure 5: Diagram that characterizes the average Q-value of the model while playing Demon Attack. Source: Prepared by the authors.

Figure 6: Graph representing the average reward of the agent while playing Demon Attack. Source: Prepared by the authors.

When observing the network's performance in the new Atari game, it is undeniable that the performance was much more enjoyable. In Figure 5, it is evident that the average Q-value has a slow and consistent growth, indicating that the agent is learning more stably than in Space Invaders within the same exploration time, as both situations spend 20% of the training time

exploring new scenarios. Furthermore, compared to the average Q-value graph from the first game, the Q-values are much lower, which could suggest that the second game gives fewer rewards or has a different scoring system. It is also important to note that there are no drops in Figure 5, which might indicate that the agent didn't face abrupt changes in strategy. Regarding Figure 6, it is clear that, just like in the previous game, it starts by exploring ineffective actions, leading to poor performance at the beginning. Then, the agent starts learning better strategies, increasing its average reward. This suggests that it is adapting to the game and discovering more advantageous actions. The slight drop following the growth might be occurring due to late exploration; that is, probably the policy still allows for exploration, and the agent might be testing new actions, which could cause a slight decrease in the average reward.

Therefore, the performance of the DQN architecture, along with the convolutional neural network and the MCTS algorithm, was divergent for the two games analyzed. For Space Invaders, the performance was faster but less stable, while for Demon Attack, the opposite was true. Thus, the results of the built model were more satisfactory in terms of efficiency for the second game of the classic Atari 2600 console.

Moreover, it is known that as the number of training timesteps increases, the model's performance tends to improve. In this context, by increasing the number of timesteps in the training of the first game to 500,000, the model reaches satisfactory Q-value numbers, as shown in Figure 7. However, it was not possible to do the same for Demon Attack due to issues with the game's environment. Therefore, the results of this scenario were not addressed in this work.

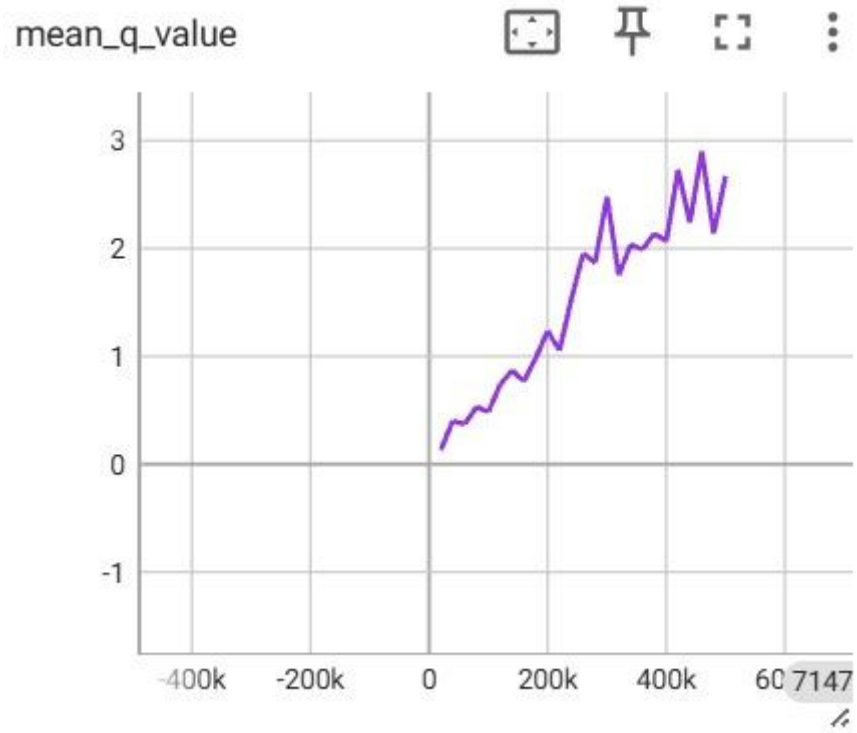

Figure 7: Diagram that characterizes the average Q-value of the model while playing Space Invaders with 500,000 timesteps. Source: Prepared by the authors.

5. CONCLUSION

Therefore, it is notable that using a second network (target network) and an epsilon-greedy (ε-greedy) policy with epsilon (ε) decay provided a good initial balance between exploration and exploitation for the game Space Invaders. However, we were unable to achieve stable training, causing the machine to explore extensively at the beginning and yield an optimal average reward, but this number decreases significantly over time.

Nevertheless, the artificial intelligence demonstrated superior performance in the game

Demon Attack. Despite experiencing an initial drop, resulting from a primary exploration phase necessary to learn the game's patterns and strategies, it was later able to reach an optimal reward value. Additionally, it managed to maintain this high performance consistently, achieving and sustaining a relatively high average score throughout the matches.

Thus, it is evident that the model's generalization capability was achieved satisfactorily. Although the performance in the challenging Space Invaders was not ideal, the DQN architecture, along with the convolutional neural network and the MCTS algorithm, demonstrated great ability in learning and efficiently playing Demon Attack, which is easier to play than the former.

To improve the results and enhance the agent's performance consistency across different games, it is extremely important to train the machine with a larger replay buffer, allowing for a more diverse sample of past experiences, reducing bias, and improving the generalization of learned policies, as well as with more timesteps to bring more stability, efficiency, and complexity to the training.

As previously observed, the machine played Demon Attack better than Space Invaders. The reason for this can be studied in future research; however, we conclude that Space Invaders is a more difficult game due to the instability of the environment, the increasing speed of the invaders as the game progresses, and the fact that learning can be slower since the machine only receives rewards when it eliminates an invader.

6. BIBLIOGRAPHIC REFERENCES

1. Mnih, Volodymyr. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
2. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533. doi:10.1038/nature14236.
3. M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents, Journal of Artificial Intelligence Research, Volume 47, pages 253-279, 2013.
4. Agarwal, R., Schuurmans, D. & Norouzi, M.. (2020). An Optimistic Perspective on Offline Reinforcement Learning International Conference on Machine Learning (ICML).