Comparação e Otimização de Algoritmos de Aprendizagem por Reforço Profundo aplicados ao simulador do jogo Enduro do Atari 2600

Felipe Juvêncio Mendes, Gustavo Henrique de Moraes Pessa, João Leonardo Andrade Morganti Silva & Marcelo Vinícius Cysneiros Aragão

***Abstract*— The growing use of artificial intelligence in contexts ranging from industrial environments to domestic applications has brought economic growth and improvements in people's quality of life, in addition to boosting research and development initiatives. In this sense, the present work is situated in the context of machine learning, performing the analysis, execution, optimization, and comparison of learning algorithms by reinforcing the DQN, PPO A2C algorithms applied to the Enduro game of Atari 2600. The performances are presented. of the algorithms submitted to the game in different scenarios, a discussion is made about the influence of the optimization of hyperparameters of the algorithm that stood out in relation to the others, and, finally, a comment about the feasibility of this process when the sample space is vast.**

***Index Terms*— Deep Reinforcement Learning, DQN, PPO, A2C, General Video Game Artificial Intelligence, Autonomous Racing, OpenAI Gym, Atari Enduro**

***Resumo*— O crescente emprego da inteligência artificial em contextos que vão desde ambientes industriais a aplicações domésticas, vem trazendo crescimento econômico e melhoria na qualidade de vida das pessoas, além de impulsionar iniciativas de pesquisa e desenvolvimento. Nesse sentido, o presente trabalho situa-se no contexto de aprendizagem de máquina, realizando a análise, execução, otimização e comparação dos algoritmos de aprendizado por reforço dos algoritmos DQN, PPO A2C aplicados ao jogo Enduro do Atari 2600. São apresentados os desempenhos dos algoritmos submetidos ao jogo em diferentes cenários, é feita uma discussão sobre a influência da otimização de hiperparâmetros do algoritmo que se sobressaiu em relação aos outros e, finalmente, um comentário a respeito da exequibilidade deste processo quando os espaço amostral é vasto.**

***Palavras Chave*— Aprendizagem por Reforço Profundo, DQN, PPO, A2C, General Video Game Artificial Intelligence, Corrida Autônoma, OpenAI Gym, Atari Enduro**

I. Introdução

Nos últimos quatro anos, o crescente mercado de tecnologia tem voltado sua atenção à crescente relevância da inteligência artificial – comumente chamada de IA – em pesquisas e aplicações. Devido ao aumento de acessibilidade e democratização do acesso a ambientes de alto poder computacional, bem como à utilização de IA em soluções tanto industriais quanto do cotidiano, o Fórum Econômico Mundial estima que o PIB (Produto Interno Bruto), em âmbito global, será 14% maior no ano de 2030, sendo equivalente US$ 15,7 trilhões. [1]

De acordo com o Índice de Nível de Inovação e Crescimento IA (INICIA)[1], o uso de tecnologias de inteligência artificial cresceu de 32% para 48% do ano de 2018 para 2020 na América Latina. Nesta região, o Brasil concentra o maior número de empresas no ramo, correspondendo a 42% delas e obtendo um salto de 120 para 206 empresas de IA neste período de tempo. [1]

Apesar de todo o crescimento apresentado, a inteligência artificial ainda está em fase inicial na América Latina, devido a três principais fatores: [1]

- A maior parte dessas empresas, mais especificamente 55% delas, ainda possuem pouco tempo de mercado, com seu ano de fundação compreendido entre 2014 e 2017;
- Tais empresas possuem investimento inicial médio de grandeza pequena, girando em torno de US$ 528 mil;
- A insistência no uso de técnicas de aprendizagem de máquina tradicionais, em detrimento das mais rebuscadas, como as de aprendizagem profunda.

Outro estudo, dessa vez realizado pela consultoria americana FrontierView, apresentou o crescimento econômico adicional que a adoção massiva de IA até 2030 poderá causar em alguns países da América devido à aceleração da transformação digital que ocorreu durante a pandemia da COVID-19[2]. Na Figura 1 pode-se observar as estimativas de crescimento econômico nos cenários conservador e otimista para cada país. [2]

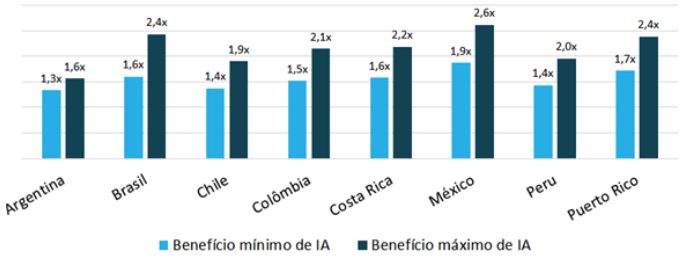


Fig. 1. Gráfico do crescimento econômico adicional em cenário conservador e otimista caso ocorra adoção massiva de IA até 2030 para diferentes países (baseado em [2]).

Com potencial para aumentar a produtividade dos mais variados setores da economia e influenciar diretamente no cres-

Trabalho de Conclusão de Curso apresentado ao Instituto Nacional de Telecomunicações, como parte dos requisitos para a obtenção do Grau de Bacharel em Engenharia de Computação. Orientador: Prof. Me. Marcelo Vinícius Cysneiros Aragão.

[1]Metodologia desenvolvida pela empresa Everis em parceria com a Endeavor Intelligence, que visa medir o nível de inovação e crescimento da IA nas empresas baseando-se em critérios como ano de fundação, grau de maturidade, volumes de vendas, empregos gerados e investimentos recebidos. [1]

[2]Doença infecciosa causada pelo vírus SARS-CoV-2.

cimento do PIB, a IA também pode possibilitar avanços científicos em áreas hoje limitadas, como na descoberta de tratamentos e vacinas para doenças, além de desempenhar tarefas cognitivas, antes prerrogativas apenas da espécie humana, como a condução de veículos. [3] [4] [5]

Portanto, tendo em vista que a inteligência artificial – assim como a eletricidade e a computação – é uma tecnologia capaz de afetar outras, causando grandes impactos na geração de riqueza e nas relações sociais, esta pode ser considerada uma tecnologia de propósito geral. [3]

É justamente por este motivo que empresas e organizações vem fomentando o aprofundamento no estudo e desenvolvimento dos diferentes ramos que a IA abrange para fortalecer a formação de uma comunidade de acesso democrático voltada ao tema, evitando que apenas um grupo de grandes empresas tenha domínio da tecnologia.

Um desses ambientes de fomento são as competições de jogos baseados em IA. Nelas, em vez dos programadores desenvolverem uma inteligência artificial para um conjunto de dados, os programadores desenvolvem agentes para atuar de forma autônoma em simulações de jogos, treinados a partir de técnicas de IA com a finalidade de satisfazer determinados objetivos prescritos pela competição.

O presente trabalho, portanto, apresenta inicialmente uma comparação de diferentes algoritmos para treinamento de um agente autônomo para a simulação do jogo Enduro do Atari 2600 (cuja visualização pode ser observada na Figura 2), seguida da tentativa de aperfeiçoamento do modelo que apresentar os melhores resultados.

Fig. 2. Captura de tela do simulador do jogo Enduro do *console* Atari 2600.

O artigo está estruturado da seguinte forma: na seção II. é feita uma revisão teórica dos temas que embasam o presente estudo; na sequência, a seção III. traz uma revisão bibliográfica, indicando importantes aspectos de outros trabalhos relacionados ao tema; a seguir, a seção IV. apresenta a proposta que caracteriza este estudo, de fato. Posteriormente, na seção V., é possível compreender os procedimentos e metodologias experimentais; por fim, são expostas na seção VI. as conclusões finais acerca do trabalho desenvolvido.

II. Revisão da teoria

O presente trabalho relaciona determinados conceitos que formam a base de sua proposta, sendo eles: Inteligência Artificial, Aprendizagem de Máquina, Aprendizagem por Reforço, Aprendizagem Profunda, Aprendizagem por Reforço Profundo e *General Video Game AI*. Dessa forma, será exposta a seguir uma breve revisão teórica sobre cada um deles.

A. Inteligência Artificial

Antes de definir inteligência artificial, é importante separar e entender o significado das palavras que formam esse conceito, começando com "inteligência". O termo origina-se do latim *intelligentia*, derivado da palavra *inter* (que significa "entre") e *legere* (que significa "escolher"), explicando, portanto, que inteligência é a capacidade de escolher, dado um conjunto de opções, a melhor (em termos de alguma métrica) ou a mais correta delas. Já a palavra "artificial" traz uma sensação de algo não humano, ou mais precisamente, não natural, no sentido de ser produzido pelo homem e não pela natureza. Entretanto, ao produzir o termo concatenado "inteligência artificial", limitar-se a apenas uma definição mostra-se insuficiente, visto que a IA abrange diferentes dimensões, incluindo comportamento e raciocínio, segundo Brewka [6].

A listagem a seguir apresenta diferentes definições para IA dentro de cada dimensão na qual ela pode ser descrita, com definições envolvendo raciocínio e comportamento.

- Sistemas que pensam como humanos:
  - "*O novo trabalho para fazer máquinas pensarem... máquinas com mentes, literalmente.*" [7]
  - "*Automação de atividades relacionadas ao pensamento humano, como tomada de decisão, resolução de problemas e aprendizagem.*" [8]
- Sistemas que pensam racionalmente:
  - "*O estudo das faculdades mentais através do uso de modelos computacionais.*" [9]
  - "*O estudo computacional que torna possível perceber, raciocinar e agir.*" [10]
- Sistemas que agem como humanos:
  - "*A arte de criar máquinas que executam funções que requerem inteligência quando realizadas por pessoas.*" [11]
  - "*O estudo que busca fazer com que computadores façam atividades nas quais humanos são melhores atualmente.*" [12]
- Sistemas que agem racionalmente:
  - "*Um campo de estudo onde buscam explicar e emular comportamentos inteligentes em termos de processos computacionais.*" [13]
  - "*O ramo da ciência da computação que busca automatizar comportamentos inteligentes.*" [14]

A área de estudo de IA é bastante ampla, e costuma ser subdividida em subáreas – como evolucionista, simbólica, conexionista e probabilística –, que fundamentalmente representam diferentes abordagens e paradigmas para resolução de problemas. Neste sentido, a aprendizagem de máquina, que será abordada a seguir, consiste em uma "consequência" da IA, agregando algoritmos e modelos capazes de aprender (por meio de amostras de dados, por exemplo), generalizar e resolver problemas.

B. *Aprendizagem de Máquina*

Os modelos de aprendizagem de máquina são usados para atingir um determinado resultado a partir de uma entrada de dados significantemente grande. No trabalho de Kaliszyk et al [15], por exemplo, foram usadas mais de duas milhões de amostras de dados para testes. São frequentemente empregados no emergente contexto do *Big Data*, onde a partir de uma grande quantidade de dados, a máquina começa a compreender, generalizar e tomar decisões cada vez melhores de acordo com os exemplos que lhe foram apresentados [16]. São algoritmos auto-reflexivos, isto é, capazes de se auto-avaliar e, então, se auto-otimizar de modo a melhorar e alcançar os resultados desejados da forma mais fiel possível. O aprendizado é feito partindo tanto de dados pré-processados como também dentro de um ambiente designado para que a máquina possa adquirir experiências, e então, aprender com elas.

A ideia principal por trás da aprendizagem de máquina é emular a forma como humanos aprendem a processar informações para assim realizar alguma ação ou alcançar algum objetivo [16]. Partindo para a definição, o termo recebeu diferentes definições na literatura no decorrer dos anos. Samuel [17] estabelece que "é um campo de estudos que dá aos computadores a habilidade de aprender sem serem explicitamente programados", enquanto Alpaydin [18] a define como o campo de estudos no qual "programa-se computadores para otimizar o critério de desempenho usando dados de exemplo ou experiências passadas".

Existem três grandes áreas dentro da aprendizagem de máquina, que serão brevemente definidas na listagem a seguir.

- Aprendizagem supervisionada: consiste na produção de uma função que associa as entradas de dados às saídas previamente conhecidas, discernindo amostras cujas saídas (classes) sejam diferentes. Neste caso, é necessário conhecer, *a priori*, as saídas associadas aos dados de entrada (ou seja, as amostras devem ser rotuladas) [19]. Como exemplo, tem-se as redes neurais: modelos que consistem basicamente em uma camada de entrada de dados, uma ou mais camadas neurais intermediárias e uma camada de saída. Cada conexão entre as unidades é ponderada por peso sináptico, que são utilizados para calcular a saída da rede diante de uma entrada de dados. A Figura 3 exibe a arquitetura básica de uma rede neural. [20]

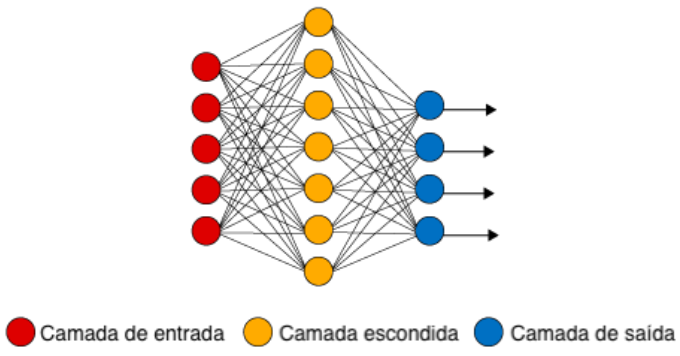


Fig. 3. Arquitetura básica de uma rede neural (baseada em [21]).

- Aprendizagem não supervisionada: metodologia na qual os dados de entrada não estão rotulados, ou seja, não existe um conhecimento prévio da relação da saída com as amostras de entrada; desta forma, o resultado do processamento consiste em gerar agrupamentos ("*clusters*") de dados semelhantes de acordo com alguma medida de similaridade [22]. Dentre os algoritmos famosos desta abordagem, podem-se citar o k-Means, DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) e OPTICS (*Ordering Points To Identify the Clustering Structure*) [23].
- Aprendizagem por reforço: é o caso onde a máquina aprende por conta própria como atuar. Na aprendizagem por reforço, o agente não recebe nenhuma informação sobre como ela deve agir, ele deve descobrir sozinho qual ação deve tomar para receber o valor máximo de recompensa [24]. Esse conceito será discutido de forma mais aprofundada nas subseções seguintes.

C. *Aprendizagem por Reforço*

A aprendizagem por reforço é um tipo de aprendizagem de máquina onde o agente – entidade que interage com o sistema, recebendo informações do ambiente e enviando qual decisão deve ser tomada – é sujeito a um ambiente incerto e complexo, isto é, um local sobre o qual ele ainda não possui nenhuma informação, enfrentando situações nas quais deve-se tomar uma ação e, a partir de tentativas e erros, encontra uma solução para o problema a fim de atingir uma meta preestabelecida. Para que a meta seja atingida, a inteligência artificial recebe recompensas e penalidades a cada ação, estabelecendo assim, seu padrão de confiança para realizar mais ações que maximizem a recompensa total. [25]

As políticas de recompensas e penalidades, associadas às regras do jogo, são definidas previamente; porém, cabe ao modelo estabelecer táticas de ganho de recompensas que iniciam com ações aleatórias mas que, a cada passo de aprendizagem, possibilitam sequências complexas de ações no final do processo de treinamento.

A aprendizagem por reforço é útil para tarefas em que não há métodos adequados para solução do problema e os cenários em que o agente irá atuar são imprevisíveis, deixando à máquina a responsabilidade de encontrar sua própria solução e limitando a intervenção humana apenas às configurações do ambiente e à elaboração da política de recompensas e penalidades. [25]

Diante disso, a aprendizagem por reforço possui três principais desafios de implementação, sendo eles: [25]

- A preparação dos ambientes de simulação interferem drasticamente no treinamento dos modelos. Casos mais simples, como simulação de jogos, podem atingir resultados satisfatórios, mas casos de aplicações no ambiente real (como carros autônomos) podem acarretar consequências sérias; logo, é extremamente importante a construção de um simulador realista para a execução dos treinamentos.
- É extremamente importante ajustar, quando usada no modelo, a rede neural que controla o agente. A única forma de comunicação existente com a rede é por meio do sistema de recompensas. Esse fator pode levar a um esquecimento trágico, isto é, a aquisição de novos conhecimento pode resultar na remoção de alguns dos antigos. Ou seja, é necessário garantir o gerenciamento correto da "memória" na

qual as experiências são armazenadas.

- Por fim, estabelecer um um critério de parada aceitável, mesmo que subótimo, para assim evitar que o agente execute alguma ação exigida de forma incorreta. Ou seja, é importante garantir que o modelo siga os critérios que foram projetados, pois, mesmo com execuções indesejáveis, é possível obter recompensas.

D. *Aprendizagem Profunda*

A aprendizagem profunda é uma das formas de conduzir o processo de aprendizagem de máquina, sendo considerada uma subcategoria desta e apelidada de "aprendizagem de representação de dados". Ela também apresenta as vertentes supervisionada, não-supervisionada e por reforço.

Sendo uma evolução natural das redes neurais, a aprendizagem profunda consiste no emprego de algoritmos que criam modelos de abstrações a partir de um grande volume de amostras (dados) de entrada, que são submetidas a um grafo profundo, com diversas camadas de processamento, com a finalidade de extrair características desde baixo até alto nível. [26]

Viana [26] traz as seguintes definições em sua publicação:

- Camada de entrada: pode ser composta pelos *pixels* de uma imagem ou por dados de uma série temporal (ou seja, uma série valores amostrados periodicamente);
- Camada(s) oculta(s): neurônios responsáveis por aprender características incrementalmente abstratas dos dados usados durante o treinamento;
- Camada de saída: produz uma saída (ou seja, um rótulo) para uma entrada desconhecida que foi apresentada à rede.

Dessa forma, Viana [26] também afirma que "a rede neural consiste em uma função de aproximação que tenta aprender os parâmetros (pesos) em camadas ocultas que, quando multiplicadas pela entrada, fornecem uma saída prevista próxima da saída desejada". Sendo assim, o aprendizado profundo se caracteriza por "concatenar" múltiplas camadas ocultas entre as de entrada e saída, criando uma rede profunda, como mostra a Figura 4.

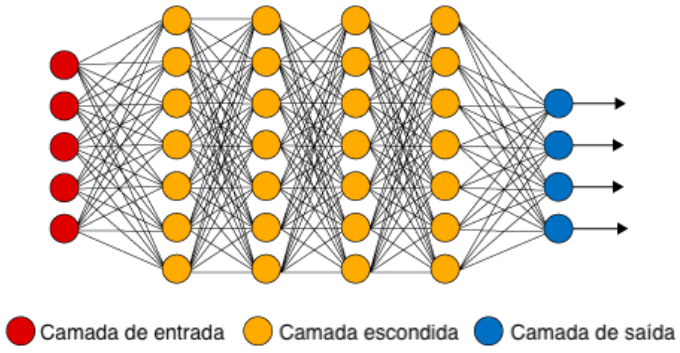


Fig. 4. Estrutura de uma rede neural profunda (baseada em [21]).

A aprendizagem profunda pode ser aplicada a uma variedade de tarefas, e diferentes questões – como suas aplicações de saída, seus dados de entrada e seu tipo de aprendizagem (supervisionada ou não-supervisionada) – interferem na escolha do método apropriado. Dentre os mais comuns, podem ser citadas as Redes Neurais Convolucionais (CNN – *Convolutional Neural Networks*, Redes Neurais Recorrentes (RNN – *Recurrent Neural Networks*, Memória Longa de Curto Prazo (LSTM – *Long Short-Term Memory* [27], *Autoencoders* [28] e Redes Adversárias Generativas (GANs – *Generative Adversarial Networks*) [29].

E. *Aprendizagem por Reforço Profundo*

Os avanços das aplicações que exigem aprendizagem de máquina dependem de treinamentos eficientes de redes neurais profundas em conjunto de treinamentos maiores do que quando usadas redes neurais simples. Diante disso, dá-se o nome de "aprendizagem por reforço profundo" à técnica de aprendizagem de máquina que combina métodos de aprendizagem por reforço e aprendizagem profunda de forma simultânea.

Obtém-se, então, esse tipo de aprendizado a partir do momento em que são utilizadas redes neurais profundas como parâmetros de peso para seus algoritmos. Quando utilizam-se modelos rasos de aprendizagem de máquina, isto é, que não possuem camadas profundas em suas redes neurais, obtém-se uma aprendizagem por reforço rasa devido à utilização dos parâmetros de peso de modelos como funções lineares, árvores de decisão, dentre outros. Difere-se assim esse tipo de aprendizado da aprendizagem por reforço nos pesos considerados em cada modelo. [30]

Nesse estudo são abordados três algoritmos de aprendizado por reforço profundo, sendo eles:

- *Deep Q-Network* (DQN): algoritmo que combina o Q-learning – que busca a maior recompensa dentro do cenário de possíveis ações – com uma rede neural profunda possivelmente convolucional. [31]
- *Proximal Policy Optimization* (PPO): versão modificada do algoritmo *Trust Region Policy Optimization* – modelo que busca atualizar a política, mas sem deixar que ela mude de forma brusca em relação a anterior, e para isso, é introduzida uma restrição – onde, apenas com uma política escolhida, é possível controlar a atualização da lógica e da zona de confiança. [32] Foi proposto em 2017 e mostrou um ótimo desempenho no contexto do OpenAI. [33]
- *Advantage Actor-Critic* (A2C): versão síncrona do *Asynchronous Advantage Actor-Critic*, este que consiste de múltiplos agentes independentes possuindo seus próprios pesos que interagem em paralelo com uma cópia do ambiente. Com isso, por ser síncrono, o A2C espera todos seus agentes finalizarem suas tarefas para atualizar os pesos globais e então, reiniciar todos seus atuadores. [34]

Nesta subseção, foram apresentados os três algoritmos que servirão como base para os experimentos conduzidos neste trabalho. A seguir, será abordado o conceito que faz, justamente, a ligação entre a IA do ponto de vista conceitual, com jogos, do ponto de vista prático.

F. *General Video Game AI*

A *General Video Game AI* (GVGAI) trata-se de um *framework* para *General Video Game Playing* (GVGP)[3] de jogos do tipo *arcade*[4] bidimensionais oferecendo uma interface comum para *bots*[5] – sendo estes agentes ou controladores do

[3] Arquitetura para programas de IA capazes de jogar mais de um jogo.
[4] Gênero de jogo de ação em ritmo acelerado que requer habilidade de coordenação motora para jogar.
[5] Programa autônomo capaz de interagir com sistemas ou usuários.

próprio jogo – e pessoas jogarem em um ou dois participantes os jogos disponíveis. [35]

Esses jogos são descritos pela *Video Game Description Language* (VGDL), linguagem projetada por Schaul [36], e definidos em algumas dezenas de linhas de código simples, numa divisão em quatro seções, que contém, respectivamente segundo a documentação do *framework*:

- A definição de todas as *sprites*[6] disponíveis para o jogo, incluindo seus parâmetros e configurações de exibição;
- A definição dos relacionamentos entre personagens, usados nas definições de nível, e as *sprites* disponíveis;
- A especificação de quais eventos ocorrem quando duas *sprites* de um determinado tipo colidem no jogo;
- A especificação das condições finais do jogo, indicando se/quando o jogador ganha ou não.

No estudo apresentado por Torrado et al. [35], é apresentado o funcionamento interno do *framework*, que será brevemente explicado a seguir.

Para que a interação do jogo dê-se por meio de agentes, devem ser implementados dois métodos: um construtor, que é capaz de inicializar qualquer estrutura necessária para jogar, e um método de ação que, por sua vez, é chamado em todo *frame*[7] do jogo, esperando uma ação de retorno para ser executada referente àquele momento. Como os jogos e a autoaprendizagem funcionam em tempo real, os agentes devem responder dentro de um predeterminado tempo para a pontuação não ser prejudicada com penalizações.

A partir desses métodos, é possível informar aos agentes o dados atuais do jogo, como: [35]

- Estado do jogo (concluído ou em execução);
- Estado do jogador (pontuação, posição, orientação e recursos coletados);
- Informações anônimas sobre outras *sprites* do jogo.

Também é possível personalizar retornos avançados e capturas de telas do estado atual do jogo e fornecê-las aos agentes.

Além do seu uso em competições anuais, Torrado et al. [35] aborda que o *framework* GVGAI tem sido utilizado em pesquisas relacionadas a agentes de inteligência virtual, geração procedural de conteúdo[8], *design* de jogos autônomos, aprendizagem por reforço profundo, dentre outros.

III. Revisão da bibliografia

Como já foi dito, este trabalho tem como objetivo estudar, entender, comparar e otimizar modelos de Aprendizado por Reforço aplicados ao jogo Enduro. A realização de testes dentro de ambientes de jogos é uma técnica amplamente utilizada para o estudo do comportamento de algoritmos de aprendizado. Logo, nesta seção são apresentados e discutidos brevemente diferentes trabalhos onde seus autores buscaram aplicar modelos dentro desse mesmo ambiente.

Mnih et al. [38] apresentam problemas da aprendizagem por reforço pela perspectiva da aprendizagem profunda. Dentre eles, são citados, por exemplo, a necessidade de aprender por meio de sinais de frequência de recompensas que, por muitas vezes, tem algum tipo de ruído. Junto a isso, há o atraso referente ao estado do algoritmo com sua recompensa, que consideram ser "assustador" quando comparado às entradas de dados de algoritmos supervisionados. Diante dos problemas citados, o trabalho apresenta uma proposta onde demonstram como o uso de uma rede neural convolucional que, associada ao algoritmo Q-learning, pode superar tais desafios e aprender políticas de controle a partir das entradas usadas (que, no caso específico, tratam-se de dados de vídeo dentro de um ambiente complexo de aprendizagem por reforço). A proposta foi avaliada em sete jogos do Atari 2600, tendo sido capaz de aprender em seis desses, e de produzir resultados sobre-humanos em três deles.

Such et al. [39], em seu trabalho, citam o Atari Learning Environment (ALE)[9] como sendo uma grande ferramenta dentro para a aprendizagem por reforço, permitindo testar diferentes algoritmos de forma simples em diversos jogos do *console*. Além disso, também é explicado como o ALE foi importante para a popularização da aprendizagem por reforço profundo. Os autores também comentam que entender o funcionamento do algoritmo (isto é, como ele aprende e processa as informações), é tão importante quanto buscar a otimização do desempenho, e mostram como essas questões não são entendidas apenas fazendo uso da ferramenta buscando resultados quantitativos. Com isso, o trabalho traz à tona o Atari Zoo, um compilado de modelos já treinados usando algoritmos da família de aprendizagem por reforço profundo, aliados a um pacote de código aberto, que facilita a análise dos resultados.

No estudo de Torrado et al. [35] enfatiza-se o quão impactantes os jogos têm sido para o desenvolvimento de inteligências artificiais, principalmente quando o ramo tratado é o de aprendizagem de máquina. Além disso, esse contexto é abordado citando o GVGAI, mostrando o quão poderosa essa ferramenta pode ser para o desenvolvimento de IAs, visto que foi criada para prover grande versatilidade dentro desse contexto. Entretanto, mesmo sendo uma ferramenta que oferece grandes capacidades, os autores afirmam que sua infraestrutura pode deixar a desejar. Com isso, buscando otimizar e usufruir o potencial máximo do GVGAI em seu trabalho, eles conectam a ferramenta com a interface do OpenAI Gym[10], que proporciona diversos algoritmos dentro da área de aprendizagem por reforço. Usando os algoritmos DQN, A2C e *Prioritized Dueling DQN* (PDDQN), os resultados obtidos se diferem completamente dentre os jogos testados. O A2C por exemplo demonstrou melhores pontuações em comparação ao DQN e PDDQN em 6 dos 8 jogos. Vale mencionar também que, quando o resultado é binário, (ou seja, vitória ou derrota), tanto DQN como A2C obtiveram desempenho ruins.

No trabalho de Holubar et al. [40], é apresentado um novo ambiente de corrida para o OpenAI Gym: um simulador no qual o objetivo principal é finalizar o percurso o mais rápido possível. Dentro do ambiente, o agente deve aprender e executar as ações de dirigir, acelerar e frear quando necessário. Neste trabalho, foram usados dois algoritmos: *Sampled Policy Gradi-*

[6] Um conjunto de imagens digitais que podem ser movimentadas na tela ou serem manipuladas como uma entidade única.

[7] Intervalo de tempo em que cada imagem dentro de uma animação é criada.

[8] Processo de criação automática e programática de conteúdos como sons, textura, fases de jogos, dentre outros, em arquivos de tamanhos mínimos. [37]

[9] Ferramenta que possibilita desenvolver agentes de inteligência artificial para jogos do Atari 2600.

[10] Ferramenta para desenvolvimento e comparação de algoritmos de aprendizado por reforço.

ent (SPG) e PPO. No estudo, os resultados obtidos em duas diferentes configurações do algoritmo SPG foram comparados com o PPO em duas vertentes: uma sem um registro de experiências, e outra com uma alteração implementada no algoritmo que possibilita o registro das experiências obtidas. Com os testes feitos, os autores mostram que, para o algoritmo SPG, os treinamentos – mesmo quando comparado com episódios diferentes – não acarretam uma diferença brusca nos resultados, mas, quando usada priorização[11], os resultados acabam sendo ligeiramente superiores. No PPO, o uso de experiências passadas não trouxe uma melhora de desempenho para o agente. Em conclusão, o algoritmo PPO mostrou-se bastante proficiente quando faz uso de novas experiências, enquanto o SPG mostrou-se promissor ao tirar proveito de experiências passadas.

Islam et al. [31], em seu estudo, abordam o assunto de carros autônomos, citando alguns dados importantes como, por exemplo, o fato da maioria dos acidentes de carro serem causados por erros humanos. Com isso, tem como proposta desenvolver um sistema usando o algoritmo DQN. Como é inviável treinar o algoritmo no mundo real, usa o simulador do OpenAI Gym CarRacing-v0[12] para treinar o agente. Neste sentido, no desenvolvimento do trabalho, foram usadas quatro imagens como dados de entrada para que o algoritmo fosse capaz de tomar sua decisão ótima. Adicionalmente, a rede neural consistia em uma rede convolucional, usando as mesmas imagens da primeira camada. Como resultado, dentro dos parâmetros do simulador do OpenAI Gym, os autores alcançaram o resultado de 820 pontos. Mesmo que, para o OpenAI Gym, apenas resultados a partir de 900 pontos sejam considerados satisfatórios, o valor obtido também foi considerado devido às limitações citadas pelo autor, como as de *hardware*, por exemplo.

Os trabalhos apresentados nesta seção trouxeram diferentes jogos para aplicação de diferentes modelos de Aprendizado de Máquina, além de ferramentas desenvolvidas para auxiliar aqueles que pretendem estudar esse ramo. Os resultados obtidos em cada projeto, sejam positivos ou negativos, possibilitaram demonstrar o funcionamento dos modelos dentro desse ambiente de testes.

IV. Proposta

Neste trabalho é proposto um estudo comparativo entre três diferentes algoritmos de aprendizagem por reforço profundo e, em uma etapa posterior, a tentativa de aperfeiçoamento do algoritmo que apresentar os melhores resultados por meio da otimizaçção de seus hiperparâmetros[13]. Esses três algoritmos foram selecionados considerando sua aplicabilidade ao jogo Enduro do *console* Atari 2600. Na Figura 5 é possível observar as etapas envolvidas na realização do estudo, detalhadas na sequência.

[11]Dar prioridade a certas amostras durante a execução do modelo.
[12]Ambiente de simulação de corrida para aplicação de modelos de aprendizagem de máquina da plataforma OpenAI Gym.
[13]São parâmetros ajustáveis que permitem controlar o processo de treinamento de um modelo de aprendizado de máquina.

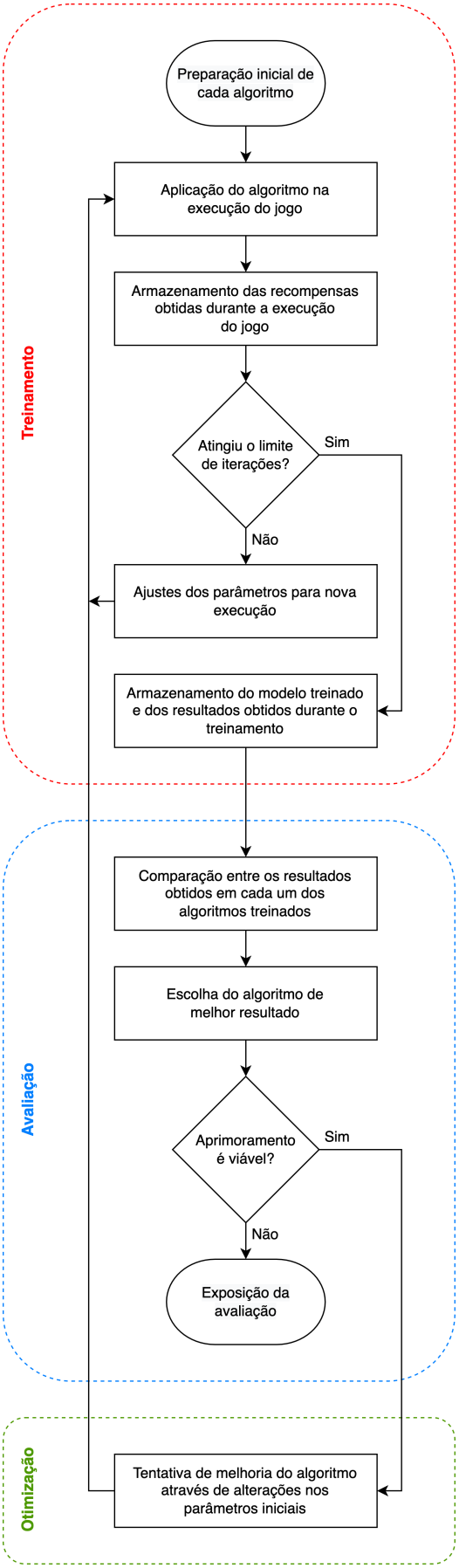


Fig. 5. Fluxograma contendo os processos necessários à execução da proposta.

A. Treinamento

O fluxo seguido para a realização do estudo se deu a partir do ciclo de treinamento de cada um dos algoritmos. Ele consiste na exposição de cada um à execução do jogo, armazenando os resultados obtidos e retroalimentando o modelo em uma nova exposição do algoritmo ao jogo. A repetição desse ciclo de treinamento, retroalimentado por uma determinada quantidade de vezes, resulta no aprendizado que possibilita à máquina obter resultados cada vez melhores na sua execução. Este bloco é dividido em cinco etapas:

a.1. Preparação inicial de cada algoritmo

Na etapa inicial é feita a construção do ambiente de treinamento considerando uma quantidade de iterações mínimas para que seja possível validar o funcionamento correto do ambiente e dos algoritmos.

a.2. Aplicação do algoritmo na execução do jogo

Após a preparação dos algoritmos, deve-se expô-los à execução do jogo para gerar resultados baseados nos parâmetros pré-treinados (ou aleatoriamente inicializados, caso seja a primeira execução). A execução só termina quando ocorre o evento de fim de jogo, ou seja, o momento em que o jogador não atinge a meta mínima de carros a serem ultrapassados dentro do tempo determinado pelo sistema do próprio jogo.

a.3. Armazenamento das recompensas obtidas durante a execução do jogo

O armazenamento das recompensas obtidas durante a execução do jogo é realizado para uma análise posterior. Caso seja atingido o número de iterações definidas no início, valor que pode variar entre centenas de milhares e dezenas de milhões, o treinamento é redirecionado para a etapa a.5. para que seja finalizado.

a.4. Ajustes dos parâmetros para nova execução

É realizada uma análise das recompensas obtidas até então, para que seja feito o ajuste dos parâmetros e iniciada uma nova exposição do algoritmo ao jogo a partir da etapa a.2..

a.5. Armazenamento do modelo treinado e dos resultados obtidos durante o treinamento

Quando o número total de iterações for atingido, é feito o armazenamento de todos os dados e resultados produzidos durante o treinamento para serem utilizados durante a etapa de avaliação.

B. Avaliação

No estágio de avaliação são recapitulados todos os resultados obtidos nos treinamentos dos modelos e feitas as considerações. Este bloco contempla duas etapas:

b.1. Comparação entre os resultados obtidos de cada um dos algoritmos utilizados no treinamento

Os resultados dos algoritmos treinados até então são comparados a partir das seguintes métricas:

- Valor máximo de recompensa, encontrado como uma constante ao final do treinamento;
- Tempo gasto, encontrado como um valor constante ao final do treinamento;
- Valor médio das recompensas obtidas durante o treinamento através da Equação (1), na qual r são as recompensas obtidas em n execuções do jogo:

$$M(r) = \frac{\sum_1^n r_i}{n} \tag{1}$$

- Desvio padrão considerando todo o treinamento, dado pela Equação (2), , na qual r são as recompensas obtidas em n execuções do jogo e $\bar{r}$ é a recompensa média:

$$DP(r) = \sqrt{\frac{\sum_1^n (r_i - \bar{r})^2}{n}} \tag{2}$$

b.2. Escolha do algoritmo com o melhor resultado

Baseado nas métricas propostas, um algoritmo é eleito como a melhor escolha para a execução do jogo. Caso sua otimização seja viável, o fluxo é redirecionado para a etapa c.1..

b.3. Exposição da avaliação

Estágio final onde são apresentados os resultados de cada um dos modelos treinados e as avaliações de desempenho.

C. Otimização

A partir da escolha do algoritmo que apresenta os melhores resultados, esse bloco de etapa única, busca aperfeiçoá-lo com o objetivo de alcançar um desempenho ainda superior no jogo.

c.1. Tentativa de melhoria do algoritmo por meio de de alterações nos parâmetros iniciais

Nessa etapa, os parâmetros iniciais do algoritmo são alterados e um novo ciclo de treinamento deve ser inciado a partir da etapa a.2..

V. Experimentos

Como forma de validar a proposta definida, foram realizadas experiências práticas utilizando ferramentas que auxiliam na execução das tarefas necessárias para realizar testes, validações e apresentação de resultados. Após toda a experimentação, também foi feita uma discussão sobre os resultados obtidos.

A. Metodologia

A experimentação tomou como base o ambiente de desenvolvimento voltado ao aprendizado de máquina por reforço OpenAI Gym e as implementações fornecidas pela biblioteca de código Stable-Baselines3. Todos os códigos desenvolvidos e modelos treinados foram desenvolvidos com a linguagem de programação Python. As características de cada uma das principais ferramentas são descritas a seguir.

- OpenAI Gym: ambiente com vasta quantidade de ferramentas para o desenvolvimento, treinamento e teste de modelos de aprendizado de máquina por reforço, com interfaces de código aberto de uso simplificado. [41]
- Stable-Baselines3: biblioteca que fornece diversas implementações dos principais algoritmos existentes para o aprendizado de máquina por reforço. Além disso, ela é mantida com uma grande cobertura de testes, incluindo os voltados para desempenho, que garantem alta confiabilidade na utilização de seus algoritmos. [42]
- RL Baselines3 Zoo: projeto disponibilizado para a utilização do Stable-Baselines3 que provê uma coleção de agentes pré-treinados, *scripts* para simplificar treinamentos, meios de avaliação dos agentes, melhorias de parâmetros e apresentação de resultados. Além disso, o projeto também contém resultados de execuções previamente feitas para comparação de resultados obtidos durante seu uso. [43]
- Optuna: *framework* com ferramentas para otimização de hiperparâmetros. É amplamente utilizada na busca em espaço de valores para otimizar parâmetros presentes em modelos de aprendizado de máquina. [44]

Nos experimentos foram selecionados três algoritmos para comparação – PPO, DQN e A2C – a partir das implementações presentes na biblioteca Stable-Baselines3, utilizando o critério de quais melhores se encaixam na natureza do problema fornecido pelo jogo Enduro do *console* Atari 2600.

Para a execução dos experimentos, foi preparado um ambiente utilizando o sistema operacional Ubuntu 20.04.3 LTS (*Long-term Support*), com a instalação do Python na versão 3.8.10. Todas as ferramentas citadas anteriormente foram instaladas e testadas para garantir o funcionamento correto. O ambiente ficou exclusivamente voltado aos experimentos e teve como especificações:

- CPU Intel Core i7-7500U CPU @ 2.70GHz;
- 8GB de memória;
- 480GB de armazenamento em SSD (*Solid State Drive*);
- GPU (*Graphics Processing Unit*) dedicada Nvidia GeForce 920MX.

A seleção da quantidade de iterações a serem utilizadas nos treinamentos baseou-se no *benchmark* disponibilizado no projeto RL Baselines3 Zoo, como exibido na da Tabela I.

TABELA I
ADAPTAÇÃO DA TABELA DE *benchmark* (BASEADA EM [43]).

Algoritmo	Recompensas Médias	Desvio Padrão das Recompensas	Número de Iterações (em milhões)
A2C	0	0	10
DQN	830.929	194.544	10
PPO	996.364	176.090	10

Para todos os algoritmos selecionados neste estudo o número de iterações foi de dez milhões. Na etapa inicial de testes, com o objetivo de validar o funcionamento da implementação sem grande uso computacional, esse número foi reduzido a 10%, ou seja, um milhão de iterações.

Além da quantidade de iterações para o treinamento, também foi aplicada, a cada mil iterações, uma etapa de teste do modelo atual em cinco rodadas do jogo. Os gráficos de recompensa, que serão apresentados na subseção a seguir, contemplam a média no desvio padrão de cada uma dessas etapas de teste.

Os hiperparâmetros iniciais também foram selecionados com base no projeto RL Baselines3 Zoo. Pôde-se observar, como descrito no arquivo de configurações, que os hiperparâmetros pré-definidos para o Enduro não estão totalmente otimizados.

B. Resultados e Discussão

Nesta subseção são apresentados os resultados obtidos por meio da execução prática dos experimentos utilizando todas as ferramentas apresentadas anteriormente. A experimentação foi dividida em quatro partes, visando organizar melhor as tarefas necessárias.

b.1. Validação do Ambiente

A validação inicial se deu a partir da execução do treinamento de cada um dos algoritmos, iniciando-se com um modelo-base aleatório. Para cada um dos algoritmos, foram executados um total de um milhão de iterações. Os resultados são apresentados no gráfico da Figura 6.

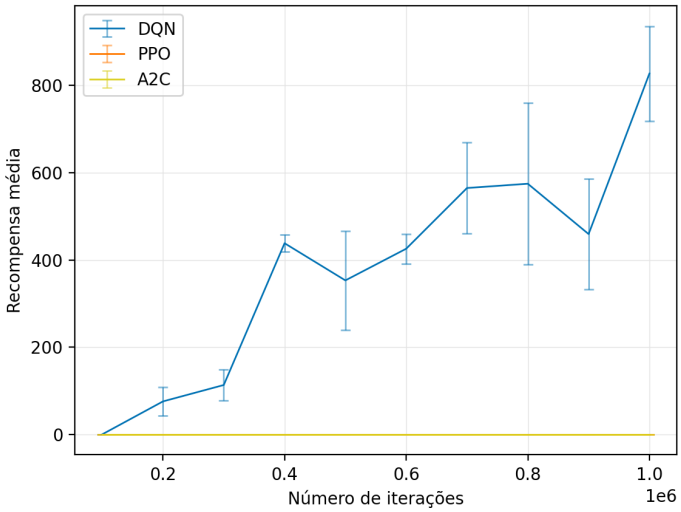


Fig. 6. Recompensas médias obtidas por número de iterações ao longo da etapa de treinamento, considerando um milhão de iterações. As barras de erro representam o desvio padrão.

Observando tal gráfico, pode-se verificar que os algoritmos A2C e PPO não geram nenhuma recompensa para quantidades de iterações inferiores a um milhão. Já o algoritmo DQN apresentou uma tendência aparentemente crescente de recompensa, de certa forma proporcional à quantidade de iterações. Contudo, em suas etapas de teste, ele apresentou grande variação nos resultados, representadas graficamente pela amplitude das barras de erro. A execução total dessa etapa demandou os tempos apresentados na Tabela II.

TABELA II
TEMPOS DE EXECUÇÃO PARA UM MILHÃO DE ITERAÇÕES.

Algoritmo	Tempo (em horas e minutos)
DQN	04:24
PPO	01:50
A2C	01:26

b.2. Treinamento dos Modelos

Feita a validação do ambiente, seguiu-se então para o treinamento completo, considerando dez milhões de iterações dos algoritmos. Os resultados da segunda etapa são apresentados no gráfico da Figura 7:

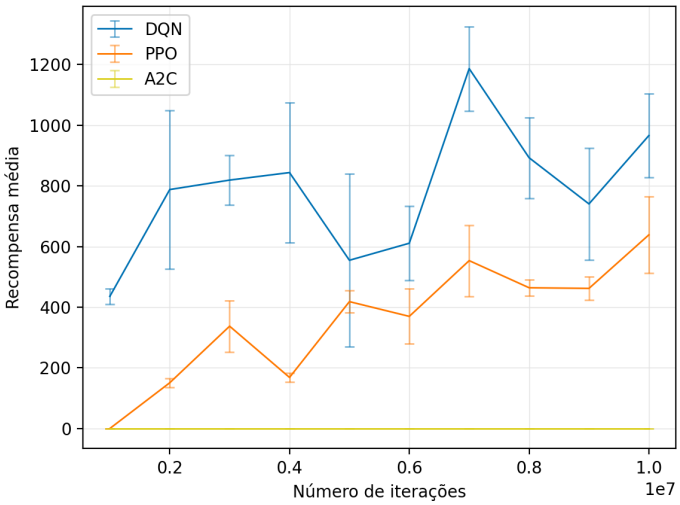


Fig. 7. Recompensas médias por número de iterações, considerando dez milhões de iterações. As barras de erro representam o desvio padrão.

Como pode-se observar, o A2C continuou sem produzir resultados, portanto foi descartado para continuação do estudo. O PPO por sua vez começou a obter recompensas. Ambos PPO e DQN apresentaram resultados crescentes à medida que as iterações foram sendo incrementadas, mesmo com algumas variações negativas. Também é possível notar que durante todo experimento o DQN aparenta fornecer recompensas mais elevadas em relação ao PPO. A execução total dessa etapa levou o tempo apresentado na Tabela III:

TABELA III
TEMPOS DE EXECUÇÃO PARA DEZ MILHÕES DE ITERAÇÕES.

Algoritmo	Tempo (em horas e minutos)
DQN	73:40
PPO	38:00
A2C	24:09

b.3. Avaliação dos Resultados

Com os resultados em mãos, iniciou-se a etapa de avaliação, visando selecionar o melhor algoritmo para tentar otimizar seus hiperparâmetros. À primeira vista, o DQN parece produzir os melhores resultados. Porém, uma análise mais detalhada mostra uma maior aceleração nos ganhos de recompensa do PPO ao longo das iterações. Para verificar este comportamento, foi realizada uma regressão linear nas distribuições de resultados apresentadas pelos dois algoritmos, produzindo as Equações (3) e (4) para o DQN e o PPO, respectivamente. A Figura 8 representa graficamente tais funções, onde i é a i-ésima iteração.

$$DQN(i) = 35,718 \times 10^{-6} \cdot i + 587,468 \tag{3}$$

$$PPO(i) = 58,586 \times 10^{-6} \cdot i + 34,093 \tag{4}$$

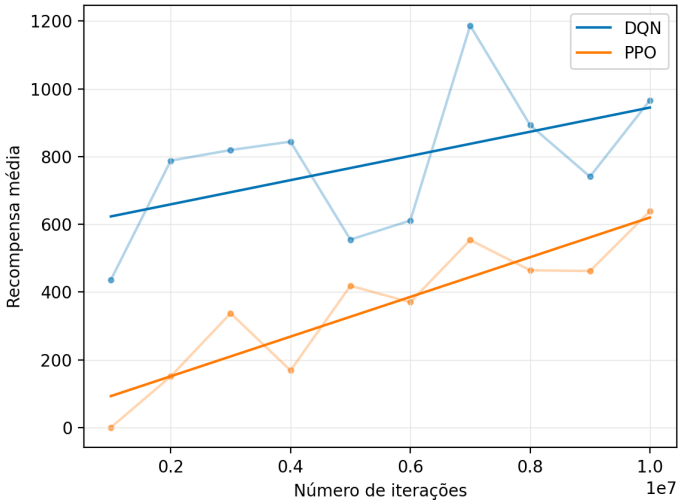


Fig. 8. Regressão linear realizada para dez milhões de iterações.

Portanto, igualando as equações (3) e (4), pode-se obter o número médio (teórico) de iterações para que o algoritmo PPO forneça resultados médios superiores aos do DQN:

$$\begin{aligned} DQN(i) &= PPO(i) \\ 35,718 \times 10^{-6} \cdot i + 587,468 &= 58,586 \times 10^{-6} \cdot i + 34,093 \\ 22,868 \times 10^{-6} \cdot i &= 553,375 \\ i &= 24,199 \times 10^{6} \end{aligned} \tag{5}$$

Diante deste raciocínio, pode-se concluir que, em média, após aproximadamente vinte e quatro milhões de iterações, o PPO começa produzir recompensas superiores ao DQN. Tal fato pode ser visualizado no gráfico da Figura 9.

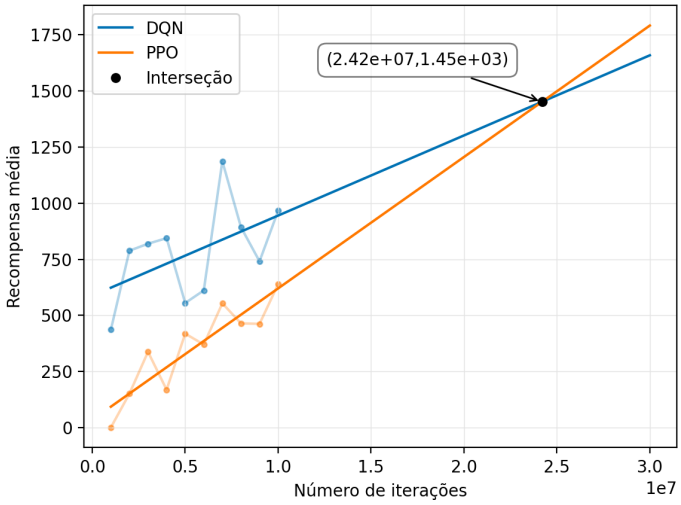


Fig. 9. Regressão linear extrapolada para 30 milhões de iterações.

Considerando também que o algoritmo PPO leva, aproximadamente, metade do tempo para a execução do treinamento se comparado ao DQN, pode-se concluir que, para quantidades maiores de iterações, espera-se que este algoritmo mostre-se como o melhor dentro da comparação.

b.4. Otimização dos Hiperparâmetros

Para a última etapa do estudo, foi realizado, dentro do projeto RL Baselines3 Zoo e utilizando a ferramenta Optuna, uma

tentativa de otimização dos hiperparâmetros do algoritmo PPO. A função de otimização disponibilizada pela ferramenta depende do fornecimento dos espaços amostrais de cada um dos hiperparâmetros que deseja-se otimizar. Um exemplo de hiperparâmetro é a topologia das redes neurais que será aplicada (ou seja, a quantidade de camadas e de neurônios em cada camada). Também é necessário fornecer um número de vezes que a função deverá ser executada e a quantidade de iterações do algoritmo de aprendizado de máquina irá executar.

O espaço amostral fornece à ferramenta uma matriz de possíveis combinações dos hiperparâmetros para que sejam testadas e avaliadas. Quando é fornecida uma matriz com todas as combinações possíveis, pode-se (teoricamente) encontrar os melhores hiperparâmetros; porém, ao custo de um tempo de execução impraticavelmente alto. Portanto, é necessário encontrar um equilíbrio dentro das possibilidades criadas no espaço amostral para resultar em uma bateria de otimização com um custo computacional pragmaticamente exequível.

Em uma primeira tentativa, realizou-se a execução de dez rodadas de otimização para cem mil iterações do algoritmo em cada uma delas. Os espaços amostrais utilizados foram os fornecidos por dentro do padrão de otimização do algoritmo no RL Baselines3 Zoo. Foram-se obtendo, então, os valores para os hiperparâmetros em cada uma das rodadas de execução. Na finalização das rodadas, sabe-se qual delas produziu mais recompensas. Os hiperparâmetros adotados nessa rodada são selecionados como os melhores. Ao final da otimização aplicou-se os hiperparâmetros em uma nova etapa de treinamento de um novo modelo. Com o intuito de identificar o comportamento da otimização ao longo do treinamento, foi selecionada a quantidade de dois milhões de iterações para essa etapa. A princípio, os resultados não se demonstraram promissores. A partir de determinada quantidade de iterações, o treinamento parou de obter recompensas e então a etapa foi abortada, como mostrado no gráfico da Figura 10:

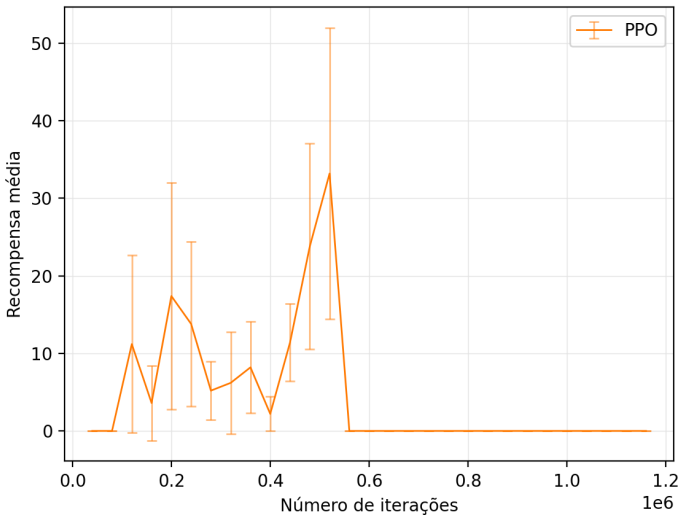


Fig. 10. Recompensas médias por número de iterações após a otimização dos hiperparâmetros.

Com os valores obtidos, pode-se observar que o treinamento produziu resultados com uma quantidade muito inferior de iterações. Porém, depois de aproximadamente meio milhão de iterações, houve uma queda abrupta nas recompensas até chegar a zero. Concluiu-se, então, que a tentativa de otimização baseando-se em cem mil iterações fez com que houvesse uma produção de resultados imediatamente "bons", mas que, ao longo prazo, atrapalham a convergência do algoritmo e, consequentemente, o impeça de alcançar recompensas relevantes. Cada rodada do experimento levou, em média, 39 minutos e 33 segundos.

Em consequência dos resultados demonstrados, conduziu-se então uma nova sequência de rodadas de otimização. Foi decidido que a realização da nova sequência deveria ser realizada com duas milhões de iterações. A decisão foi tomada com base no objetivo de produzir resultados em escalas superiores. Adicionalmente, foi levado em consideração que, com os hiperparâmetros iniciais, o algoritmo começou a gerar recompensas com iterações entre um e meio e dois milhões de iterações, como visto na média da Figura 7.

Além da mudança no número de iterações, também pode-se concluir que dever-se-ia excluir a maioria dos hiperparâmetros presentes no espaço amostral fornecido por padrão, de modo a evitar a "explosão combinatória". Foram então selecionados os espaços amostrais que, a princípio, gerassem consequências mais impactantes no resultado do aprendizado. Essa modificação veio com o objetivo de diminuir consideravelmente a demanda de tempo e custo computacional, já que o número de iterações fora aumentado drasticamente. Os hiperparâmetros foram selecionados a partir dos mais utilizados em estudos de aprendizado de máquina, mais especificamente associados às redes neurais, sendo eles:

- Taxa de aprendizagem: representa um fator multiplicativo da magnitude do ajuste que será feito nos hiperparâmetros (isto é, pesos sinápticos) do modelo a cada iteração do treinamento. A modificação deste parâmetro pode gerar grandes variações nos resultados, conforme Ebhota et al. [45];
- Arquitetura da rede neural: esse hiperparâmetro foi selecionado com o objetivo de verificar se haveria grandes mudanças nos resultados devido a variação na quantidade de neurônios presentes nas camadas das redes neurais;
- Função de ativação: O último parâmetro selecionado também é um dos alvos de estudos de desempenho em algoritmos de aprendizado de máquina. De acordo com Hayou et al. [46], a modificação desse hiperparâmetro pode impactar consideravelmente os resultados do treinamento de um modelo inteligente.

Os demais hiperparâmetros disponibilizados por padrão para rodadas de otimização dentro do RL Baselines3 Zoo foram omitidos e os espaços amostrais para os hiperparâmetros selecionados foram mantidos sem alteração.

Após a execução da nova tentativa de otimização, utilizando a quantidade de iterações com um aumento de 2.000%, não foi possível obter nenhum resultado superior aos representados na Figura 10. Contudo, o tempo médio das rodadas foi de 759 minutos e 36 segundos (aproximadamente 13 horas), o que representa um aumento médio de 1.920% na duração de cada uma delas. Com os novos dados obtidos, chegou-se à conclusão de que não seria viável encontrar uma otimização dentro do período de tempo previsto para a execução do presente estudo.

VI. Conclusão

A inteligência artificial vem se mostrando o quão impactante pode ser dentro de uma grande variedade de contextos. Além disso, a alta aplicabilidade da tecnologia (e suas ramificações, como a aprendizagem de máquina) em diferentes áreas a torna ainda mais relevante. Porém, só foi possível alcançar tamanho potencial graças a anos de estudos das mais diversas formas como, por exemplo, o as aplicações de IA no contexto de jogos – uma poderosa forma de estudo que mostrou-se promissora para o desenvolvimento de modelos competentes para solucionar uma vasta gama de problemas, como no caso dos algoritmos de aprendizado por reforço. Diante disto, o presente trabalho buscou estudar e entender o funcionamento de três algoritmos em particular, analisando seus comportamentos em cenários de teste e otimização controlados para que, com os resultados obtidos, pudessem ser tiradas conclusões robustas.

Diversos trabalhos usaram e documentaram o uso dos modelos estudados neste artigo. Cada um abordou diferentes jogos, o que mostra não apenas a versatilidade que a aprendizagem de máquina possui, como também confirma o quão impactante os jogos têm sido na evolução desta área. Assim sendo, neste estudo foram aplicados os algoritmos PPO, A2C e DQN no jogo Enduro da plataforma Atari, onde o A2C foi rapidamente descartado devido à falta de resultados. Nas iterações posteriores, o DQN mostrou-se melhor que o PPO, de um modo geral.

Buscando uma análise mais detalhada, foi aplicada uma técnica de regressão linear, que mostrou que o PPO evolui mais rápido (isto é, com um menor número de iterações), se comparado ao seu competidor (DQN). Unindo este fato a sua rápida execução, conclui-se que, dentre os algoritmos testados, o PPO ocupa ao primeiro lugar do pódio. Na etapa seguinte, o objetivo era otimizar os valores até então obtidos. Para isso, foram feitas mudanças de hiperparâmetros, que rapidamente trouxeram resultados promissores mas, ao mesmo tempo, limitados, pois após aproximadamente 500 mil iterações, houve uma queda abrupta de recompensa, que inclusive atingiu valor zero. Foi considerada uma nova bateria de testes com novas alterações aplicadas ao modelo, que não produziu resultados satisfatórios.

Por fim, é possível afirmar que, apesar dos resultados limitados, eles não são desanimadores. Pôde-se perceber que a otimização dos algoritmos é viável, empregando-se as devidas adequações em seus hiperparâmetros. Com isso, apresenta-se como sugestão natural para trabalhos futuros a continuidade e aprofundamento nas experimentações envolvendo otimização de hiperparâmetro, norteadas por modelos de regressão ainda mais robustos (como o polinomial, por exemplo).

Portanto, de forma geral, ambos os algoritmos PPO e DQN provaram-se alternativas adequadas para resolver o problema estudado; também, mostram-se como candidatos a trazerem soluções ainda melhores quando encontrados submetidos ao devido ajuste de hiperparâmetros.

Referências

[1] noomis. *Inteligência artificial cresce na América Latina, mas desafio ainda é grande*. Mar. de 2021. URL: https://noomis.febraban.org.br/temas/inteligencia-artificial/inteligencia-artificial-cresce-na-america-latina-mas-desafio-ainda-e-grande.

[2] Microsoft. *A adoção de inteligência artificial pode adicionar 4,2 pontos percentuais de crescimento adicional ao PIB do Brasil até 2030*. Dez. de 2020. URL: https://news.microsoft.com/pt-br/a-adocao-de-inteligencia-artificial-pode-adicionar-42-pontos-percentuais-de-crescimento-adicional-ao-pib-do-brasil-ate-2030/.

[3] André Gualtieri e Eugênio V. Garcia. *Inteligência artificial para o desenvolvimento: hora de virar o jogo no Brasil*. Ago. de 2021. URL: https://www.jota.info/opiniao-e-analise/colunas/regulacao-e-novas-tecnologias/inteligencia-artificial-para-o-desenvolvimento-hora-de-virar-o-jogo-no-brasil-07082021.

[4] Redação IT Forum. *7 áreas que estão sendo transformadas pela inteligência artificial*. Jun. de 2018. URL: https://itforum.com.br/noticias/7-areas-que-estao-sendo-transformadas-pela-inteligencia-artificial/.

[5] Igor Pimenta. *[Guia Completo] Inteligência Artificial: o que é, conceito e métodos de IA*. Ago. de 2021. URL: https://www.take.net/blog/tecnologia/inteligencia-artificial/.

[6] Gerd Brewka. "Artificial intelligence—a modern approach by Stuart Russell and Peter Norvig, Prentice Hall. Series in Artificial Intelligence, Englewood Cliffs, NJ." Em: *The Knowledge Engineering Review* 11.1 (1996), pp. 78–79.

[7] Ira Pohl. "John Haugeland. Artificial intelligence: the very idea. Bradford books. The MIT Press, Cambridge, Mass., and London, 1985, ix 289 pp." Em: *Journal of Symbolic Logic* 53.2 (1988), pp. 659–660.

[8] Richard Bellman. *An introduction to artificial intelligence: can computer think?* 1978.

[9] Eugene Charniak. *Introduction to artificial intelligence*. Pearson Education India, 1985.

[10] Patrick Henry Winston. *Artificial Intelligence (3rd Ed.)* USA: Addison-Wesley Longman Publishing Co., Inc., 1992. ISBN: 0201533774.

[11] Ray Kurzweil, Robert Richter, Ray Kurzweil e Martin L Schneider. *The age of intelligent machines*. Vol. 580. MIT press Cambridge, 1990.

[12] Elaine Rich, Kevin Knight, Pedro Antonio González Calero e Fernando Trescastro Bodega. *Inteligencia artificial*. Vol. 1. McGraw-Hill, 1994.

[13] R.J. Schalkoff. *Artificial Intelligence: An Engineering Approach*. McGraw-Hill Computer science series. McGraw-Hill, 1990. ISBN: 9780070550841.

[14] George F Luger. *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education, 2005.

[15] Cezary Kaliszyk, François Chollet e Christian Szegedy. "Holstep: A machine learning dataset for higher-order logic theorem proving". Em: *CoRR* (2017).

[16] Issam El Naqa e Martin J. Murphy. "What Is Machine Learning?" Em: *Machine Learning in Radiation Oncology: Theory and Applications*. Ed. por Issam El Naqa, Ruijiang Li e Martin J. Murphy. Springer International Publishing, 2015, pp. 3–11.

[17] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". Em: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229.

[18] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[19] C. Crisci, B. Ghattas e G. Perera. "A review of supervised machine learning algorithms and their applications to ecological data". Em: *Ecological Modelling* 240 (2012), pp. 113–122. ISSN: 0304-3800.

[20] Sun-Chong Wang. "Artificial Neural Network". Em: *Interdisciplinary Computing in Java Programming*. Boston, MA: Springer US, 2003, pp. 81–100. ISBN: 978-1-4615-0377-4.

[21] Favio Vázquez. *Deep Learning made easy with Deep Cognition*. Dez. de 2017. URL: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351.

[22] Quoc V. Le. "Building high-level features using large scale unsupervised learning". Em: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 8595–8598.

[23] Hari Krishna Kanagala e VV Jaya Rama Krishnaiah. "A comparative study of K-Means, DBSCAN and OPTICS". Em: *2016 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE. 2016, pp. 1–6.

[24] Richard S Sutton e Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[25] Data Science Academy. *Capítulo 62 – O Que é Aprendizagem Por Reforço?* Dez. de 2019. URL: https://www.deeplearningbook.com.br/o-que-e-aprendizagem-por-reforco/.

[26] Gabi Viana. *Capítulo 62 – O Que é Aprendizagem Por Reforço?* Nov. de 2018. URL: https://medium.com/@gabi.viana11/diferenca-entre-aprendizado-de-maquina-machine-learning-aprendizagem-profunda-deep-learning-3035e95ba1d1.

[27] Sepp Hochreiter e Jürgen Schmidhuber. "Long short-term memory". Em: *Neural computation* 9.8 (1997), pp. 1735–1780.

[28] Geoffrey E Hinton e Richard S Zemel. "Autoencoders, minimum description length, and Helmholtz free energy". Em: *Advances in neural information processing systems* 6 (1994), pp. 3–10.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville e Yoshua Bengio. "Generative Adversarial Networks". Em: *Communications of the ACM* 63.11 (out. de 2020), pp. 139–144. ISSN: 0001-0782.

[30] Yuxi Li. "Deep Reinforcement Learning: An Overview". Em: *CoRR* abs/1701.07274 (2017).

[31] Sadman Saumik Islam e Samia Binta Alam. *Autonomous Virtual Vehicle Using Reinforcement Learning*. 2019.

[32] Praphul Singh. *Reinforcement Learning: Proximal Policy Optimization (PPO)*. Abr. de 2020. URL: https://blogs.oracle.com/ai-and-datascience/post/reinforcement-learning-proximal-policy-optimization-ppo.

[33] user sidsen99 user e user avtarkumar719 user. *A Brief Introduction to Proximal Policy Optimization*. Nov. de 2021. URL: https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/.

[34] Sergios Karagiannakos. *The idea behind Actor-Critics and how A2C and A3C improve them*. Nov. de 2018. URL: https://theaisummer.com/Actor_critics/.

[35] Ruben Rodriguez Torrado, Philip Bontrager, Julian Togelius, Jialin Liu e Diego Perez-Liebana. "Deep Reinforcement Learning for General Video Game AI". Em: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. 2018, pp. 1–8.

[36] The General Video Game AI Competition. *The Video Game Definition Language (VGDL)*. 2018. URL: http://www.gvgai.net/vgdl.php.

[37] Wendell Araújo e Eduardo Aranha. *Geração Procedural de Conteúdo para Criação de Fases de Jogos Educativos usando Gramática*. 2018. URL: https://br-ie.org/pub/index.php/sbie/article/view/8031.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra e Martin A. Riedmiller. "Playing Atari with Deep Reinforcement Learning". Em: *CoRR* abs/1312.5602 (2013).

[39] Felipe Petroski Such, Vashisht Madhavan, Rosanne Liu, Rui Wang, Pablo Samuel Castro, Yulun Li, Ludwig Schubert, Marc G. Bellemare, Jeff Clune e Joel Lehman. "An Atari Model Zoo for Analyzing, Visualizing, and Comparing Deep Reinforcement Learning Agents". Em: *CoRR* abs/1812.07069 (2018).

[40] Mario S. Holubar e Marco A. Wiering. "Continuous-action Reinforcement Learning for Playing Racing Games: Comparing SPG to PPO". Em: *CoRR* abs/2001.05270 (2020).

[41] J K Terry e Kevin Corder. *gym README.md*. Nov. de 2021. URL: https://github.com/openai/gym.

[42] Antonin Raffin et al. *stable-baselines3 README.md*. Out. de 2021. URL: https://github.com/DLR-RM/stable-baselines3.

[43] Antonin raffin, Quentin Gallouédec, Martiño Crespo, Darío Hereñú, Carl Boettiger e Marc. *rl-baselines3-zoo README.md*. Nov. de 2021. URL: https://github.com/DLR-RM/rl-baselines3-zoo.

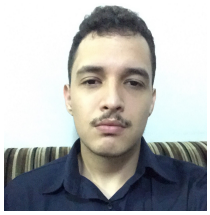
[44] Toshihiko Yanase et al. *optuna README.md*. Out. de 2021. URL: https://github.com/optuna/optuna.

[45] Virginia Ebhota, Joseph Isabona e Viranjay Srivastava. "Effect of Learning Rate on GRNN and MLP for the Prediction of Signal Power Loss in Microcell Sub-Urban Environment". Em: *International Journal on Communications Antenna and Propagation (IRECAP)* 9 (fev. de 2019), p. 36.

[46] Soufiane Hayou, Arnaud Doucet e Judith Rousseau. "On the Impact of the Activation function on Deep Neural Networks Training". Em: *Proceedings of the 36th International Conference on Machine Learning*. Ed. por Kamalika Chaudhuri e Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, jun. de 2019, pp. 2672–2680.

AUTORES

Felipe Juvêncio Mendes é Técnico em Eletrônica com ênfase em Automação Industrial pela Escola Técnica de Eletrônica Francisco Moreira da Costa (ETE FMC) e graduando em Engenharia de Computação pelo Instituto Nacional de Telecomunicações (Inatel). Atualmente compõe a equipe de desenvolvimento de *software* na organização autônoma descentralizada Psychedelic. Possui interesse na área de engenharia de *software*.



Gustavo Henrique de Moraes Pessa é graduando em Engenharia de Computação pelo Instituto Nacional de Tecnologias (Inatel). Durante a graduação foi bolsista de Iniciação Científica voltada a área de Digital Twins dentro de indústrias. Atualmente atua como desenvolvedor de *software* na empresa Keymax. Possui interesse nas áreas de desenvolvimento *frontend*, tanto *web* como *mobile*.

João Leonardo Andrade Morganti Silva é Técnico em Eletrônica com ênfase em Automação Industrial pela Escola Técnica de Eletrônica Francisco Moreira da Costa (ETE FMC) e graduando em Engenharia de Computação pelo Instituto Nacional de Telecomunicações (Inatel). Durante a graduação foi bolsista de Iniciação Científica voltada às tecnologias IoT (*Internet of Things*) e também membro da CP2eJr (Empresa Júnior do Inatel), mais recentemente como presidente. Atualmente atua como desenvolver iOS na empresa Toodoo. Possui interesse nas áreas de desenvolvimento *mobile* e gestão de projetos.

Marcelo Vinícius Cysneiros Aragão é graduado em Engenharia de Computação pelo Instituto Nacional de Telecomunicações (Inatel) em 2014, Mestre em Ciência e Tecnologia da Computação pela Universidade Federal de Itajubá em 2018. Trabalhou de 2011 a 2018 no Inatel Competence Center, onde atuou principalmente como desenvolvedor de soluções de *Business Support Systems* (BSS) em ambiente de integração contínua. Atualmente, trabalha como professor do ensino superior no Inatel, ministrando disciplinas como Inteligência Computacional e Matemática Discreta, e também como coordenador do curso de pós-graduação em Desenvolvimento de Aplicações para Dispositivos Móveis e *Cloud Computing*. Possui interesse nas áreas de aprendizado de máquina, ciência de dados e engenharia de *software*.