SCIENTIA PLENA VOL. 21, NUM. 01 2025
www.scientiaplena.org.br doi: 10.14808/sci.plena.202X.XXXXXX

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

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

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

Departamento de Computação, Universidade Federal de Sergipe - UFS, 49100-000, São Cristóvão-Sergipe, Brasil

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

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.

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.

1. INTRODUÇÃO

Nos últimos anos, o Aprendizado por Reforço tem se destacado por sua eficiência no treinamento de máquinas para a resolução de problemas. Essa técnica é um paradigma de *machine learning* no qual um agente aprende a tomar decisões através da interação com o ambiente no qual ele está inserido, ou seja, por meio da experiência. Dessa maneira, o sistema de inteligência artificial recebe recompensas por ações que o levem a atingir o seu objetivo e penalidades por atividades que o afastem da completude desta meta. Consequentemente, por meio da tentativa e erro, o agente aprende a maximizar as recompensas e diminuir as penalidades, desenvolvendo assim uma política ótima para manipular aquele ambiente. Tendo

em vista tal contexto, é sabido que o *Q-learning* é um poderoso algoritmo de RL, tanto que este foi utilizado, no trabalho de Mnih (2013) [1], juntamente de Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNNs), para o desenvolvimento do primeiro modelo de *deep learning* para aprender com sucesso políticas de controle diretamente de entradas sensoriais de alta dimensão usando Aprendizado por Reforço Profundo. O sistema foi responsável por conseguir jogar 7 jogos de Atari 2600.

Neste contexto, é indubitável que o DRL tem se destacado como uma alternativa poderosa e relevante para a solução de problemas complexos. Em especial, a aplicação do Aprendizado por Reforço Profundo em jogos eletrônicos tem sido abundantemente estudada, motivada especialmente pelo trabalho de Mnih et al. (2015) [2], que aprimorou a arquitetura *Deep Q-Network*, desenvolvida no trabalho anterior de Mnih (2013) [1], e demonstrou a capacidade que esta rede possui de sobrepujar o desempenho de humanos profissionais em jogos de Atari 2600. Tal abordagem também utiliza CNNs para extrair características úteis dos pixels de determinado jogo e assim, passar esses atributos a um algoritmo *Q-learning* para que este aprenda uma política ótima.

Neste trabalho, exploramos a construção de um agente baseado em DQN para jogar o jogo *Space Invaders*, mundialmente famoso e um clássico do console Atari, mesmo sabendo que é um jogo difícil para treinamento. Após o devido treinamento e avaliação deste algoritmo, investigamos sua capacidade de generalização ao transferirmos o mesmo atuador para outro jogo do mesmo *videogame*, o *Demon Attack*, que possui certa semelhança com o jogo anterior, mas tem mudanças significativas na dinâmica de movimento e no comportamento dos inimigos. Tais divergências serão essenciais para os experimentos. O modelo baseado em DRL, para garantir a constância na aprendizagem da arquitetura, segue os fundamentos tradicionais de replay de experiência (*experience replay*) e rede alvo (*target network*). Primeiramente, as experiências do agente ficam armazenadas em um buffer de memória que, durante o treinamento, amostra aleatoriamente pequenos conjuntos de dados (mini-batches) de experiências desse buffer para atualizar a rede neural. Subsequentemente, duas redes neurais são atualizadas, uma a cada passo e outra a cada N passos com os pesos da rede principal. Ademais, a rede também utilizará o algoritmo MCTS para auxiliar no treinamento do agente. O *Monte Carlo Tree Search* realiza a expansão heurística utilizando a simulação de novos estados, de modo que este funciona com base em 4 etapas subsequentes, a seleção de um estado aleatório, a expansão do mesmo, a simulação de um jogo randomicamente e a retropropagação (*backpropagation*).

Portanto, o objetivo do seguinte artigo é analisar a capacidade de generalização do sistema. De maneira que - uma vez que o modelo seja treinado no *Space Invaders* e, em seguida, seja aplicada no *Demon Attack* - será possível comparar e avaliar o desempenho desta em ambos os jogos, através de métricas previamente estabelecidas e descritas. Dessa forma, será concebível compreender até que ponto o sistema de DQN desenvolvido pode adaptar-se a novos cenários de jogos do clássico console Atari 2600, evidenciando assim, seus limites e potenciais. Tais fatores demonstrados neste trabalho também podem ser úteis para trabalhos futuros que busquem estudar mais profundamente o escopo deste artigo.

2. METODOLOGIA

O agente utilizado para as experimentações que serão descritas neste trabalho foi desenvolvido por meio de uma abordagem mista, com a utilização de DQN, rede neural convolucional e, como uma forma de acelerar o aprendizado do modelo, aproveitamos o algoritmo MCTS para a geração de um conjunto inicial de experiência. Nesse sentido, após o pré-processamento das observações e, com o objetivo de capturar a temporalidade, o empilhamento dos frames, utilizamos um tensor de forma (88, 80, 4) como entrada da CNN, correspondendo aos 4 frames de 88x80 pixels. Além disso, foram escolhidas 3 camadas convolucionais ocultas com ativação ReLU, onde a primeira possui 32 filtros de tamanho 8x8 com stride de 4, a segunda, 64 filtros de tamanho 4x4 com stride de 2, e a terceira, também com 64 filtros, mas de tamanho 3x3 e stride de 1. Ademais, uma última camada oculta totalmente

conectada com 512 neurônios e ativação ReLU e, por fim, uma camada totalmente conectada de saída com um neurônio para cada ação possível, representando os valores de Q.

Outros fatores determinantes utilizados para o treinamento do nosso agente podem ser mencionados: uma segunda rede (*target network*), que é uma cópia da rede principal (*main network*), com os pesos atualizados periodicamente com base na rede principal, possibilitando um treinamento mais estável; uma política *epsilon-greedy* (*ε-greedy*) com decaimento do valor de *epsilon* (ε) ao longo do tempo, permitindo um melhor balanceamento entre exploração e explotação; a técnica de *experience replay*, aproveitando as experiências armazenadas em um buffer para a realização de uma amostragem aleatória durante o treinamento, o que favorece a quebra de correlações temporais no treinamento.

Para o treinamento do agente explorando a utilização dos recursos mencionados anteriormente nesta seção, o algoritmo *Deep Q-Learning* representado pela Figura 1.

Algorithm 1 deep Q-Learning (DQN) com replay de experiência

Inicialize o buffer de replay D com capacidade N
Inicialize a função de valor-ação Q com pesos aleatórios θ
Inicialize a função de valor-ação alvo $\hat{Q}$ com pesos $\theta^{-} = \theta$
para episódio $= 1$ até M **faça**
    Inicialize o estado inicial $s_1 = \{x_1\}$ e uma sequência pré-processada $w_1 = w(s_1)$
    para $t = 1$ até T **faça**
        Com probabilidade ε, escolha uma ação aleatória a_t
        Caso contrário, escolha $a_t = \arg\max_a Q(w(s_t), a; \theta)$
        Execute a_t no ambiente e observe a recompensa r_t e a imagem x_{t+1}
        Defina $s_{t+1} = s_t, a_t, x_{t+1}$ e pré-processe $w_{t+1} = w(s_{t+1})$
        Armazene a transição (w_t, a_t, r_t, w_{t+1}) em D
        Amostre um minibatch aleatório (w_j, a_j, r_j, w_{j+1}) de D
        Calcule o valor alvo y_j:
        se o episódio termina no passo $j + 1$ **então**
            $y_j = r_j$
        senão
            $y_j = r_j + \gamma \cdot \max_{a'} \hat{Q}(w_{j+1}, a'; \theta^{-})$
        fim se
        Realize um passo de gradiente descendente em $(y_j - Q(w_j, a_j; \theta))^2$ com relação aos parâmetros da rede θ
        A cada C passos, atualize $\hat{Q} = Q$
    fim para
fim para

Figura 1: Algoritmo deep Q-learning com replay de experiência. Fonte: Mnih et al. (2015) [2].

Com relação às principais bibliotecas e frameworks que foram utilizados no desenvolvimento, podem ser mencionados o Stable-Baselines3 (SB3), que disponibiliza implementações estáveis de algoritmos de aprendizado por reforço, a ferramenta TensorBoard, que permite a visualização de métricas de treinamento, gráficos de modelos e outras estatísticas, o Gymnasium, que provê o acesso a um conjunto de ambientes baseados em jogos de Atari para o treinamento dos agentes, e o PyTorch, que oferece uma abordagem dinâmica na construção de redes neurais. Já em relação ao dataset utilizado, o processo de aprendizagem da arquitetura foi feito exclusivamente por meio da interação com o ambiente, ou seja, nenhuma experiência foi obtida por meio de um dataset externo. Outrossim, o ambiente de desenvolvimento e de realização de testes foi o Google Colab, utilizando uma máquina com a GPU NVIDIA T4 e 16

GB de memória RAM. Apesar da máquina do Google Colab ter uma boa performance, tínhamos limitações para o tempo de uso desta, dificultando a experimentação.

3. EXPERIMENTOS

Em nossos experimentos implementamos um agente *Reinforcement Learning* com uma esquema baseado em *Deep Q-Networks* com uma *Convolutional Neural Network* para o processamento de dados visuais com arquitetura *Nature CNN*, mais adequada para processar as imagens de entrada dos jogos Atari. Com o intuito de melhorar a qualidade das experiências armazenadas e, consequentemente, a eficiência do aprendizado por reforço, utilizamos o algoritmo de *Monte Carlo Tree Search* para auxiliar na geração de dados iniciais do *buffer* de replay.

Durante os testes, iniciamos o treinamento do agente no ambiente do *Space Invaders*, ajustando as configurações até alcançarmos um estado que indicasse uma projeção clara de melhoria no desempenho dentro das nossas limitações. Uma vez obtido esse resultado satisfatório, aplicamos as mesmas configurações ao treinamento do agente no jogo *Demon Attack*, mantendo o agente e os hiperparâmetros previamente definidos. Ao final do processo, realizamos uma análise comparativa dos resultados obtidos em ambos os jogos, avaliando a eficácia do modelo e sua capacidade de adaptação a um ambiente com mecânicas semelhantes, porém distintas.

3.1. Configuração dos testes

O agente foi inicialmente treinado no ambiente do jogo *Space Invaders*, utilizando MCTS para gerar amostras iniciais de alto valor para o *buffer* de replay, acelerando o aprendizado. O agente foi treinado por um número fixo de iterações, com ajustes nos hiperparâmetros para estabilizar a convergência da política ótima. O treinamento foi conduzido por 100.000 passos de tempo (*timesteps*), com uma taxa de aprendizado (α) de 0,0003, fator de desconto (γ) de 0,99, e tamanho do *buffer* de replay de 50.000 transições.

No treinamento do agente para o *Demon Attack*, todas as configurações dos testes do *Space Invaders* foram mantidas, a fim de entender o comportamento do agente em um jogo muito semelhante.

3.2. Métrica de desempenho

Durante os testes, foi usado como métrica de aprendizado a média do Q-valor por iteração. Essa escolha foi baseada no artigo *Playing Atari with Deep Reinforcement Learning* de Mnih (2013) [1], o qual destaca essa métrica como sendo mais estável em comparação com a recompensa média por iteração. Enquanto a recompensa média pode apresentar variações significativas devido à natureza estocástica do ambiente e às mudanças abruptas na política do agente, o gráfico da média do Q-valor tende a crescer de maneira muito mais suave, refletindo de forma mais consistente o progresso do aprendizado ao longo do tempo.

3.3. Treinamento offline

Devido a problemas de compatibilidade entre as dependências, não usamos o treinamento offline com o conjunto de dados pré-gerados do *DQN Replay Dataset*, elaborado por Agarwal et al. (2020) [4], que poderia acelerar e otimizar o processo de aprendizado. Com isso, tivemos que trabalhar com recursos computacionais limitados pelo tempo de uso, o que restringiu o nosso número de iterações a cada teste em um valor de 100.000 iterações e 50.000 transições no *buffer* de replay.

3.4. Comparação de abordagens

Durante os testes, diversas abordagens foram testadas, porém, as que mais surtiram efeito no aprendizado do agente foram a priorização de ações de ataque e movimento nos dados gerados pelo MCTS para o buffer, o aumento da taxa de aprendizado do DQN e a adição de um pré-treinamento dos dados que melhorou a qualidade dos dados que estavam no buffer de replay.

Antes dessas alterações, o agente frequentemente ficava preso em estados subótimos, caracterizados por movimentação limitada e, em alguns casos, incapacidade de aprender a atacar. Esses comportamentos possuíam um desempenho insatisfatório, com recompensas médias baixas e pouca progressão no jogo. Após as modificações, observamos uma melhoria significativa: o agente passou a se movimentar mais e a atirar nos inimigos com maior frequência, o que resultou em um aumento notável no desempenho geral. Esses ajustes não apenas permitiram que o agente escapasse de estados subótimos, mas também aceleraram a convergência do modelo, demonstrando a eficácia das abordagens adotadas.

4. RESULTADOS E DISCUSSÃO

É indubitável que os experimentos feitos foram de suma importância pois, a partir deles, a análise dos resultados pode ser feita, servindo assim de inspiração para a discussão. Neste contexto, após o treinamento inicial do agente jogando *Space Invaders*, os seguintes resultados foram obtidos:

Figura 2: Diagrama da recompensa média da arquitetura ao jogar *Space Invaders*. Fonte: Elaborado pelos autores.

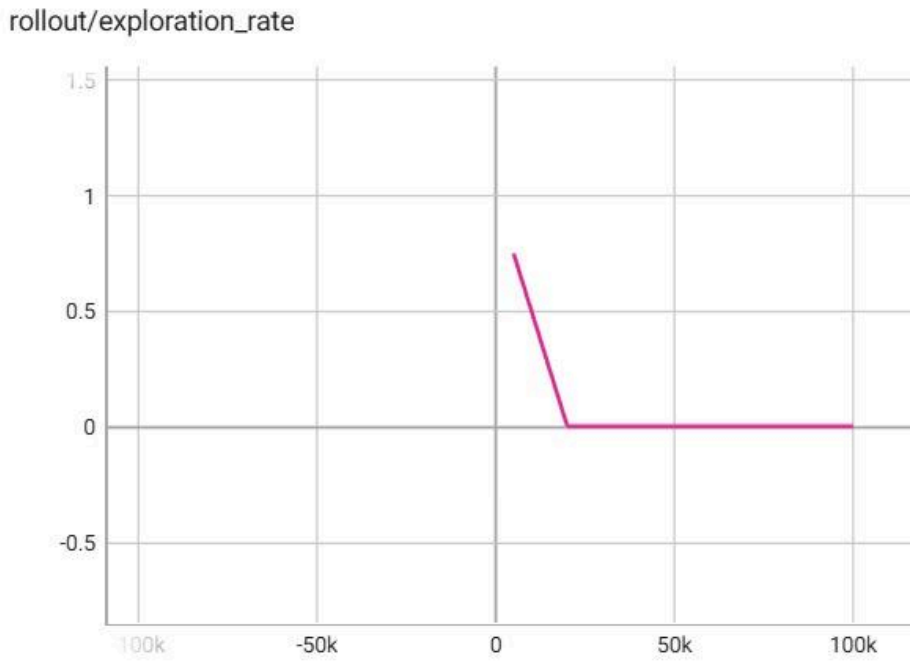

Figura 3: Gráfico *epsilon*/*e-greedy* representa o tempo de exploração do agente ao jogar *Space Invaders* e *Demon Attack*. Fonte: Elaborado pelos autores.

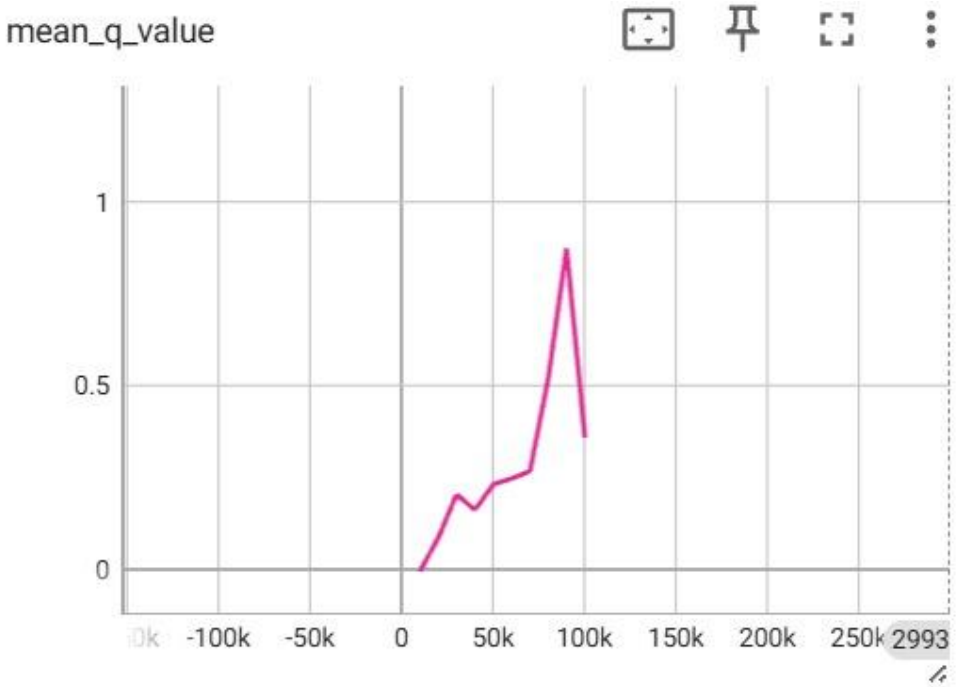

Figura 4: Esquema que caracteriza a média do Q-valor do modelo ao jogar *Space Invaders*. Fonte: Elaborado pelos autores.

Primeiramente, é perceptível que de acordo com a Figura 2, o agente apresenta flutuações significativas no desempenho, o que indica que ele está aprendendo, porém com inconsistências. Além disso, é conhecido que esta queda inicial indica que o sistema está explorando diferentes estratégias e ajustando o seu comportamento, fato que pode ser observado na Figura 3. O modelo passa 20% do tempo de treinamento explorando novos cenários. Depois desta queda, há uma melhora considerável nas recompensas que o sistema baseado em DQN recebe, indicando assim que este está aprendendo a jogar eficientemente com o tempo. O ápice, acima 300 de

recompensa, indica que para esse cenário a estratégia eficaz foi encontrada. Apesar disso, a queda posterior ao topo demonstra uma dificuldade do modelo em manter um desempenho constante. Logo, é indubitável que para ajustar este fator, a arquitetura pode precisar de alterações, seja na taxa de aprendizado ou no método de exploração.

Seguidamente, ao analisar o gráfico exposto na Figura 4, é notável que a média dos valores Q ao longo do treinamento do agente de *Space Invaders* também apresenta quedas. O valor Q começa próximo de 0 e aumenta gradualmente, sinalizando que a configuração está aprendendo a associar ações com recompensas altas. Apesar disso, em torno dos 100.000 passos, há um pico seguido de queda, que demonstra problemas no aprendizado do agente. Por fim, comparando o diagrama com a Figura 2, ambos mostram um crescimento seguido de uma queda. Tal veracidade pode indicar que, quando o agente estava aprendendo e ganhando mais recompensas, os valores Q aumentaram. No entanto, se o agente começou a explorar menos ou a aprender políticas subótimas, os valores Q e a recompensa média diminuíram.

Depois de realizados os experimentos do agente no jogo *Space Invaders*, o executor - para avaliar a capacidade de generalização do mesmo - ficou encarregado de tentar jogar o jogo *Demon Attack*, sendo este, de aprendizado mais simples que o anterior. Dessa forma, as experiências realizadas resultaram no seguinte:

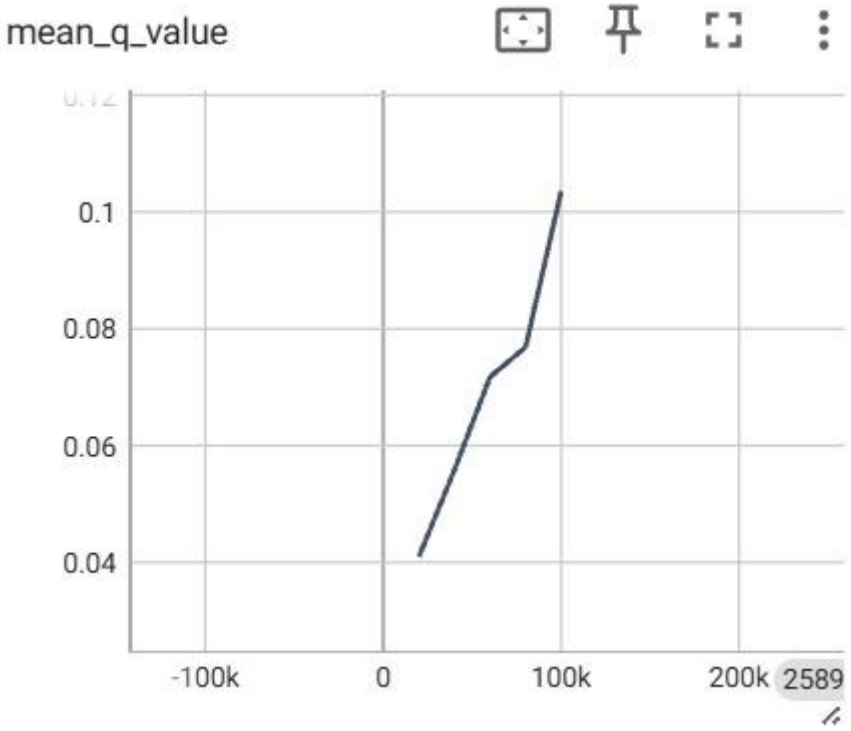

Figura 5: Esquema que caracteriza a média do Q-valor do modelo ao jogar *Demon Attack*. Fonte: Elaborado pelos autores.

Figura 6: Gráfico que representa a recompensa média do agente ao jogar *Demon Attack*. Fonte: Elaborado pelos autores

Ao observar o rendimento da rede no novo jogo de Atari, é indubitável que o desempenho foi bem mais prazeroso. Já que na Figura 5, é cognoscível que o Q-valor médio possui um crescimento lento e consistente, indicando que o agente está aprendendo de maneira mais estável do que no *Space Invaders* no mesmo tempo de exploração, já que esses duas situações passam 20% do tempo de treinamento explorando novos cenários. Ademais, comparado ao gráfico de valor Q médio do primeiro jogo, os Q-valores estão bem mais baixos, o que pode sugerir que o segundo jogo dá menos recompensas ou tem um sistema de pontuação diferente. Também é importante observar que não há quedas na Figura 5, o que pode indicar que o agente não enfrentou mudanças abruptas na estratégia. A respeito da Figura 6, é evidente que, assim como no jogo anterior, o seguinte começa explorando ações ineficazes, levando a um desempenho ruim no início. Depois, o agente começa a aprender estratégias melhores, aumentando sua recompensa média. Isso sugere que ele está se adaptando ao jogo e descobrindo ações mais vantajosas. A leve queda que sucede o crescimento pode estar ocorrendo pela exploração tardia, ou seja, provavaelmente a política ainda permite exploração e o agente pode estar testando novas ações, o que pode causar uma pequena redução na recompensa média.

Portanto, a performance da arquitetura DQN, junto da rede neural convolucional e ao algoritmo MCTS foi divergente para os dois jogos analisados. Sendo que para o *Space Invaders*, o desempenho foi mais rápido, mas menos estável e para o *Demon Attack*, o contrário. Dessa maneira, os resultados do modelo construído foram mais satisfatórios, em termos de eficiência, para o segundo jogo do clássico console Atari 2600.

Outrossim, é sabido que à medida que o número de passos de tempo de treinamento aumenta, a performance do modelo tende a aumentar. Neste contexto, ao aumentar o número de *timesteps* ao treinamento do primeiro jogo ao patamar de 500.000, o mesmo atinge números satisfatórios de Q-valor, como demonstrado pela Figura 7. Porém, não foi possível fazer o mesmo no *Demon Attack* devido a problemas do ambiente do próprio jogo. Assim, os resultados deste cenário não foram abordados neste trabalho.

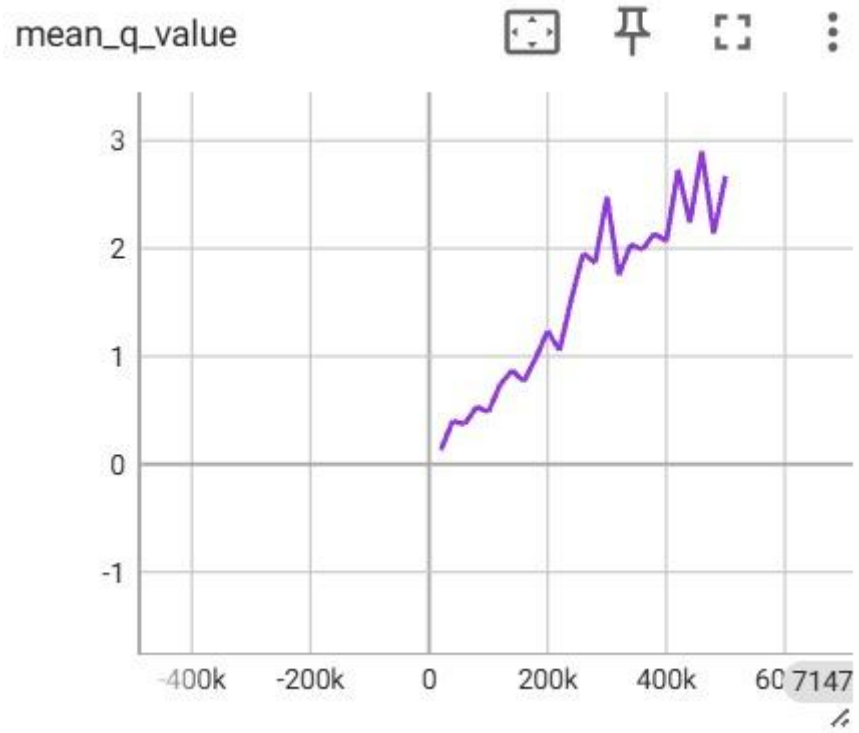

Figura 7: Esquema que caracteriza a média do Q-valor do modelo ao jogar *Space Invaders* com 500.000 *timesteps*. Fonte: Elaborado pelos autores.

5. CONCLUSÃO

Portanto, é notável que usar uma segunda rede (*target network*) e usar uma política de *epsilon-greedy* (*ε-greedy*) com decaimento do valor de *epsilon* (*ε*) proporcionou um bom balanceamento inicial entre exploração e explotação para o jogo *Space Invaders* porém não conseguimos alcançar um treinamento estável, fazendo a máquina explorar bastante no início e render média ótima de recompensas, mas cai bastante esse número com o passar do tempo.

Entretanto, a Inteligência Artificial demonstrou um desempenho superior no jogo *Demon Attack*. Apesar de ter enfrentado uma queda inicial, resultado de uma fase de exploração primária necessária para aprender os padrões e estratégias do jogo, ela conseguiu, posteriormente, atingir um valor ótimo de recompensa. Além disso, foi capaz de manter essa performance elevada de forma consistente, alcançando e sustentando uma média de pontuação relativamente alta ao longo das partidas.

Dessa maneira, é cognoscível que a capacidade de generalização do modelo foi atingida de maneira satisfatória. Apesar do desempenho evidenciado ao jogar o árduo *Space Invaders* não ter sido o ideal, a arquitetura DQN, junto da rede neural convolucional e ao algoritmo MCTS demonstrou grande poder ao aprender e jogar de modo eficiente o *Demon Attack*, o qual é mais fácil de se jogar do que o anterior.

Assim, para uma melhora nos resultados e aprimorar a consistência de desempenho do agente em diferentes jogos, é de extrema importância treinar a máquina com um buffer de replay maior, para ter uma amostra mais diversificada de experiências passadas reduzindo o viés e melhorando a generalização das políticas aprendidas, e com mais timesteps, a fim de trazer mais estabilidade, eficiência e complexidade para o treinamento.

Como visto anteriormente, a máquina jogou melhor o *Demon Attack* do que o *Space Invaders.* A causa disso pode ser estudada em uma nova pesquisa, porém chegamos a conclusão que o *Space Invaders* é um jogo mais difícil por conta da instabilidade do ambiente, a velocidade que os invasores tomam conforme o jogo avança e o aprendizado pode se tornar mais lento já que a máquina só ganha recompensas quando mata um invasor.

6. REFERÊNCIAS BIBLIOGRÁFICAS

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).