Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise de Políticas de Exploração no Aprendizado por Reforço aplicado a Jogos de Atari

Antônio Carlos de Souza Junior

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof.a Dr.a Roberta Barbosa Oliveira

Brasília
2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise de Políticas de Exploração no Aprendizado por Reforço aplicado a Jogos de Atari

Antônio Carlos de Souza Junior

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.a Dr.a Roberta Barbosa Oliveira (Orientador)
CIC/UnB

Prof. Dr. Díbio Leandro Borges
CIC/UnB

Prof. Dr. Vinícius Ruela Pereira Borges
CIC/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 06 de maio de 2022

Dedicatória

Eu dedico este trabalho à minha família e amigos.

Agradecimentos

Agradeço a professora Roberta Barbosa Oliveira por ter dedicado seu tempo e atenção para me auxiliar na confecção desta monografia.

Agradeço aos meus amigos e companheiros de curso que sempre estiveram próximos para compartilhar os momentos e os momentos de diversão.

Agradeço aos meus familiares porque sem eles eu não estaria aqui. Obrigado por sempre me apoiarem e me motivarem a continuar seguindo em frente.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

No aprendizado por reforço o dilema *exploration-exploitation* ainda é uma lacuna, sendo um dos maiores desafios da área. O dilema está relacionado à tomada de decisão, onde se deve decidir entre repetir o que foi feito no passado ou tentar novas ações que podem melhorar o resultado final. Ainda há carência de estudos que comparem as técnicas clássicas de exploração para aprendizado por reforço com técnicas mais atuais desenvolvidas para *deep learning*. O presente trabalho faz a comparação entre tais técnicas, com objetivo de compreender quais são mais eficientes na tarefa de auxiliar no aprendizado da inteligência artificial (agente). Para isso é proposta uma metodologia para o treinamento de diversas instâncias do agente, cada uma utilizando um dos jogos de Atari (ambiente): (a) *Pong*, (b) *Breakout*, e (c) *Space Invaders*, que são problemas desafiadores e com diferentes níveis de complexidade. Combinados com uma das seis diferentes políticas de exploração: (1) Aleatória, (2) *Greedy*, (3) *Epsilon Greedy*, (4) Boltzmann, (5) *Decaying Epsilon Greedy*, e (6) *Random Noise*. Todos os agentes são treinados utilizando o mesmo processo de aprendizagem utilizando *Deep Q-learning*, sem nenhum conhecimento prévio do ambiente e recebendo como estímulo do ambiente apenas imagens da tela do jogo. O desempenho de cada combinação ambiente-política foi avaliado com base na média de pontuação do agente obtida após interagir por 100 episódios com um dos jogos. De todas as políticas utilizadas para avaliação, a *Decaying Epsilon Greedy* foi a que obteve a maior média de pontuação nos jogos *Pong* e *Space Invaders* e a segunda maior no *Breakout*, perdendo somente para *Random Noise*, sendo, portanto, a melhor estratégia encontrada para auxiliar o agente a conseguir o melhor desempenho nos jogos de Atari.

Palavras-chave: inteligência artificial, redes neurais convolucionais, aprendizado profundo, aprendizado por reforço, deep q-learning, políticas de exploração

Abstract

In reinforcement learning, the dilemma exploration-exploitation is still a gap, being one of the biggest challenges in the field. The dilemma is related to decision making, where one must decide between repeating what was done in the past or trying new actions that can improve the final result. There is still a lack of studies comparing classical exploration techniques in reinforcement learning with techniques developed for deep learning. The present work makes a comparison between such techniques, in order to understand which ones are more efficient in the task of assisting in the learning of an artificial intelligence (agent). For this, a methodology is proposed for training several instances of the agent, each one using one of the Atari games (environment): (a) Pong; (b) Breakout; and (c) Space Invaders; which are challenging problems with different levels of complexity. Combined with one of six different exploration policies: (1) Random; (2) Greedy; (3) Epsilon Greedy; (4) Boltzmann; (5) Decaying Epsilon Greedy; and (6) Random Noise. All agents are trained using the same learning process using deep q-learning, without any prior knowledge of the environment and receiving only images of the game screen as an input from the environment. The performance of each environment-policy combination was evaluated based on the average agent score obtained after interacting for 100 episodes with one of the games. Of all the policies used for evaluation, Decaying Epsilon Greedy had the highest average score in Pong and Space Invaders and the second highest in Breakout, second only to Random Noise, being, therefore, the best strategy found to reach best performance in Atari games.

Keywords: artificial intelligence, convolutional neural network, deep learning, reinforcement learning, deep q-learning, exploration policies

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ALE *Arcade Learning Environment*.

AR Aprendizado por Reforço.

CNN *Convolutional Neural Network*.

DNN Deep Neural Network.

DQN *Deep Q-Network*.

IA Inteligência Artificial.

MDP *Markov Decision Process*.

MP *Markov Process*.

RNA Rede Neural Artificial.

Capítulo 1

Introdução

Durante muito anos a Inteligência Artificial (IA) foi vista como apenas obra de ficção científica. Era impensável que máquinas poderiam desempenhar funções que lembrassem a inteligência exclusivamente humana [11]. No entanto, recentemente, a IA já se tornou uma realidade, sendo utilizada diariamente em diversos tipos de aplicações, como jogos [12], assistentes pessoais [13] e reconhecimento de imagens [14]. A IA é considerada um ramo da Ciência da Computação, e tem como objetivo pesquisar e desenvolver dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, ou seja, a capacidade de comportamentos inteligentes [15]. Para desenvolver IAs com capacidades parecidas com as humanas é necessário entender como funciona o aprendizado humano.

O aprendizado humano é realizado em constante interação com o ambiente. Por exemplo, quando o ser humano aprendeu sobre o fogo ele não tinha um professor, apenas uma conexão sensorial com o ambiente. Por meio dessa conexão é possível "aprender sobre causa e efeito, sobre as consequências das ações, e como atingir um objetivo" [16]. Considerando o mesmo exemplo citado anteriormente, para aprender sobre o fogo, o ser humano precisou interagir com o ambiente diversas vezes. Portanto, a interação constante em conjunto com as experiências e estímulos adquiridos permitiu perceber como o fogo funciona e como ele pode ser melhor utilizado. À medida que o tempo passa mais conhecimento é acumulado a respeito do ambiente, sendo possível tomar melhores decisões no futuro. Inspirados nesse modelo, ao longo dos anos, foram desenvolvidas diversas pesquisas para aproximar, a forma de aprendizado por meio de experiência, da IA para serem capazes de realizar tarefas complexas como, reconhecimento de padrões e processamento de imagem [17]. Dessa forma, nasceu a área de aprendizado de máquina, ou *machine learning*, que tem como objetivo extrair informações de um conjunto de dados e representá-las de formas úteis para solução de problemas. Para auxiliar nessa tarefa, foi criada a Rede Neural Artificial (RNA) [18] que surgiu da tentativa de simular o funcionamento do cére-

bro humano, capaz de encontrar relações complexas em um grupo de dados e processar uma grande quantidade de informação. Com objetivo de expandir a capacidade da RNA foi criada a Rede Neural Convolucional, em inglês *Convolutional Neural Network* (CNN), que tem a estrutura similar a RNA, mas com a capacidade de compreender características de imagens de forma mais eficiente.

Existem três principais tipos de aprendizado de máquina: (1) aprendizado supervisionado, (2) aprendizado não-supervisionado, e (3) aprendizado por reforço [15]. O primeiro tipo, tem como objetivo encontrar relações entre entradas e saídas, baseado em amostras fornecidas para o modelo, ou seja, aprende a partir de exemplos fornecidos como chegar em um resultado. O segundo, tem como objetivo encontrar semelhanças ou padrões nos dados, sem conhecer como deveriam ser as saídas. O último, aprende a partir de estímulos (recompensas) gerados por interações entre a IA (agente) e um ambiente [16]. Por meio desses estímulos podem ser definidos objetivos, ações que se aproximam do objetivo resultam em recompensas positivas e ações que se afastam do objetivo resultam em recompensas negativas.

O Aprendizado por Reforço (AR) consiste em ensinar para um agente como tomar decisões para alcançar a maior recompensa possível. Por meio de tentativa e erro o agente deve encontrar a melhor forma de agir. Nessa estratégia, os ambientes são simulados, a máquina toma decisões a cada passo e cada decisão produz um estímulo, como no aprendizado humano. Algumas das aplicações do AR são: (1) controle de tráfico adaptativo [19], nesse ambiente o agente tem como objetivo diminuir o tempo de espera de motoristas; (2) inferência de diagnóstico [20], o agente tem como objetivo maximizar a taxa de acerto de conceitos médicos e diagnósticos; e (3) video games [21], o agente tem como objetivo maximizar a pontuação recebida ao final do jogo. Para solucionar problemas de aprendizado por reforço no contexto de jogos de Atari, métodos tradicionais podem ser utilizados, como o *SARSA* [22] para aprender políticas lineares a partir das imagens do jogo. O maior problema desse método é a necessidade de escolher as características da imagem que serão utilizadas pelo modelo para aprender sobre o jogo. Essa escolha é complexa e pode ser uma limitação, por exemplo, quando se quer utilizar um mesmo agente para aprender diferentes jogos. Esse problema pode ser superado com o *Deep Q-learning*, algoritmo de aprendizado por reforço que utiliza CNNs, capaz de extrair características das imagens de forma automática, i.e. sem a necessidade de interferência humana. No *Deep Q-learning* [10], a CNN é chamada de *Deep Q-Network* (DQN), que é utilizada para estimar o quão bom é para um agente escolher uma ação em determinado estado. Com essa informação é possível resolver qualquer problema escolhendo sempre a melhor ação para todos os estados.

Os principais motivos para aplicação de aprendizado por reforço em jogos é porque

eles foram feitos para serem desafiadores e precisam de habilidade para serem resolvidos por isso são propícios para algoritmos de AR; outro motivo é o fato de jogos possuírem um ambiente com regras bem definidas e um conjunto finito de estados, pré-requisito necessário para utilizar AR. Além disso, o aprendizado por reforço aplicado no contexto de jogos vem obtendo resultados muito animadores nos últimos anos. Com o AlphaGo [23] a DeepMind mostrou para o mundo que era possível construir um sistema capaz de ganhar de jogadores humanos profissionais de Go. Ao mesmo tempo, outros pesquisadores da DeepMind demonstraram que é possível criar um sistema capaz de aprender a jogar jogos de Atari sem nenhum conhecimento prévio [10]. Apesar dos bons resultados, o maior desafio do aprendizado por reforço é o fato do treinamento demorar muito tempo para produzir bons resultados porque os algoritmos fazem um uso ineficiente dos dados. Para minimizar esses problemas, políticas de exploração podem ser utilizadas durante o treinamento para guiar o agente no sentido de obter experiências que irão contribuir para o seu aprendizado mais eficiente.

No presente trabalho é aplicada a técnica de AR *Deep Q-learning* [10] para ensinar a IA a jogar Atari. Para isso, as DQNs serão utilizadas como agente, responsável por receber imagens da tela do jogo e decidir qual a melhor ação a ser tomada naquela situação. No processo de tomada de decisão é muito importante definir a estratégia utilizada para aprender sobre o ambiente e escolher como utilizar o conhecimento já adquirido [24]. No AR as políticas definem como o agente toma decisões e explora o ambiente. Portanto, é importante a avaliação da efetividade de diferentes estratégias de exploração do ambiente por meio da análise de como elas influenciam na pontuação final dos jogos de Atari. Serão comparadas cinco estratégias de exploração comuns na literatura clássica de AR [16, 24]: (1) Aleatória; (2) *Greedy*; (3) *Epsilon Greedy*; (4) *Boltzmann*; e (5) *Decaying Epsilon Greedy*; com uma estratégia mais recente, que é a *Random Noise*, proposta por Fortunato et al. [25]. As estratégias clássicas adicionam aleatoriedade nas ações da IA para estimular a exploração do ambiente. Enquanto, a proposta mais recente [25] insere ruído nas DQNs para estimular a exploração. A hipótese fundamental do trabalho é que todas as políticas de exploração irão melhorar o desempenho do agente em comparação com a escolha de ação de forma aleatória. Os ambiente em que serão aplicadas as políticas de exploração são: (a) *Pong*, (b) *Breakout*, e (c) *Space Invaders*. O primeiro jogo, tem nível de dificuldade fácil, porque a forma de obter pontuação exige pouco planejamento do agente. O segundo jogo, tem nível de dificuldade intermediária, porque possui as mesmas ações que o primeiro, mas necessita de um nível de planejamento maior para obter pontos. O último jogo, tem nível de dificuldade difícil, porque possui mais possibilidade de ações, o que exige mais planejamento do agente em comparação com os jogos anteriores.

1.1 Objetivo Geral e Específicos

O objetivo principal desse trabalho é analisar como as políticas de exploração influenciam no desempenho do agente nos jogos de Atari: (a) *Pong*, (b) *Breakout*, e (c) *Space Invaders*; utilizando modelos de *Deep Learning* para construir o agente e *Deep Q-learning* para treiná-lo. Para isso, foram definidos os seguintes objetivos específicos:

1. Verificar o impacto dos hiperparâmetros de cada política no desempenho do agente, e em cada jogo;
2. Verificar se as políticas conseguem melhorar o desempenho do agente em todos os jogos;
3. Verificar a diferença de desempenho de cada uma das políticas nos jogos;
4. Verificar se existe uma técnica que se saiu melhor que as outras em todos os jogos.

1.2 Organização do Trabalho

O presente trabalho está estruturado da seguinte maneira: o Capítulo 2 introduz os conceitos teóricos necessários para a compreensão dos experimentos realizados, como Aprendizado por reforço, *Deep Q-learning* e políticas de exploração; o Capítulo 3 apresenta os artigos que serviram de inspiração e como base teórica para desenvolvimento da metodologia; O Capítulo 4 descreve quais métodos são utilizados e como é realizada a avaliação dos experimentos; o Capítulo 5 apresenta a análise dos resultados obtidos nos experimentos; e por fim, o Capítulo 6 discute as conclusões do trabalho e propõe trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os conceitos teóricos utilizados ao longo do presente trabalho. Primeiramente são abordados os temas relacionados à RNA e à CNN. Em seguida são detalhados os principais elementos do aprendizado por reforço, como é realizado o aprendizado e qual a importância da exploração nesse processo.

2.1 Redes Neurais Artificiais

As RNAs são estruturas que têm como inspiração o cérebro humano, e possui como unidade mais básica o neurônio artificial (também chamado de Nó ou Perceptron). Os Nós da RNA (perceptrons) possuem uma série de conexões entre si como o cérebro biológico, principalmente a sua capacidade de se desenvolver ao longo do tempo com aprendizado [17]. A RNA possui três tipos de camadas: a entrada, a escondida e a saída. A primeira camada recebe os dados de entrada, a segunda é responsável pelo processamento da informação e a última retorna o resultado do processamento. Na Figura 2.1, apenas uma camada escondida é considerada, porém várias podem ser combinadas para aumentar a capacidade de abstração da rede. As RNAs que possuem mais de duas camadas escondidas geralmente são conhecidas como Redes Neurais Profundas, em inglês Deep Neural Network (DNN) [26].

Na Figura 2.1 os círculos representam os perceptrons e as linhas representam as sinapses, conexões entre os perceptrons. As sinapses são responsáveis por realizar a multiplicação da entrada com o peso, retornando o resultado para o perceptron. O perceptron ilustrado na Figura 2.2(b) são responsáveis por realizar a soma dos resultados das sinapses e em seguida aplicar uma função de ativação para produzir a saída [27]. A saída (axônio) do perceptron é determinada utilizando a Equação 2.1:

$$y = f(x_1 * w_1 + ... + x_n * w_n + b), \tag{2.1}$$

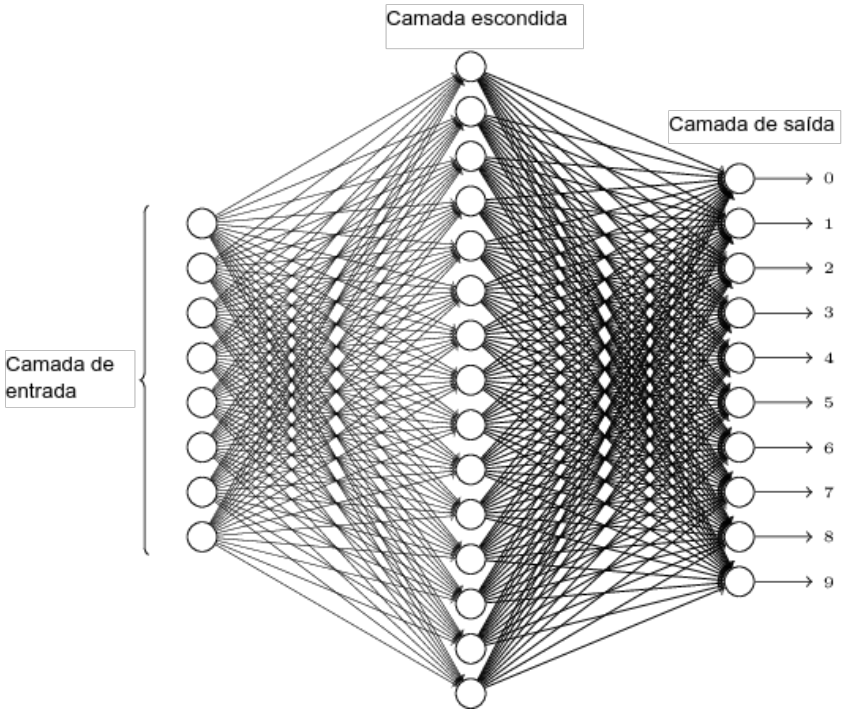

Figura 2.1: Exemplo de um tipo de estrutura de RNA (adaptado de Michael [1]).

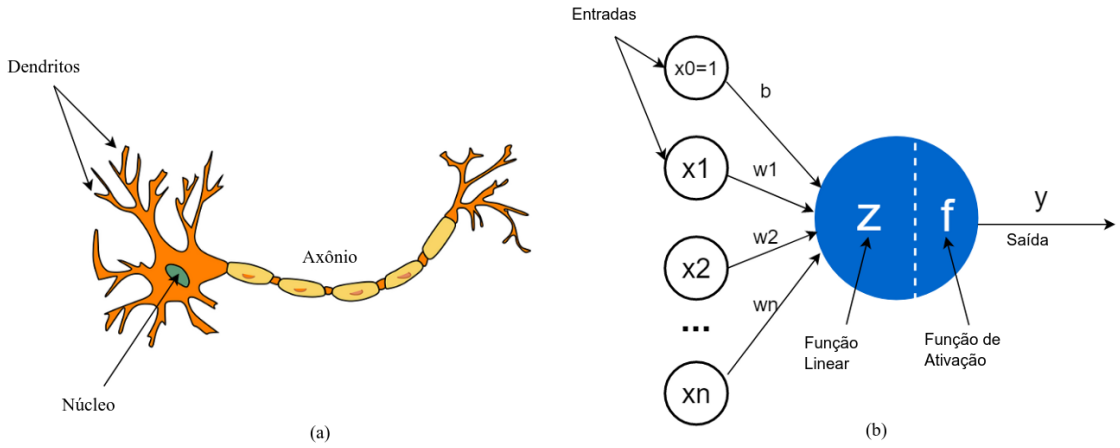

Figura 2.2: Comparação de similaridade entre as estruturas do neurônio humano e do neurônio artificial (adaptado de Rukshan [2]): (a) a estrutura do neurônio humano e (b) a estrutura do perceptron; Ambos recebem um sinal pela entrada ou dendrito, processam o sinal no núcleo e retornam o resultado da computação pela saída ou axônio.

sendo $n \in \mathbb{N}$, $x_1, ..., x_n$ as entradas (dendritos), $w_1, ..., w_n$ os pesos de cada entrada, b o bias, entrada com valor 1, e f a função de ativação que determina se um perceptron será ativado ou não. Além disso, a função de ativação também permite aprender relações não

lineares entre os dados tornando possível a solução de problemas complexos [17]. A mais comum é a função de retificação (ReLU) [26], determinada pela Equação 2.2:

$$ReLU(x) = max(0, x), \tag{2.2}$$

que é utilizada em todas as camadas escondidas da RNA no presente trabalho.

A RNA inicialmente não tem nenhum conhecimento, por isso precisa ser treinada. O processo de treinamento tem como objetivo ensinar para o modelo os valores ideias dos pesos, de forma que seja encontrada uma relação entre as entradas fornecidas e as saídas esperadas. Durante o treinamento, a RNA é alimentada com uma determinada entrada, que é processada e produz uma estimativa de resposta y sendo que a resposta esperada é $\hat{y}$. Uma função de perda (*loss* em inglês) é utilizada para calcular o erro entre o valor esperado e o estimado, essa medida indica o quão distante a RNA está da resposta. Baseado no valor do erro são ajustados os pesos de cada neurônio, reforçando conexões positivas, que diminuem o erro, e enfraquecendo conexões negativas, esse processo pode ser realizado por um otimizador, utilizando o algoritmo de *backpropagation*. Quanto maior a quantidade e variedade dos dados utilizados durante o treinamento melhor será capacidade da RNA de produzir o resultado esperado [17].

Em resumo, a RNA recebe como entrada um conjunto de dados de treinamento por vez, e para cada um é feita uma predição (estimativa). Para cada neurônio, são reforçadas as conexões com as entradas que contribuíram para a predição correta. Nesse processo um hiperparâmetro *learning rate* pode ser utilizado para controlar o ritmo de atualização dos pesos da RNA, i.e. determinar como a RNA deve se adaptar aos dados utilizando o erro. Valores muito baixos de *learning rate* podem resultar em um processo de treinamento mais longo enquanto um valor grande pode resultar em um treinamento instável ou com valores sub-ótimos para os pesos da RNA porque ela terá uma tendência de esquecer os dados mais antigos [7]. Em essência as RNAs são funções flexíveis que podem ser utilizadas para resolver qualquer problema apenas variando seus pesos [28]. O teorema da aproximação universal [29] prova que as RNAs podem modelar qualquer função não linear, em teoria, o que comprova essa propriedade.

2.2 Redes Neurais Convolucionais

No presente trabalho serão utilizadas RNAs como agente, tomador de decisão. O agente recebe como entrada a imagem da tela do jogo e retorna como saída uma ação a ser tomada por ele no ambiente. Para o processamento de imagens serão utilizadas as CNNs, que associam as RNAs com camadas convolucionais. A razão para utilizar essa arquitetura é que ela leva em consideração a estrutura espacial das imagens [17]. As CNNs são utilizadas

para o processamento de imagens porque usam uma arquitetura adaptada para essa tarefa. Elas são capazes de atribuir importância aos objetos presentes na imagem e de diferenciar um do outro [17]. Na arquitetura das CNNs as camadas são divididas em duas partes (Figura 2.3): (1) camadas convolucionais, responsáveis por processar os dados da imagem, extraindo suas características; e (2) as camadas densas, que diferente das convolucionais são compostas por neurônios totalmente conectados, e são responsáveis por aprender como os dados processados da imagem se relacionam com as saídas esperadas.

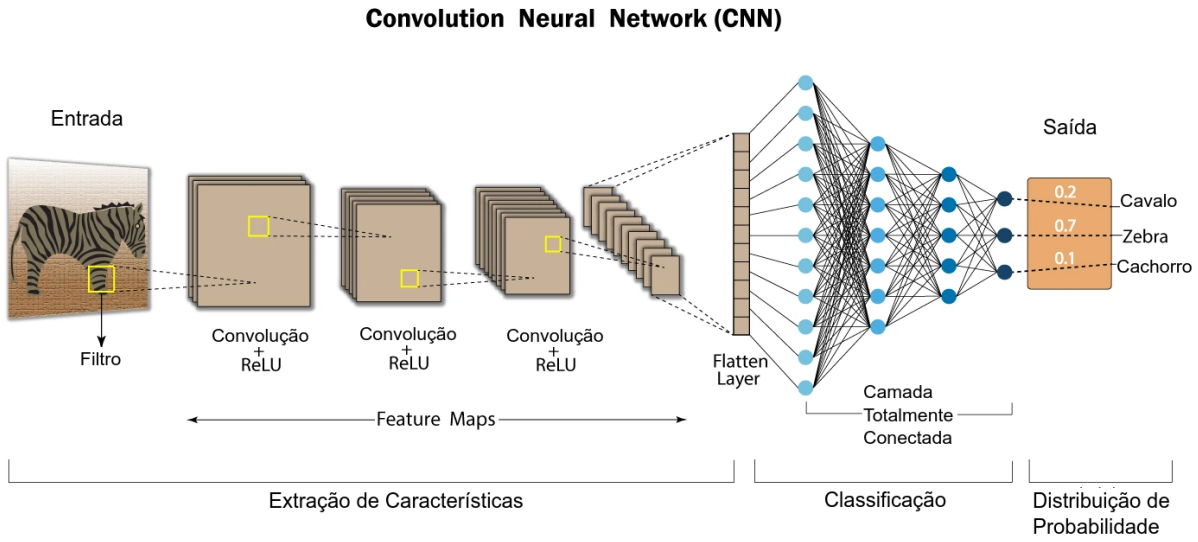

Figura 2.3: Camadas de uma Rede Neural Convolucional: exemplo de classificação de animais com base na imagem (Adaptado de Swapna [3]). Camadas convolucionais fazem a extração de características e camadas densas a classificação.

Na Figura 2.3, o conjunto de camadas convolucionais é responsável por reduzir a imagem para uma forma simples, sem perder características que seriam essenciais para a classificação. Para isso é realizado o processo de convolução, que consiste em deslizar um filtro sobre uma imagem de entrada e realizar o produto escalar dos valores que se sobrepõem criando uma imagem de saída chamada *feature map*. Durante esse processo o tamanho da imagem inicial se reduz por causa da limitação de movimento do filtro sobre a imagem de entrada, além disso a matriz resultante pode ser ainda menor se a convolução utilizar *stride*, argumento que regula o tamanho do deslocamento do filtro, maior que um. A Figura 2.4 mostra como o processo de convolução com *stride* dois diminui as dimensões da entrada.

Durante a convolução podem ser utilizados diferentes filtros, a primeira camada convolucional captura características simples, como linhas horizontais e verticais, que em camada convolucionais profundas são combinadas para obter características mais comple-

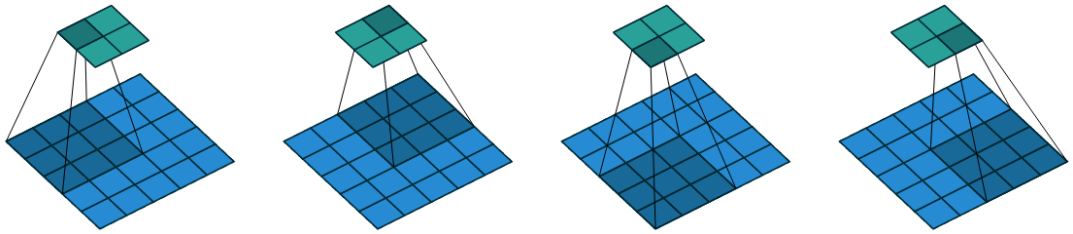

Figura 2.4: Processo de convolução utilizando filtro 3×3 com imagem 5×5 usando *stride* 2×2 [4].

xas, permitindo a CNN ter melhor entendimento da imagem. Após a filtragem, os *feature maps* podem ser aplicados em um função de ativação como a ReLU, para enfatizar as características selecionadas. As camadas densas possuem o mesmo funcionamento das RNA. Treinar uma rede convolucional significa encontrar os melhores valores para utilizar nos filtros das camadas convolucionais e nos pesos das camadas densas. O processo de treinamento é o mesmo utilizado na RNA como apresentado anteriormente.

No processamento de *frames* do jogo, as CNNs podem ser responsáveis por extrair da imagem, as posições dos jogadores e a posição da bola, por exemplo. De posse dessas caraterísticas, as camadas densas são capazes de compreender o ambiente e determinar qual ação deve ser tomada pelo agente [10].

2.3 Aprendizado por reforço

No AR, o objetivo é, por meio da interação com o ambiente, aprender por tentativa e erro, como alcançar a maior recompensa em longo prazo [16]. O agente deve aprender como suas ações interferem no estado do ambiente, percebendo que algumas ações são mais importantes, em certos estados, do que outras, para conseguir alcançar a maior pontuação possível, por exemplo, no final de um jogo. O agente não recebe diretamente as instruções de qual ação deve tomar, ele deve decidir baseado em experiências passadas, obtidas pela interação com o ambiente. Em cada interação com o ambiente, o agente tenta novas estratégias procurando uma melhor forma de resolver o problema. Para avaliar a qualidade da solução, cada ação tomada recebe uma recompensa imediata, que contribui para a pontuação final.

Alguns componentes importantes para a definição e compreensão do problema de aprendizado por reforço são [16, 30]:

1. Agente: capaz de tomar decisões e realizar ações em um ambiente. Na prática, é o algoritmo a ser desenvolvido;
2. Ambiente: tudo com o que o agente interage. Representa o problema a ser resolvido. No presente trabalho, é o jogo de Atari;
3. Ação: movimento possível de ser realizado no ambiente pelo agente. É limitada pelas regras do ambiente, no presente trabalho, cada jogo tem ações possíveis predefinidas (discretas);
4. Observação (estado): estado do ambiente visível pelo agente em determinado tempo. É a forma do agente verificar como suas ações afetam o ambiente;
5. Recompensa: valor numérico recebido após cada ação, que pode ser positivo ou negativo. O objetivo da recompensa é indicar ao agente o quão boa são suas ações, reforçando comportamentos de forma positiva ou negativa;
6. Política: estratégia aplicada pelo agente na escolha de uma ação em determinado estado.

O agente interage com o ambiente por meio de ações e o ambiente interage com o agente por meio de observações e recompensas. A interação contínua entre o ambiente e o agente descreve um problema de aprendizado por reforço. O agente percebe o ambiente por meio de observação, e com base nela seleciona ações que produzem algum efeito no ambiente. A partir dessas ações o ambiente, se modifica, transita para um novo estado, e retorna para o agente uma recompensa. Por meio dessa interação o agente tem como objetivo maximizar o seu ganho ao longo do tempo (ganho cumulativo total). O relacionamento entre agente e ambiente pode ser observado na Figura 2.5 sendo A_t a ação executada pelo agente no tempo t, S_{t+1} e R_{t+1} são, respectivamente, o estado e recompensa do ambiente após a ação A_t. Um dos desafios desse relacionamento é que o sinal de recompensa é frequentemente esparso, com ruídos e atrasada em relação à ação [21].

2.4 Algoritmo *Q-learning*

O principal desafio do AR é o agente conseguir atingir um objetivo sem antes compreender como funciona o ambiente, e como suas ações influenciam nas mudanças de estado. O algoritmo *Q-learning* [7, 30, 31] é utilizado pelo agente para armazenar o resultado de todas suas experiências no ambiente, dessa forma ele constrói um mapa de boas ações para cada situação, e é capaz de tomar decisões inteligentes para maximizar a recompensas total.

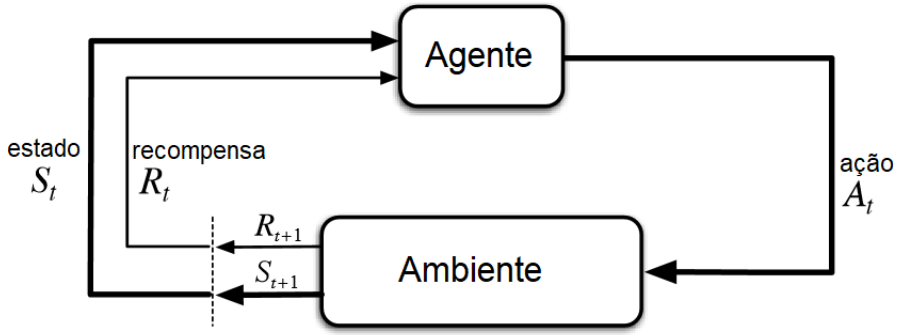

Figura 2.5: Ciclo de interação entre agente e ambiente no aprendizado por reforço [5].

Em sua forma mais simples o *Q-learning* (Figura 2.6) é uma tabela que mapeia estado e ações para um Q-valor (*Quality value*), que é o valor de tomar uma ação em determinado estado. Esse valor é definido como a recompensa total esperada de um estado [30]. O Q-valor é calculado baseado em interações anteriores do agente com o ambiente e leva em conta as possíveis ações futuras que o agente pode tomar. Quanto maior o Q-valor maiores as chances de se obter maiores recompensas no futuro. O *Q-learning* funciona observando um agente jogar e gradualmente melhora suas estimativas de Q-valores [7]. Para entender melhor como o *Q-learning* funciona é necessário compreender as propriedades das cadeias de Markov.

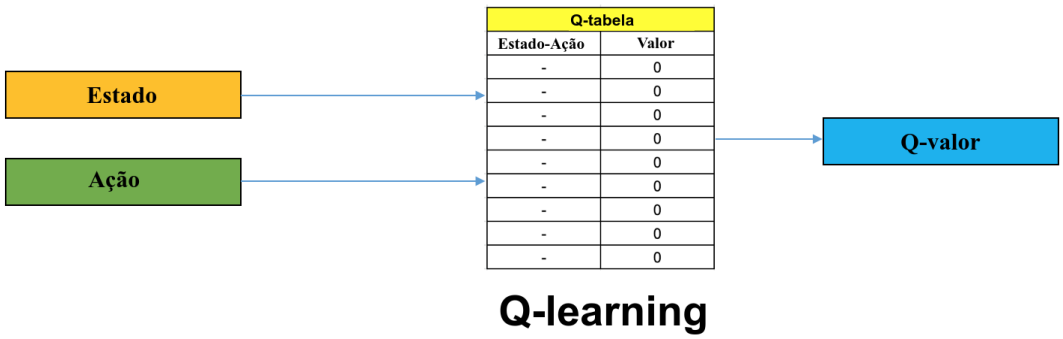

Figura 2.6: Tabela do *Q-learning*: recebe como entrada um estado e uma ação e tem como saída um Q-valor (adaptado de Choudhary [6]).

As Cadeias de Markov ou Processos de Markov, em inglês *Markov Process* (MP) [16], são modelos matemáticos utilizados para descrever como um sistema transita de um estado para o outro ao longo do tempo. Essas cadeias representam cada estado e as probabilidades de transição entre um estado e outro, como pode ser observado na Figura 2.7. Por definição, esses processos são estocásticos sem memória, possuem um número finito de

estados e a transição de um estado para o outro é feito de forma aleatória. Além disso, consideram a propriedade de Markov que diz que a probabilidade de transição para o próximo estado do sistema depende apenas do estado atual e não dos estados passados [32]. Essa propriedade permite que durante o aprendizado o agente seja capaz de tomar decisões apenas com a observação atual do ambiente. Na Figura 2.7 está ilustrado um sistema que possui 4 estados (S_0, S_1, S_2, S_3) e as probabilidades de transição para cada estado, por exemplo, no estado S_0 o sistema tem probabilidade de 70% de se manter no mesmo estado, probabilidade de 20% de transitar para o estado S_1 e probabilidade de 10% de chance de transitar para o estado S_3. Para o MP ser utilizado no aprendizado por reforço é necessário estender a definição adicionando a decisão do agente e recompensas na transição dos estados.

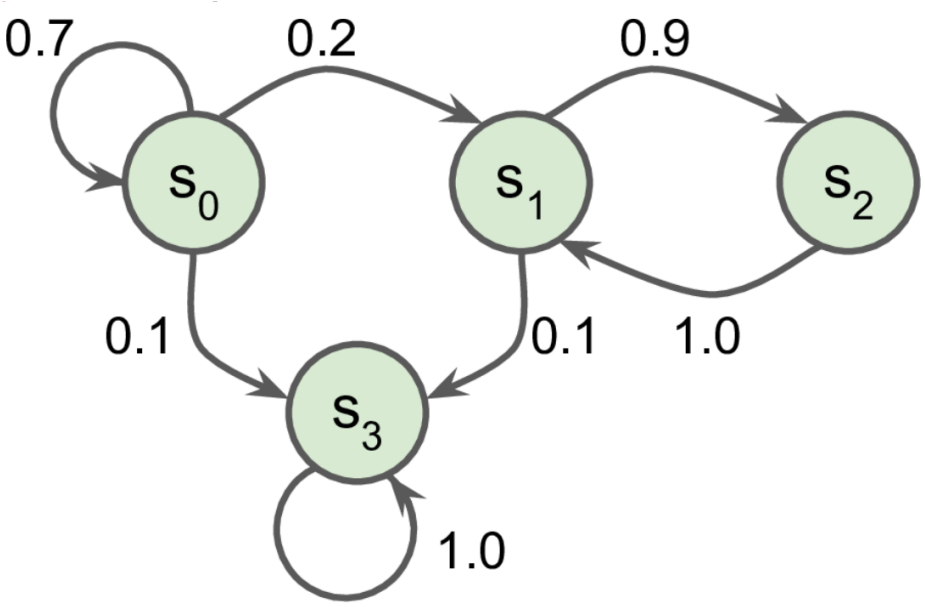

Figura 2.7: Exemplo de Cadeia de Markov: os círculos (S_0, S_1, S_2, S_3) representam os estados e as setas as probabilidades de transição de um estado para o outro [7].

O matemático Richard Bellman foi responsável pela descrição dos Processos de Decisões de Markov, em inglês *Markov Decision Process* (MDP) [33]. MDPs são formulações de tomada de decisão em sequência, onde ações influenciam não só as recompensas imediatas, mas também estados e recompensas futuras [16]. As MDPs (Figura 2.8) são muito parecidas com as MPs, mas a cada etapa o agente pode escolher entre diferentes ações que afetam diretamente as probabilidades de transição de estado. Além disso, as transições de estado retornam recompensas para o agente com o objetivo de maximizar a recompensa final ao longo do tempo [7]. A equação de Bellman [7, 33] (Equação 2.3) estima qual deve ser o valor ótimo para cada par estado-ação, chamados de Q-valores. Essa equação

é utilizada para atualizar os Q-valores durante o processo de treinamento do agente. A equação se baseia na intuição: conhecendo-se o Q-valor do próximo estado $Q(s', a')$ para todas as ações, a melhor estratégia seria selecionar a ação a', que maximiza a recompensa esperada [21]. A partir da determinação dos Q-valores é possível definir uma política ótima para obter recompensas, selecionando sempre a ação com maior Q-valor.

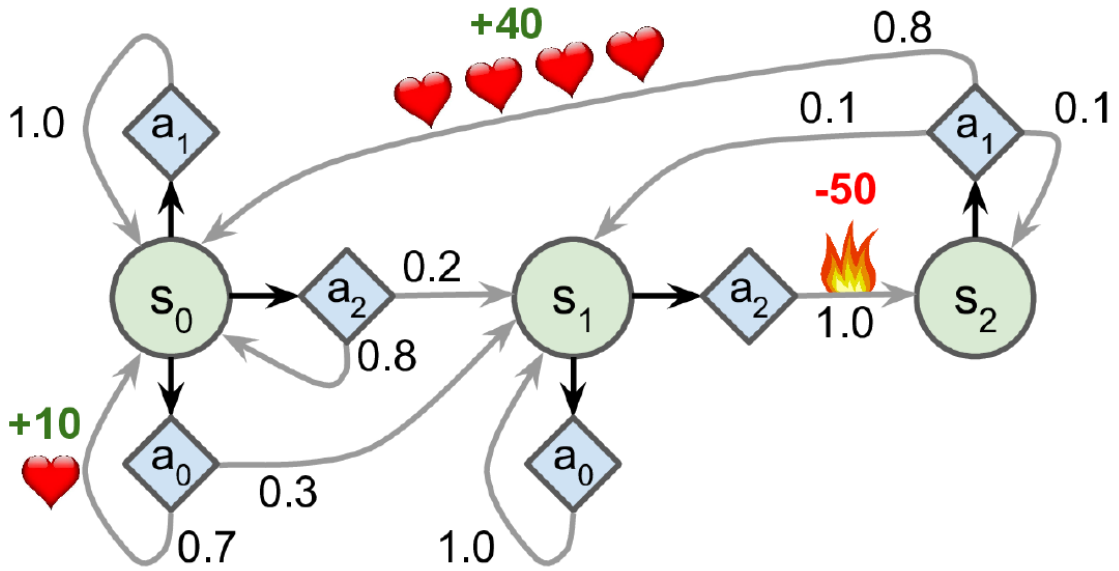

Figura 2.8: Exemplo de Processo de Decisão de Markov: os círculos (S_0, S_1, S_2) representam os estados, as setas as probabilidades de transição de um estado para o outro condicionadas a uma ação (A_0, A_1, A_2) e em alguns casos a transição recebe recompensa positiva (coração) ou negativa (fogo) [7].

$$Q(s, a) = r + \gamma(max(Q(s', a')). \tag{2.3}$$

Na Equação 2.3, um Q-valor $Q(s, a)$ no estado s e ação a é obtido pela soma da recompensa atual r mais a recompensa máxima futura $max(Q(s', a'))$ no próximo estado s' descontado γ a cada passo. O valor de γ indica o grau de importância dado para as recompensas futuras, quanto mais perto do valor 1 (um) maior a importância, já no caso do valor 0 (zero) significa que apenas recompensas imediatas são consideradas.

Os algoritmos de aprendizado por reforço aprendem a determinar os Q-valores por meio da exploração do ambiente. O *Q-learning* é denominado como *off-policy*, isso significa que a política de seleção de ações utilizada durante a etapa de treinamento (aprendizado) do agente pode ser diferente da utilizada durante a etapa de avaliação. Normalmente, durante a etapa de avaliação do agente, é utilizada a política de decisão *Greedy*, que sempre escolhe

a melhor ação conhecida. Isso ocorre porque jogando várias vezes o mesmo jogo, na média boas ações terão retorno maior que as ações ruins [16].

Existem algumas limitações para o algoritmo *Q-learning*, são elas: (1) funcionar apenas para ambientes que possuem um conjunto de ações discretas e pequeno; (2) não ser aplicável para ações contínuas; (3) não aprender políticas que se modificam ao longo do tempo (limitação das MDPs [30]); (4) precisar de muitas amostras para o algoritmos convergir; (5) não é garantido que o algoritmo convirja [7]; e (6) com um número de estados muito grande no ambiente, se torna impossível manter uma tabela que mapeia estados e ações em memória. Para resolver esse último podemos utilizar redes neurais para substituir as tabelas [21, 16].

2.5 *Deep Q-learning*

Na DQN, a CNN substitui a tabela utilizada no algoritmo tradicional de AR. A DQN recebe como entrada um estado do ambiente e retorna como saída um Q-valor para cada uma das ações possíveis. Esse processo evita ter que utilizar a DQN para estimar um Q-valor para cada conjunto estado-ação de forma individual. Dessa forma o Q-valor é avaliado uma única vez por estado, o que torna o treinamento do agente mais eficiente. A Figura 2.9 ilustra como é realizada a avaliação de ação baseada no estado, a DQN recebe como entrada uma observação s da tela do jogo e tem como saída N Q-valores $Q(s, a_n)$.

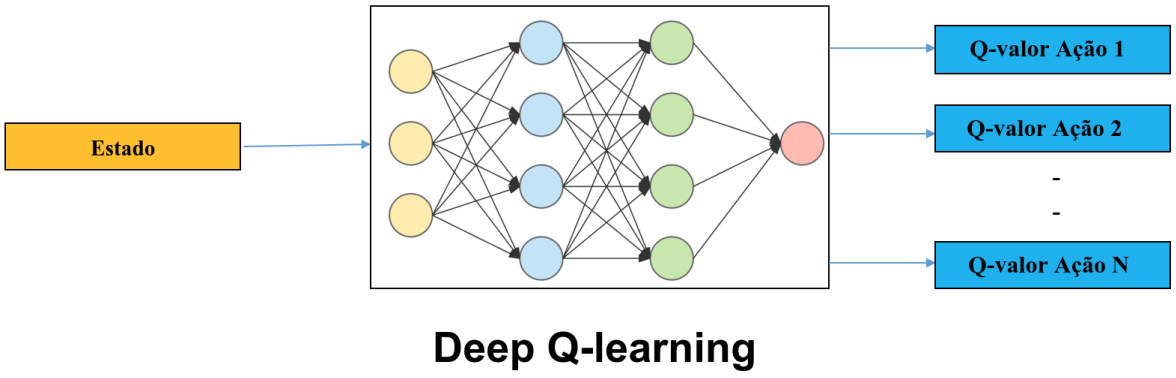

Figura 2.9: Avaliação da DQN: recebe como entrada uma observação s da tela do jogo e tem como saída N Q-valores $Q(s, a_n)$ (adaptado de Choudhary [6]).

No aprendizado por reforço o treinamento é realizado em ambientes dinâmicos, portanto, não se utiliza uma base de dados. Através da interação com o ambiente são obtidos: estado, ação e recompensa. Essas informações são utilizadas para atualizar a DQN e para calcular os Q-valores. O processo de treinamento do agente consiste nos seguintes passos:

1. Observação do Ambiente;
2. Decisão de ação a partir de uma política;
3. Execução da ação;
4. Recebimento de recompensa: positiva ou negativa;
5. Aprendizado com a experiência (treinamento da RNA);
6. Iteração até chegar em uma estratégia ótima.

Durante o treinamento da DQN, a função de perda, que define como calcular o erro entre o Q-valor estimado e o Q-valor real, é utilizado para atualizar os pesos da DQN [10]. O objetivo da DQN é estimar com exatidão os Q-valores para todos os estados possíveis, para que o agente consiga selecionar as melhores ações e obter a maior. A Equação 2.3 representa esse objetivo (*target*), que é o valor ideal para os Q-valores. O *Q-loss* $\mathcal{L}_i(\theta_i)$ descrito na Equação 2.4 é a função de perda utilizada na DQN para definir como calcular o erro quadrático médio em relação aos Q-valores estimados pela DQN $Q(s, a; \theta_i)$ e o ideal (*target*) $r + \gamma(max(Q(s', a'; \theta_i^-)$. O objetivo do treinamento é definir os pesos da DQN de forma que esse erro seja o menor possível. O erro é reduzido dando um passo na direção do objetivo (*target*), que representa a direção que deseja se mover. Na Equação 2.4 o valor estimado do Q-valor é representado por $Q(s, a; \theta_i)$ sendo que θ_i representa os pesos da DQN sendo treinada no momento i e θ_i^- representa os pesos utilizados na rede ideal (*target*) no momento i [10]. Os parâmetros da rede *target* θ_i^- são atualizados a cada N passos e se mantém constante durante outras atualizações da DQN.

$$\mathcal{L}_i(\theta_i) = \mathbb{E}[||(r + \gamma(max(Q(s', a'; \theta_i^-)) - Q(s, a; \theta_i)||^2] \tag{2.4}$$

Para superar algumas das limitações do *Q-learning*, apresentadas na seção anterior, principalmente o problema de conversão, e também para possibilitar a utilização do *Q-learning* no ambiente do Atari, foram propostas por Mnih et al. [10] duas melhorias para o *Deep Q-learning*: (1) *Experience Replay*, e (2) *Target Network*.

No *Deep Learning* se assume que os dados de treinamento são independentes entre si, enquanto no AR os dados são muito relacionados entre si. Além disso, no AR a distribuição dos dados muda à medida que o algoritmo aprende novos comportamentos, o que se torna um problema para o *Deep Learning* que assume uma distribuição fixa. Para aliviar esses problemas um mecanismo de *experience replay* é utilizado para amostrar transições anteriores de forma aleatória. Esse processo diminui as variações de distribuição nos dados de treino [21], ou seja, todo o treinamento é realizado utilizando experiências anteriores do agente, que durante a interação com o ambiente foram armazenadas no

replay memory após cada ação. Uma experiência consiste em um estado inicial, ação, recompensa e estado final. No treinamento de uma experiência, o estado inicial é utilizado como entrada da DQN, o Q-valor estimado da ação escolhida é comparado com o Q-valor esperado, calculado utilizando o próximo estado e recompensa. Com base no erro gerado pela comparação são ajustados os pesos da DQN, dessa forma o treinamento se aproxima do paradigma supervisionado. O algoritmo completo para o treinamento da *deep Q-network* está representado no Algoritmo 1 sendo $\mathcal{M}$ e $\mathcal{T}$ o número de episódios e o número de *frames*, respectivamente.

Algorithm 1: *Deep Q-learning* com *experience replay* (adaptado de Mnih et al. [10])

Inicializa *replay memory* $\mathcal{D}$ com capacidade C
Inicializa função de ação-valor Q com pesos aleatórios θ
Inicializa função de ação-valor $\hat{Q}$ com pesos aleatórios $\theta^- = \theta$
for *episodio=1, M* **do**
    Inicializa emulador com estado x_1 e realiza o preprocessamento $\phi_1 = \phi(x_1)$
    for *t=1, T* **do**
        Seleciona ação a_t a partir de uma política de exploração, abordado na próxima seção
        Executa ação a_t no emulador e observa recompensa r_t e novo estado x_{t+1}
        Preprocessa $\phi_{t+1} = \phi(x_{t+1})$
        Armazena experiência $(\phi_t, \mathrm{a}_t, r_t, \phi_{t+1})$ em $\mathcal{D}$
        Amostra *mini-batch* de experiências $(\phi_j, \mathrm{a}_j, r_j, \phi_{j+1})$ de $\mathcal{D}$
        Define $y_j = r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)$
        Atualiza os pesos θ da rede Q utilizando a equação $(Q(x_j, a_j; \theta) - y_j)^2$
        A cada N passos atualiza $\hat{Q} = Q$
    end
end

As vantagens de se utilizar o *experience replay* são [21]: (1) cada experiência pode ser utilizada potencialmente para várias atualizações de peso da DQN, o que ajuda no uso eficiente dos dados; (2) o fato das experiências serem selecionadas de forma aleatória quebra as correlações entre elas e reduz a variância nas atualizações da DQN; (3) a distribuição de ações é suavizada com os vários estados anteriores, tornando o treinamento mais estável; (4) a estratégia de atualização iterativa da DQN ajusta os Q-valores estimados na direção dos Q-valores da *target network*, uma vez que reduz a relação entre os estimados e o ideal

(*target*), considerando que a *target network* é atualizada apenas a cada N passos.

2.6 Políticas de Exploração

Um dos problemas mais recorrentes do aprendizado por reforço é o problema de *exploration-exploitation*, em que o agente deve ser capaz de escolher quando deve explorar o ambiente e quando deve selecionar a melhor ação já conhecida [16]. O algoritmo de exploração é o que influencia o agente em sua tomada de decisão, quanto mais se explora um ambiente mais conhecimento é adquirido sobre ele, ou seja, melhor serão ajustados os Q-valores da DQN para os estados conhecidos.

A exploração no aprendizado por reforço é uma tentativa de adquirir mais conhecimento a respeito do ambiente, mas esse conhecimento possui um custo associado. No melhor dos casos, o agente usando o conhecimento já adquirido pode determinar quais partes do ambiente devem ser exploradas. O aprendizado eficiente não pode ser realizado sem um grau de exploração para que os comportamentos ótimos possam ser identificados pelo agente[24]. Na prática as estratégias que serão apresentadas a seguir podem ser modificadas em diferentes graus de exploração, por meio de hiperperâmetros. Além disso, é importante notar que as estratégias podem ter eficiência diferente dependendo do tipo de ambiente [16, 24].

Uma das motivações de utilizar uma política de exploração eficiente é diminuir o tempo de aprendizado do agente. Se o agente tem uma forma eficaz e eficiente de aprender sobre o ambiente, ele será capaz de atingir o objetivo em menor tempo e com melhores recompensas [24]. Por outro lado, excesso de exploração pode gastar muito tempo e recurso no reconhecimento de partes do ambiente que são irrelevantes para o resultado final. Um exemplo de excesso de exploração, seria um agente ficar olhando fixamente para uma televisão com ruídos, ele sempre estará vendo um novo estado do ambiente, mas isso não auxilia no seu aprendizado [34].

Existem várias políticas de exploração propostas na literatura [10, 16, 24, 25], algumas delas são: (1) Aleatória, (2) *Greedy*, (3) *Epsilon Greedy*, (4) *Boltzmann*, (5) *Decaying Epsilon Greedy*, e (6) *Random Noise*. As políticas de 1 a 5 são estratégias clássicas da literatura e a política 6 foi proposta para DQNs.

2.6.1 Aleatória

Essa política consiste em selecionar ações aleatórias para o agente. Essa forma de exploração não é eficiente, mas serve como base para comparação com as outras formas de exploração apresentadas [10]. Por ser aleatória, totalmente exploratória, o agente apenas aprende de forma superficial sobre o ambiente, nunca aplicando o que foi aprendido,

tornando essa a estratégia menos efetiva [16]. O ideal é que ao longo do tempo o agente utilize seu conhecimento para tomar melhores decisões.

2.6.2 *Greedy*

Essa política consiste em sempre tomar a melhor decisão conhecida para determinada situação para maximizar a recompensa imediata, ou seja, o agente sempre seleciona a ação com o maior Q-valor. Se existir mais de uma ação *greedy*, então uma é selecionada de forma aleatória. Essa política pode ser descrito da seguinte maneira Equação 2.5:

$$A_t = max(Q_t(a_t)), \tag{2.5}$$

sendo que A_t denota a ação a_t que tem o maior Q-valor $max(Q_t(a_t))$ no tempo t [16].

As desvantagens dessa política é o agente poder ficar preso em um resultado ótimo local. Isso pode ocorrer, porque essa estratégia explora um grupo de ações específico enquanto ignora todas as outras possibilidades. Uma das consequência disso é que o agente pode não descobrir as ações ótimas para o ambiente ou pode sofrer *overfitting*, obtendo boa pontuação apenas em algumas configurações do ambiente [16].

2.6.3 *Epsilon Greedy*

Essa política consiste em agir de forma *greedy* (Seção 2.6.2) a maioria do tempo e com probabilidade ϵ agir de forma aleatória (Equação 2.6). Na Equação 2.6 é possível visualizar como o agente escolhe uma ação, com probabilidade ϵ ele age de forma *greedy* (Seção 2.6.2) e com probabilidade $1 - \epsilon$ ele age de forma aleatória. A desvantagem dessa política é executar muitas ações aleatórias quando pode existir uma melhor opção, ou seja, essa estratégia explora muito, até quando deveria optar por ações mais conservadoras, perdendo oportunidade de pontuações maiores [24]. Sendo que quanto mais ϵ está próximo de 1, maior a probabilidade de movimentos aleatórios. Apesar da desvantagem é uma ótima estratégia por conta da sua simplicidade e bons resultados para ambientes mais simples [16].

$$A_t = \begin{cases} max(Q_t(a_t)), & \text{com probabilidade } 1 - \epsilon \\ \text{qualquer ação}, & \text{com probabilidade } \epsilon \end{cases} \tag{2.6}$$

2.6.4 Boltzmann

A política de Boltzmann normaliza os Q-valores, Q_{a_i}, de cada ação utilizando a função *softmax* (Equação 2.7), e usa os valores resultantes como probabilidades para selecionar

as ações. As probabilidades são calculadas dividindo o exponencial do Q-valor de cada ação, $\exp(Q_{a_i})$, pela soma dos exponenciais dos Q-valores de todas as ações, sendo K o número de ações. A diferença da Equação 2.7 para a equação de Boltzmann (Equação 2.8) é que a última possui um parâmetro de temperatura τ que regula a distribuição das probabilidades. A medida que τ se aproxima de zero, o agente tende a exploração *Greedy* (Seção 2.6.3), e a medida que τ se aproxima de um o agente tende a exploração aleatória. É uma boa estratégia quando a melhor ação é mais evidente que as outras, mas sofre quando são muito parecidas [24]. Pode convergir de forma lenta dependendo da temperatura τ [24]. Na Figura 2.10 é possível observar como seria a distribuição de probabilidade para cada ação em um jogo que possui seis ações possíveis, com variações no valor de τ.

$$P(a_i) = \frac{\exp(Q_{a_i})}{\sum_{j=1}^{K} \exp(Q_{a_j})}, \tag{2.7}$$

$$P(a_i) = \frac{\exp(Q_{a_i}/\tau)}{\sum_{j=1}^{K} \exp(Q_{a_j}/\tau)}. \tag{2.8}$$

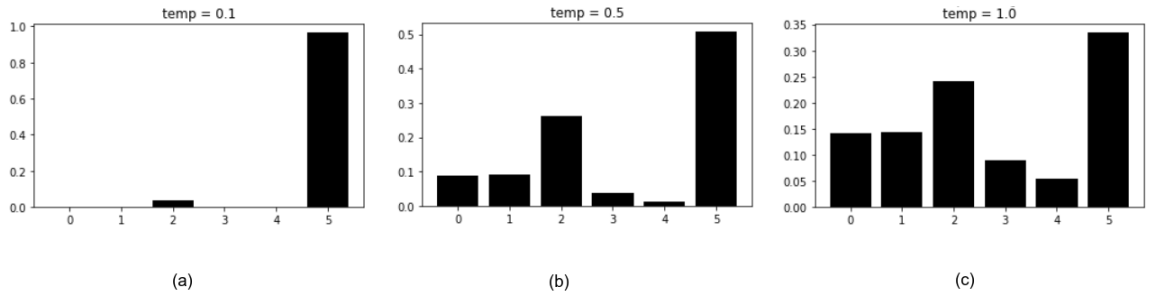

Figura 2.10: Comparação das distribuições de probabilidade com diferentes valores de τ utilizados na política de Boltzmann (adaptado de Meneghetti [8]): (a) $\tau = 0.1$, (b) $\tau = 0.5$ e (c) $\tau = 1.0$.

2.6.5 *Decaying Epsilon Greedy*

Essa política é semelhante ao *Epsilon Greedy* (Seção 2.6.3). A principal diferença entre elas é que na *Decaying Epsilon Greedy* ocorre a diminuição gradual do valor de ϵ com o passar do tempo até atingir um valor mínimo pré definido. Essa estratégia possui vantagem em relação às políticas anteriores, porque possui grande taxa de exploração no início do aprendizado e com o tempo diminui essa taxa, sendo capaz de aproveitar as oportunidades à medida que o agente aprende sobre o ambiente [30]. O maior desafio da política se torna descobrir como variar esse processo de decaimento do ϵ [16].

2.6.6 *Noise-based Exploration (Random Noise)*

A política *Random Noise* utiliza o método *Noisy Net* proposto por Fortunato et al. [25]. Nessa política ruídos paramétricos são adicionadas nos pesos das camadas densas da DQN para estimular a exploração do ambiente. Durante o treinamento, os parâmetros de ruído são apreendidos em conjunto com os pesos da DQN. A ideia principal é que uma mudança nos pesos induz uma mudança consistente e relacionada ao estado na política, que pode influenciar em muitos passos do agente [25]. A vantagem dessa exploração é a auto-regulação, que ocorre devido ao nível de ruído ser ajustado pelos pesos da DQN a medida que ocorre o aprendizado. Essa estratégia possui como parâmetro σ_0 que influencia no cálculo do valor inicial de σ, que é um dos parâmetros que influencia o nível de ruído nas camadas densas da DQN. Os algoritmos citados até agora são baseados em perturbações aleatórias nas ações do agente, o que é menos eficiente para seleção de ações em ambiente que necessitam de padrões complexos de comportamento [25].

Para melhor compreender como funciona essa estratégia, considere uma camada densa da DQN que possui p entradas e q saídas, representado pela Equação 2.1, sendo $x \in \mathbb{R}^p$ a entrada, $w \in \mathbb{R}^{q \times p}$ os pesos e $b \in \mathbb{R}$ o bias. A camada correspondente *Noisy Net* linear pode ser representada pela Equação 2.9:

$$y = (\mu^w + \sigma^w \odot \epsilon^w)x + \mu^b + \sigma^b \odot \epsilon^b, \tag{2.9}$$

sendo $\mu^w + \sigma^w \odot \epsilon^w$ e $\mu^b + \sigma^b \odot \epsilon^b$ utilizadas para a substituição de w e b na Equação 2.1. Os parâmetros $\mu^w, \sigma^w \in \mathbb{R}^{q \times p}$ e $\mu^b, \sigma^b \in \mathbb{R}^q$ são treinados, enquanto $\epsilon^w \in \mathbb{R}^{q \times p}$ e $\epsilon^b \in \mathbb{R}^q$ são ruídos aleatórios que são gerados pela Equação 2.10 [25]:

$$f(x) = sgn(x)\sqrt{|x|}, \tag{2.10}$$

sendo x um valor gerado aleatoriamente.

Capítulo 3

Revisão Literária

O objetivo deste capítulo é explorar vários trabalhos que utilizam o aprendizado por reforço para solução de jogos, apresentar as suas contribuições e os principais marcos para evolução das técnicas que foram responsáveis por popularizar esse ramo de pesquisa.

Silver et al. [23] foram responsáveis pela criação do AlphaGo, o primeiro programa de computador capaz de ganhar do campeão mundial de Go. Go é conhecido como um dos jogos mais desafiadores para inteligência artificial devido a sua complexidade. De acordo com os autores, trabalhos anteriores que utilizavam algoritmos de árvore de busca, conseguiam apenas jogar no nível de jogadores amadores, por conta do enorme número de possíveis movimentos do Go. Para superar esses desafios, o AlphaGo combina técnicas de árvores de busca com RNA. As RNAs recebem como entrada uma descrição do tabuleiro do jogo Go, em seguida, uma das redes a *policy network* seleciona o próximo movimento e a outra rede, *value network*, prediz qual jogador será o vencedor. As RNAs foram treinadas utilizando uma combinação do aprendizado supervisionado com dados de *experts*, e aprendizado por reforço jogando contra si mesmo. A técnica de busca utilizada combina as avaliações das RNAs com o algoritmo de Monte Carlo. Depois de um longo tempo de treinamento o AlphaGo melhorou e se tornou cada vez melhor em aprender e tomar decisões. O AlphaGo adquiriu uma porcentagem de vitória de 99,8% ganhando 494 dos 495 jogos contra outros jogadores, sendo capaz de ganhar até mesmo dos campeões mundiais de Go. Segundo os autores, o AlphaGo se tornou um dos melhores jogadores de Go de todos os tempos.

Mnih et al. [10] foi um dos primeiros trabalhos a adquirir bons resultados utilizando aprendizado por reforço combinado com *deep learning* no Atari. O objetivo dos autores é criar um único agente capaz de jogar uma variedade de jogos disponíveis no Atari 2600. O artigo propõe uma forma de aplicar o aprendizado por reforço em um domínio, parcialmente observável e com grande número de estados para adquirir um desempenho comparável a jogadores profissionais nos 49 jogos testados. Para isso foi proposto a

utilização de três técnicas do *Deep Q-learning*: 1) CNN; 2) *experience replay*; e 3) *target deep Q-network* conhecida como DQN. Segundo os autores, essa metodologia foi utilizada porque torna o aprendizado mais estável e auxilia a CNN a convergir. O artigo mostra que os agentes foram capazes de aprender políticas de forma robusta para um número variado de jogos, utilizando somente as entradas e sem conhecimento prévio dos jogos. Todos os agentes foram treinados utilizando o mesmo algoritmo de aprendizado por reforço, arquitetura da DQN e hiperparâmetros. Os agentes foram treinados por 200 milhões de *frames* para cada jogo, utilizando a política de exploração *decaying e-greedy*. A métrica principal de avaliação foi a pontuação adquirida pelo agente durante a fase de avaliação. A fase de avaliação consiste em rodar 30 jogos, com duração máxima de cinco minutos cada, sendo a média das pontuações normalizada, em relação a pontuação humana, que é a métrica de comparação entre os jogos. Segundo os autores, o agente DQN proposto foi capaz de superar os trabalhos anteriores, em que o agente possuía conhecimento prévio do ambiente, na maioria dos jogos e desempenhou de forma parecida com um jogador humano na maior parte dos jogos.

Fortunato et al. [25] introduziram a *NoisyNet*, uma estratégia de exploração baseada em modificações na arquitetura da DQN. A proposta dos autores é adicionar ruído paramétrico aos pesos da DQN do agente de aprendizado por reforço. O objetivo é obter uma forma mais eficiente de exploração do ambiente. Os ruídos da DQN são utilizados para guiar a exploração. Essa estratégia é eficiente porque no início do treinamento a DQN terá grande influência do ruído, portanto irá explorar mais. Com o tempo a rede irá se adaptar ao ruído em algumas partes, tornando a exploração mais dinâmica em relação ao algoritmos clássicos de *Q-learning* que adicionam aleatoriedade nas ações do agente para estimular a exploração do ambiente. Essa estratégia adiciona maior dinamicidade na tomada de decisão do agente. Fortunato et al. utilizaram as mesmas técnicas de aprendizado por reforço e metodologia de avaliação do artigo Mnih et al. [10] com a diferença da estratégia de exploração. Como fonte de dados de treino foi utilizado o Atari 2600, uma versão diferente da *NoisyNet* foi treinada nos 57 diferentes jogos. Segundo Fortunato et al. [25], substituindo a forma de exploração convencional *decaying e-greedy* utilizada no trabalho de Mnih et al. [10] pela *NoisyNet* o desempenho do agente aumenta de forma substancial nos jogos de Atari. Os resultados obtidos utilizando *NoisyNet* tiveram mediana de pontuação normalizada 48% maior em relação ao artigo Mnih et al. [10].

Hessel et al. [12] exploram como a combinação de diferentes técnicas que expandem a capacidade do *Deep Q-learning* podem afetar o aprendizado do agente nos jogos de Atari. Os autores verificaram que combinando seis técnicas de *Deep Q-learning*: (1) *Double DQN*, (2) *Prioritized DQN*, (3) *Dueling DQN*, (4) *multi-step learning*, (5) *Distributional Q-learning* e (6) *Noisy DQN* é possível alcançar um desempenho muito superior, em

comparação com seu uso isolado. O desempenho superior em termos de eficiência seria em relação ao menor uso de dados durante o aprendizado, e em termos de eficácia seria em relação à avaliação de desempenho nos jogos. A avaliação do novo agente que combina todas as técnicas, denominada *Rainbow* pelos autores, foi realizada em 57 jogos do Atari 2600. As estratégias de treinamento e avaliação utilizadas são as mesmas do Mnih et al. [10] e van Hasselt et al. [35]. Durante o treinamento os agentes são avaliados a cada 1 milhão de passos, suspendendo o treinamento e avaliando o último agente por 500K *frames*, durante essa etapa os episódios foram truncados para um máximo de 108K *frames* [35]. Em comparação com Mnih et al. [10], o algoritmo *Rainbow* atinge o desempenho do DQN após 7 milhões de *frames*. Sua pontuação final nos jogos foi substancialmente maior que apenas utilizando as técnicas isoladas, obteve mediana normalizada de 231%, 152% maior que o DQN. Para avaliar a relevância de cada técnica, os autores treinaram vários agentes cada um sem uma das técnicas para perceber como essa falta afeta o aprendizado do agente. Em termos de desempenho mediano, o agente obteve resultados melhores com *Noisy Nets*, uma observação relevante é que a remoção dessa estratégia produziu uma perda de desempenho em alguns jogos, mas também produziu uma pequena melhora em outros jogos. Os autores demonstraram que várias expansões do DQN podem ser combinadas com sucesso em uma única técnica, produzindo resultados superiores aos das técnicas isoladas, nos jogos de Atari.

Os artigos apresentados neste capítulo foram os que mais influenciaram no presente trabalho, na definição do tema, definição da metodologia e métricas de avaliação. Mnih et al. [10] por ser pioneiro no desenvolvimento de *deep reinforcement learning* para jogos de Atari auxiliou na definição da metodologia de treinamento do agente, construção da arquitetura da rede e definição de hiperparâmetros. Hessel et al. [12] ajudou da decisão de comparar técnicas relacionadas ao *Q-learning*. Fortunato et al. [25], propõem uma das formas de exploração que serão utilizadas durante os experimentos. O diferencial do presente trabalho dos trabalhos abordados anteriormente é a comparação realizada entre mais de uma política de exploração do ambiente, comparando estratégias clássicas com uma mais moderna, proposta por Fortunato et al. Essa análise é relevante porque aborda de forma teórica e prática como é o impacto de estratégias de exploração na pontuação do agente nos jogos de Atari. Um resumo dos trabalhos citados acima é apresentado na Tabela 3.1, incluindo os autores, as técnicas utilizadas, o ambiente e a política de exploração.

Tabela 3.1: Comparação das técnicas recentes de aprendizado por reforço.

Autores	**Técnicas**	**Ambiente**	**Política de exploração**
Silver et al. [23] (2016)	*DL + Monte Carlo*	Go	*Monte Carlo*
Mnih et al. [10] (2015)	*DQL* + DQN	Atari 2600	*decaying e-greedy*
Fortunato et al. [25] (2017)	*DQL* + DQN + *NoiseNet*	Atari 2600	*Random Noise*
Hessel et al. [12] (2018)	*DQL + Rainbow*	Atari 2600	*Random Noise*

DQL - *Deep Q-learning*, DL - *Deep Learning*

Capítulo 4

Metodologia

Este capítulo apresenta as etapas da metodologia proposta para a realização dos experimentos representados na Figura 4.1. Na primeira seção são apresentados os ambientes utilizados, na segunda seção será descrito a etapa de pré-processamento das imagens das telas de jogos de Atari. Em seguida, é apresentada a arquitetura da DQN considerada e como é realizado o seu treinamento utilizando aprendizado por reforço, bem como, o processo de avaliação. Por fim, são apresentadas as políticas de exploração e os hiperparâmetros que serão utilizados nos experimentos.

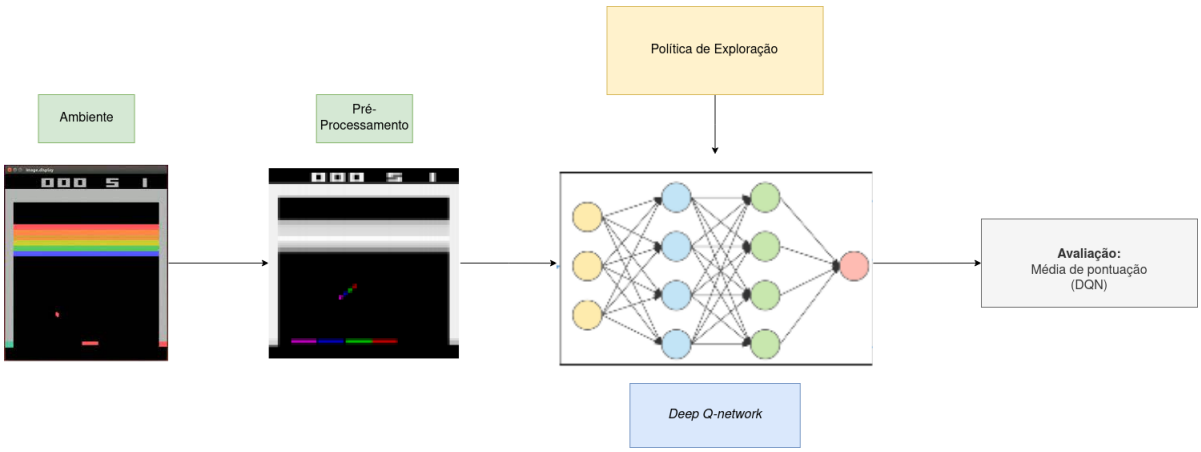

Figura 4.1: Fluxograma da metodologia utilizada.

O objetivo principal do presente trabalho é verificar como as diferentes políticas de exploração do ambiente afetam o desempenho do agente. Para isso, as seguintes políticas de exploração foram consideradas: (1) aleatória, (2) *greedy*, (3) *e-greedy*, (4) *decaying e-greedy*, (5) *boltzmann*, e (6) *random noise*. Cada uma das políticas foi utilizada para treinar um agente diferente a partir de jogos de Atari: (1) *Pong*; (2) *Breakout*; e (3) *Space*

Invaders. Elas serão comparadas utilizando como métrica a pontuação adquirida em cada um dos jogos.

4.1 Ambientes

Para analisar políticas de exploração do aprendizado por reforço, foram considerados três ambientes diferentes dos jogos do Atari 2600, implementados no *Arcade Learning Environment* (ALE) [22]. Como interface para o ALE foi utilizada a ferramenta OpenAI gym[1] [36]. Essa ferramenta padroniza a forma de interação com diversos *benchmarks* (ambientes) utilizados no aprendizado por reforço. A vantagem da interface simplificada é permitir que o agente interaja com qualquer ambiente sempre da mesma forma, sem necessidade de fazer mudanças no algoritmo quando se troca de jogo. As formas de interação são as mesmas do aprendizado por reforço: (1) o agente observa o estado do ambiente; (2) executa ações no ambiente; (3) recebe recompensas após cada ação; e (4) verifica o resultado da ação por meio de uma nova observação do ambiente. O OpenAI gym é um *benchmarks* utilizado para algoritmos de aprendizado por reforço, porque apresenta ao agente estímulos visuais e um conjunto de tarefas diverso e interessante, que foram feitas para serem desafiadoras para jogadores humanos [21]. No presente trabalho, serão utilizados três jogos de Atari: (1) *Pong*, (2) *Breakout*, e (3) *Space Invaders*.

4.1.1 *Pong*

Pong é um jogo em duas dimensões que simula um tênis de mesa. A Figura 4.2 apresenta a um *frame* do jogo durante uma partida, o agente controla a paleta (barra vertical) da direita, o adversário controlado pelo computador controla a paleta da esquerda, e o ponto branco representa a bola. Quando a bola é rebatida pela paleta de qualquer um dos jogadores a sua velocidade aumenta, reiniciando sua velocidade quando não é acertada. No *Pong* o agente deve fazer mais pontos do que o seu adversário para vencer. O objetivo do jogo é fazer a bola passar pelo oponente, não permitindo que o oponente consiga retornar a bola para o outro lado. Durante o jogo, o agente deve escolher quando utilizar cada uma das três possíveis ações do jogo, que são ficar parado ou mover-se para cima ou para baixo. Além disso, é importante levar em conta que a bola aumenta de velocidade cada vez que é rebatida. A cada ponto feito pelo agente ele recebe uma recompensa +1, a cada ponto feito pelo adversário a recompensa é -1, e em todos os outros estados a recompensa é 0. O jogador que atingir primeiro a pontuação de 21 vence o jogo. A recompensa final

[1] `https://gym.openai.com/`

do agente é calculada da seguinte forma: $pontuação_{agente} - pontuação_{adversário}$. Portanto, a recompensa final varia de -21 a 21.

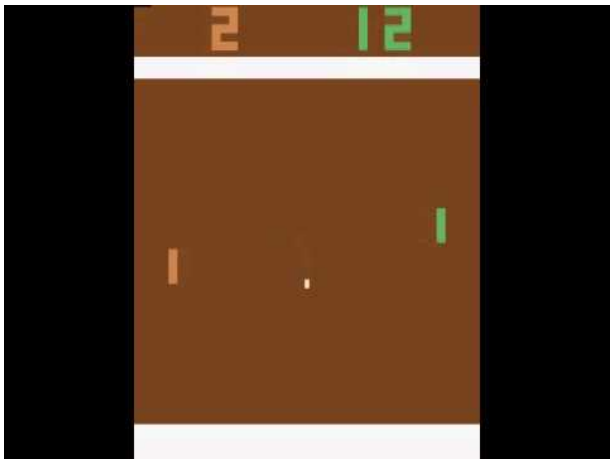

Figura 4.2: Um frame da tela do jogo *Pong*.

4.1.2 *Breakout*

No jogo *Breakout* o objetivo do agente é quebrar todos os blocos para obter a maior pontuação possível. A Figura 4.3 apresenta um *frame* do jogo durante uma partida o agente controla a pá (barra horizontal) e o ponto vermelho representa a bola. Quando a bola é rebatida pela pá a sua velocidade aumenta, se não for acertada a sua velocidade é reiniciada. Para adquirir pontos, o agente deve mover a pá para evitar que a bola caia para fora do jogo, ao mesmo tempo em que a direciona para atingir os blocos. Ao atingir um bloco a pontuação do jogo é incrementada, quanto mais distante a fileira de blocos, maior a pontuação obtida. Existem três possíveis ações no jogo: ficar parado, mover-se para direita, mover-se para esquerdo. A recompensa é zero em todos os estados, exceto no estado em que um dos blocos é atingido. O jogo termina quando todas as cinco vidas do agente são utilizadas. Uma vida é perdida toda vez que a bola cai para fora do jogo.

4.1.3 *Space Invaders*

No jogo *Space Invaders* o agente é uma nave, seu objetivo é destruir os *aliens* que avançam em sua direção enquanto se defende dos ataques desferidos pelos invasores, escondendo-se atrás de asteroides. A velocidade em que os *aliens* se aproximam aumenta à medida que o jogo passa. Ao destruí-los o agente adquire pontos, para avançar nas fases, todos os *aliens* devem ser destruídos. A Figura 4.4 apresenta a tela do jogo, a nave verde é controlada pelo agente, os blocos vermelhos são os asteroides e os *aliens* estão representados em cinza.

Figura 4.3: Um frame da tela do jogo *Breakout*.

Existem quatro possíveis ações no jogo: ficar parado, mover-se para direita ou esquerda e atirar. A recompensa é zero em todos os estados, exceto naquele em que um dos *aliens* é destruído. O jogo termina quando todas as três vidas do agente são utilizadas, uma vida é perdida toda vez que o agente é destruído.

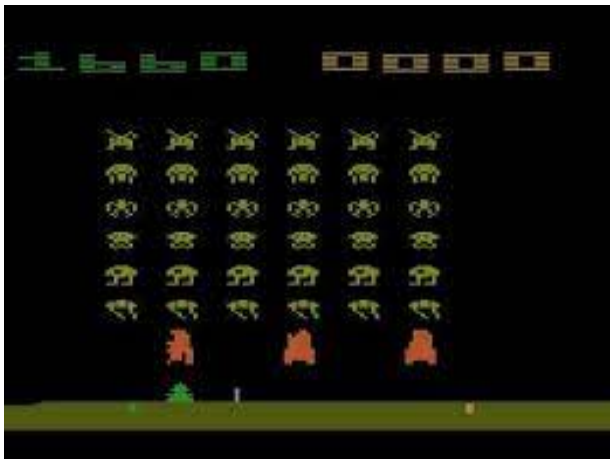

Figura 4.4: Um frame da tela do jogo *Space Invaders*.

4.2 Pré-processamento

Por ser custoso em termos de memória e custo computacional, os *pixels* coloridos da tela do jogo do Atari, recebidos do emulador, devem ser pré-processados para simplificar

o nível de informação processada pela DQN [21]. As estratégias de pré-processamento utilizadas no presente trabalho foram baseadas nas etapas propostas por Mnih et al. [10]: (1) conversão de escala de cor, (2) redimensionamento, e (3) empilhamento de *frames*.

Primeiro, foi realizada a conversão de escala de cor da imagem para níveis de cinza, visando diminuir a quantidade de dados e simplificar o processamento. Em seguida, a dimensão das imagens foram reduzidas utilizando interpolação [21] de 210×160 *pixels* para 84×84 *pixels* para diminuir o uso de memória. Por fim, quatro *frames* do jogo são empilhados para produzir uma observação com dimensões $84 \times 84 \times 4$ *pixels*, utilizada como entrada da DQN. O empilhamento de *frames* é necessário para o agente adquirir um conjunto de informações mais completo sobre o estado do ambiente, incluindo velocidade e direção de movimento dos objetos, essencial para os jogos. Além disso, permite que as estratégias de aprendizado por reforço possam ser utilizadas, por respeitar a propriedade de Markov, que diz que a transição de estados deve depender apenas do estado atual e não do histórico de estados passados [32]. Na Figura 4.5 é possível visualizar como é um estado observado pelo agente.

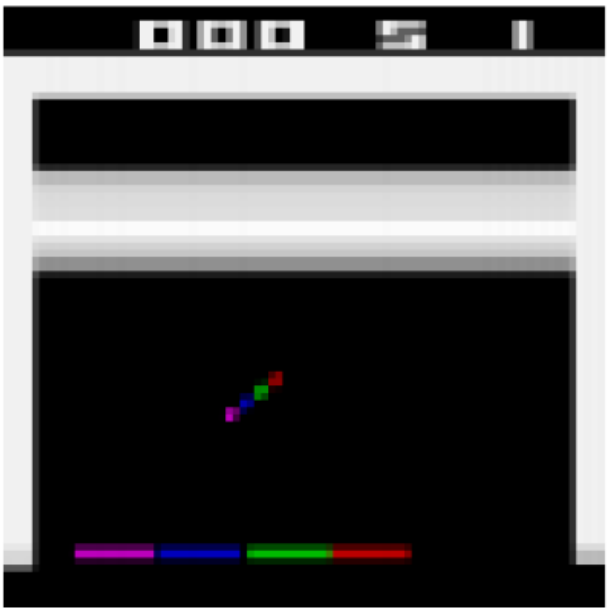

Figura 4.5: Exemplo de *frames* empilhados para o jogo *Breakout*: cada uma das cores (rosa, azul, verde e vermelho) representa um *frame* [7] diferente.

4.3 Arquitetura da *Deep Q-network*

A arquitetura da *Deep Q-network* utilizada no presente trabalho foi proposta por Mnih et al. [10], como ilustrada na Figura 4.6. A arquitetura recebe uma imagem de entrada

com dimensões de $84 \times 84 \times 4$ *pixels*, resultado da etapa de pré-processamento (Seção 4.2). A primeira camada escondida faz a convolução de 32 filtros 8×8 com *stride* 4 e função de ativação Relu, produzindo uma saída 20×20. A segunda camada escondida faz a convolução de 64 filtros 4×4 com *stride* 2 e função de ativação Relu, produzindo uma saída 7×7. A terceira camada escondida faz a convolução de 64 filtros 3×3 com *stride* 1 e função de ativação Relu, produzindo uma saída 7×7. A última camada escondida é totalmente conectada e possui 512 células. A camada de saída é uma camada totalmente conectada com um neurônio para cada ação. O número de ações varia entre três (*Pong* e *Breakout*) e quatro (*Space Invaders*) nos jogos utilizados durante o experimento. No presente trabalho, o otimizador Adam (Seção 2.1) foi utilizado durante o treinamento porque ele é menos sensível à escolha do *learning rate* (Seção 2.1) [12].

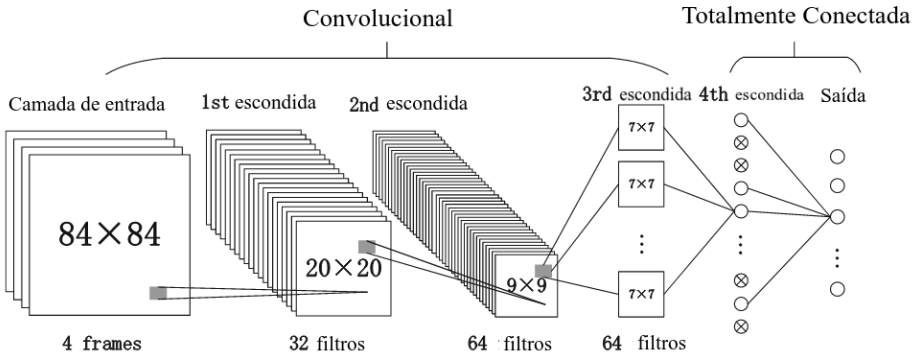

Figura 4.6: Arquitetura *Deep Q-Network* implementada no presente trabalho (adaptada de Zhai et al. [9]).

4.4 Treinamento

O agente inicia o jogo sem nenhum conhecimento prévio do ambiente e através de múltiplas experiências no ambiente adquire conhecimento para controlar o jogo a fim de adquirir a maior pontuação possível. Nesta seção são descritos como foi realizado o treinamento do agente e quais métricas foram utilizadas para avaliação. A Figura 4.7 resume o processo de treinamento utilizado. Uma DQN diferente foi treinada para cada conjunto jogo-política: utilizando a mesma arquitetura, processo de aprendizado e hiperparâmetros fixos de configuração da Tabela 4.1 [10, 12]. No total serão treinados 36 agentes, levando em conta que cada política pode ser treinada mais de uma vez por jogo, mas com hiperparâmetros diferentes.

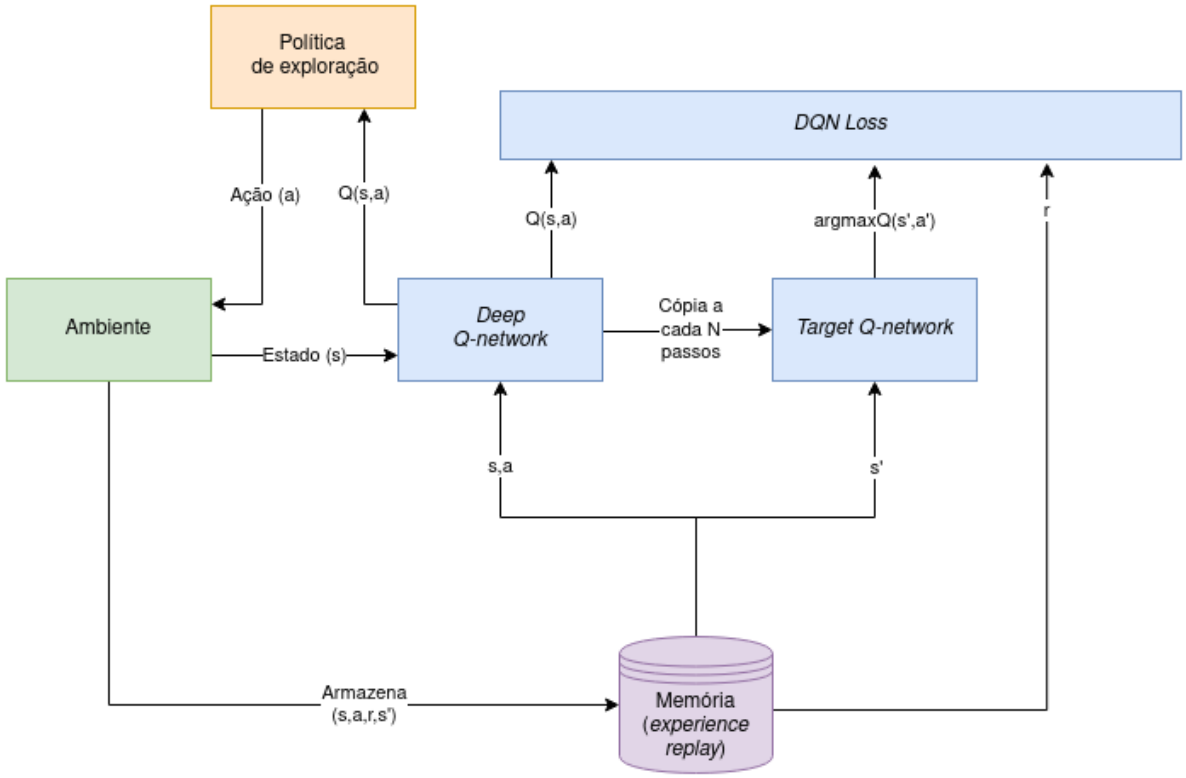

Figura 4.7: Diagrama de treinamento utilizando o algoritmo *Deep Q-learning*, baseado em Mnih et al. [10].

Durante o treinamento foram realizadas mudanças na forma como o agente recebe recompensa do ambiente e nas condições para finalizar um episódio, para ajudar a DQN a estimar melhor os Q-valores [21]. Dado que as pontuações são diferentes para cada jogo, as recompensas foram modificadas para ser sempre -1 quando a recompensa for negativa, 1 quando a recompensa for positiva ou 0. Truncar a recompensa dessa forma limita a escala das derivadas do erro e permite utilizar uma taxa de aprendizado similar em todos os jogos. Em alguns jogos o agente possui mais de uma vida, para esses casos a perda de uma vida é considerada como o fim do episódio, essa estratégia tem como objetivo incentivar o melhor desempenho do agente. Além disso, durante a etapa de treinamento e avaliação será utilizada a técnica de *frame skipping*, no qual cada ação do agente é executada por quatro *frames* consecutivos, ou seja, a interação do agente com o ambiente ocorre a cada quatro *frames*. Portanto, um passo do agente equivale a 4 *frames* do jogo. A vantagem dessa estratégia é permitir que o agente tenha mais experiências sem uma grande diferença de tempo de treinamento, já que avançar o emulador é mais rápido que a decisão do agente [21]. Além disso, a interação do agente com o ambiente possui tempo máximo de duração por episódio de 30 minutos, para evitar interações infinitas [35].

Tabela 4.1: Lista de hiperparâmetros fixos considerados no presente trabalho para DQN.

Parâmetro	**Valor**
tamanho do *minibatch*	32
Adam *learning Rate*	0.0000625
tamanho do *replay memory*	1000000
replay inicial	50000
discount factor	0.99
frequência de atualização *target Q-network*	10000
quantidade de passos	3000000
repetição de ações	4
frames por observação	4
máximo de *no-op*	30

Para aumentar a estabilidade da DQN durante o treinamento foram utilizadas duas técnicas principais [10]: *experience replay* e *target Q-Network* explicada em detalhes na Seção 2.5. A técnica de *experience replay* consiste na criação de uma memória, em que são armazenadas as experiências do agente a cada passo de interação com o ambiente. Na memória são armazenados o estado atual s, a ação realizada a, a recompensa adquirida r e a estado após a ação s', como representado na Figura 4.7. Durante o treinamento, são aplicadas atualizações na DQN utilizando *mini-batches*, i.e. um subconjunto dos dados existentes, amostrados aleatoriamente das experiências anteriores. Nos experimentos foi utilizada uma memória com capacidade de 1 milhão de experiências, tamanho baseado no trabalho de Mnih et al. [10, 12, 25, 35]. A técnica de *target Q-Network* consiste em utilizar uma segunda DQN para gerar os Q-valores esperados (*target*) durante a atualização do Q-Learning, como representado em azul na Figura 4.7. A *target Q-Network* é atualizada com a cópia dos pesos da *Q-network* a cada 10 mil passos, em todos os outros momentos ela mantém os mesmo pesos.

Na literatura mais recente [12, 21, 25, 35] de *Deep Q-learning*, são utilizados 200 milhões de *frames* durante o treinamento. Portanto, para realizar os experimentos baseados nessa metodologia seria necessário em média 10 dias por jogo [35]. Por essa razão, foi utilizado uma quantidade menor de *frames* baseado no trabalho do Mnih et al. [21], em que os autores utilizam 10 milhões de *frames*. A desvantagem de diminuir o número de *frames* é que em alguns jogos o agente pode não convergir para um resultado, por necessitar de mais tempo de aprendizado.

No presente trabalho, o treinamento dos agentes foi realizado por 12 milhões de *frames* (3 milhões de passos). Ao longo do treinamento são realizados estágios de avaliação intermediários. A metodologia de avaliação intermediária do presente trabalho foi baseada na proposta por Mnih et al. [10]. A duração entre as avaliações foi reduzida pela metade para simplificar o processo de treinamento e avaliação. A avaliação do desempenho do

agente ao longo do treinamento é realizada para verificar se o aprendizado está ocorrendo, i.e. se sua pontuação aumenta no decorrer do treinamento, conforme o agente interage com o ambiente. A cada 400000 *frames* (100000 passos) é realizada uma avaliação do agente com duração de 250000 *frames* (62500 passos). A medida de desempenho utilizada foi a média de recompensas obtidas pelo agente nos jogos terminados durante o período de treinamento. Se algum episódio não for terminado nos 250000 *frames* o episódio é descartado. Essas avaliações possuem duração em *frames* para garantir que independente do jogo, o treinamento tenha um tempo de duração semelhante, além de auxiliar na reprodução de resultados e ser uma forma justa de comparação entre os jogos [37].

Como mencionado anteriormente, os dados de treinamento no aprendizado por reforço são gerados por meio da interação com o ambiente. O objetivo do agente é, por meio da interação com o ambiente, obter a maior pontuação em longo prazo. A interação do agente com o ambiente pode ser representada como uma sequência de ações, observações e recompensas [10]. A cada passo o agente seleciona uma ação, executa a ação e modifica o ambiente. O ambiente retorna para o agente uma sequência de imagens do emulador e uma recompensa que representam, respectivamente, uma observação e a mudança na pontuação. Como critério de seleção de ação, o agente utiliza uma das políticas pré-definidas, diferente para cada experimento.

O treinamento da DQN só começa após 50000 passos, para o agente já ter carregado em memória um bom número de experiências para serem amostradas. A cada passo de um episódio, um *mini-batch* de experiências é selecionado. O estado atual s e a ação a das experiências são utilizados pela DQN para estimar o Q-valor de cada uma das ações $Q(s, a)$. O estado posterior s' e a recompensa r da experiência são utilizados pela *Target Q-network* para estimar a ação com máximo Q-valor no próximo estado $max(Q(s', a'))$. Os valores de $Q(s, a)$ e $max(Q(s', a'))$ são utilizados no cálculo da função de perda *Q-loss* (Equação 2.4), o erro resultante é utilizado para ajustar os pesos da *Q-network*. Esse processo é repetido até o fim do episódio que termina quando uma das seguintes condições é satisfeita: (1) o agente ganha o jogo; (2) o agente perde o jogo; ou (3) o limite de tempo de interação com o ambiente é atingido. O diagrama da Figura 4.7 representa esse processo.

4.5 Avaliação das políticas de exploração

O método de avaliação das políticas de exploração foi baseado nos trabalhos de Mnih et al. [10] e van Hasselt et al. [35]. Após o treinamento, a validação dos agentes foi feita por meio da avaliação do desempenho das políticas de ação em 100 episódios para cada jogo. Cada episódio tem tempo total de duração de 5 minutos e condição inicial aleatória. Nessa etapa

do experimento, a pontuação dos jogos não foi modificada e o jogo não acaba ao se perder uma vida como ocorre durante o treinamento. Para cada episódio iniciar em um momento inicial diferente, foi executada uma ação *no-op* que não afeta o ambiente no máximo 30 vezes por episódio. Esse processo é importante porque todos os jogos são determinísticos. Testando o agente com vários momentos iniciais diferentes é possível verificar se ele é capaz de generalizar a solução, dessa forma promovendo um desafio para as políticas testadas [35]. A política de exploração *e-greedy* com $\epsilon = 0.001$ foi utilizada durante a avaliação para evitar o *overfitting* e para produzir aleatoriedade nos experimentos. As métricas utilizadas durante a avaliação são a média (DQN), desvio padrão, valor máximo (DQN Máx) e valor mínimo (DQN Min) das recompensas considerando os 100 episódios. Além disso, como forma de comparação entre os jogos foram utilizados uma pontuação normalizada (DQN Normalizada), que é calculada utilizando a Equação 4.1 [10, 35]. Os valores de pontuação humana foram retirados do trabalho de Hasselt et al. [35], a pontuação humana é a média de pontuação obtida por um jogador humano após jogar 20 episódios de cada jogo com duração máxima de 5 minutos. Uma pontuação normalizada de 100% corresponde a pontuação humana e 0% corresponde ao agente que utiliza a política aleatória.

$$pontuação_{normalizada} = \frac{pontuação_{agente} - pontuação_{aleátoria}}{pontuação_{humano} - pontuação_{aleátoria}} \tag{4.1}$$

O objetivo do presente trabalho é avaliar as políticas de exploração para seleção de ação utilizadas pelo agente para aprender sobre o ambiente. Serão avaliadas as seguintes políticas de exploração: (1) aleatória, (2) *greedy*, (3) *e-greedy*, (4) *boltzmann*, (5) *decaying e-greedy* e (6) *random noise*. As políticas de um a cinco são clássicas do aprendizado por reforço [24] e a política seis foi criada para ser utilizada com RNAs no *Deep Q-learning* [25].

4.5.1 Aleatória

A primeira avaliação foi realizada considerando a política de exploração aleatória. O objetivo desse experimento é estabelecer um desempenho inicial para o agente, para servir como base de comparação com as outras políticas. Isso é importante para verificar se as outras políticas possuem um desempenho melhor que o agente que apenas realiza movimentos aleatórios. Além disso, a pontuação do agente aleatório será utilizada no cálculo da pontuação normalizada das outras políticas (Equação 4.1). Espera-se que essa política de exploração tenha o pior desempenho em comparação com as outras porque o agente apenas explora o ambiente e não usa o que foi aprendido durante o treinamento [16, 24].

4.5.2 *Greedy*

A segunda avaliação foi realizada considerando a política de exploração *Greedy*. Nessa política o agente sempre escolhe a ação com maior Q-valor, i.e. sempre utiliza o conhecimento adquirido, para maximizar a recompensa imediata [16]. O objetivo dessa avaliação é analisar como é o resultado do agente que sempre escolhe a melhor alternativa conhecida, ou seja, não explora o ambiente.

4.5.3 *Epsilon Greedy*

A terceira avaliação foi realizada considerando a política de exploração *Epsilon Greedy*. Nessa política o agente seleciona a alternativa com maior Q-valor com probabilidade 1-ϵ. O objetivo dessa avaliação é analisar como seria o resultado do agente que explora o ambiente com a mesma probabilidade durante todo o jogo. Será realizado um experimento para cada um dos valores definidos para $\epsilon = 0.3; 0.5; 0.7$. Estes valores foram escolhidos com o objetivo de analisar uma probabilidade menor, intermediária e maior para a exploração, respectivamente. Eles foram definidos de forma empírica.

4.5.4 *Boltzmann*

A quarta avaliação foi realizada considerando a política de exploração *Boltzmann*. Nessa política o agente seleciona as ações a partir de um modelo de distribuição probabilístico das ações. O objetivo dessa avaliação é analisar como seria o resultado para o agente que explora o ambiente de forma probabilística normalizada em relação a temperatura. Foi realizado um experimento para cada um dos valores selecionados para $\tau = 0.3; 0.5; 0.7$, esses valores foram escolhidos com o objetivo de analisar uma probabilidade menor, intermediária e maior para a exploração, respectivamente. Eles foram definidos de forma empírica.

4.5.5 *Decaying Epsilon Greedy*

A quinta avaliação foi realizada considerando a política de exploração *Epsilon Greedy*. Nessa política o agente seleciona a alternativa com maior Q-valor com probabilidade 1-ϵ, com valor de ϵ decrescente ao longo do treinamento. O objetivo dessa avaliação é analisar como seria o resultado para o agente que explora mais o ambiente durante o início do treinamento e menos em episódios mais avançados, utilizando mais a melhor estratégia conhecida. O ϵ tem valor inicial de 1, esse valor decai linearmente ao longo do primeiro 1 milhão de *frames* até atingir o valor final de 0.01, se mantendo fixo em 0.01 para o resto do treinamento. O resultado esperado para essa avaliação é que ela tenha um desempenho

melhor que as políticas de exploração citadas anteriormente, por possuir um equilíbrio maior entre explorar o ambiente e selecionar a melhor ação conhecida. Os valores de ϵ foram baseados no trabalho Hasselt et al. [35].

4.5.6 *Random Noise*

A sexta avaliação foi realizada considerando a política de exploração *Random Noise*. Nessa política o agente seleciona as ações de forma *Greedy*, o que gera a exploração são ruídos adicionados a rede neural. O objetivo dessa avaliação é analisar como seria o resultado para o agente que utiliza ruído aleatório adicionado a camada densa da DQN como forma de exploração. Será realizado um experimento para cada um dos valores selecionados para $\sigma_0 = 0.3; 0.4; 0.5$. O objetivo é analisar se essa política de exploração tem o melhor desempenho que as citadas anteriormente, por possuir uma metodologia de exploração com regulação automática do nível de exploração durante o treinamento. O valor do hiperparâmetro 0.5 foi retirado do trabalho do Fortunato et al. [25]. Os valores de $\sigma_0 = 0.3$ e 0.4 também são considerados nos experimento para verificar se diminuindo o valor do hiperparâmetro, utilizado no trabalho de referência [25], ocorre um aumento na velocidade de aprendizado do agente, já que no presente trabalho serão utilizados menos *frames* durante o treinamento. A ideia é com um valor de σ_0 menor que 0.5 a DQN se acostume com o ruído em menos *frames*, já que a influência dele, regulada por σ_0, será menor.

Capítulo 5

Resultados Experimentais

Com o objetivo de avaliar diferentes políticas de exploração considerando o aprendizado por reforço e arquitetura de *deep learning*, seis experimentos foram realizados: o primeiro experimento para avaliar a política aleatória (Seção 4.5.1); o segundo para a política *greedy* (Seção 4.5.2); o terceiro para a política *epsilon greedy* (Seção 4.5.3); o quarto para a política de Boltzmann (Seção 4.5.4); o quinto para a política de *decaying epsilon greedy* (Seção 4.5.5); e por último, a avaliação da política *random noise* (Seção 4.5.6).

Neste capítulo, são abordados os resultados e as discussões de cada um dos experimentos aplicados para os jogos: *Pong*; *Breakout*; e *Space Invaders*, como apresentados na Seção 4.1. Estes jogos representam tarefas com diferentes níveis de complexidade que devem ser aprendidas pelo agente. Utilizando diferentes ambientes é possível comparar como as políticas de exploração conseguem generalizar para diferentes tipos de tarefa. Os experimentos foram implementados com o auxílio da biblioteca *PFRL* [38]. A principal vantagem da biblioteca é a facilidade de modificar as políticas de exploração sem a necessidade de criar uma DQN para cada experimento. Os resultados foram avaliados e discutidos considerando 3 milhões de passos do agente (12 milhões de *frames*) para o treinamento e 100 episódios para a avaliação do agente.

5.1 *Pong*

Nesta seção serão discutidos os resultados da aplicação das políticas de exploração no jogo *Pong*, serão comparados os resultados utilizando diferentes hiperparâmetros para cada política. A pontuação do jogo *Pong* varia de -21 a 21. O agente ganha recompensa positiva toda vez que faz um ponto e ganha recompensa negativa quando perde um ponto. A pontuação humana nesse jogo foi de 9.3 pontos [35].

5.1.1 Avaliação das políticas de exploração

Aleatória

Como descrito na Seção 4.5.1, esse experimento tem como objetivo servir como base de referência para os experimentos seguintes porque esse agente age sem inteligência ao selecionar as ações. Representa a medida do desempenho do agente sem a utilização de uma política de planejamento. Na Figura 5.1 a linha azul representa a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política aleatória. A linha preta representa um limiar zero em relação a média de pontuação, se o agente obtém uma pontuação maior que esse limiar significa que ele conseguiu ganhar do adversário no jogo. Durante o treinamento, o agente foi capaz de aprender apenas com movimentos aleatórios, atingiu pontuação superior a zero nos passos: $2,0 \times 10^6; 2,5 \times 10^6; 3,0 \times 10^6$. Esses resultados mostram que o agente foi capaz de ganhar do adversário em alguns jogos.

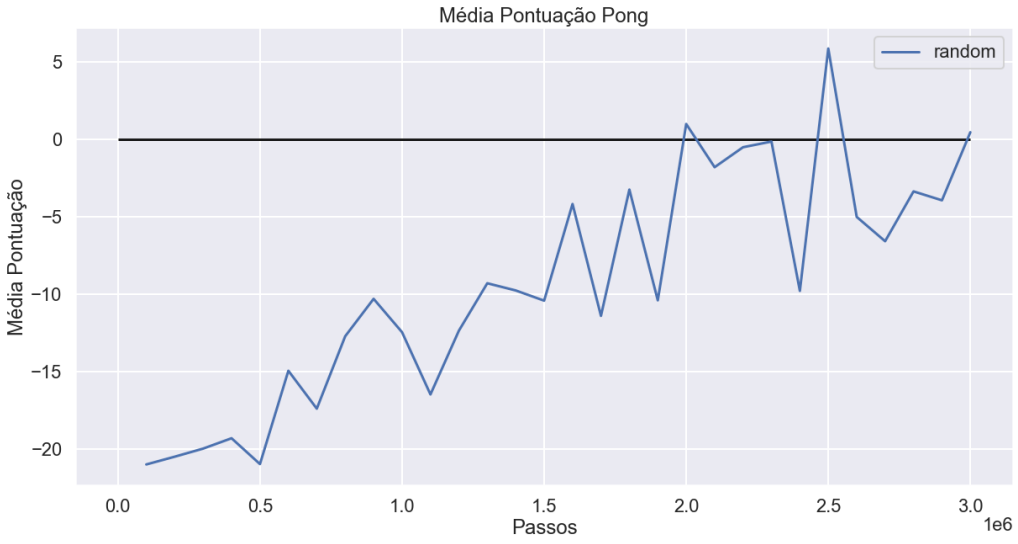

Figura 5.1: Média de pontuação por passo considerando a política aleatória no treinamento do jogo *Pong*.

A Tabela 5.1 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração aleatória. A partir desta política o agente obteve uma média de pontuação (DQN) de 4.85 pontos e desvio padrão de 5.36 pontos. O valor de média da pontuação foi utilizado para calcular a pontuação normalizada nos outros experimentos do jogo *Pong* utilizando a Equação 4.1. O agente obteve pontuação máxima (DQN Máximo) de 11.0 pontos e pontuação mínima (DQN Mínimo) de -17.0 pontos.

Tabela 5.1: Resultados da avaliação da política aleatória para o jogo *Pong*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
4.85 ± 5.36	**-**	**11.0**	**-17.0**

Greedy

Como descrito na Seção 4.5.2, esse experimento tem como objetivo representar quando o agente não explora o ambiente e escolhe sempre a melhor ação conhecida. Representa o desempenho do agente que planeja apenas com o que conhece sem nunca procurar por outras alternativas. Na Figura 5.2 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *greedy*, e a linha preta representa a média de pontuação zero. No gráfico da Figura 5.2, a média de pontuação do agente começa negativa e aumenta aos poucos até o passo 1,25 milhões, a partir desse ponto o crescimento da pontuação é sempre crescente até o passo 1,5 milhões. A partir do passo 2,5 milhões a pontuação se estabiliza na média por volta de 20 pontos. Estes resultados mostram que o agente encontrou uma estratégia consistente para adquirir uma melhor pontuação a longo prazo. O agente que utiliza a política *greedy* foi capaz de aprender muito bem como escolher as melhores ações para conseguir ganhar do adversário e obteve uma alta pontuação, considerando que a pontuação máxima é 21 pontos.

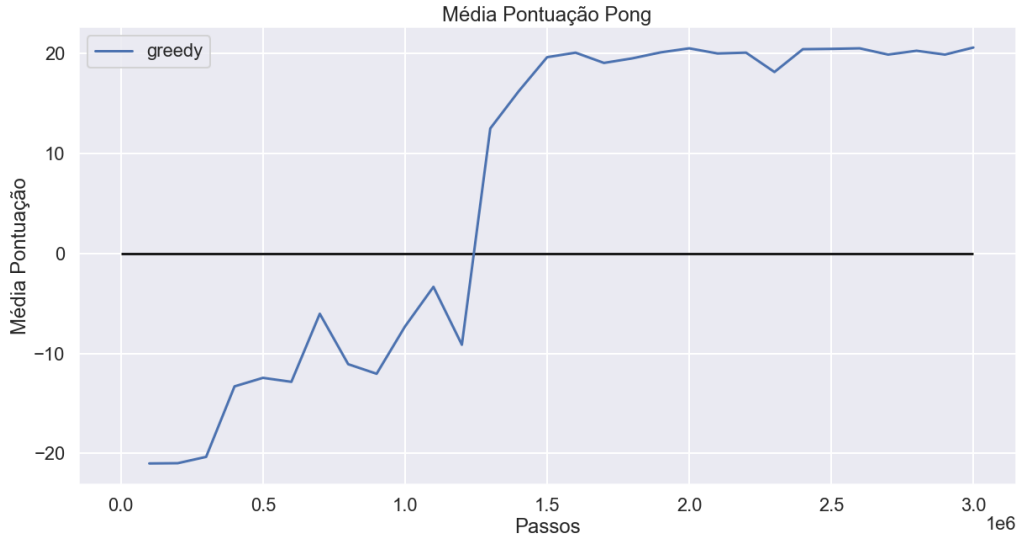

Figura 5.2: Média de pontuação por passo considerando a política *greedy* no treinamento do jogo *Pong*.

A Tabela 5.2 apresenta os resultados da avaliação obtidos do agente que age utilizando a política *greedy*. Com essa política o desempenho do agente foi maior que o humano, ob-

Tabela 5.2: Resultados da avaliação da política *greedy* para o jogo *Pong*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
20.42 ± 2.30	**349.9%**	**21.0**	**-1.0**

tendo pontuação três vezes maior, como representado pela pontuação normalizada (DQN Normalizado). Apesar da pontuação média (DQN) alta do agente, o valor mínimo foi negativo, ou seja, o agente perdeu do adversário em alguns jogos. O desvio padrão dessa política foi menor que utilizando a política aleatória. A pontuação máxima foi 21 pontos, isso mostra que em alguns casos o agente foi capaz de ganhar o jogo sem o adversário conseguir fazer pontos.

Epsilon Greedy

Como descrito na metodologia, esse experimento tem como objetivo representar quando o agente explora o ambiente dependendo do valor de ϵ. Ele representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo. A Figura 5.3 representa a média de pontuações por passo recebida durante a avaliação da política *Epsilon Greedy (e-greedy)*, a linha preta representa a média de pontuação zero. Todas as estratégias foram capazes de ganhar do adversário. No gráfico da Figura 5.3, é possível observar o agente considerando o valor de $\epsilon = 0.3$ (cor azul) obteve um melhor resultado de treinamento quando comparado com os outros valores de ϵ. Além disso, esse agente possui menor quantidade de quedas repentinas do que os outros valores de ϵ, porque valores de ϵ próximos de zero se assemelham a estratégia *greedy*. Enquanto valores de ϵ próximos de 1 se assemelham a estratégia aleatória, como ocorre para os outros valores de ϵ, por isso possuem mais variação na média de pontuação durante o treinamento. As quedas repentinas no aprendizado ocorrem, normalmente, quando o agente descobre uma nova estratégia, i.e. quando ele aprende sobre uma nova parte do ambiente a atualização da política do agente pode fazê-lo esquecer o que foi aprendido anteriormente sobre outras partes do ambiente [7].

A Tabela 5.3 apresenta os resultados obtidos pelo agente utilizando a política de exploração *e-greedy* durante a avaliação. Na Tabela 5.3 é possível observar que quanto menor o valor de epsilon ou quanto menor a exploração, maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\epsilon = 0.3$, isto é, maior pontuação máxima, pontuação mínima, pontuação média e pontuação normalizada. Contudo, em todas as variações de epsilon o agente obteve desempenho maior que o humano como representado pela pontuação normalizada (DQN Normalizado), e sempre foi capaz de ganhar do adversário, como pode ser observado pelos valores máximos e mínimos de

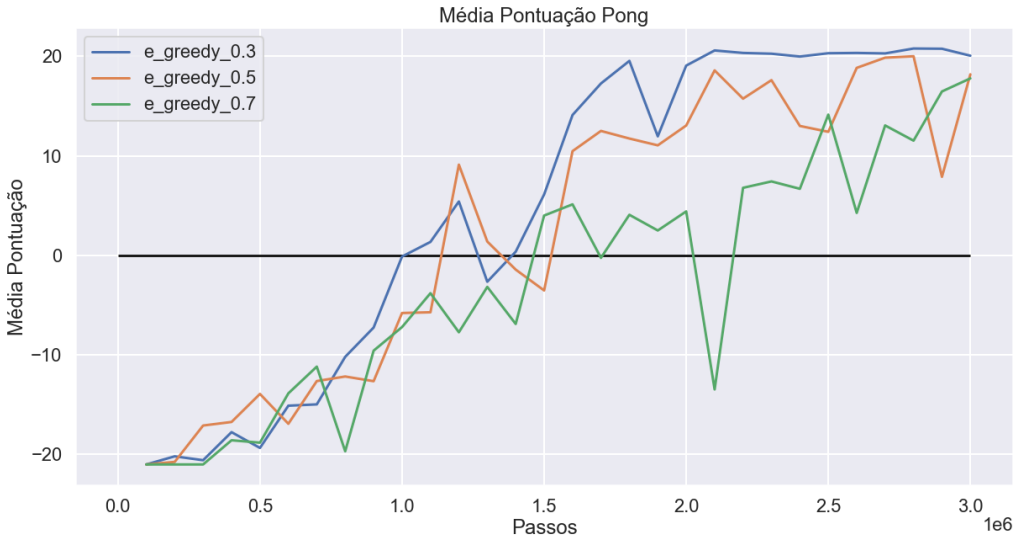

Figura 5.3: Média de pontuação por passo considerando a política *e-greedy* no treinamento do jogo *Pong*.

Tabela 5.3: Resultados da avaliação da política *e-greedy* para o jogo *Pong*.

ϵ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**20.66 ± 0.52**	**355.3%**	**21.0**	**19.0**
0.5	19.95 ± 0.30	339.3%	20.0	18.0
0.7	17.41 ± 1.44	282.2%	**21.0**	15.0

pontuação sempre maior que zero. Além disso, o desvio padrão é menor que a política *greedy* para todos os valores de epsilon.

Boltzmann

Como descrito na Seção 4.5.4, esse experimento tem como objetivo representar quando o agente explora o ambiente com maior probabilidade dependendo do valor de τ escolhido. Representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo usando uma forma de escolha diferente da *e-greedy*. Na Figura 5.4 as linhas representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política de exploração Boltzmann. A linha preta representa um limiar zero em relação a média de pontuação, se o agente obtém uma pontuação maior que esse limiar significa que ele conseguiu ganhar do adversário no jogo. No gráfico da Figura 5.4 é possível observar que as curvas possuem muitas quedas de pontuação e não aparentam convergir. Com essa estratégia o agente não foi capaz de aprender de forma tão eficiente quanto as políticas citadas anteriormente. O agente considerando o valor de

Tabela 5.4: Resultados da avaliação da política Boltzmann para o jogo *Pong*.

τ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**14.14 ± 3.28**	**208.8%**	**19.0**	**5.0**
0.5	6.23 ± 2.94	31.0%	11.0	-3.0
0.7	4.87 ± 4.81	0.4%	13.0	-5.0

$\tau = 0.3$ (cor azul) foi o que obteve o melhor resultado e maior estabilidade de treinamento quando comparado com os outros valores de τ. Isso ocorre porque valores de τ próximos de zero exploram menos e portanto se aproximam da estratégia *greedy*. Enquanto valores de τ próximos de um exploram mais e portanto se aproximam da estratégia aleatória.

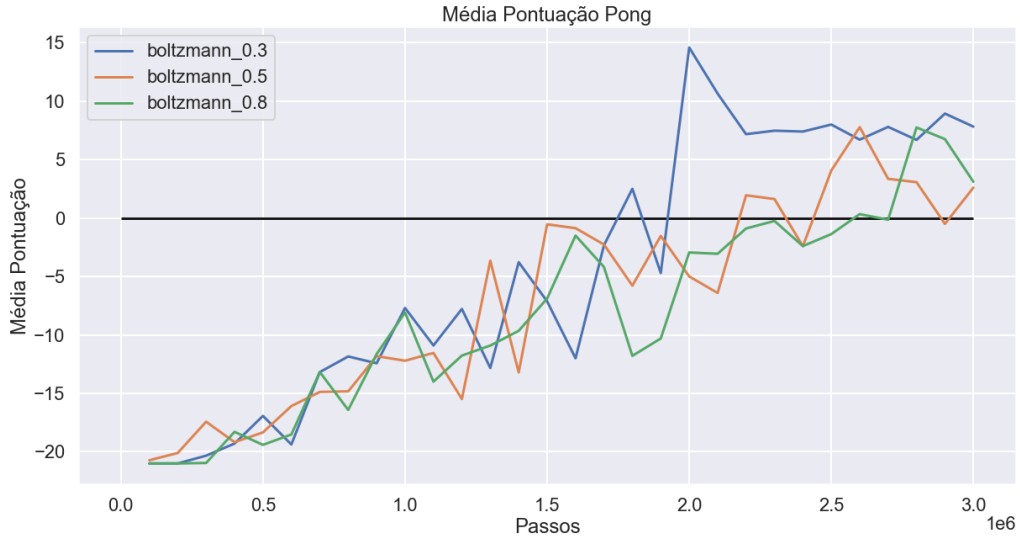

Figura 5.4: Média de pontuação por passo considerando a política Boltzmann no treinamento do jogo *Pong*.

A Tabela 5.4 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração Boltzmann. Na Tabela 5.4 é possível observar que quanto menor o valor de τ ou quanto menor a exploração maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\tau = 0.3$, sendo o único que obteve desempenho melhor que o humano, i.e. pontuação normalizada (DQN Normalizado) maior que 100%. A pontuação mínima considerando o valor de $\tau = 0.5$ e 0.7 foi negativa, o que mostra que esses agente perdem do adversário em alguns casos. Os desvio padrões foram muito maiores que a política *e-greedy*. Como $\tau = 0.7$ a pontuação normalizada foi muito inferior à humana, a média e desvio padrão da pontuação foi muito próxima da política aleatória.

Decaying Epsilon Greedy

Como descrito na Seção 4.5.5, esse experimento tem como objetivo analisar o efeito de decrementar o ϵ ao longo do treinamento do agente. Representa o desempenho do agente que começa explorando muito o ambiente e à medida que seu conhecimento sobre o ambiente aumenta, a sua exploração diminui. Na Figura 5.5 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *Decaying Epsilon Greedy (decaying e-greedy)*, a linha preta representa um limiar zero em relação a média de pontuação e a linha vermelha representa o valor de epsilon durante o treinamento. No gráfico da Figura 5.5 é possível observar que o agente explora mais o ambiente nos primeiros 1 milhão de passos, isso ocorre porque o epsilon ainda está decaindo com o tempo. Antes do valor de epsilon ficar fixo em 0.01 a pontuação do agente não é muito alta, mas a partir do passo 1 milhão a pontuação começa a aumentar muito até o passo 1.6 milhões e depois a média de pontuação se estabiliza por volta de 20.5 pontos. O agente que utiliza a política *decaying e-greedy* convergiu (aprendeu) mais rápido do que a política *greedy* isso mostra que explorar é importante para auxiliar na conversão da média de pontuação.

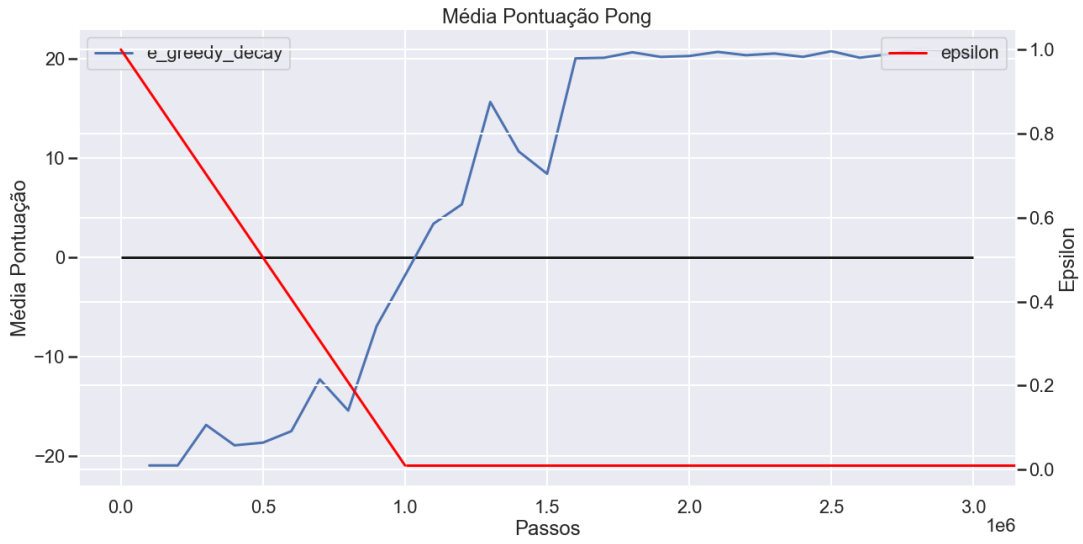

Figura 5.5: Média de pontuação por passo considerando a política *decaying epsilon greedy* no treinamento do jogo *Pong*.

Os resultados da avaliação do agente após 100 episódios de avaliação são apresentados na Tabela 5.5. Como é possível observar, houve um grande ganho em relação à política aleatória. As pontuações média, máxima e mínima foram muito próximas do resultado

Tabela 5.5: Resultados da avaliação da política *decaying epsilon greedy* para o jogo *Pong*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
20.96 ± 0.20	**362.0%**	**21.0**	**20.0**

Tabela 5.6: Resultados da avaliação da política *random noise* para o jogo *Pong*.

σ_0	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**20.17 ± 0.95**	**344.3%**	**21.0**	**14.0**
0.4	18.18 ± 3.08	299.6%	**21.0**	6.0
0.5	16.61 ± 4.20	264.3%	**21.0**	3.0

máximo possível, que é 21 pontos, o desvio padrão foi muito baixo, e a pontuação normalizada foi a maior em comparação com as políticas anteriores.

Noise-based Exploration (Random Noise)

Como descrito na Seção 4.5.6, o objetivo deste experimento é analisar o efeito da exploração utilizando ruídos na DQN. Na Figura 5.6 as linhas (cor azul, amarelo e verde) representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política *random noise* e a linha preta representa a média de pontuação zero. Todas as estratégias foram capazes de ganhar do adversário. Na Figura 5.6 é possível observar que o agente com valor de $\sigma_0 = 0.3$ (cor azul) possui aprendizado gradual e incremental. O agente com valor de $\sigma_0 = 0.5$ aprendeu de forma mais lenta comparado com os outros valores de σ, mas no final obteve pontuação final próxima do agente com valor de $\sigma_0 = 0.4$. O agente considerando o valor de $\sigma_0 = 0.4$, aprendeu com muita estabilidade durante o treinamento, isso pode ser observado no passo 1,5 milhões quando atinge resultados maiores que o agente com $\sigma_0 = 0.3$, mas a partir do passo 2,5 milhões sua média de pontuação decai, obtendo resultados piores que o agente com $\sigma_0 = 0.3$. O agente com $\sigma_0 = 0.4$ possui certa estabilidade, aumentando e diminuindo a pontuação ao longo de todo o aprendizado.

A Tabela 5.6 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração utilizando a política *random noise*. Na Tabela 5.6 é possível observar que quanto menor o valor de σ_0 ou quanto menor a exploração maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\sigma_0 = 0.3$. Apesar disso, em todas as variações de σ_0 o desempenho foi maior que o humano e a pontuação máxima foi de 21.0 pontos. A maior diferença entre eles foi o desvio padrão que diminui à medida que σ_0 diminui e a pontuação mínima que aumenta à medida que σ_0 diminui.

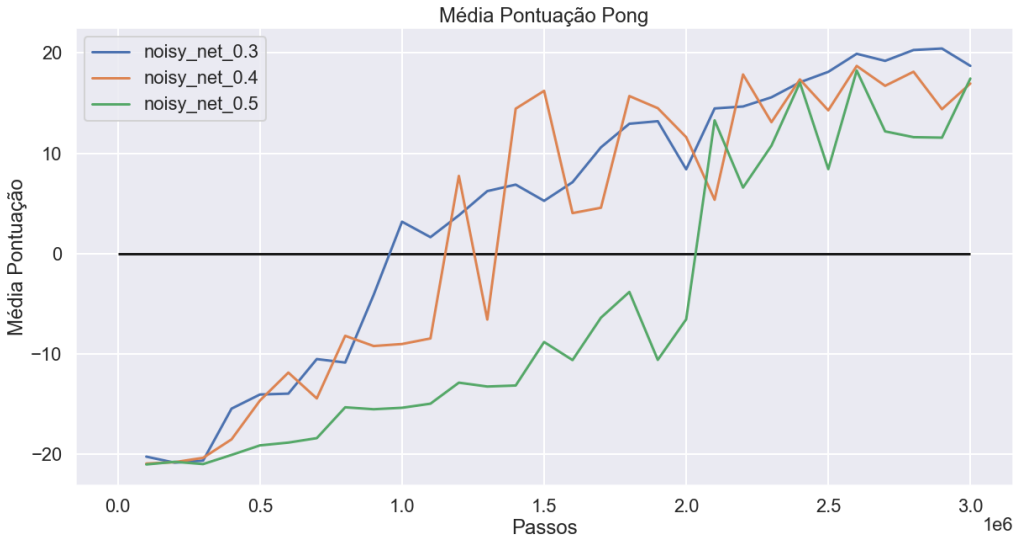

Figura 5.6: Média de pontuação por passo considerando a política *random noise* no treinamento do jogo *Pong*.

5.1.2 Comparação dos melhores hiperparâmetros

Esta seção tem como objetivo comparar as políticas de exploração com os hiperparâmetros que obtiveram os melhores resultados no jogo *Pong*. No gráfico da Figura 5.7 é possível observar que os agentes das políticas *decaying e-greedy* (cor verde), *e-greedy* (cor amarela), *greedy* (cor vermelha) e *random noise* (cor roxa) obtiveram um resultado muito próximo e convergiram para uma pontuação média de 20 pontos. Além disso, todas as políticas obtiveram um resultado melhor que a política aleatória e a política de Boltzmann obteve um dos piores resultados.

Na Tabela 5.7 é possível observar que os resultados obtidos durante a avaliação dos agentes que agem utilizando as políticas de exploração *decaying e-greedy*, *e-greedy*, *greedy* e *random noise* obtiveram média de pontuação próximas, desempenho maior que o humano, como pode ser observado pelo valor da pontuação normalizada, e pontuação máxima de 21. Das políticas citadas, a política *greedy* foi a que obteve desvio padrão muito maior que as outras. A política de exploração que obteve o melhor desempenho foi a *decaying e-greedy*, porque possui a maior média e menor desvio padrão entre todas as outras estratégias. A política *e-greedy* possui desempenho próximo, mas tem maior dispersão de pontuação. As políticas *greedy* e *random noise* obtiveram pontuação com média na faixa de 20 pontos, mas a estratégia *greedy* possui dispersão de resultados muito maior que a *random noise*. A política de Boltzmann obteve o pior resultado, com média de pontuação muito abaixo e dispersão muito alta em comparação com as estratégias anteriormente citadas, sendo

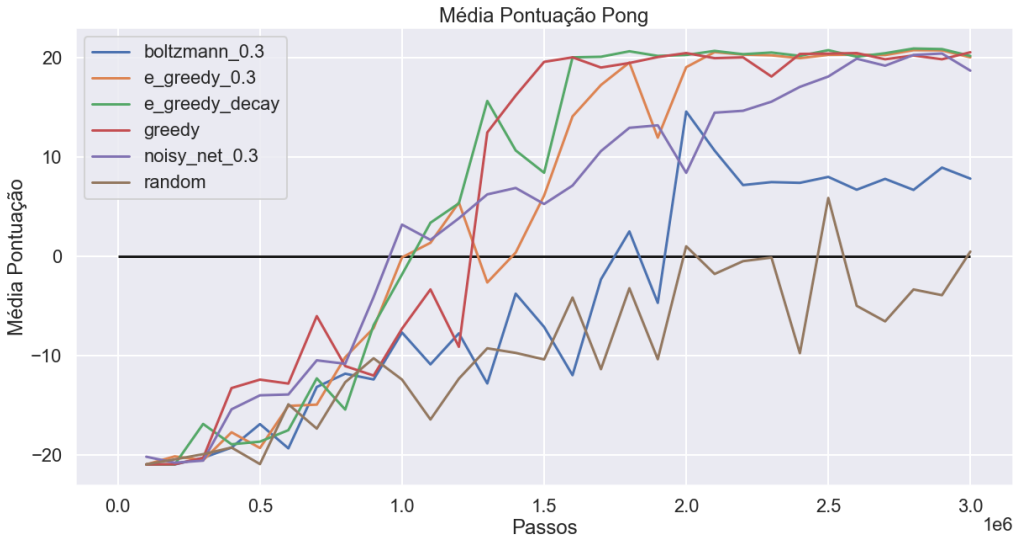

Figura 5.7: Média de pontuação por passo nas melhores políticas no treinamento do jogo *Pong*.

Tabela 5.7: Melhores resultados em ordem decrescente por política para o jogo *Pong*.

Experimento	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
decaying e-greedy	**20.96 ± 0.20**	**362.0%**	**21.0**	**20.0**
e-greedy 0.3	20.66 ± 0.52	355.3%	**21.0**	19.0
greedy	20.42 ± 2.30	349.9%	**21.0**	-1.0
random noise 0.3	20.17 ± 0.95	344.3%	**21.0**	14.0
boltzmann 0.3	14.14 ± 3.28	208.8%	19.0	5.0
aleatório	4.85 ± 5.36	0.0%	11.0	-17.0

melhor apenas que a aleatória.

5.2 *Breakout*

Nesta seção serão discutidos os resultados da aplicação das políticas de exploração no jogo *Breakout*, serão comparados os resultados utilizando diferentes hiperparâmetros para cada política. A pontuação do jogo *Breakout* começa em zero e aumenta toda vez que um bloco é destruído. O agente ganha recompensa positiva toda vez que faz um ponto e ganha recompensa negativa quando deixa a bola cair. A pontuação humana nesse jogo foi de 31.8 pontos [35].

5.2.1 Avaliação das políticas de exploração

Aleatória

Como descrito na Seção 4.5.1, esse experimento tem como objetivo servir como base de referência para os experimentos seguintes porque esse agente age sem inteligência ao selecionar as ações. Representa a medida do desempenho do agente sem a utilização de uma política de planejamento. Na Figura 5.8 a linha azul representa a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política aleatória. Durante o treinamento, a pontuação do agente aleatório a partir do passo 0.5 milhão se torna caótica, i.e. aumentando e caindo ao longo do treinamento. Esse comportamento é esperado porque o agente aleatório faz escolhas de ações aleatórias, sem utilizar o conhecimento adquirido com experiências anteriores. Apesar das perdas de desempenho ao longo do treinamento, a média de pontuação do agente aleatório tem uma tendência de crescimento.

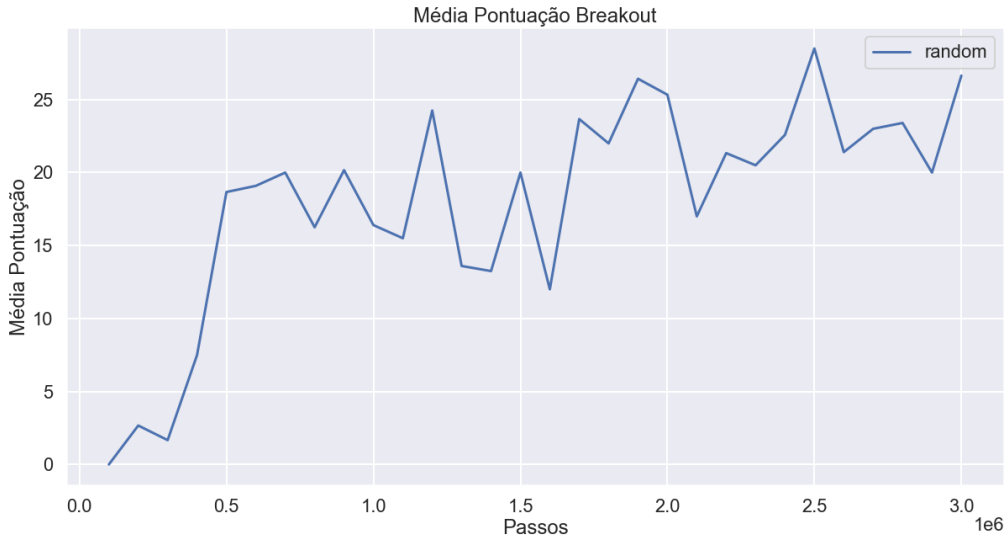

Figura 5.8: Média de pontuação por passo considerando a política aleatória no treinamento do jogo *Breakout*.

A Tabela 5.8 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração aleatória. Com essa política o agente obteve média de pontuação (DQN) de 22.26 e o desvio padrão foi alto, indicando que os resultados foram muito dispersos. A pontuação máxima do agente aleatório foi 36.0 pontos e a pontuação mínima foi 4.0 pontos. O esperado é que todos os outros experimentos obtenham maior média de pontuação e sejam menos dispersos.

Tabela 5.8: Resultados da avaliação da política aleatória para o jogo *Breakout*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
22.26 ± 9.93	**0.0%**	**36.0**	**4.0**

Greedy

Como descrito na Seção 4.5.2, esse experimento tem como objetivo representar quando o agente não explora o ambiente e escolhe sempre a melhor ação conhecida. Representa o desempenho do agente que planeja apenas com o que conhece sem nunca procurar por outras alternativas. Na Figura 5.9 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *greedy*. No gráfico da Figura 5.9, o agente tem crescimento gradual até o passo 2,5 milhões. A partir desse ponto a média de pontuação do agente sofre uma sequência de quedas repentinas e depois se recupera um pouco, mas o desempenho que ele perde não é recuperado totalmente. Durante o treinamento do agente *greedy* a pontuação não converge. A divergência pode ter acontecido por falta de exploração, i.e. o agente não consegue encontrar uma estratégia ótima para o ambiente.

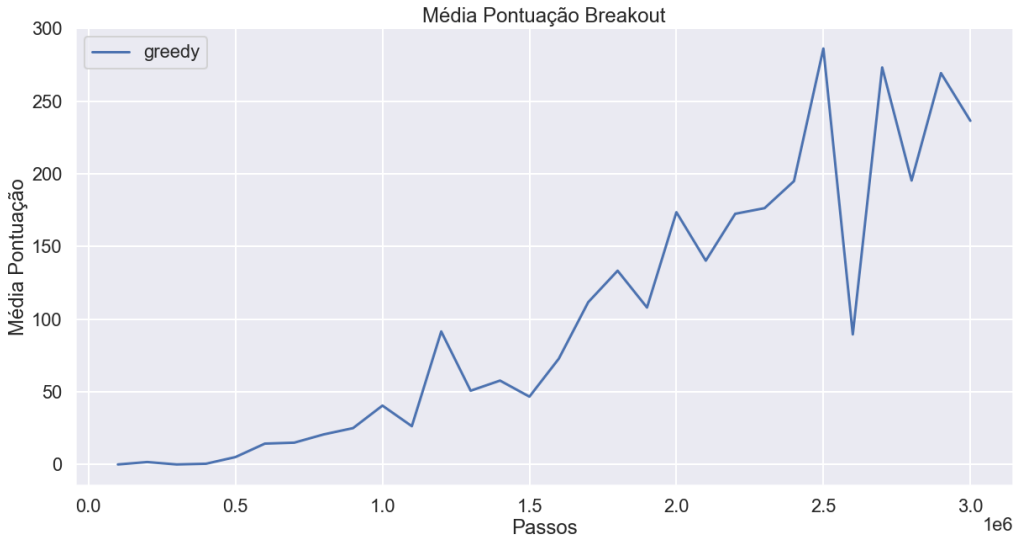

Figura 5.9: Média de pontuação por passo considerando a política *greedy* no treinamento do jogo *Breakout*.

A Tabela 5.9 apresenta os resultados da avaliação obtidos do agente que age utilizando a política *greedy*. Com essa política o desempenho do agente foi muito maior que o humano (DQN normalizado) e a política aleatória (DQN). Apesar disso, o desvio padrão foi maior que a política aleatória. A pontuação máxima foi 325.0 e a pontuação mínima foi 11.0.

Tabela 5.9: Resultados da avaliação da política *greedy* para o jogo *Breakout*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
219.28 ± 84.97	**2065.2%**	**325.0**	**11.0**

Epsilon Greedy

Como descrito na Seção 4.5.3, esse experimento tem como objetivo representar quando o agente explora o ambiente dependendo do valor de ϵ. Ele representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo. Na Figura 5.10 as linhas representam a média de pontuações por passo recebida durante a avaliação da política *Epsilon Greedy (e-greedy)*. No gráfico da Figura 5.10 é possível observar que o processo de aprendizagem foi lento para todos os valores de epsilon. O crescimento das linhas foi gradual e em pequenos passos, com exceção do agente com valor de $\epsilon = 0.3$ (cor azul) que teve mais variações na pontuação ao longo do treinamento. A partir do passo 2,3 milhões o agente com valor de $\epsilon = 0.3$ (cor azul) obteve um melhor resultado de treinamento quando comparado com os outros valores de ϵ.

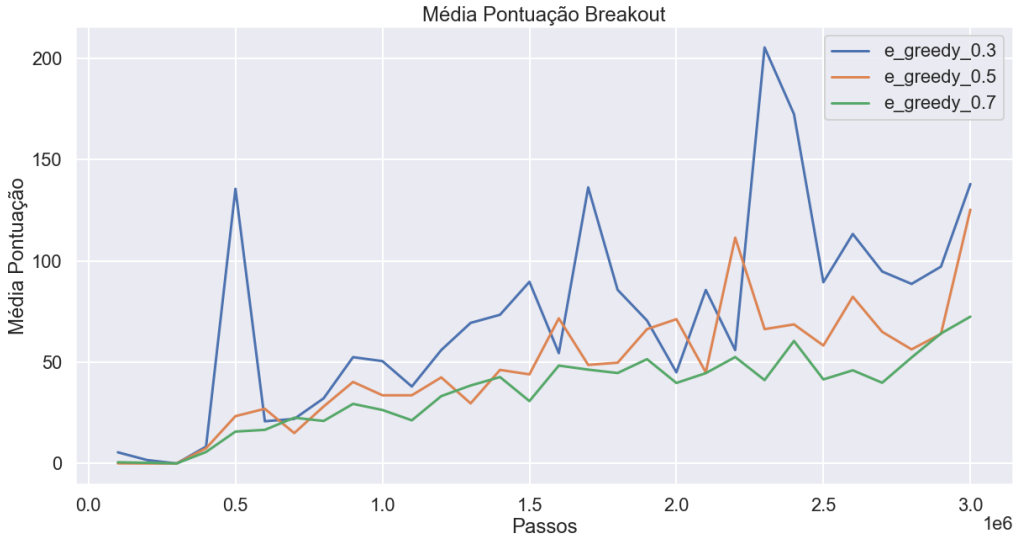

Figura 5.10: Média de pontuação por passo considerando a política *e-greedy* no treinamento do jogo *Breakout*.

A Tabela 5.10 apresenta os resultados obtidos pelo agente utilizando a política de exploração *e-greedy*. A média de pontuação sofreu grande influência de ϵ. Na Tabela 5.10 é possível observar que quanto menor o valor de epsilon ou quanto menor a exploração, maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\epsilon = 0.3$, isto é, maior pontuação máxima, pontuação mínima, pontuação média e pontuação normalizada.

Tabela 5.10: Resultados da avaliação da política *e-greedy* para o jogo *Breakout*.

ϵ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**164.85 ± 90.49**	**1494.7%**	**329.0**	**31.0**
0.5	63.31 ± 26.47	430.3%	228.0	15.0
0.7	50.65 ± 33.45	297.6%	131.0	5.0

Tabela 5.11: Resultados da avaliação da política Boltzmann para o jogo *Breakout*.

τ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**133.51 ± 67.48**	**1166.1%**	**256.0**	**27.0**
0.5	65.31 ± 18.66	451.3%	96.0	15.0
0.7	68.07 ± 14.98	480.2%	123.0	22.0

Contudo, em todas as variações de epsilon o agente obteve desempenho maior que o humano como representado pela pontuação normalizada (DQN Normalizado). O desvio padrão foi maior que o agente *greedy*.

Boltzmann

Como descrito na Seção 4.5.4, esse experimento tem como objetivo representar quando o agente explora o ambiente com maior probabilidade dependendo do valor de τ escolhido. Representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo usando uma forma de escolha diferente da *e-greedy*. Na Figura 5.11 as linhas representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política de exploração Boltzmann. No gráfico da Figura 5.11 é possível observar que as curvas possuem muitas quedas de pontuação e não aparentam convergir. Além disso, o aumento de pontuação é muito pequeno de um passo para o outro. Com essa estratégia o agente não foi capaz de aprender de forma tão eficiente quanto as políticas citadas anteriormente. O agente considerando o valor de $\tau = 0.3$ (cor azul) foi o que obteve a maior média de pontuação ao final do treinamento quando comparado com os outros valores de τ.

A Tabela 5.11 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração Boltzmann. Na Tabela 5.11 é possível observar que quanto menor o valor de τ ou quanto menor a exploração maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\tau = 0.3$ A pontuação normalizada foi maior que a humana em todos os valores de τ. As pontuações médias, máxima e mínima do agente Boltzmann foram menores do que a política *e-greedy* apenas de possuir um desvio padrão menor.

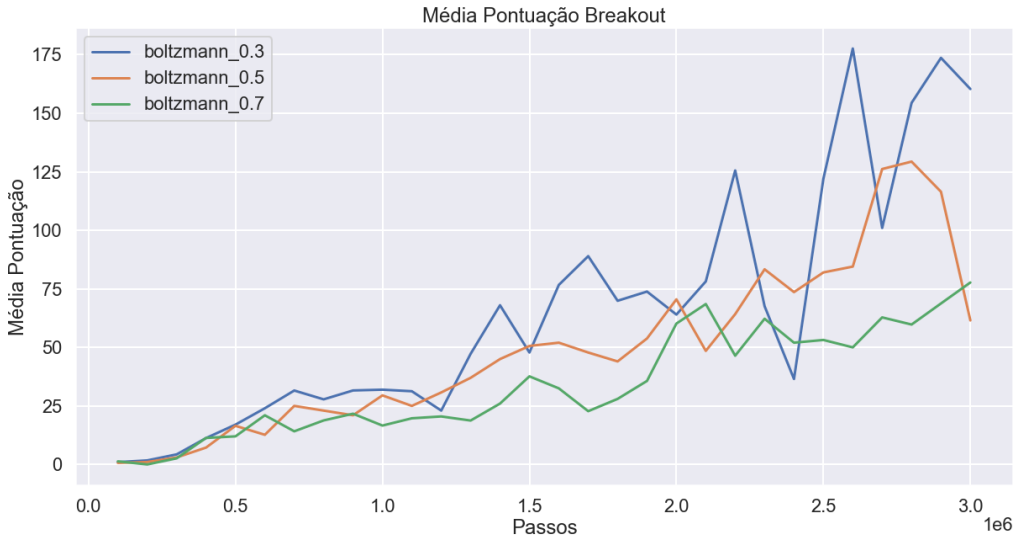

Figura 5.11: Média de pontuação por passo considerando a política Boltzmann no treinamento do jogo *Breakout*.

Decaying Epsilon Greedy

Como descrito na Seção 4.5.5, esse experimento tem como objetivo analisar o efeito de decrementar o ϵ ao longo do treinamento do agente. Representa o desempenho do agente que começa explorando muito o ambiente e à medida que seu conhecimento sobre o ambiente aumenta, a sua exploração diminui. Na Figura 5.12 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *Decaying Epsilon Greedy (decaying e-greedy)*. No gráfico da Figura 5.12 é possível observar que o agente explora mais o ambiente nos primeiros 1 milhão de passos, isso ocorre porque o epsilon ainda está decaindo com o tempo. Antes do valor de epsilon ficar fixo em 0.01 a pontuação do agente é baixa, mas a partir do passo 1 milhão a pontuação começa a aumentar em saltos até o passo 2,6 milhões e depois decai. A média de pontuação não converge.

Os resultados da avaliação do agente após 100 episódios de avaliação são apresentados na Tabela 5.12. Como é possível observar, houve um grande ganho em relação à política aleatória. A pontuação média foi maior que nas políticas anteriores, mas possui grande dispersão de valores por conta do alto desvio padrão. O agente que utiliza a política *decaying e-greedy* obteve uma das menores pontuações mínimas. Apesar disso, o desempenho foi muito maior que o humano (DQN Normalizado).

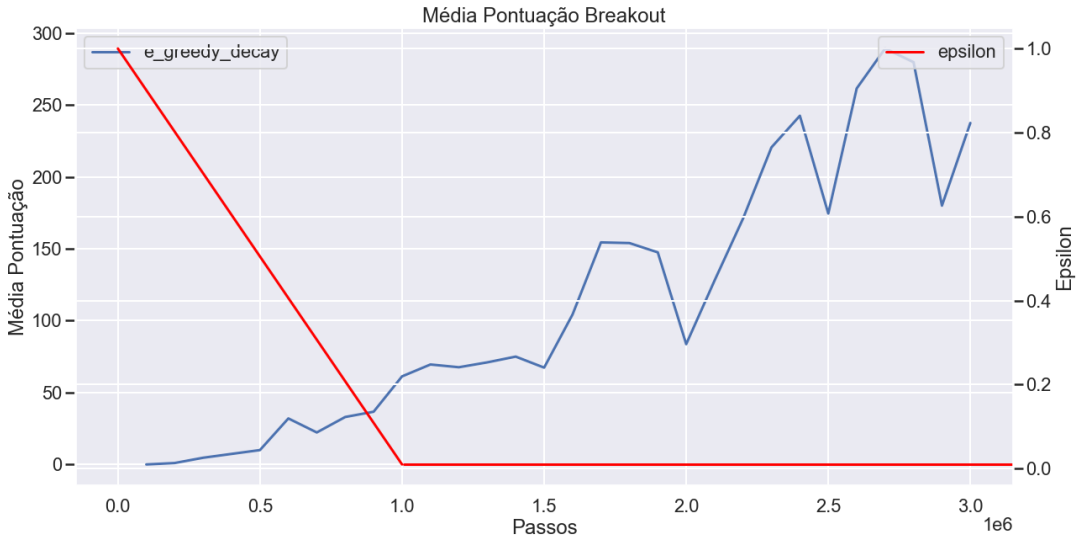

Figura 5.12: Média de pontuação por passo considerando a política *decaying epsilon greedy* no treinamento do jogo Breakout.

Tabela 5.12: Resultados da avaliação da política *decaying epsilon greedy* para o jogo *Breakout*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
242.7 ± 70.46	**2310.7%**	**311.0**	**2.0**

Noise-based Exploration (Random Noise)

Como descrito na Seção 4.5.6, o objetivo deste experimento é analisar o efeito da exploração utilizando ruídos na DQN. Na Figura 5.13 as linhas (cor azul, amarelo e verde) representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política *random noise*. Na Figura 5.13 é possível observar que para todos os valores de σ_0 o crescimento foi próximo até o passo 2,7 milhões, a partir desse ponto o agente com $\sigma_0 = 0.3$ teve um salto de pontuação, enquanto os outros não aumentaram tanto, em comparação. Em comparação com a política *decaying e-greedy* as linhas do gráfico da Figura 5.13 tem um crescimento mais estável, i.e. com menores quedas na média da pontuação.

A Tabela 5.13 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração *random noise*. Na Tabela 5.13 é possível observar que quanto menor o valor de σ_0 ou quanto menor a exploração maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\sigma_0 = 0.3$. Apesar disso, em todas as variações de σ_0 o desempenho foi maior que o humano. Os valores de pontuação máxima possuem valores próximos, mas os valores de pontuação mínimos são diferentes. Todas as

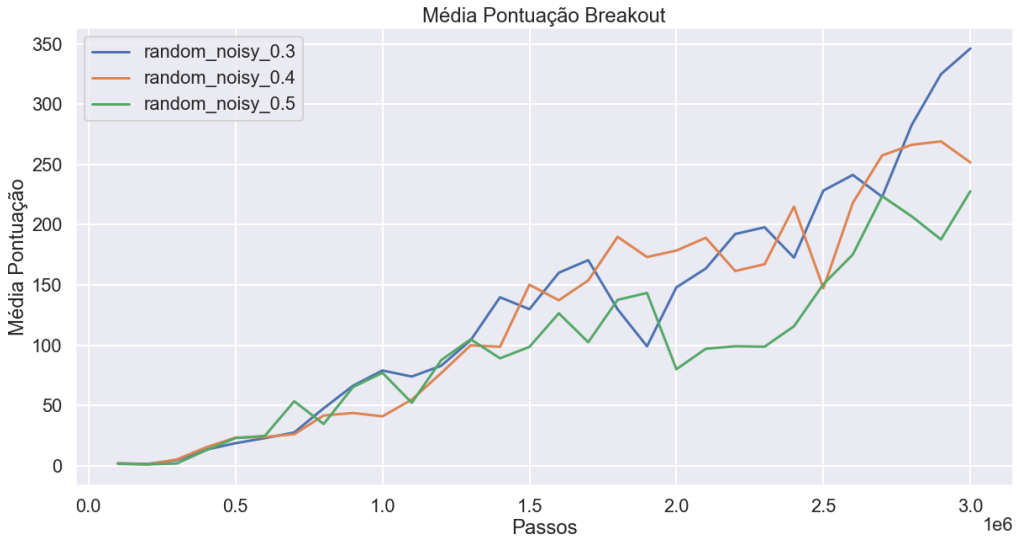

Figura 5.13: Média de pontuação por passo considerando a política *Random Noise* no treinamento do jogo *Breakout*.

Tabela 5.13: Resultados da avaliação da política *random noise* para o jogo *Breakout*.

σ_0	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	**349.76 ± 71.05**	**3432.9%**	**431.0**	29.0
0.4	257.36 ± 72.99	2464.4%	401.0	**38.0**
0.5	180.35 ± 108.31	1657.1%	415.0	1.0

variação da política de *random noise* possuem desvio padrão maior que 70.0.

5.2.2 Comparação dos melhores hiperparâmetros

Esta seção tem como objetivo comparar as políticas de exploração com os hiperparâmetros que obtiveram os melhores resultados no jogo *Breakout*. No gráfico da Figura 5.14 é possível observar que o agente com a política *random noisy* (cor roxa) possui o treinamento mais estável em comparação com as outras políticas, i.e. possui menores quedas na média da pontuação. Além disso, possui a maior média de pontuação no final do treinamento. As *decaying e-greedy* (cor verde) e *greedy* (cor vermelha) terminam o treinamento com a segunda maior pontuação. Com os menores resultados estão as políticas Boltzmann (cor azul), *e-greedy* (cor amarela) e a aleatória (cor marrom), respectivamente.

Na Tabela 5.14 é possível observar os resultados das políticas de exploração com os melhores hiperparâmetros no jogo *Breakout*. O agente que utiliza a política *random noise* obteve o melhor resultado, com média de pontuação muito maior que a segunda política. A política *decaying e-greedy* obteve pontuação abaixo da estratégia anterior, mas possui

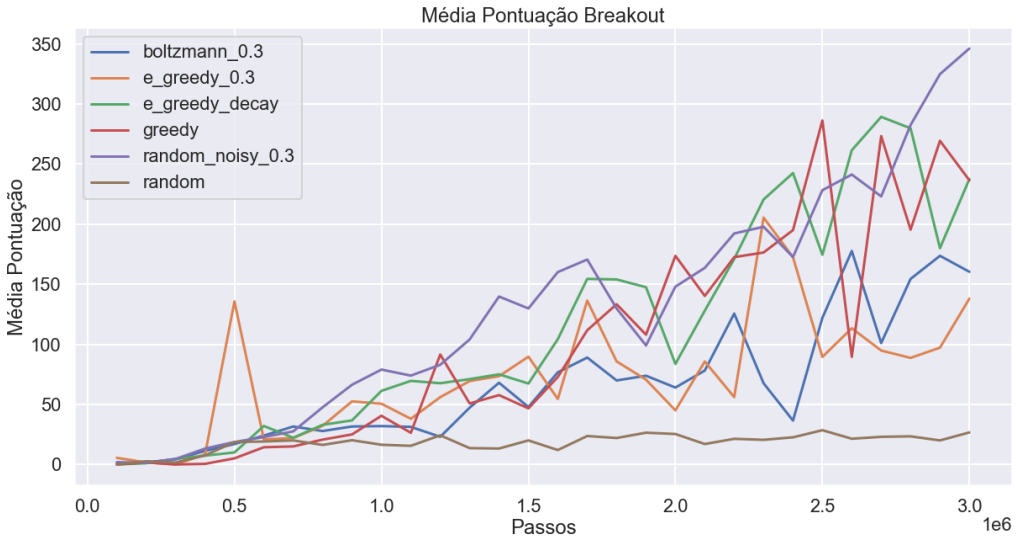

Figura 5.14: Média de pontuação por passo melhores políticas no treinamento do jogo Breakout.

Tabela 5.14: Melhores resultados em ordem decrescente por política para o jogo *Breakout*.

Experimento	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
random noise 0.3	**349.76 ± 71.05**	**3432.9%**	**431.0**	29.0
decaying e-greedy	242.7 ± 70.46	2310.7%	311.0	2.0
greedy	219.28 ± 84.97	2065.2%	325.0	11.0
e-greedy 0.3	164.85 ± 90.49	1494.7%	329.0	**31.0**
boltzmann 0.3	133.51 ± 67.48	1166.1%	256.0	27.0
aleatória	22.26 ± 9.93	0.0%	36.0	4.0

resultado um pouco menos disperso. A política *greedy* obteve resultado mediano em relação às outras estratégias, mas alto valor de dispersão entre os dados. As políticas *e-greedy* e *boltzmann* obtiveram média de resultado próximo, apesar da estratégia *e-greedy* possuir maior dispersão. As duas últimas políticas foram as que obtiveram as menores médias de pontuação.

5.3 *Space Invaders*

Nesta seção serão discutidos os resultados da aplicação das políticas de exploração no jogo *Space Invaders*, serão comparados os resultados utilizando diferentes hiperparâmetros para cada política. A pontuação do jogo *Space Invaders* começa em zero e aumenta toda vez que um *alien* é destruído. O agente ganha recompensa positiva toda vez que faz um ponto

e ganha recompensa negativa quando é atingido por um *alien*. A pontuação humana nesse jogo foi de 1652.3 pontos [35].

5.3.1 Avaliação das políticas de exploração

Aleatória

Como descrito na Seção 4.5.1, esse experimento tem como objetivo servir como base de referência para os experimentos seguintes porque esse agente age sem inteligência ao selecionar as ações. Representa a medida do desempenho do agente sem a utilização de uma política de planejamento. Na Figura 5.15 a linha azul representa a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política aleatória. A pontuação do agente aleatório é bem caótica, i.e. aumentando e caindo durante todo o treinamento. Esse comportamento é esperado porque o agente aleatório faz escolhas de ações aleatórias, sem utilizar o conhecimento adquirido com experiências anteriores. Apesar disso, o agente foi capaz de aprender o suficiente para conseguir uma pontuação significativa durante a avaliação. Além disso, é possível perceber que ao longo do tempo a pontuação do agente sobe e cai diversas vezes, e não se estabiliza em um valor.

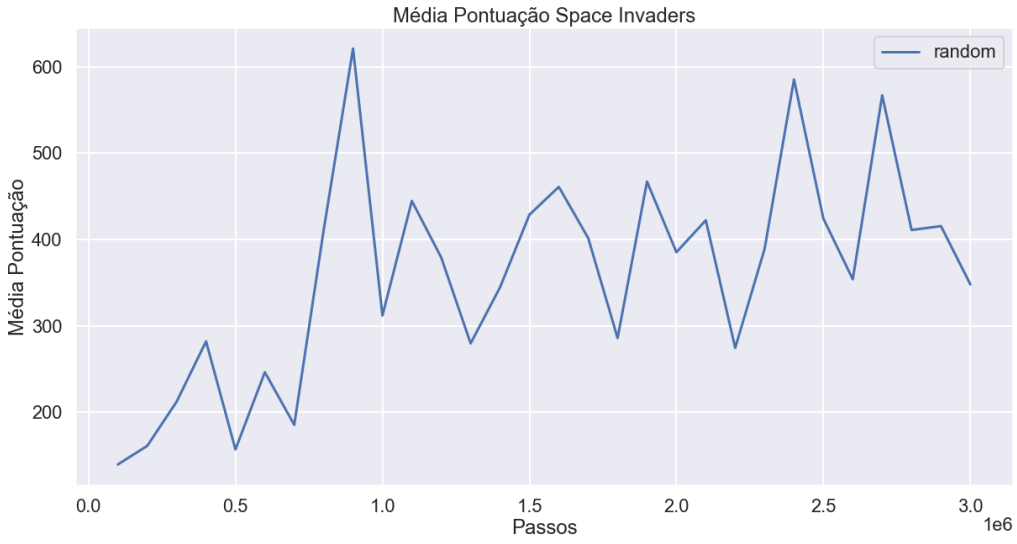

Figura 5.15: Média de pontuação por passo considerando a política aleatória no treinamento do jogo *Space Invaders*.

A Tabela 5.15 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração aleatória. Com essa política o agente obteve média de pontuação de 574.1, e o desvio padrão é muito alto, indicando que os resultados foram

Tabela 5.15: Resultados da avaliação da política aleatória para o jogo *Space Invaders*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
574.1 ± 302.06	**0.0%**	**1010.0**	**165.0**

muito dispersos. A pontuação máxima do agente aleatório foi 1010.0 pontos e a pontuação mínima foi 165.0 pontos. O esperado é que em todos os outros experimentos a média de pontuação seja maior e os resultados sejam menos dispersos.

Greedy

Como descrito na Seção 4.5.2, esse experimento tem como objetivo representar quando o agente não explora o ambiente e escolhe sempre a melhor ação conhecida. Representa o desempenho do agente que planeja apenas com o que conhece sem nunca procurar por outras alternativas. Na Figura 5.16 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *greedy*. No gráfico da Figura 5.16, o agente tem crescimento gradual até o passo 1,7 milhões. A partir desse ponto a média de pontuação do agente começa a perder a estabilidade, sofrer variações e decair. Durante o treinamento do agente *greedy* a pontuação não converge. A divergência pode ter acontecido por falta de exploração, i.e. o agente não consegue encontrar uma estratégia ótima para o ambiente.

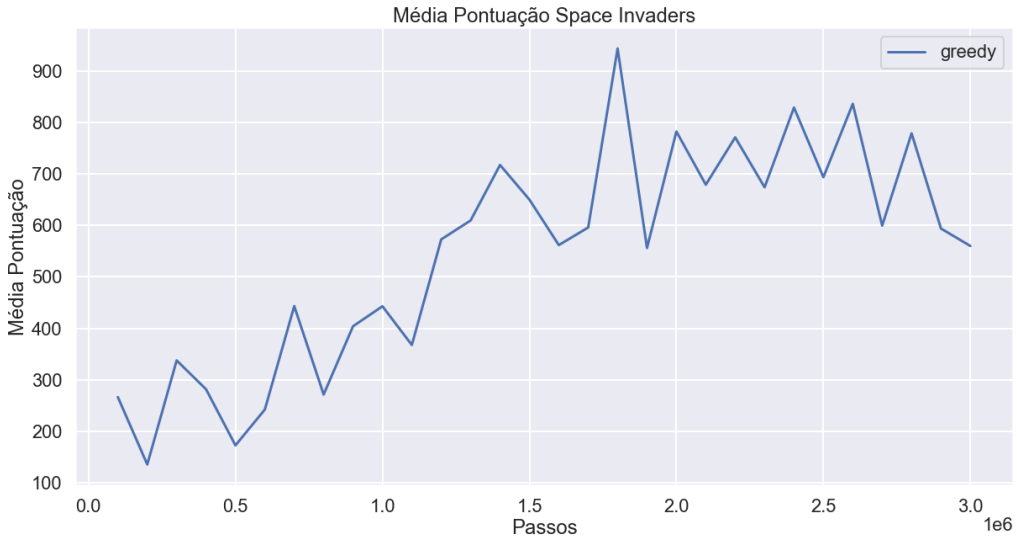

Figura 5.16: Média de pontuação por passo considerando a política *greedy* no treinamento do jogo *Space Invaders*.

Tabela 5.16: Resultados da avaliação da política *greedy* para o jogo *Space Invaders*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
891.2 ± 269.68	**29.4%**	**1720.0**	**510.0**

A Tabela 5.16 apresenta os resultados da avaliação obtidos do agente que age utilizando a política *greedy*. Com essa política o desempenho do agente é maior que a política aleatória, mas não superou o desempenho humano (DQN normalizado). O desvio padrão do agente foi menor que a política aleatória. A pontuação máxima foi 1720.0 e a pontuação mínima foi 510.0.

Epsilon Greedy

Como descrito na Seção 4.5.3, esse experimento tem como objetivo representar quando o agente explora o ambiente dependendo do valor de ϵ. Ele representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo. Na Figura 5.17 as linhas representam a média de pontuações por passo recebida durante a avaliação da política *Epsilon Greedy (e-greedy)*. No gráfico da Figura 5.17 é possível observar que para todos os valores de ϵ as curvas têm um comportamento parecido. No final do treinamento os agente com valor de $\epsilon = 0.3$ e 0.5 têm a mesma média de pontuação enquanto o agente do valor de $\epsilon = 0.7$ sofreu uma queda brusca.

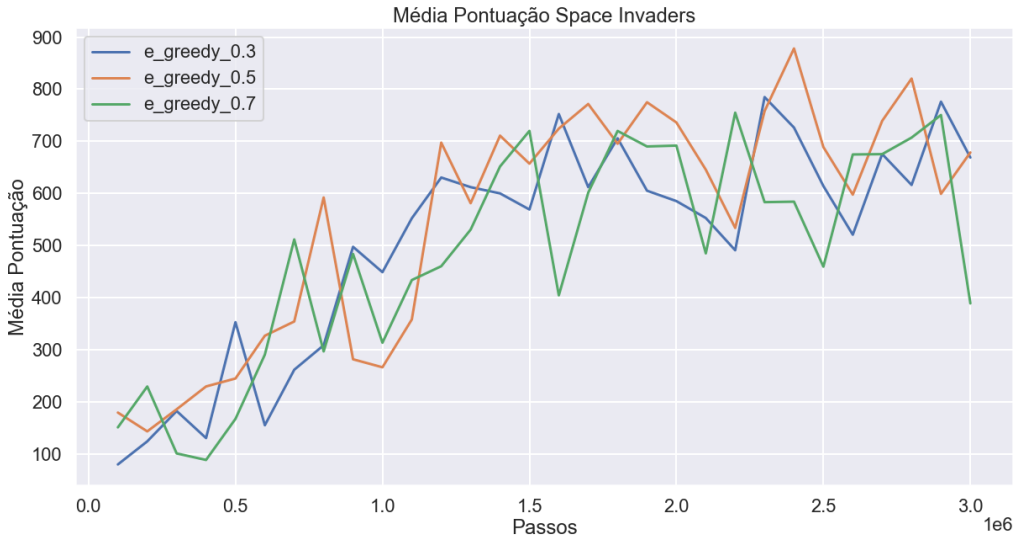

Figura 5.17: Média de pontuação por passo considerando a política *e-greedy* no treinamento do jogo *Space Invaders*.

Tabela 5.17: Resultados da avaliação da política *e-greedy* para o jogo *Space Invaders*.

ϵ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	790.15 ± 270.75	20.0%	1165.0	495.0
0.5	**820.8 ± 361.44**	**22.9%**	**1765.0**	**515.0**
0.7	782.75 ± 251.53	19.4%	1140.0	355.0

Tabela 5.18: Resultados da avaliação da política Boltzmann para o jogo *Space Invaders*.

τ	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	821.9 ± 325.14	23.0%	1470.0	330.0
0.5	**911.45 ± 245.45**	**31.3%**	**1920.0**	**520.0**
0.7	620.4 ± 202.25	4.3%	1195.0	285.0

A Tabela 5.17 apresenta os resultados obtidos pelo agente utilizando a política de exploração *e-greedy*. Na Tabela 5.17 é possível observar que com valor intermediário de ϵ ($\epsilon = 0.5$) ou com equilíbrio na exploração o agente obteve a maior pontuação média. Os melhores resultados foram obtidos com $\epsilon = 0.5$, isto é, maior média de pontuação e a maior dispersão entre os resultados. Contudo, em todas as variações de ϵ o agente obteve desempenho pior que o humano.

Boltzmann

Como descrito na Seção 4.5.4, esse experimento tem como objetivo representar quando o agente explora o ambiente com maior probabilidade dependendo do valor de τ escolhido. Representa o desempenho do agente que explora o ambiente e planeja com a mesma proporção durante todo o jogo usando uma forma de escolha diferente da *e-greedy*. Na Figura 5.18 as linhas representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política de exploração Boltzmann. No gráfico da Figura 5.18 é possível observar que o ganho de pontuação da linha com $\tau = 0.5$ (cor laranja) cresce mais rápido que todas as outras. A linha com $\tau = 0.3$ cresce de forma mais instável, i.e. ocorrem grandes ganhos de pontuação seguido de quedas repentinas. A linha com $\tau = 0.7$ cresce mais lentamente em comparação com os outros valores de τ.

A Tabela 5.18 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração Boltzmann. Na Tabela 5.18 é possível observar que os resultados utilizando essa política tendem a ser melhores com o valor intermediário de τ ($\tau = 0.5$), ou seja, com um equilíbrio entre os níveis de exploração durante o jogo é possível obter a maior pontuação. Os resultados foram muito discrepantes, $\tau = 0.5$ obteve o melhor resultado, enquanto $\tau = 0.7$ obteve resultados pouco melhores que o aleatório. A pontuação normalizada foi menor que a humana em todos os valores de τ.

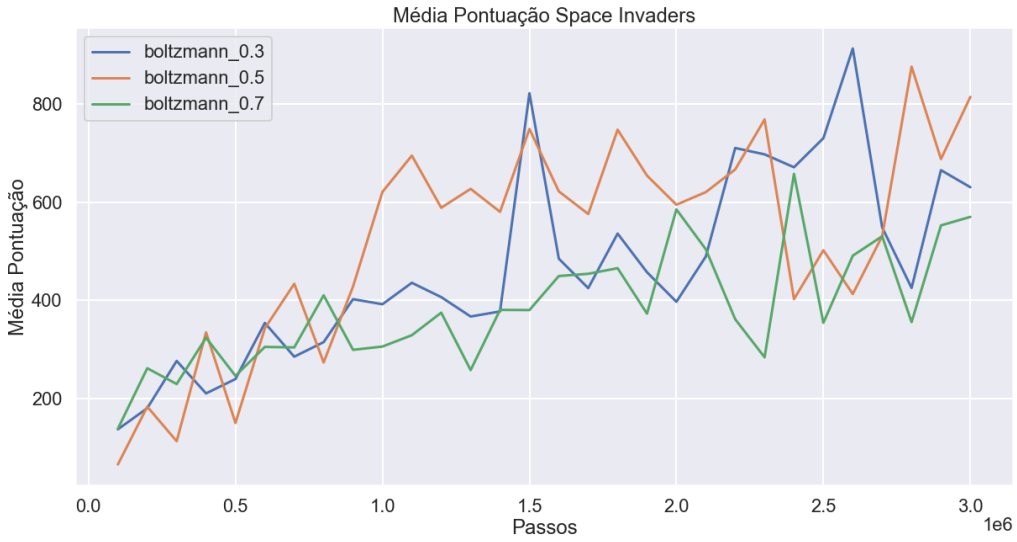

Figura 5.18: Média de pontuação por passo considerando a política Boltzmann no treinamento do jogo *Space Invaders*.

Decaying Epsilon Greedy

Como descrito na Seção 4.5.5, esse experimento tem como objetivo analisar o efeito de decrementar o ϵ ao longo do treinamento do agente. Representa o desempenho do agente que começa explorando muito o ambiente e à medida que seu conhecimento sobre o ambiente aumenta, a sua exploração diminui. Na Figura 5.19 a linha azul representa a média de pontuação por passo recebida durante a avaliação do agente durante o treinamento da política *decaying e-greedy*. No gráfico da Figura 5.19 é possível observar que o ganho de aprendizado do agente é constante e gradual, começando a aumentar a pontuação mesmo com alto grau de exploração. A partir do passo 1 milhão, quando a taxa de exploração atinge o seu mínimo, o crescimento começa a aumentar de forma mais acelerada, apesar disso a pontuação não se estabiliza em um valor.

A Tabela 5.19 apresenta os resultados obtidos pelo agente utilizando a política *decaying e-greedy*. Ela obteve resultado melhor que o aleatório, mas desempenho inferior ao humano.

Os resultados da avaliação do agente após 100 episódios de avaliação são apresentados na Tabela 5.19. Como é possível observar, houve um grande ganho em relação à política aleatória, mas um desempenho inferior ao humano. A pontuação média do agente foi maior que nas políticas anteriores, mas possui grande dispersão de valores por conta do alto desvio padrão.

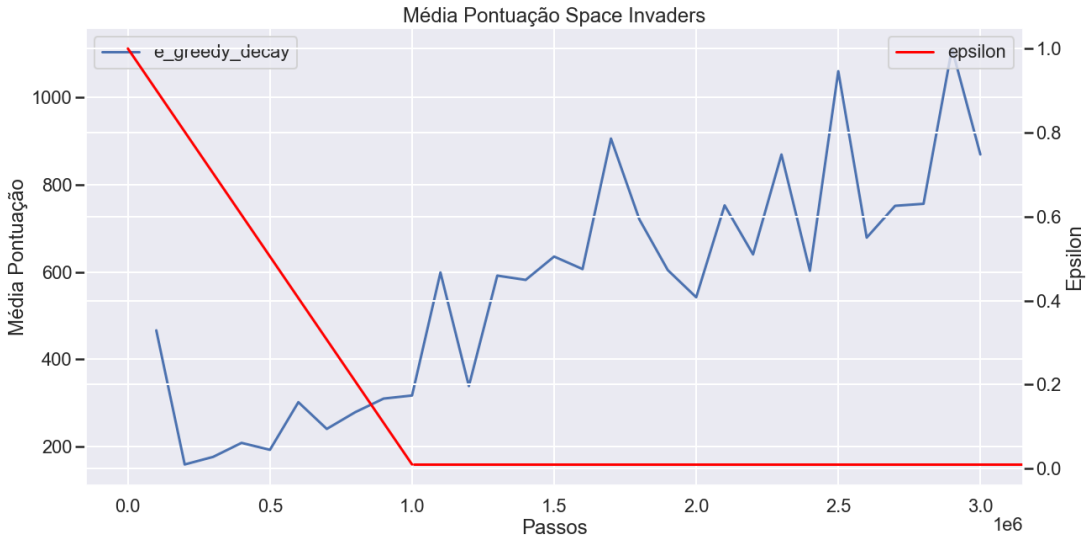

Figura 5.19: Média de pontuação por passo considerando a política *decaying epsilon greedy* no treinamento do jogo *Space Invaders*.

Tabela 5.19: Resultados da avaliação da política *decaying epsilon greedy* para o jogo *Space Invaders*.

DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
946.55 ± 325.98	**34.5%**	**1520.0**	**370.0**

Noise-based Exploration (Random Noise)

Como descrito na Seção 4.5.6, o objetivo deste experimento é analisar o efeito da exploração utilizando ruídos na DQN. Na Figura 5.20 as linhas (cor azul, amarelo e verde) representam a média de pontuação por passo recebida na avaliação do agente durante o treinamento da política *random noise*. As curvas de todos os valores de σ_0 crescem no mesmo ritmo até o passo 1,5 milhões, a partir desse ponto ocorrem variações nas curvas. No final do treinamento os agente com valores $\sigma_0 = 0.4$ e 0.5 possuem média de pontuação muito próximos.

A Tabela 5.20 apresenta os resultados obtidos durante a avaliação do agente que age utilizando a política de exploração *random noise*. Na Tabela 5.6 é possível observar que quanto maior o valor de σ_0 ou quanto maior a exploração maior a pontuação obtida pelo agente. Os melhores resultados foram obtidos com $\sigma_0 = 0.5$. Em todas as variações de σ_0 o desempenho foi pior que o humano. A dispersão dos valores de pontuação foram diretamente proporcionais ao σ_0.

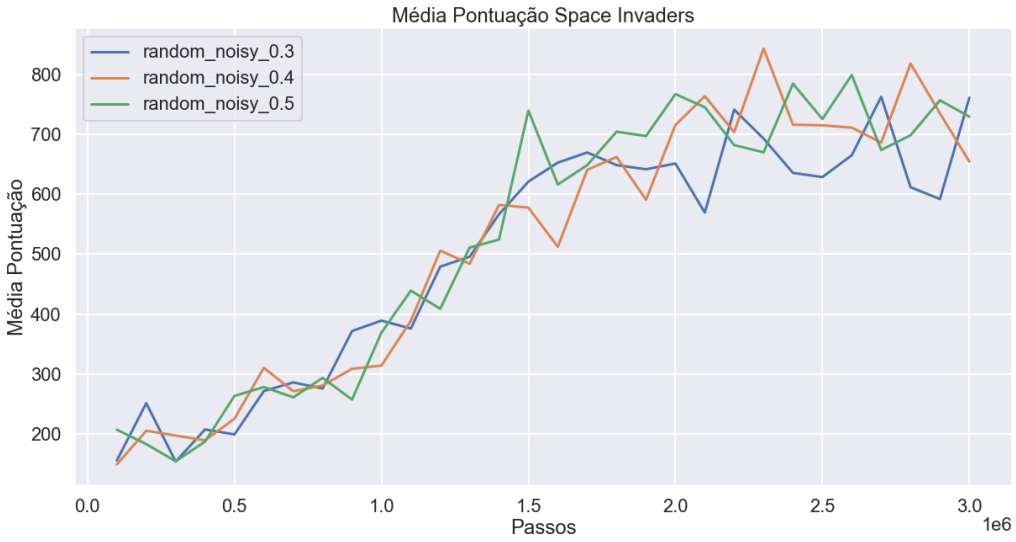

Figura 5.20: Média de pontuação por passo considerando a política *random noise* no treinamento do jogo *Space Invaders*.

Tabela 5.20: Resultados da avaliação da política *random noise* para o jogo *Space Invaders*.

σ_0	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
0.3	736.2 ± 296.41	15.0%	1575.0	430.0
0.4	753.05 ± 251.96	16.6%	1545.0	455.0
0.5	**814.15 ± 322.13**	**22.3%**	**2035.0**	**480.0**

5.3.2 Comparação dos melhores hiperparâmetros

Esta seção tem como objetivo comparar as políticas de exploração com os hiperparâmetros que obtiveram os melhores resultados no jogo *Space Invader*. No gráfico da Figura 5.14 é possível observar que o agente com a política *decaying e-greedy* (cor verde) obteve o melhor resultado durante o treinamento em comparação com as outras políticas. As políticas Boltzmann (cor azul) e *random noisy* (cor roxa) terminam o treinamento com a segunda maior pontuação. Com os menores resultados estão as políticas *e-greedy* (cor amarela), *greedy* (cor vermelha) e a aleatória (cor marrom), respectivamente.

Na Tabela 5.14 é possível observar os resultados das políticas de exploração com os melhores hiperparâmetros no jogo *Space Invaders*. O agente utilizando a política *decaying e-greedy* obteve a maior média de pontuação. A política *boltzmann* obteve pontuação um pouco menor, mas com menor dispersão. A política *greedy* apresentou desempenho mediano em relação às outras estratégias. As estratégias *e-greedy* e *random noise* obtiveram pontuação muito próxima e tiveram o pior desempenho em relação às outras. Todas as

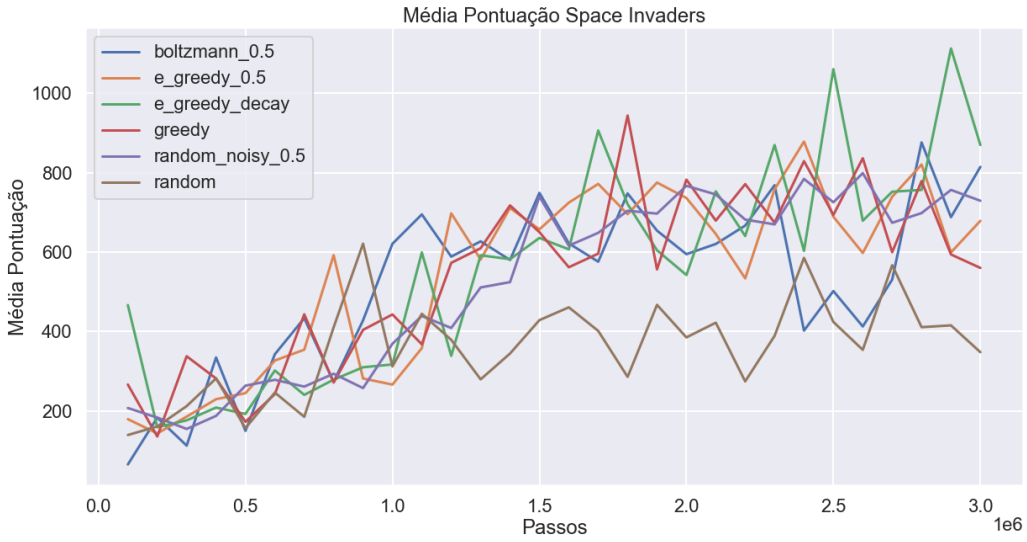

Figura 5.21: Média de pontuação por passo melhores política no treinamento do jogo *Space Invaders*.

Tabela 5.21: Melhores resultados em ordem decrescente por política para o jogo *Space Invaders*.

Experimento	DQN	DQN Normalizado	DQN Máximo	DQN Mínimo
decaying e-greedy	**946.55 ± 325.98**	**34.5%**	1520.0	370.0
boltzmann 0.5	911.45 ± 245.45	31.3%	1920.0	**520.0**
greedy	891.2 ± 269.68	29.4%	1720.0	510.0
e-greedy 0.5	820.8 ± 361.44	22.9%	1765.0	515.0
random noise 0.5	814.15 ± 322.13	22.3%	**2035.0**	480.0
aleatória	574.1 ± 302.06	0.0%	1010.0	165.0

políticas obtiveram desempenho maior que a política aleatória, mas menor que o desempenho humano.

5.4 Discussão

No ambiente *Pong*, as políticas com menor grau de exploração, i.e. que possuem hiperparâmetro com valor de 0.3, (Tabela 5.7) alcançaram médias de pontuação muito próximas, obtendo desempenho superior ao humano e a política aleatória. Com base na Tabela 5.7 é possível concluir que o jogo *Pong* não exige muito planejamento por parte do agente e portanto não é muito complexo. Por isso, estratégias que utilizam muita exploração obtiveram resultados piores que os que priorizam a estratégia *greedy*. Isso pode ser no-

tado pela média de pontuação menor obtido pelas políticas de exploração que possuem hiperparâmetros maiores que 0.3. Apesar disso, a estratégia *decaying e-greedy* que possui exploração que decai ao longo do treinamento foi a que obteve o melhor resultado. Com performance logo abaixo da política anterior, a política *e-greedy* com 30% de chance de exploração ($\epsilon = 0.3$) obteve melhor desempenho, uma vez seus resultados são mais consistente do que a política *greedy*. A estratégia *greedy* (sem exploração) é eficiente para maximizar a pontuação imediata, mas com exploração pode ser produzido um resultado total maior no fim do jogo [16]. Portanto, para conseguir uma alta pontuação no ambiente foi necessário um grau de exploração, ainda que o ambiente seja simples. A política de Boltzmann e a aleatória foram as que obtiveram os piores resultados, enquanto as outras políticas obtiveram resultados muito próximos.

No ambiente *Breakout*, as políticas que se saíram melhores foram a *random noise* e *decaying e-greedy* (Tabela 5.14). Essas políticas de exploração têm em comum a capacidade de planejar a longo prazo; começam o treinamento explorando mais o ambiente e terminam com menor grau de exploração, ou seja, sabem decidir quando devem explorar e quando devem seguir uma estratégia. O que reforça essa ideia é perceber que a estratégia ótima encontrada por esses agentes consiste em fazer um buraco nos blocos laterais, para que a bola fique presa na parte de trás dos blocos e o agente possa ganhar pontos sem se mover, esse fenômeno pode ser observado na Figura 5.22. A estratégia *greedy* obteve um desempenho mediano, apesar de ser capaz de planejar a longo prazo, não é capaz de encontrar a estratégia mais eficiente por falta de exploração. As políticas *e-greedy* e *boltzmann*, que possuem um mesmo nível de exploração durante todo o jogo, obtiveram resultados inferiores por possuírem menor capacidade de planejamento a longo prazo, escolhendo ações aleatórias quando deveriam escolher a melhor ação conhecida. Apesar disso, todas as estratégias da Tabela 5.14 tiveram desempenho maior que o humano e a política aleatória.

No ambiente *Space Invaders* (Tabela 5.21), ao contrário dos ambientes apresentados anteriormente, foi necessário mais exploração para obter alta pontuação. A estratégia que obteve o melhor resultado foi a *decaying e-greedy*, porque possui um grau de exploração que decai ao longo do treinamento, fazendo melhor uso do conhecimento à medida que aprende sobre o jogo. Não muito distantes dessa estratégia a política de Boltzmann se sai muito bem neste ambiente apesar de ter tido desempenho pobre nos outros. A política *greedy* obteve resultado mediano e as outras estratégias *e-greedy* e *random noise* obtiveram a pior média de pontuação. Nenhuma das estratégias obteve desempenho maior que o humano, enquanto no artigo de Mnih et al. [10] o desempenho foi maior que o humano, mas treinou o agente por 200 milhões de *frames*. Isso contribui para a ideia que ambientes mais complexos precisam de mais exploração e consequentemente demandam mais tempo

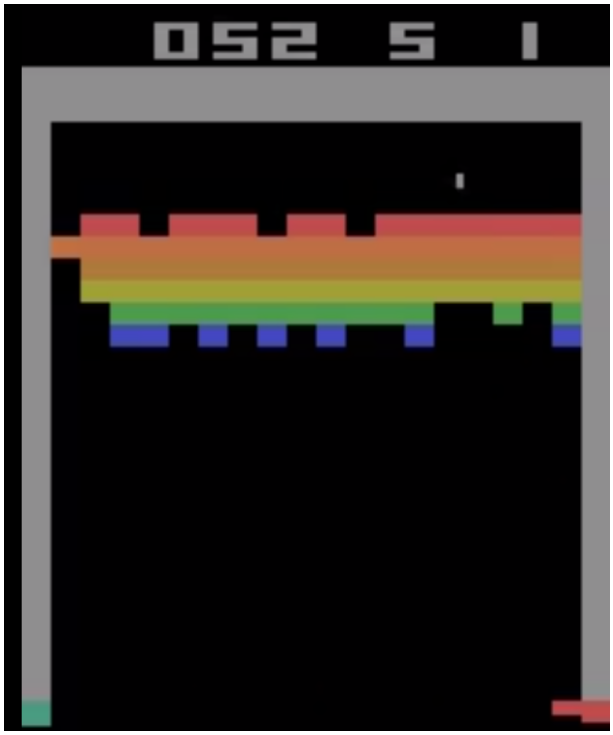

Figura 5.22: Estratégia de prender a bola nos blocos traseiro no treinamento do jogo *Breakout*.

para serem resolvidos, portanto, treinar o agente por mais tempo pode beneficiar nos resultados.

A política *random noise* obteve o melhor resultado apenas para 1 dos 3 jogos considerados no presente trabalho, e em todos os outros o desempenho foi menor que a estratégia *greedy*. Segundo Fortunato et al. [25] essa estratégia é vantajosa em ambientes que necessitam de padrões complexos de comportamento. O *Pong* é um jogo muito simples, então não se beneficia dessa estratégia. Por outro lado, no *Breakout* foi a política que obteve o melhor resultado por ser um ambiente que necessita de planejamento mais elaborado. No caso do *Space Invaders*, por ser uma estratégia que adiciona ruídos a DQN, seria necessário aumentar o número de passos durante o treinamento para que o agente se adapte melhor ao ruído e, portanto melhore seu resultados. Apesar disso, está política é pouco sensível, pois obteve um bom desempenho considerando diferentes valores de hiperparâmetros.

A estratégia de Boltzmann obteve um bom resultado apenas no *Space Invaders*, enquanto nos outros jogos estava entre os piores resultados, em alguns casos obteve média de pontuação pior que o agente aleatório. Na maioria dos casos, com $\tau = 0.3$ seu resultado se aproximou da política *greedy* e com $\tau = 0.7$ seu resultado se aproximou política aleatória, mas em ambos os casos convergiu de forma mais lenta [24]. Portanto, é a política mais volátil, pois possui a maior variação de resultados e em sua maioria obteve resulta-

dos ruins. Além disso, tem maior custo computacional e complexidade que a estratégia *e-greedy*, apesar de possuir resultados inferiores na maioria dos jogos.

A estratégia *greedy* obteve resultado intermediário, i.e. sempre está entre as duas melhores políticas e as duas piores políticas, em todos os jogos. Para o jogo mais simples, *Pong*, sofreu *overfitting* não sendo capaz de generalizar para diferentes variações do jogo, perdendo para o adversário. Para jogos mais complexos essa estratégia não foi muito eficiente porque pode cair em um valor máximo local, deixando de melhorar sua pontuação e até divergindo, não encontrando uma boa estratégia para o jogo. O método *greedy* tem um performance significativa pior a longo prazo, porque tem a tendência de ficar preso executando ações sub-ótimas [16].

A estratégia *e-greedy* obteve bom desempenho apenas quando a quantidade de exploração necessária não era alta, como no *Pong*. Nos outros jogos seu desempenho foi muito baixo, próximo do aleatório, isso pode ter ocorrido devido a falta de equilíbrio de quando deve fazer uma escolha de forma *greedy* e quando deve explorar o ambiente. Esse resultado foi diferente da literatura clássica de aprendizado por reforço [16], em que essa estratégia é melhor que a *greedy* na maioria dos casos. Isso pode ter ocorrido pelo fato dos ambientes possuírem um número muito grande de estados, e o tempo de treinamento não ser o suficiente para essa técnica convergir no *Breakout* e no *Space Invaders*. O método *e-greedy* tende a desempenhar melhor porque ele continua explorando, o que aumenta suas chances de reconhecer uma ação ótima.

A estratégia *decaying e-greedy* foi a que obteve bons resultados em todos os jogos, sendo que em 2 deles obteve a maior média de pontuação. Portanto, é possível concluir que essa estratégia foi a que melhor obteve resultado em ambientes de diferentes complexidades, sendo portanto uma estratégia de simples implementação e que apresenta ótimos resultados. Usando essa estratégia o agente utiliza ações aleatórias no início do treinamento quando as estimativas dos Q-valores é ruim e a medida que o treinamento progride menos ações aleatória são utilizadas dando espaço para a utilização dos Q-valores para decidir como agir [30].

Uma das limitações da metodologia do presente trabalho é o pequeno número de *frames* utilizado durante o treinamento dos agentes, não sendo possível comparar com os artigos mais recentes de aprendizado por reforço. Além disso, a quantidade de *frames* pode influenciar na melhora dos resultados, quanto mais tempo o agente interage com o ambiente mais conhecimento ele consegue obter. Outra limitação é não ter utilizado mais de uma política de exploração que foi criada exclusivamente para o *deep learning*. O trabalho de Yang et al. [39] faz a comparação de várias políticas de exploração utilizando apenas *deep learning* e serve de complemento para as políticas que não foram abordadas, além de expandir a discussão do problema de exploração.

Capítulo 6

Conclusões e Trabalhos Futuros

No presente trabalho foram apresentadas seis políticas de exploração: *random*, *greedy*, *e-greedy*, Boltzmann, *decaying e-greedy*, *random noise*. Um agente foi treinado utilizando cada uma dessas políticas em conjunto com três diferentes ambientes do Atari: *Pong*, *Breakout* e *Space Invaders*. Os agentes utilizaram a mesma arquitetura, hiperparâmetros de configuração da DQN e método de treinamento. Durante os experimentos foram estabelecidos que cada ambiente se beneficiava de estratégias de exploração de forma diferente.

O ambiente *Pong* é o mais simples, mesmo com baixa taxa de exploração os agentes foram capazes de obter bons resultados com todas as políticas. No jogo *Space Invaders* foi necessário mais planejamento e portanto mais exploração para se obter um melhor resultado. O jogo *Breakout* se beneficiou de estratégia de exploração mais que *Pong*, mas menos que o *Space Invaders* possuindo um resultado intermediário. Nos jogos *Pong* e *Breakout* os melhores agentes foram capazes de obter pontuação maior ou igual a de jogadores humanos, enquanto no *Space Invaders* isso não foi possível, sendo, portanto um jogo mais complexo que seria necessário maior tempo de treinamento para atingir o desempenho humano. De forma geral, a estratégia que obteve um bom desempenho em todos os jogos foi a *decaying e-greedy*, porque essa política consegue simular a melhor estratégia de jogo, que é explorar mais o ambiente enquanto se conhece pouco sobre ele e a medida que esse conhecimento aumenta, diminuir a exploração e agir de forma gulosa com maior frequência. Essa política de exploração clássica superou a política *random noise* em 2 dos 3 jogos considerados no presente trabalho. A principal contribuição do presente trabalho foi comparar as diferentes políticas de exploração citadas anteriormente, avaliando o impacto de cada uma na pontuação do agente nos jogos de Atari e determinando qual política tem melhor desempenho em diferentes condições.

Como proposta de trabalhos futuros, seria interessante adicionar mais estratégias de exploração, com estruturas mais complexas utilizando *deep learning* como: motivação

intrínseca [40] e exploração por curiosidade [41]. Outro ponto de melhoria é realizar o treinamento do agente durante 200 milhões de *frames* para ter resultado comparável com outros trabalhos [10, 12, 35] e também para obter melhores resultados em ambientes que são mais complexos como o *Space Invaders*, e portanto se beneficiam com mais tempo de treinamento. Outra adição é fazer a comparação utilizando mais jogos do Atari 2600. Além disso, verificar qual política de exploração é a mais vantajosa para cada estilo de jogo.

Referências

[1] Nielsen, Michael A.: *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/, acesso em 2022-03-20. ix, 6

[2] Pramoditha, Rukshan: *The concept of artificial neurons (perceptrons) in neural networks*, Dec 2021. https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc, acesso em 2022-03-17. ix, 6

[3] Swapna, K E: *Convolution neural network (cnn)*. https://developersbreach.com/convolution-neural-network-deep-learning/, acesso em 2022-04-18. ix, 8

[4] Dumoulin, Vincent e Francesco Visin: *A guide to convolution arithmetic for deep learning*. arXiv preprint arXiv:1603.07285, 2016. https://arxiv.org/abs/1603.07285. ix, 9

[5] Neves, Enzo Cardeal: *Aprendizado por reforço 1- introdução*, 2020. https://medium.com/turing-talks/aprendizado-por-refor%C3%A7o-1-introdu%C3%A7%C3%A3o-7382ebb641ab, acesso em 2022-03-18. ix, 11

[6] Choudhary, Ankit: *Deep q-learning: An introduction to deep reinforcement learning*, Apr 2019. https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/, acesso em 2022-04-23. ix, 11, 14

[7] Gron, Aurlien: *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edição, 2017, ISBN 1491962291. ix, x, 7, 10, 11, 12, 13, 14, 29, 40

[8] Meneghetti, Douglas De Rizzo, May 2020. https://douglasrizzo.com.br/boltzmann-policy-temperature/, acesso em 2022-05-08. ix, 19

[9] Zhai, Jianwei, Quan Liu, Zongzhang Zhang, Shan Zhong, Haijun Zhu, Peng Zhang e Cijia Sun: *Deep Q-Learning with Prioritized Sampling*. Em Hirose, Akira, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee e Derong Liu (editores): *Neural Information Processing*, páginas 13–22. Springer International Publishing, 2016, ISBN 978-3-319-46687-3. x, 30

[10] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski *et al.*: *Human-level control through deep reinforcement learning*. nature, 518(7540):529–533, 2015. x, 2, 3, 9, 15, 16, 17, 21, 22, 23, 24, 29, 30, 31, 32, 33, 34, 63, 67

[11] Fia: *Inteligência artificial: O que É, como funciona e exemplos*, Oct 2021. https://fia.com.br/blog/inteligencia-artificial/, acesso em 2022-03-20. 1

[12] Hessel, Matteo, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar e David Silver: *Rainbow: Combining improvements in deep reinforcement learning.* Em *Thirty-second AAAI conference on artificial intelligence*, 2018. 1, 22, 23, 24, 30, 32, 67

[13] Wu, Yunhan, Daniel Rough, Anna Bleakley, Justin Edwards, Orla Cooney, Philip R Doyle, Leigh Clark e Benjamin R Cowan: *See what i'm saying? comparing intelligent personal assistant use for native and non-native language speakers.* Em *22nd international conference on human-computer interaction with mobile devices and services*, páginas 1–9, 2020. 1

[14] Xie, Ning, Farley Lai, Derek Doran e Asim Kadav: *Visual entailment: A novel task for fine-grained image understanding.* arXiv preprint arXiv:1901.06706, 2019. 1

[15] RUSSEL, Stuart e Peter NORVIG: *Inteligência Artificial*, volume 3. Rio de Janeiro: Elsevier, 2013. 1, 2

[16] Sutton, Richard S e Andrew G Barto: *Reinforcement learning: An introduction.* MIT press, 2018. 1, 2, 3, 9, 11, 12, 14, 17, 18, 19, 34, 35, 63, 65

[17] Academy, Data Science: *Deep learning book*, 2021. https://www.deeplearningbook.com.br/, acesso em 2022-02-20. 1, 5, 7, 8

[18] Rosenblatt, Frank: *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 65(6):386, 1958. 1

[19] El-Tantawy, Samah, Baher Abdulhai e Hossam Abdelgawad: *Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto.* IEEE Transactions on Intelligent Transportation Systems, 14(3):1140–1150, 2013. 2

[20] Ling, Yuan, Sadid A. Hasan, Vivek Datla, Ashequl Qadir, Kathy Lee, Joey Liu e Oladimeji Farri: *Diagnostic inferencing via improving clinical concept extraction with deep reinforcement learning: A preliminary study.* Em Doshi-Velez, Finale, Jim Fackler, David Kale, Rajesh Ranganath, Byron Wallace e Jenna Wiens (editores): *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 de *Proceedings of Machine Learning Research*, páginas 271–285. PMLR, 18–19 Aug 2017. 2

[21] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra e Martin Riedmiller: *Playing atari with deep reinforcement learning.* arXiv preprint arXiv:1312.5602, 2013. 2, 10, 13, 14, 15, 16, 26, 29, 31, 32

[22] Bellemare, Marc G, Yavar Naddaf, Joel Veness e Michael Bowling: *The arcade learning environment: An evaluation platform for general agents.* Journal of Artificial Intelligence Research, 47:253–279, 2013. 2, 26

[23] Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot *et al.*: *Mastering the game of go with deep neural networks and tree search*. nature, 529(7587):484–489, 2016. 3, 21, 24

[24] McFarlane, Roger: *A survey of exploration strategies in reinforcement learning*. McGill University, 2018. 3, 17, 18, 19, 34, 64

[25] Fortunato, Meire, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin *et al.*: *Noisy networks for exploration*. arXiv preprint arXiv:1706.10295, 2017. 3, 17, 20, 22, 23, 24, 32, 34, 36, 64

[26] *Deep learning*. Nature, 521(7553):436–444, 2015. 5, 7

[27] Welch, Stephen: *Neural networks demystified*. `https://www.youtube.com/watch?v=bxe2T-V8XRs`, acesso em 2022-03-12. 5

[28] Howard, J. e S. Gugger: *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated, 2020, ISBN 9781492045526. `https://books.google.no/books?id=xd6LxgEACAAJ`. 7

[29] Hornik, Kurt, Maxwell Stinchcombe e Halbert White: *Multilayer feedforward networks are universal approximators*. Neural networks, 2(5):359–366, 1989. 7

[30] Lapan, Maxim: *Deep Reinforcement Learning Hands-On*. Packt Publishing, 2018, ISBN 978-1-78883-424-7. 9, 10, 11, 14, 19, 65

[31] D, P.W.P. e P. Winder: *Reinforcement Learning: Industrial Applications of Intelligent Agents*. O'Reilly Media, Incorporated, 2020, ISBN 9781098114831. 10

[32] Chan, Ka, C. Lenard e Terence Mills: *An introduction to markov chains*. dezembro 2012. 12, 29

[33] Bellman, Richard: *A markovian decision process*. Journal of mathematics and mechanics, páginas 679–684, 1957. 12

[34] Weng, Lilian: *Exploration strategies in deep reinforcement learning*. lilianweng.github.io, 2020. `https://lilianweng.github.io/posts/2020-06-07-exploration-drl/`. 17

[35] Van Hasselt, Hado, Arthur Guez e David Silver: *Deep reinforcement learning with double q-learning*. Em *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 23, 31, 32, 33, 34, 36, 37, 46, 55, 67

[36] Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang e Wojciech Zaremba: *Openai gym*. arXiv preprint arXiv:1606.01540, 2016. 26

[37] Machado, Marlos C, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht e Michael Bowling: *Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents*. Journal of Artificial Intelligence Research, 61:523–562, 2018. 33

[38] Fujita, Yasuhiro, Prabhat Nagarajan, Toshiki Kataoka e Takahiro Ishikawa: *Chainerrl: A deep reinforcement learning library*. Journal of Machine Learning Research, 22(77):1–14, 2021. `http://jmlr.org/papers/v22/20-376.html`. 37

[39] Yang, Tianpei, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, Peng Liu e Zhen Wang: *Exploration in Deep Reinforcement Learning: A Comprehensive Survey*. páginas 1–24, Jan. `http://arxiv.org/abs/2109.06668`. 65

[40] Bellemare, Marc G., Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton e Remi Munos: *Unifying count-based exploration and intrinsic motivation*, 2016. 67

[41] Pathak, Deepak, Pulkit Agrawal, Alexei A. Efros e Trevor Darrell: *Curiosity-driven exploration by self-supervised prediction*, 2017. 67