OBSERVANDO O COMPORTAMENTO DO ALGORITMO PROXIMAL POLICY OPTIMIZATION NO AMBIENTE DE FLAPPY BIRD

OBSERVING THE BEHAVIOR OF THE PROXIMAL POLICY OPTIMIZATION ALGORITHM IN THE FLAPPY BIRD ENVIRONMENT

Sérgio Hiroyuki Inoue Furuta [1]
Mário Henrique Akihiko da Costa Adaniya [2]

RESUMO

Os video games se tornaram grande parte da cultura popular no mundo desde sua popularização, assim como os avanços em Inteligência Artificial. Neste trabalho, exploramos a utilização do algoritmo Proximal Policy Optimization (PPO) em um jogo. Avaliamos como o algoritmo se comporta e seu seus resultados aplicados no jogo Flappy Bird. Executamos diversos experimentos e analisamos a pontuação no jogo como métrica de desempenho. Com esses resultados, foi possível estabelecer um entendimento melhor sobre o algoritmo e como ele se comporta no ambiente de Flappy Bird.

Palavras-chave: video games; inteligência artificial; algoritmos.

ABSTRACT

Video games have become a significant part of popular culture worldwide since their popularization, along with advancements in Artificial Intelligence. In this work, we explore the use of the Proximal Policy Optimization (PPO) algorithm in a game. We evaluate how the algorithm behaves and its outcomes when applied to the game Flappy Bird. We conducted several experiments and analyzed the game score as a performance metric. With these results, it was possible to gain a better understanding of the algorithm and how it behaves in the Flappy Bird environment.

Keywords: video games; artificial intelligence; algorithm.

[1] Discente do curso de Engenharia de Software Centro Universitário Filadélfia de Londrina - UniFil
[2] Docente do curso de Engenharia de Software Centro Universitário Filadélfia de Londrina - UniFil

1 INTRODUÇÃO

Um dos primeiros videogames a surgir foi o jogo Tennis for Two, lançado em 1958, por William Higinbotham, baseado em computadores militares previamente utilizados para calcular a trajetória de mísseis, modificado para calcular a trajetória da bola de tênis no lugar de mísseis de acordo com (Brookhaven National Laboratory, 2023). Desde então jogos vem se tornando cada vez mais populares e conhecidos, por exemplo, no Brasil, atraindo um público superior a 150 milhões de pessoas, mas esses dados podem se expandir para o resto do mundo, pois o Brasil é 13º no ranking de maior mercado no setor, segundo dados divulgados no site do governo em 2023 (Cultura, 2023).

Jogos de plataforma foram um dos primeiros gêneros a serem desenvolvidos. Em meados dos anos 80 é lançado o que é considerado o primeiro jogo do gênero, Space Panic no arcade desenvolvido pela Universal. No entanto, o ambiente ainda era estático, sem mudanças no cenário. Mas em 1985 o primeiro jogo de plataforma com movimentos de deslize surgiu, sendo o clássico Super Mario World do game boy, o jogo se popularizou tanto na época que o personagem principal Mario tornou-se o mascote da empresa Nintendo (Funk, 2005; Persson, 2005; Millington, 2019; Minkkinen, 2016).

O Flappy Bird é um jogo mobile lançado em maio de 2013, que rapidamente conquistou grande popularidade, mas também recebeu muitas críticas por conta de seu nível de dificuldade. Em 2014, o criador Dong Nguyen decidiu removê-lo das lojas de aplicativos devido ao seu caráter extremamente viciante. Apesar de parecer simples, o jogo exige que o jogador controle um pássaro através de aberturas entre canos dispostos regularmente, realizando apenas duas ações: deixar o pássaro descer ou fazê-lo saltar pressionando a tecla "para cima" (Wei, 2023).

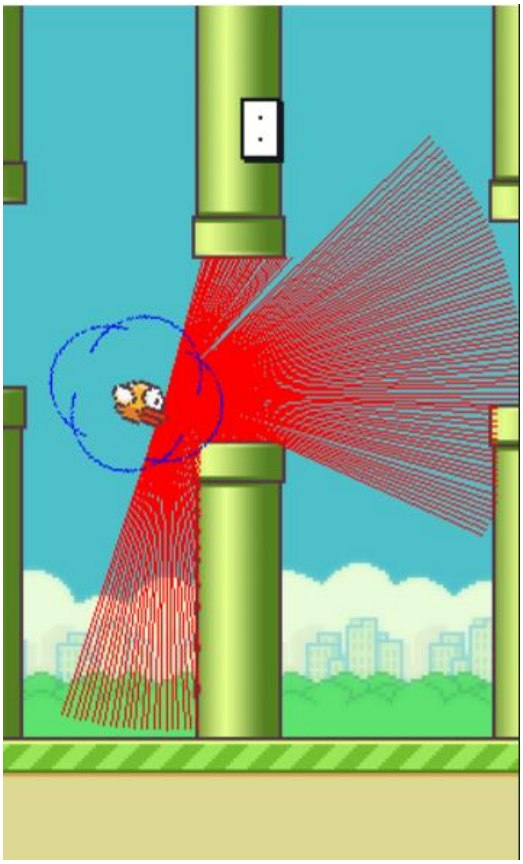
Embora as mecânicas do jogo sejam básicas, o desafio é elevado pela dinâmica rápida, pela alta variabilidade dos cenários e pelo vasto espaço de busca. Isso faz do Flappy Bird um ambiente interessante para experimentação com algoritmos, como os de aprendizagem por reforço. Para desenvolver um programa capaz de jogar com eficiência, é necessário considerar as características do problema, uma vez que os obstáculos são gerados de forma aleatória, criando variações no

ambiente que exigem adaptação do algoritmo.

O Flappy Bird é um jogo que oferece muitas oportunidades para testar algoritmos de aprendizagem por reforço. Embora seja um jogo simples, o ambiente não é fixo, fazendo com que o algoritmo passe por um ambiente semelhante, mas não idêntico ao anterior, já que os obstáculos são criados de maneira aleatória, de acordo com Wei (2023). O objetivo do jogo é levar o pássaro o mais longe possível, passando por entre dois canos que possuem um vão entre eles, e existe uma variação de altura deste vão. O jogo pode ser observado na Figura 1 (Brandão; Pires; Georgieva, 2019).

No entanto, existem diversos gêneros de jogos, como jogos de plataforma, FPS (First Person Shooter), puzzles, mundo aberto e RPG (Role Playing Game). Neste artigo, o foco foi nos jogos de plataforma, visando explorar o comportamento do algoritmo de Inteligência Artificial (IA) em jogos (Funk, 2005; Persson, 2005; Millington, 2019).

Figura 1 – Imagem do jogo Flappy Bird.



Fonte: Captura de tela dos autores.

A IA, de forma simplificada, busca encontrar soluções para problemas que são

considerados necessários e inteligentes para serem executados, como, por exemplo, jogar jogos de videogame. No entanto, não se existe um consenso primeiro no que seria inteligência, nem em como um computador pode ser considerado inteligente. Na década de 50, Alan Turing propôs o que seria conhecido como o teste de Turing, onde um computador teria a capacidade de se passar por um ser humano, mas não é um consenso a respeito do assunto segundo (Russell; Norvig, 2016). Além de que, na época nem era chamado dessa forma ainda, pois somente em 1956, durante um workshop na universidade de Dartmouth, que o termo IA foi utilizado pela primeira vez (Moor, 2006).

A IA possui muitas áreas, como aprendizado de máquina, redes neurais, redes neurais profundas, processamento de linguagem natural, entre outras. Aprendizagem de máquina é a área focada no aspecto de aprendizagem, onde na programação clássica é necessária uma codificação minuciosa do passo a passo para se chegar no resultado desejado, com aprendizagem de máquina é possível se aproximar do resultado por ter uma série de entradas e saídas esperadas, fazendo a IA gerar o algoritmo aproximado do resultado esperado (Mahesh, 2020; Dongare *et al*., 2012; Ernst; Louette, 2024; Kaiser *et al*., 2019).

Uma das técnicas mais proeminentes na aprendizagem de máquina são as redes neurais. Redes neurais são modelos que tentam simular a maneira com que os neurônios funcionam, recebendo uma entrada, processando a informação e gerando uma saída. Para chegar a resultados satisfatórios, são testadas diversas possibilidades de entrada, processamento e saída (Dongare *et al*., 2012; Pouyanfar *et al*., 2018).

Redes neurais são eficazes em tarefas que envolvem padrões complexos ou relações não lineares, como reconhecimento de imagens, processamento de linguagem natural e previsão de séries temporais. Elas conseguem modelar e aprender essas relações a partir de grandes volumes de dados, ajustando-se continuamente para melhorar a precisão dos resultados (Uhrig, 1995).

Por outro lado, o aprendizado por reforço é projetado para resolver problemas de tomada de decisão no qual o agente deve explorar e aprender interagindo com o ambiente. Ele é útil em situações em que não há dados rotulados explícitos, mas onde o desempenho pode ser avaliado por recompensas e penalidades. Exemplos incluem

controle robótico, otimização de tráfego, jogos complexos e sistemas autônomos, onde o objetivo é aprender uma política que maximize as recompensas acumuladas ao longo do tempo.

Neste artigo, devido às características do projeto, optou-se por uma abordagem utilizando redes neurais e aprendizado por reforço, onde é necessário explorar o ambiente, sem ter uma resposta exata, mas uma aproximação do resultado desejado (Janiesch; Zschech; Heinrich, 2021; Mahesh, 2020).

Existem diversos estudos explorando o potencial de aprendizagem por reforço, pois a abordagem visa encontrar soluções para problemas que não se tem um mapeamento entre uma entrada e uma saída, como no aprendizado supervisionado. Abrindo assim a possibilidade de explorar-se ambientes dinâmicos, podendo determinar como se encontra o ambiente ao seu redor e por uma série de tentativas e erros aprender o que deve ser feito, ou se aproximar do objetivo. Para incentivar um comportamento e reprovar outros, uma série de pontos é dado ao algoritmo, nesse sentido o algoritmo visa maximizar sua pontuação ao longo do tempo, sendo bastante explorada para jogar videogames (Ernst; Louette, 2024; Kaiser *et al*., 2019; Li, 2023; Kaelbling; Littman; Moore, 1996).

O Proxy Policy Optimization (PPO), é um algoritmo de aprendizagem por reforço onde as suas decisões são baseados em uma política de decisão, para atualizar sua política de decisão é utilizado um algoritmo de gradiente ascendente. O PPO se diferencia por ter uma forma diferente de modificar no gradiente, similar a um array que auxilia na tomada de decisões, utilizando do que é chamado de *clipping* onde limita o quanto o gradiente pode ser alterado, pois em alguns casos a alteração geraria muitas oscilações nas decisões do modelo, tornando o treinamento instável. Com o *clipping* evitam-se alterações drásticas no gradiente (Engstrom *et al*., 2019; Holubar; Wiering, 2020). O PPO foi utilizado para jogos de puzzle, onde precisava passar por fases que apresentava conhecimento anterior e também como se comportava em novos ambientes (Kristensen; Burelli, 2020).

2 FUNDAMENTAÇÃO TEORICA

A IA em jogos vêm sendo explorados há algum tempo e diversos trabalhos foram desenvolvidos a respeito do tema, como no livro Artificial Intelligence for Games, Second Edition de (Millington, 2019). Onde o autor passa alguns conceitos a respeito de IA, aprendizagem por reforço, jogos, e IA aplicada em jogos.

IA em jogos eram executados somente por meio estados e ações, mas era limitado por precisar calcular uma abundância de possibilidades a ser tomada com base no estado atual do jogo segundo (Risi; Preuss, 2020). Como no jogo de xadrez, onde antes da inteligência artificial deep blue, desenvolvida pela Google, os algoritmos tinham de avaliar todas as ações possíveis para tomar a melhor decisão, agora o algoritmo descobre quais ações tendem a retornar a maior recompensa, por meio de uma série de tentativas e erros, assim adquirindo conhecimento do jogo. (Campbell; Jr; Hsu, 2002) (Schaeffer *et al*., 2007).

Segundo Holubar e Wiering (2020), os jogos oferecem uma plataforma para treinar IA por meio de modelos simplificados da realidade. Neles, é possível explorar o comportamento da IA em situações específicas, como em jogos de corrida, onde ela interage com o ambiente e responde a desafios (Holubar; Wiering, 2020).

Algoritmos de aprendizagem por reforço

Aprendizagem por reforço é uma técnica popular de aprendizagem por reforço aplicada em jogos, ao apresentar o aspecto de ao realizar uma ação recebe recompensas baseadas nelas. Tornando-se possível com que o algoritmo aprenda sem ter uma saída esperada para comparar os resultados, e que analise ambientes complexos por meio da tentativa e erro (Ernst; Louette, 2024).

Alguns autores estão explorando a possibilidade de utilizar técnicas de aprendizagem de máquina para investir, onde o algoritmo aprende os padrões que o mercado tende a realizar para automatizar ações de compra e venda. No entanto, ainda há diversos problemas como o dilema de exploração e explotação para encontrar os melhores investimentos, também ruídos que podem surgir por meio de operações dificultando a previsão (Liu *et al*., 2020).

Aprendizagem por reforço também vem sendo explorado no ambiente de indústria com o problema de escalonar as funções a serem executadas em uma fábrica. Como o quanto deve ser produzido, qual material produzir, e quando produzir, qual o melhor momento para vender (Hubbs *et al*., 2020).

Proximal Policy Optimization (PPO)

O PPO baseia suas decisões em sua política de decisão, é utilizado um algoritmo de gradiente ascendente. O PPO se diferencia por ter uma forma diferente de mexer no gradiente, utilizando do que é chamado de *clipping* onde limita o quanto o gradiente pode ser alterado, pois em alguns casos a alteração geraria muitas oscilações nas decisões do modelo, tornando o treinamento instável, com o *clipping* evitam-se drásticas no gradiente. No treinamento ele não executa uma rodada de treino e elimina as informações, ele consegue usar essa informação por múltiplas rodadas para melhor treinar o modelo. Também apresenta uma implementação flexível para uma variedade de ambientes, sendo usado em jogos e em robótica (Engstrom *et al*., 2019; Holubar; Wiering, 2020).

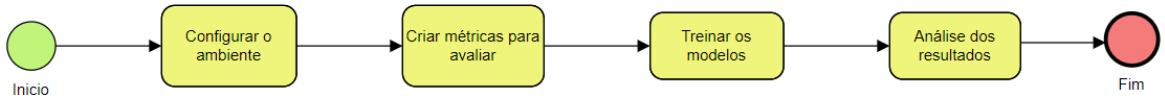
O PPO vem sendo estudado em ambientes com múltiplos agentes aprendendo, onde não se baseia somente naquilo que pode realizar, mas também no que os demais agentes podem estar fazendo. O estudo também aponta a importância do tuning para o funcionamento do PPO em tal ambiente e aponta também que se difere dos *tuning* que fazem ele performar melhor em ambientes com um único agente. Destacando que sem a necessidade de alterar a arquitetura do algoritmo ou a base dele, pode ter excelentes atuações em um novo contexto apenas com ajustes de seus parâmetros (Yu *et al*., 2022).

Outros autores apontam haver uma grande diferença de desempenho no PPO utilizando algumas otimizações no código, como limite para o clipping para modificar o quanto o gradiente é alterado, usar um clipping global e não para uma rodada, função de ativação, regularização e estabilização. Essas alterações variam conforme o problema, podendo apresentar resultados significativos no desempenho do agente (Engstrom *et al*., 2020).

3 METODOLOGIA

O objetivo deste trabalho é entende o comportamento do algoritmo de IA para aprender a jogar Flappy Bird, para auxiliar futuros trabalhos relacionados à inteligência artificial em jogos. Assim, o artigo visa aprender a respeito do comportamento do algoritmo PPO, devido à utilização em modelos comumente utilizados para jogos. Na Figura 2 abaixo tem-se o workflow do projeto a ser executado.

Figura 2 – Workflow da metodologia.



Fonte: Os autores.

Configurar o Ambiente

• foi utilizado de uma biblioteca em Python que permite a criação de instâncias do jogo.

• foi feito a integração dos algoritmos de IA com a biblioteca para facilitar a execução dos comandos no jogo.

Criar Métricas Para Avaliação

• Pontuação final no jogo: avaliada ao final do período de treinamento.

Treinar os Modelos

• Foi implementado o algoritmo PPO no ambiente de jogo.

• As execuções dos treinamentos serão em condições controladas.

• Foi registrado os dados de desempenho durante o treinamento.

Análise dos Resultados

• Foi comparado o tempo de aprendizagem do algoritmo.

• Foi avaliado a pontuação final e do tempo para concluir as fases.

• Foi feita a análise dos dados em relação ao tempo de treinamento para determinar a eficiência do algoritmo.

4 DESENVOLVIMENTO

O modelo PPO foi aplicado utilizando a biblioteca stable-baselines[3], e o jogo Flappy Bird com a biblioteca flappy-b[4]ird-gymnasium, assim foi realizado a integração de ambas, para analisar o comportamento do algoritmo no jogo. Por serem bibliotecas compatíveis, sua integração é simples, facilitando o processo de construção para o treinamento do modelo.

O ambiente no qual o modelo foi treinado é um ambiente dinâmico em que ele é alterado toda vez que o algoritmo é posto para jogar uma nova rodada. Uma variação que não foi realizada foi o treinamento em um ambiente estático no qual entre rodadas não há mudança no ambiente. No processo de treinamento, cada vez que os resultados são salvos para atualizar a política do algoritmo, assim é possível analisar o processo de aprendizagem. Para saber em que ponto o algoritmo apresentou o melhor desempenho após seu treinamento, os agentes foram postos a teste, jogando 100 rodadas cada para visualizar a capacidade de aprender, analisando qual foi a maior pontuação obtida e qual foi a média das rodadas.

Com as informações obtidas no treinamento e os resultados obtidos no final do treinamento, foi feita uma análise das informações, para melhorar a visualização foram gerados gráficos para entender melhor como funciona o aprendizado do PPO sem nenhuma alteração. Para analisar o comportamento do agente foi rodado também utilizando a técnica de rodar múltiplas instâncias do jogo simultaneamente, assim com os resultados obtidos foi gerado um gráfico. Comparando a diferença entre o treinamento com várias simulações simultâneas e com apenas uma de cada vez.

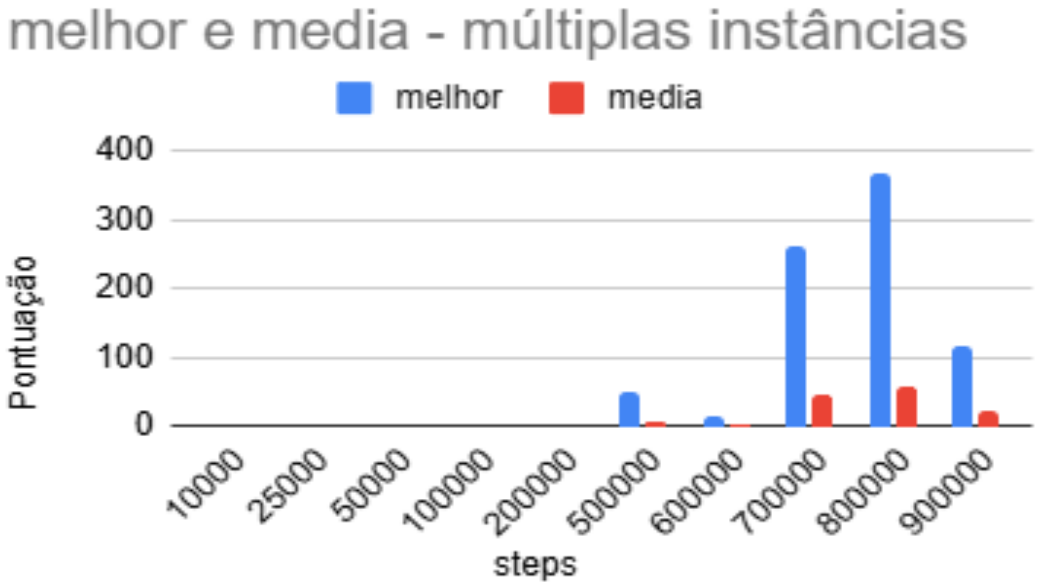
[3] https://github.com/hill-a/stable-baselines
[4] https://github.com/markub3327/flappy-bird-gymnasium.

Figura 3 – Gráfico de barras com desempenho do algoritmo com uma única instância.

Fonte: Os autores.

Com base no gráfico indicado na Figura 3 é possível notar que o comportamento do modelo, com apenas uma simulação por vez, apesar de estar melhorando, não é uma curva de aprendizagem constante, onde ele apresenta uma melhora, mas, depois, decai, para poder melhorar novamente seus resultados. Esse comportamento pode ser considerado o dilema da exploração e explotação, que consiste em explorar mais do ambiente ou aproveitar do que já é conhecido.

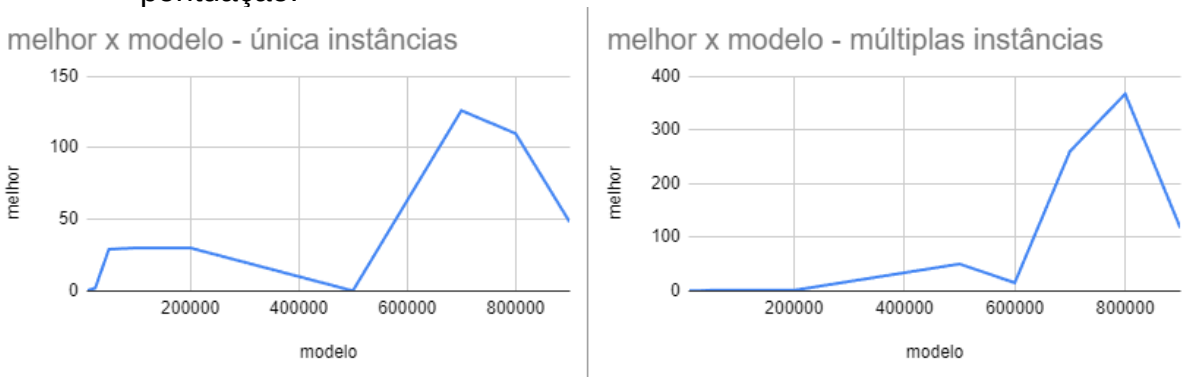
Figura 4 – Gráfico de barras com desempenho do algoritmo com múltiplas instâncias.



Fonte: Os autores.

No entanto, no caso das múltiplas instâncias, o treinamento apresentou um comportamento um pouco mais uniforme, como indicado na Figura 4, onde a única instabilidade foi ao final do treinamento, no último modelo. No entanto, como pode ser observado, o gráfico o começo foi pior que com uma única instância, no entanto, ao final os resultados foram melhores. O que pode indicar que com múltiplas instâncias atualizando a sua política simultaneamente, o PPO apresenta um aprendizado mais uniforme.

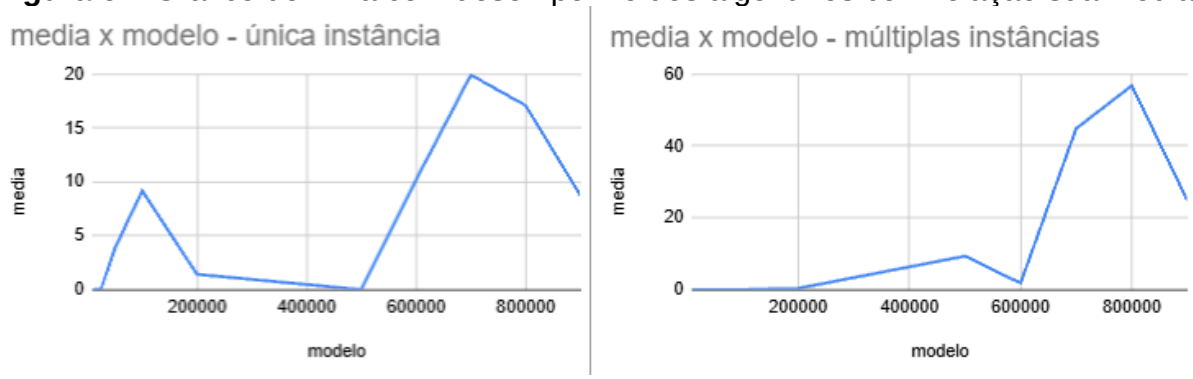
Figura 5 – Gráfico de linha com desempenho dos algoritmos com relação sua melhor pontuação.



Fonte: Os autores.

Os gráficos ilustrados na Figura 5, mostram que os resultados no início do treinamento com uma única instância foram melhores, no entanto, existe uma certa aleatoriedade para as decisões no início podendo não necessariamente indicar que é melhor. No entanto, ao final do treinamento o modelo treinado com múltiplas instâncias teve melhores resultados com o melhor resultado sendo próximo de 400 no modelo com 800.000 *steps*. O que é interessante notar é que ambos tiveram uma queda no modelo com 900.000, podendo indicar que estaria explorando mais do ambiente para aprender como maximizar sua recompensa. O mesmo pode ser observado quando considerado a média dos modelos, onde com uma única instância conseguiu alguns resultados melhores no começo, mas ao final as maiores médias foram dos modelos com múltiplas instâncias, tendo chagado perto de médias de 60 com 800.000 *steps*, como indicado na Figura 6.

Figura 6 – Gráfico de linha com desempenho dos algoritmos com relação sua média.



Fonte: Os autores

Para uma análise minuciosa dos dados, é possível observar nas tabelas qual foi a média exata e a melhor pontuação de cada modelo, como indicado nas Tabelas 1 e 2.

Tabela 1 – Resultados de Desempenho do Modelo com Uma Instância.

Modelo	Melhor	Média
10000	0	0.00
25000	2	0.07
50000	29	3.83
100000	30	9.16
200000	30	1.42
500000	0	0.00
600000	63	10.19
700000	126	19.91
800000	110	17.07
900000	48	8.74

Fonte: Os atores.

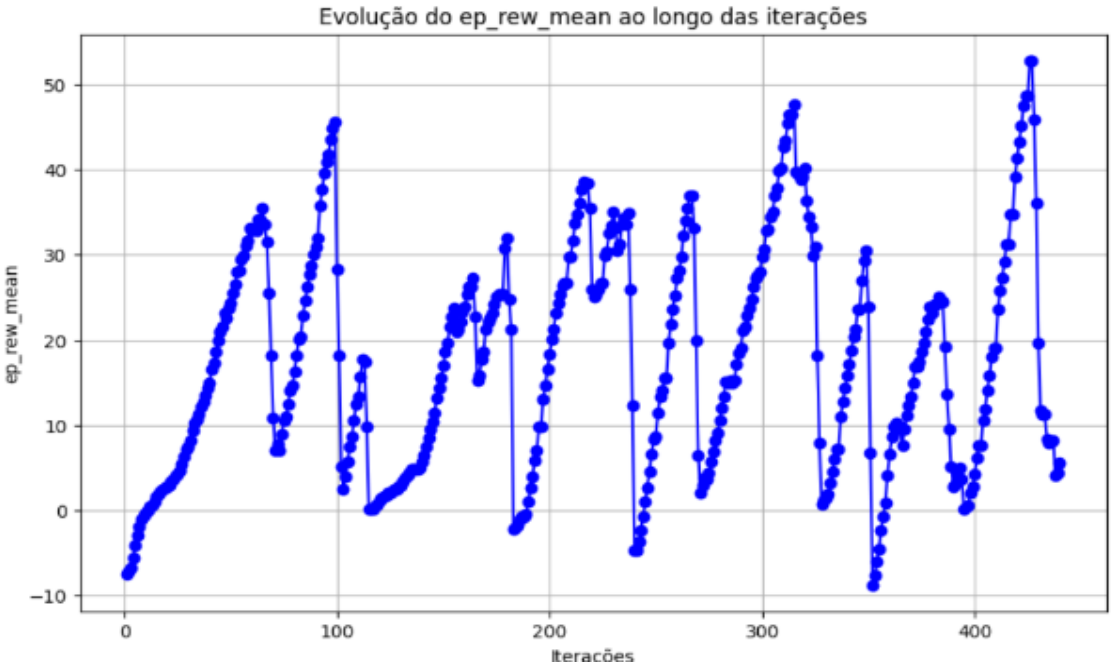
Tabela 2 – Resultados de Desempenho do Modelo com Múltiplas Instâncias.

Modelo	Melhor	Média
10000	0	0.00
25000	0	0.00
50000	1	0.01
100000	1	0.02
200000	1	0.35
500000	50	9.31
600000	15	1.82
700000	260	44.80
800000	367	56.74
900000	117	24.81

Fonte: Os autores.

O gráfico representado na Figura 7 indica que o modelo em seu treinamento melhora até atingir um limite e então decai em rendimento, possivelmente para explorar mais o ambiente. Como pode ser observado, o número de *steps* não tem relação com o número de iterações realizadas, pois esse gráfico representa o modelo com uma execução do algoritmo com 900.000 *steps*, mas apresenta aproximadamente 450 iterações. Como demonstrado pelos resultados, a política final não utiliza o ponto em que foi melhor, por não ter um "histórico", mas é atualizada constantemente e no momento em que é salvo é como se comporta ao final do treinamento, podendo não ser o melhor.

Figura 7 – Gráfico de linha com desempenho dos algoritmos no treinamento.



Fonte: Os autores.

5 CONCLUSÃO

Nesse artigo foi abordado a respeito de jogos, IA, algoritmos de aprendizagem por reforço, redes neurais e redes neurais profundas, no entanto, o foco maior do artigo foi a respeito do algoritmo de aprendizagem por reforço PPO, aplicado ao jogo Flappy Bird. Fori analisado a pontuação utilizando multiprocessamento e uma única

instância, da pontuação foram extraídos a média e o melhor resultados, com base no número de *steps*, começando dos 10.000 até 900.000.

Os resultados com multiprocessamento foram melhores, apesar de serem piores no começo, considerando que a meta do algoritmo é maximizar a sua recompensa ao longo do tempo, e os resultados ao final foram melhores. No entanto, não foi considerado os gastos computacionais para o treinamento do modelo, e se as múltiplas instâncias teriam um gasto superior ao final, mesmo considerando o tempo superior de treinamento do modelo com apenas uma execução. Sendo assim, com base nas métricas estabelecidas e os resultados obtidos, é possível concluir que o PPO utilizando o método de treinamento com múltiplas instâncias é melhor que com apenas uma instância.

Futuramente é interessante realizar uma análise com mais indicadores de monitoramento como processamento de hardware necessário para o treinamento e a energia gasta em cada modelo. Ainda, para o modelo existem diversos hiper-parâmetros que não foram explorados nesse artigo, mas que poderiam impactar no comportamento do modelo de forma tanto positiva quanto negativa. Assim, seria interessante comparar como essas mudanças impactaram o modelo ao final do treinamento. Também seria interessante testar com um ambiente estático para o treinamento, para saber se ele aprende melhor em um ambiente estático, para depois ir para um ambiente dinâmico, ou se é melhor treinar em uma ambiente dinâmico desde o início.

REFERÊNCIAS

BRANDÃO, A.; PIRES, P.; GEORGIEVA, P. Reinforcement learning and neuroevolution in flappy bird game. *In*: MORAES, A. et al. (Ed.) **Pattern Recognition and Image Analysis**. Switzerland: Springer, 2019. p. 225–236.

BROOKHAVEN NATIONAL LABORATORY. **The First Video Game**. 2023. Disponível em: https://www.bnl.gov/about/history/firstvideo.php. Acesso em: 5 mar. 2025.

CAMPBELL, M.; JR, A. J. H.; HSU, F.-h. Deep blue. **Artificial intelligence**, v. 134, n. 1-2, p. 57–83, 2002.

CULTURA, M. da. **Setores do micbr**: Jogos eletrônicos atraem mais de 150 milhões de brasileiros. Portal do Governo Brasileiro, 2023. Disponível em: https://www.gov.br/cultura/pt-br/assuntos/noticias/setores-do-micbr-jogos-eletronicos-atraem-mais-de-150-milhoes-de-brasileiros. Acesso em: 5 mar. 2025.

DONGARE, A. *et al*. Introduction to artificial neural network. **International Journal of Engineering and Innovative Technology (IJEIT),** Citeseer, v. 2, n. 1, p. 189–194, 2012.

ENGSTROM, L. *et al*. Implementation matters in deep rl: A case study on ppo and trpo. *In*: International conference on learning representations. [S.l.: s.n.], 2019

ENGSTROM, L. *et al*. Implementation matters in deep policy gradients: A case study on ppo and trpo. **arXiv preprint,** arXiv:2005.12729, 2020.

ERNST, D.; LOUETTE, A. **Introduction to reinforcement learning**. 2024.

HOLUBAR, M. S.; WIERING, M. A. Continuous-action reinforcement learning for playing racing games: Comparing SPG to PPO. **arXiv,** arXiv:2001.05270, 2020. Disponível em: https://arxiv.org/abs/2001.05270. Acesso em: 5 mar. 2025.

HUBBS, C. D. *et al*. A deep reinforcement learning approach for chemical production scheduling. **Computers & Chemical Engineering**, v. 141, p. 106982, 2020.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, v. 31, n. 3, p. 685–695, 2021.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, v. 4, p. 237–285, 1996.

KAISER, L. *et al*. Model-based reinforcement learning for atari. **arXiv preprint,** arXiv:1903.00374, 2019.

KRISTENSEN, J. T.; BURELLI, P. Strategies for using proximal policy optimization in mobile puzzle games. *In*: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020. **Proceedings** [...]. [*S.l.*]: ACM, 2020. p. 1–10.

LI, S. E. Deep reinforcement learning. *In*: LI, S. E. **Reinforcement learning for sequential decision and optimal control**. [*S.l.*]: Springer, 2023. p. 365–402.

LIU, Y. *et al*. Adaptive quantitative trading: An imitative deep reinforcement learning approach. **Proceedings of the AAAI conference on artificial intelligence**, v. 34, n. 02, p. 2128–2135, 2020.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR)**, v. 9, n. 1, p. 381–386, 2020.

MILLINGTON, I. **AI for Games**. [S.l.]: CRC Press, 2019.

MOOR, J. The dartmouth college artificial intelligence conference: The next fifty years. **Ai Magazine**, v. 27, n. 4, p. 87–87, 2006.

POUYANFAR, S. *et al*. A survey on deep learning: Algorithms, techniques, and applications. **ACM Computing Surveys** (CSUR), ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

RISI, S.; PREUSS, M. From chess and atari to starcraft and beyond: How game ai is driving the world of ai. **KI-Künstliche Intelligenz**, v. 34, n. 1, p. 7–17, 2020.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence**: a modern approach. [S.l.]: Pearson, 2016.

SCHAEFFER, J. *et al*. Checkers is solved. **Science, American Association for the Advancement of Science**, v. 317, n. 5844, p. 1518–1522, 2007.

UHRIG, R. Introduction to artificial neural networks. *In*: IECON ’95 - ANNUAL CONFERENCE ON IEEE INDUSTRIAL ELECTRONICS, 21., 1995, Orlando. **Proceedings** [...] Orlando: IEEE, 1995. v. 1, p. 33–37. vol.1.

WEI, S. Reinforcement learning for improving flappy bird game. **Highlights in Science, Engineering and Technology**, v. 34, p. 244–249, 02 2023.

YU, C. *et al*. The surprising effectiveness of ppo in cooperative multi-agent games. **Advances in Neural Information Processing Systems**, v. 35, p. 24611–24624, 2022.