Arthur Phillip Ferreira da Silva

Desenvolvimento de Inteligência Artificial em Jogos Allegro

Belo Horizonte, Minas Gerais
2021

Arthur Phillip Ferreira da Silva

Desenvolvimento de Inteligência Artificial em Jogos Allegro

Pesquisa Tecnológica

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Pedro O.S. Vaz de Melo

Belo Horizonte, Minas Gerais
2021

Sumário

1 Introdução

A Inteligência Artificial (IA) é um dos objetos de estudo mais pesquisados da atualidade por seu vasto espectro de utilização. Dado que uma grande busca das pesquisas em computação é o desenvolvimento de soluções adaptativas que não precisem de uma explicita definição de todas as possíveis situações para que se tenha uma ampla cobertura de domínio. estudo de IA tem como objetivo o desenvolvimento de algoritmos de competência geral em uma variedade de tarefas e domínios sem a necessidade de explicitar as características específicas do domínio. [Bellemare et al., 2013]

Segundo Yannakakis and Togelius, o campo da inteligência artificial e computacional em jogos não só tem visto grandes avanços como tem em seu currículo várias histórias de sucesso. Em seu livro, Yannakakis and Togelius apresenta a longa história que os jogos e a inteligência artificial têm juntos. Como também as pesquisas sobre IA para jogos e como estão relacionadas à construção de agentes que joguem, com ou sem aprendizado prévio. Em suas palavras, estes autores disseram:

> "Mesmo antes de a inteligência artificial ser reconhecida como um campo, os primeiros pioneiros da ciência da computação escreveram programas de jogo porque queriam testar se os computadores poderiam resolver tarefas que pareciam exigir inteligência".

Segundo Russell, o conceito de agência racional é considerado um dos principais candidatos a cumprir o papel de uma inteligência, os atributos para isso são ter precisão, permitir o desenvolvimento cumulativo de sistemas robustos e ser generalizável. O estudo de IA aplicado a jogos não só é antigo como se especializou em áreas que abrangem grupos de jogos ou mesmo funções dentro do jogo. Outro trabalho de Yannakakis and Togelius mostra como essas áreas podem se beneficiar do conhecimento criado em outras áreas e fazer sua própria pesquisa mais relevante para as outras áreas.

Porém, as pesquisas de IA em jogos sofrem com o fato de, além de se preocupar com o desenvolvimento da IA, devem obter uma solução a outro problema: como as decisões do agente racional serão aplicadas no jogo, ou seja, como a IA vai jogar de fato. Para viabilizar os estudos nessa área, algumas soluções foram propostas, *Arcade Learning Environment* (ALE) [Bellemare et al., 2013], por exemplo, utiliza um emulador de Atari para interfacear a interação entre os agentes racionais e os jogos desse console. *General Video Game Playing* (GVGP) [Perez-Liebana et al., 2018] expande a proposta anterior com uma plataforma de código aberto que visa criar controladores para reprodução geral de videogames, possibilitando e incentivando a pesquisa em qualquer jogo de videogame.

1.1 Objetivos Gerais

Um ramo de estudo de IA é sua aplicação em jogos, na construção de agentes racionais capazes de aprender e jogar autonomamente como é o caso do *Alfa Go* da *DeepMind* que foi capaz de derrotar o campeão mundial de Go.

Para incentivar e viabilizar o estudo de IA em jogos foram desenvolvidas estruturas e bibliotecas que auxiliam na integração do agente e o jogo. Estas soluções se dedicam a criar uma conexão com o jogo através de emuladores dos consoles permitindo que o agente simule um humano jogando videogame.

O objetivo desse trabalho é propiciar um ambiente facilitador ao estudo de soluções de IA aplicada em jogos. Uma plataforma que permita que o pesquisador possa dedicar sua atenção e tempo às estratégias de decisão sem se preocupar na integração do sistema com o(s) jogo(s) escolhido(s).

1.2 Objetivos Específicos

Este trabalho apresenta três desafios principais, sendo que o primeiro é o estado do jogo. A estrutura desenvolvida anteriormente fornece para a IA a imagem da tela em um momento escolhido, portanto deve-se transformar essa imagem em dados quantizáveis que a IA possa analisar para tomar sua decisão. Isso será feito utilizando uma rede neural convolucional que foi escolhida por demandar um menor nível de pré-processamento quando comparada a outros algoritmos de classificação de imagens.

Outro desafio, é o treinamento da AI para que possa jogar os jogos de forma eficiente, e para isso, foi decidido utilizar a aprendizagem por reforço que se preocupa com o como um agente deve agir em um ambiente de forma que maximize alguma noção de recompensa a longo do tempo.

Por último, a escolha dos jogos a serem utilizados no trabalho. Foram utilizados os jogos Frogger e Pong em suas versões simplificadas.

2 Referencial Teórico

Existem algumas soluções bem famosas de IA que conseguem derrotar campeões mundiais de alguns jogos. Até então, não se pensava que uma máquina conseguiria fazer estratégias tão sofisticadas em jogos de raciocínio lógico. *Alfa Go* [Silver et al., 2016], por exemplo, é um algoritmo que utiliza redes neurais para jogar Go, um jogo milenar muito famoso na China, e derrotou o campeão mundial Lee Sedol em 2016. *Alfa zero* [Silver et al., 2018], o sucessor de *Alfa Go*, através de uma rede muito mais ampla consegue jogar Xadrez e Shogi além do original Go. *DeepMind*, a empresa criadora dos dois anteriores, desenvolveu, recentemente, o *Alfa Star* [Vinyals et al., 2019], lançado no inicio de 2019 em um confronto com finalistas do mundial do jogo *Star Craft* e demonstrou uma grande performance vencendo a maioria das partidas.

A competição da *General Video Game AI* (GVGAI) foi fundada na crença de que a melhor maneira de impedir que os pesquisadores de IA dependam de engenharia específica de jogo em seus agentes. Os pesquisadores desenvolveriam seus agentes sem saber quais jogos iriam jogar e, depois de submeter seus agentes à competição, todos os agentes seriam avaliados usando um conjunto de jogos invisíveis [Perez-Liebana et al., 2018]. GVGP é uma proposta, também chamada de GVGAI *framework*, que possibilita o controle dos jogos, fazendo a interface entre o agente e diversos jogos de vários consoles. Um *framework* que padroniza o controle dos jogos para a criação dos agentes racionais que participarão da competição GVGAI.

ALE é uma estrutura de software para interface com ambientes de jogos Atari 2600 emulados. Ele permite que o usuário faça interface com o Atari 2600 recebendo movimentos do joystick, enviando informações de tela e/ou RAM e emulando a plataforma. O ALE também fornece uma camada de manipulação de jogos que transforma cada jogo em um problema padrão de aprendizado de reforço, identificando a pontuação acumulada e se o jogo terminou [Bellemare et al., 2013].

Os algoritmos de aprendizagem por reforço andam de mãos dadas com o desenvolvimento de ambientes desafiadores que testam os limites dos métodos atuais, segundo o artigo de Küttler et al. lançado em 2020, no qual essa estratégia foi utilizada para solucionar um jogo rougelike chamado NetHack.

3 Metodologia

3.1 Biblioteca Allegro

Allegro é uma biblioteca de código aberto para desenvolvimento de jogos em C/C++, simples e robusta. Ela fornece rotinas de baixo nível, normalmente necessárias, na programação de jogos, criação de janelas, aceitação de entrada do usuário, carregamento de dados, desenho de imagens, reprodução de sons e temporização [Leverton, acesso em 2019]. O acesso ao código fonte proporciona ao pesquisador maior controle sobre o ambiente de aprendizado dos agentes. Acessando o código fonte do jogo e provendo um canal e interação com o agente, é possível aumentar a autonomia e flexibilidade da pesquisa. O que traz uma possibilidade do pesquisador ter controle não só do agente, como também do seu ambiente de atuação que é o jogo.

3.2 Ambiente Allegro

A estrutura desenvolvida foi dividida em três atividades fundamentais que correspondem aos três módulos principais: Identificador, Atuador e Estado. O Identificador é a parte responsável por fornecer quais são as ações possíveis de atuação do agente sobre o jogo; o Atuador é aquele que transforma as decisões do agente em ações de fato no jogo; o Estado tem a função de captar o estado do jogo e fornecer para o agente informação necessária para sua tomada de decisão. Figura 1

3.2.1 Identificador

Sobre o texto do código fonte, são feitas buscas por reconhecimento de caracteres e são recuperados os comandos possíveis a serem executados durante o jogo.

As funções e variáveis da biblioteca Allegro foram usadas como marcadores. Isto facilitou a obtenção dos comandos necessários para manipulação do jogo pois, foi a partir disso, que se pensou em fazer uma busca por esses marcadores e retirar dali as informações necessárias.

Os principais marcadores foram:
Allegro_KEY_X representa o uso do teclado no jogo onde X é a tecla a ser pressionada.
mouse indica a utilização do mouse no jogo.

3.2.2 Atuador

Esta parte é responsável por traduzir as decisões da IA para ações no jogo e, para isso, foram utilizadas as bibliotecas Pyautogui e Pygetwindow. Pyautogui é uma biblioteca multiplataforma que acessa as APIs de sistema para enviar comandos de teclado e mouse, simulando os periféricos. Através da Pygetwindow é possível identificar e manipular as janelas de todos os programas abertos, porém, só funciona em Windows. O uso dessas bibliotecas permite identificar e manipular o jogo como se fosse um usuário humano, assim, o Atuador recebe as decisões do agente e executa a ação através do mouse ou teclado dentro do escopo da tela pertencente ao jogo.

3.2.3 Estado

Para que o agente possa tomar uma boa decisão, esta precisa ter ciência do atual estado do jogo, assim como qual o impacto de sua ultima ação sobre ele. A solução mais prática, e a adotada neste trabalho, é fornecer para o agente uma imagem da tela do jogo.

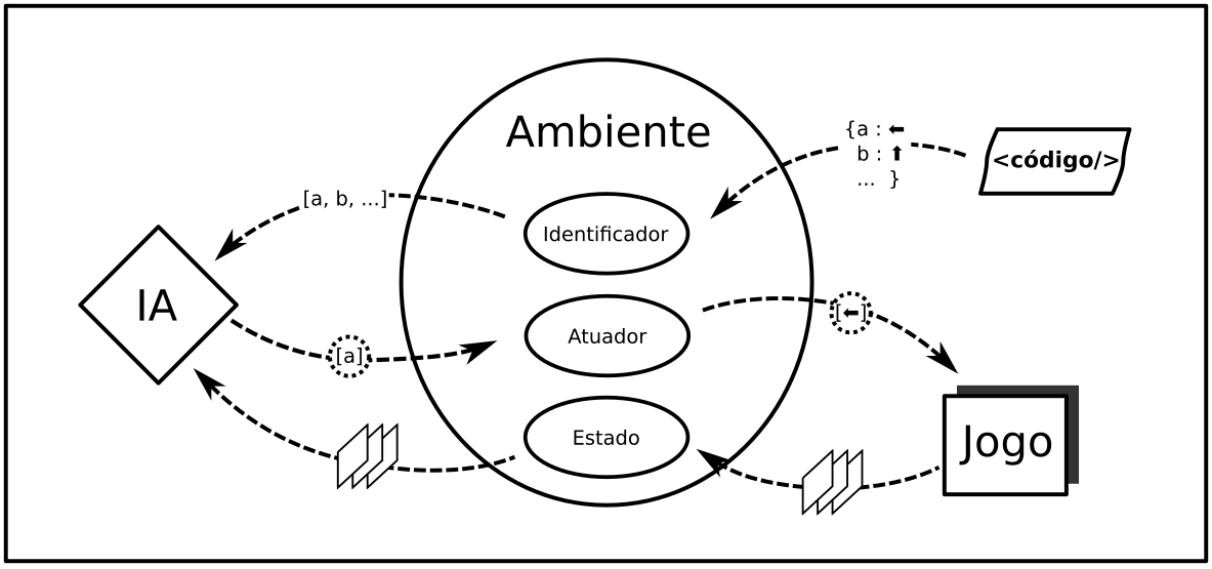

Figura 1 – Fluxo de dados

1. A estrutura lê o código e fornece informações ao agente.
2. A estrutura inicia o jogo.
3. O Agente envia comandos à estrutura.
4. A estrutura aplica no jogo os comandos recebidos.
5. O estado do jogo é enviado ao agente.
6. Enquanto o jogo não termina, volta ao passo 3.

3.3 Aprendizado por reforço

O aprendizado por reforço ganhou muita atenção pelo sucesso do sistema AlphaGo da DeepMind, derrotando o jogador Go campeão mundial. Sistema que foi treinado em parte

pelo aprendizado por reforço em redes neurais profundas. Esse tipo de aprendizado é um aspecto que se difere dos paradigmas clássicos supervisionados e não supervisionados de aprendizado de máquina [Thomas, acesso em 2021a].

Aprendizagem supervisionada consiste em aprender a partir de um conjunto de treinamento de exemplos rotulados fornecidos. Cada exemplo é um conjunto de dados com uma descrição de uma situação com uma especificação, ou rótulo, da melhor ação que o sistema deve tomar para essa situação, em muitos casos, para identificar em qual categoria uma situação pertence. O objetivo da aprendizagem supervisionada é que o sistema generalize sua resposta e que ele atue corretamente em situações que não estão presentes no conjunto de treinamento. Esse é um tipo de aprendizado importante, porém, não é adequado para aprender com a interação. Entretanto, em problemas interativos, é muito difícil obter exemplos de comportamento que abranjam todas as situações considerando a ação correta que o agente deve escolher.

O aprendizado por reforço também é diferente do aprendizado não supervisionado, pois este se refere, normalmente, à localização de estruturas ocultas em conjuntos de dados não rotulados. O aprendizado supervisionado e o aprendizado não supervisionado não são suficientes para classificar, exaustivamente, os paradigmas do aprendizado de máquina. Mesmo que a aprendizagem por reforço possa parecer um tipo de aprendizagem não supervisionada, ela se difere por não buscar uma estrutura oculta. Por outro lado, o aprendizado por reforço busca maximizar um sinal de recompensa. Encontrar uma estrutura no comportamento de um agente, embora possa ser útil, não resolve o problema do agente de aprendizagem por reforço de maximizar um sinal de recompensa. Então, o aprendizado por reforço se configura como um terceiro paradigma de aprendizado de máquina, ao lado do aprendizado supervisionado, do aprendizado não supervisionado e talvez também de outros menos explorados. [Sutton and Barto, 2015]

3.4 Rede Neural Convolucional

A Rede Neural Convolucional (CNN do inglês *Convolutional Neural Network*) teve excelentes resultados em vários campos de pesquisa relacionados ao reconhecimento de padrões, ao processamento de imagem e ao reconhecimento de voz. Seu nome deriva da operação matemática linear entre matrizes chamada convolução. Ela pertence ao grupo das Redes Neurais Artificiais (RNA) com múltiplas camadas, que é uma das ferramentas mais poderosas por ser capaz de lidar com uma grande quantidade de dados. Porém, as CNNs trazem um benefício que é a redução do número de parâmetros na RNA devido a ter camadas ocultas mais profundas. Essa conquista levou pesquisadores e desenvolvedores a abordarem modelos maiores, a fim de resolver tarefas complexas, o que não era possível com as RNAs clássicas, o que gerou muito interesse na comunidade em diferentes campos, especialmente, no reconhecimento de padrões. [Albawi et al., 2017]

A CNN tem várias camadas que incluem a camada convolucional, que pode ser mais facilmente entendida como um "filtro móvel" que passa pela imagem. Este filtro móvel, ou convolução, aplica-se a uma certa vizinhança e, então, é gerada uma combinação das coordenadas para preencher os nós de saída.

Graças a isso, em uma CNN nem todos os nós de entrada estão conectados aos nós de saída. Isso é contrário às redes neurais totalmente conectadas, em que cada nó de uma camada está conectado a todos os nós da camada seguinte. Outro ponto é que cada filtro tem parâmetros constantes. Ou seja, conforme o filtro se move ao redor da imagem, os mesmos pesos são aplicados. Cada filtro realiza uma certa transformação em toda a imagem. Isso está em contraste com as redes neurais totalmente conectadas, que têm um valor de peso diferente para cada conexão. Com isso, as CNNs podem reduzir significativamente o número de parâmetros necessários na rede, em comparação com redes neurais totalmente conectadas. [Thomas, acesso em 2021b]

3.5 Jogos

Para testar o funcionamento da interface, foram utilizados os jogos a seguir, desenvolvidos de forma simplificada em Allegro. Para este trabalho, foi necessário inserir pequenas adaptações nos jogos para que eles fornecessem a recompensa das ações do agente, o que é necessário para a Aprendizagem por reforço. Através da função inserida, os jogos emitem a pontuação total até o momento, o que é capturado pelo ambiente desenvolvido e fornecido ao agente para que este use em seu treinamento.

3.5.1 Frogger

Frogger é um jogo em que um sapo que tem que atravessar uma rua de várias pistas sem ser atropelado. A simplificação adotada nesse trabalho representa o sapo como um quadrado verde que inicia no centro do limite inferior da tela e tem que percorrer toda a tela para chegar no limite superior e, assim, vencer o jogo. Figura 2

Os carros são representados por retângulos coloridos, de diferentes comprimentos, que caminham em linha da esquerda para a direita a velocidades distintas. O jogo foi implementado de dois modos Fixo e Aleatório. No modo Fixo, apesar dos carros terem tamanhos e velocidades diferentes, eles não variam de jogo pra jogo, além de começarem todos na extrema direita da tela. No Aleatório, a cada execução do jogo os carros mudam de tamanho, velocidade e posição inicial.

A regra de recompensa utilizada foi a seguinte:

- Subir: +5 pontos.
- Descer: -5 pontos.

Figura 2 – Frogger

- Ir para os lados: -1 ponto.
- Vencer: +50 pontos

O desafio da IA foi de mover o sapo na direção e tempo correto para não causar colisões, lembrando que, ao ficar parado, ele também pode ser atropelado.

3.5.2 Pong

O Pong é um jogo que simula um tênis de mesa. O jogador controla uma barra vertical no jogo movendo-a, verticalmente, no lado esquerdo da tela, e compete contra o computador que controla uma segunda barra no lado oposto. O jogador usa sua barra para acertar o quadrado (bola) e mandá-la para o outro lado, como mostra a Figura 3.

O competidor que não conseguir acertar a bola perde o jogo. Nesta simplificação, a velocidade da bola não aumenta com o tempo, mas varia dependendo de onde bate na barra.

Figura 3 – Pong

A regra de recompensa utilizada foi a seguinte:

- Rebater: +5 pontos.

O desafio da IA foi de mover a barra na direção e tempo corretos, para não deixar que a bola bata em seu lado da tela .

3.6 Desenvolvimento

Este trabalho foi desenvolvido utilizando TensorFlow, um *framework* de código aberto que provê um vasto ferramental para *machine learning* e *deep learning*.

Para o treinamento foi utilizado um modelo Keras, um modelo de treinamento do TensorFlow. Nele foram colocadas duas CNNs de 16 e 32 camadas e duas redes densamente conectadas de 256 camadas e uma outra com o número de ações possíveis. A cada treinamento, os estados do jogo passam pelas duas CNNs e são achatados, o resultado segue dois caminhos, no primeiro ele passa por redes densamente conectadas e maiores pela rede das ações, onde se obtém a probabilidade de cada ação ser a melhor, no segundo, o resultado passa pela outra rede densamente conectada e é normalizado. Ambos os resultados são combinados visando, assim, ter um treinamento mais eficiente.

Para reduzir o custo de processamento, as imagens da tela do jogo obtidas passam por um tratamento. Primeiramente, a imagem é transformada em tons de cinza, depois reduzida quatro vezes em tamanho e, por fim, tem seus valores normalizados. Essas imagens são armazenadas numa pilha e, no treinamento, são fornecidas as três ultimas imagens, para que a rede neural tenha um senso de movimento.

Para cada jogo, o agente aplicou o algoritmo de aprendizado por 10.000 épocas, sendo que as primeiras 5.000 ações eram aleatórias, que equivalem, aproximadamente, a 500 épocas no Frogger e 150 no Pong, isso é necessário para a formação do conjunto mínimo de amostras que o treinamento precisa. Após esse período, as escolhas das ações são feitas pelo agente, porém, um percentual e de vezes a escolha ainda é aleatória visando evitar que as escolhas não fiquem enviesadas. O e representa 100% no início do Treinamento e vai decaindo progressivamente até o valor de 5%.

A cada ação executada no jogo, são armazenadas em uma memória as três últimas imagens, a ação adotada e a recompensa obtida. Após as 5.000 ações iniciais, a cada ação é feito o treinamento do agente com uma amostra aleatória de estados retirados da memória.

4 Análise dos Resultados

Os jogos foram treinados por 10.000 épocas, onde o agente foi aprendendo incrementalmente como atuar em cada situação do jogo. Os resultados de cada interação foram registrados e, a partir deles, foram gerados os gráficos a seguir. A performance do agente pode ser observada, para cada jogo, no gráfico dos resultados brutos e, no gráfico da média das últimas 100 épocas, é possível ver melhor a evolução média do agente no jogo.

4.1 Frogger

Em Frogger, o agente teve um desempenho bem significativo. Na figura 5 pode-se ver a evolução média crescente do aprendizado, quase chegando ao valor máximo do jogo, 45pontos. Na figura 4 pode-se ver que esse valor foi atingido muito mais vezes que valores negativos ou próximos de zero. Em comparação com a abordagem puramente aleatória (até a época 500 aproximadamente), pode se dizer que o agente se saiu muito bem saindo de uma média de -5 pontos para uma de 30 pontos a cada 100 jogos.

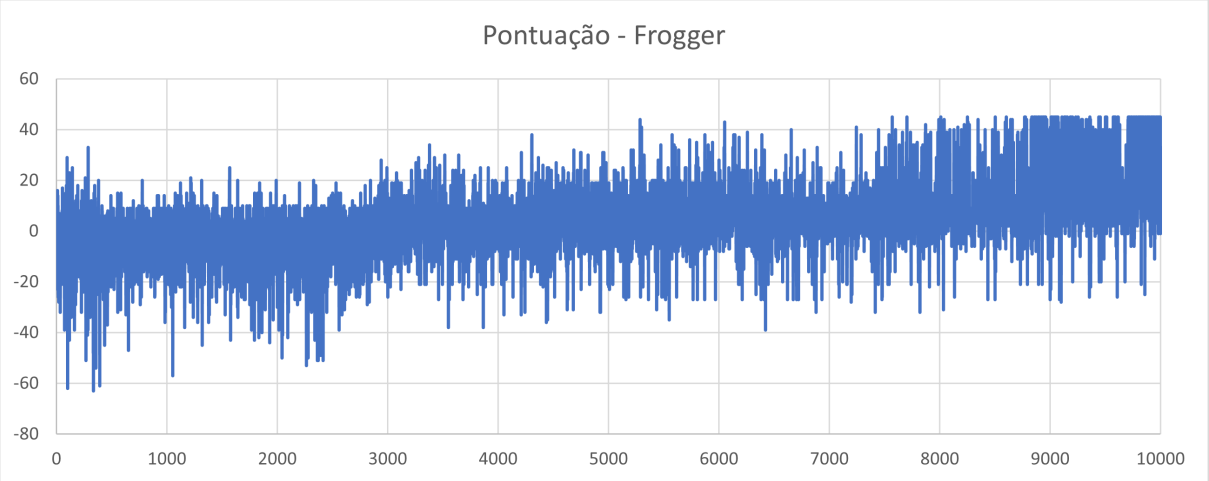

Figura 4 – Pontuação por época

4.2 Frogger - aleatório

Em comparação com o jogo anterior, onde os carros começavam sempre com as mesmas posições e velocidades de uma época para outra, pode-se dize que o agente teve um pouco mais de dificuldades para aprender o jogo. Embora a figura 6 mostre que conseguiu atingir a pontuação máxima muitas vezes, na figura 7, vê-se que a média dos resultados é bem menor.

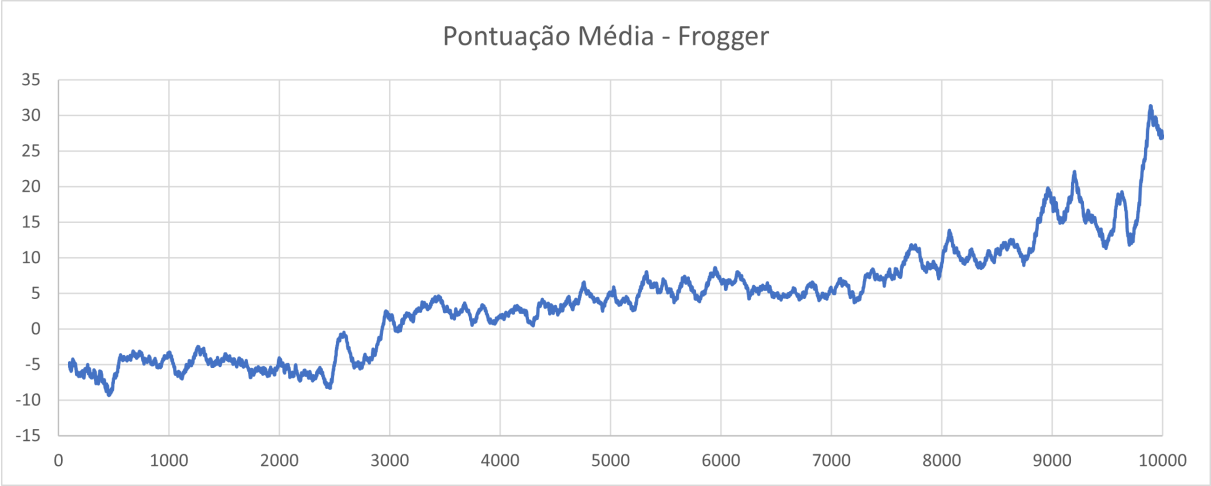

Figura 5 – Pontuação média das ultimas 100 épocas

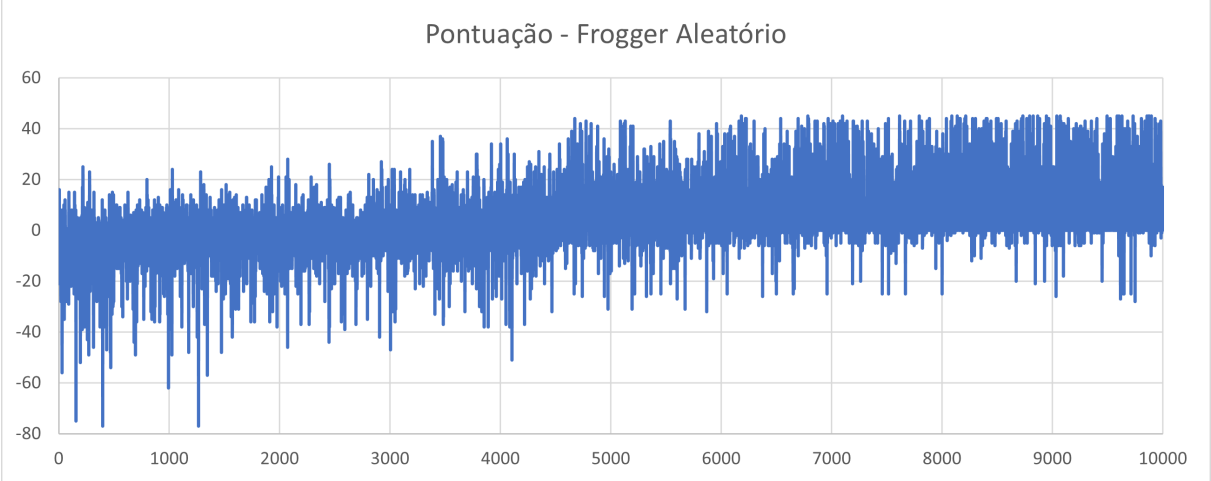

Figura 6 – Pontuação por época

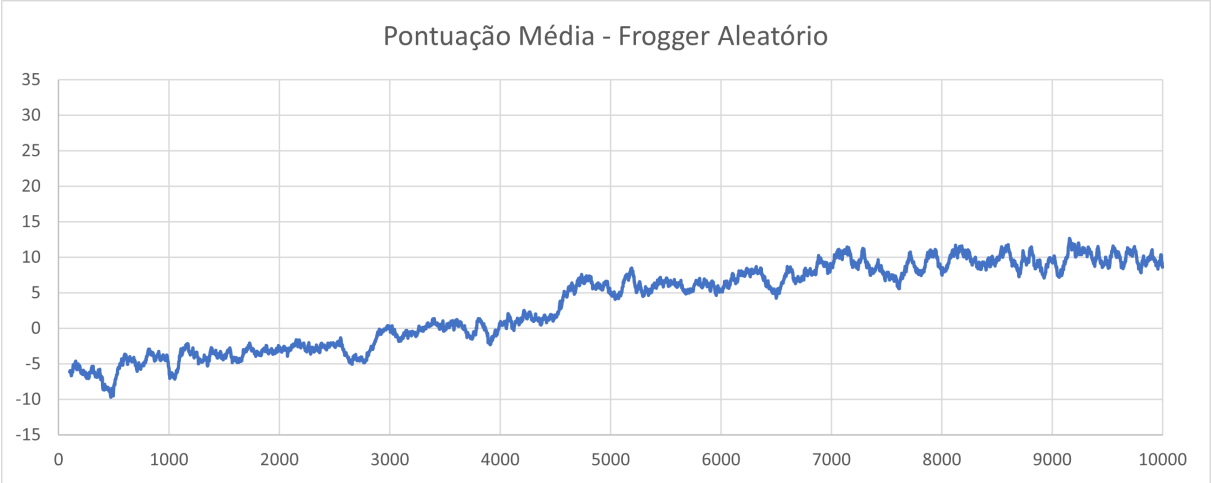

Figura 7 – Pontuação média das ultimas 100 épocas

Ainda assim, é uma melhora significativa sobre a abordagem aleatória de -5 para 5 pontos em média. Possivelmente, essa diferença aumentará com o crescimento do número de épocas, podendo ter uma curva parecida com o jogo anterior se houverem épocas suficientes.

4.3 Pong

No jogo Pong os resultados não foram satisfatórios. Na figura 9 pode-se ver como a média da pontuação cai por volta da época 3.000 e não volta a crescer, embora a figura 8 mostre que conseguiu atingir uma pontuação elevada muitas vezes. O problema foi identificado como uma limitação computacional da máquina usada para testes.

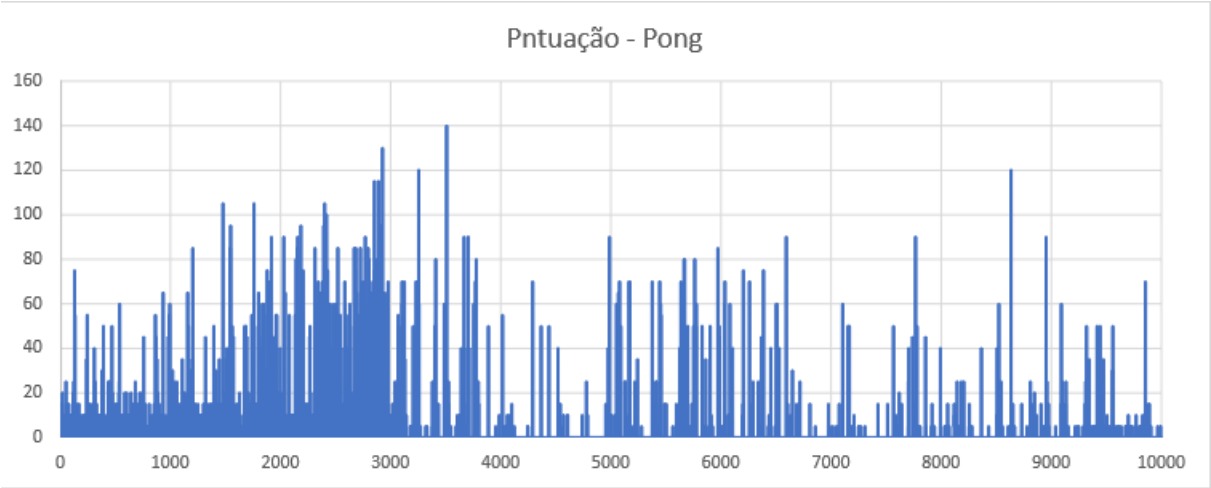

Figura 8 – Pontuação por época

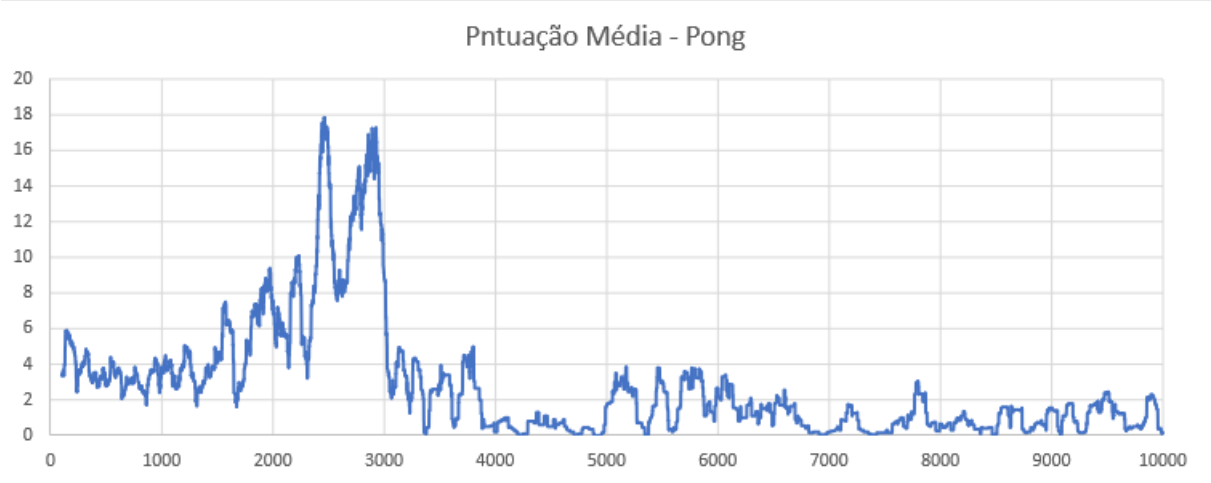

Figura 9 – Pontuação média das ultimas 100 épocas

No inicio do jogo, a bola se direciona à parte superior da tela e, quando rebatida, volta direcionada à parte inferior. O agente deve mover a barra para cima logo e, em seguida,

para baixo a fim de rebater as bolas. Devido às configurações da máquina não serem ideais a esse experimento, o jogo, muitas vezes, demora para abrir e alguns comandos do agente são perdidos. Isso faz com que a barra não suba o suficiente e o jogo termine com pontuação nula. Quando o agente consegue fazer a primeira rebatida, normalmente, ele consegue atingir altas pontuações; porém, como a máquina tem que ficar muitas horas ligada, o atraso na abertura do jogo é cada vez mais constante, o que deixa a média muito baixa.

5 Conclusão

Neste trabalho, foi apresentada uma ferramenta para auxilio em projetos e pesquisas em inteligência artificial em jogos. Apesar de algumas limitações, como o sistema operacional a ser utilizado e a necessidade da biblioteca Allegro, constituiu-se um exemplo e um incentivo às pesquisas na área.

A ferramenta mostrou-se eficaz dentro de suas restrições, embora o maior problema sejam as configurações da maquina onde o trabalho é operado, que é o maior limitante para o uso dos aprendizados de máquina.

Como trabalhos futuros, pretende-se expandir a ferramenta desenvolvida para outras plataformas, além do Windows, e desenvolver formas de extrair a recompensa do jogo da própria tela, diminuindo, assim, a interferência no código do jogo e ampliando o uso dela para um número maior de jogos.

Referências

S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

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

H. Küttler, N. Nardelli, A. H. Miller, R. Raileanu, M. Selvatici, E. Grefenstette, and T. Rocktäschel. The nethack learning environment, 2020.

M. Leverton. Site allegro. `https://www.allegro.cc/about`, acesso em 2019.

D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas. General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms. *CoRR*, abs/1802.10363, 2018.

S. J. Russell. Rationality and intelligence. *Artificial Intelligence - Elsevier Science B.V.*, (94):57–77, 1997.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2015.

A. Thomas. Reinforcement learning tutorial with tensorflow. `https://adventuresinmachinelearning.com/reinforcement-learning-tensorflow/`, acesso em 2021a.

A. Thomas. Reinforcement learning tutorial with tensorflow. `https://https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow//`, acesso em 2021b.

O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

G. N. Yannakakis and J. Togelius. A panorama of artificial and computational intelligence in games. *IEEE Transacions on Computational Inteligence and AI in Games*, 7(4): 317–335, 2015.

G. N. Yannakakis and J. Togelius. *Artificial Intelligence and Games*. Springer, 2018. `http://gameaibook.org`.