Aplicação de Técnicas de Aprendizado por Reforço Profundo no Jogo DOOM: Um Estudo com Dueling DQN e PPO

Dayon Victor Czykailo[1]
[1] *Centro Universitário UNIFACEAR*

Rosanete Grassiani dos Santos[2]
[2]*Centro Universitário UNIFACEAR*

RESUMO
Este trabalho tem como objetivo desenvolver e avaliar um agente de aprendizado por reforço profundo aplicado ao jogo DOOM, utilizando a plataforma ViZDoom como ambiente de simulação. Foram comparados os algoritmos Dueling Deep Q-Network (Dueling DQN) e Proximal Policy Optimization (PPO), com o propósito de analisar sua eficiência, estabilidade e capacidade de adaptação em diferentes cenários tridimensionais. A implementação foi realizada em Python, com uso das bibliotecas PyTorch e Stable Baselines3, e monitoramento via TensorBoard. O treinamento ocorreu nos cenários Basic, Defend the Center e Deadly Corridor, variando-se as recompensas e dificuldades para avaliar o aprendizado do agente. Os resultados obtidos buscam evidenciar o comportamento e o desempenho de cada abordagem, contribuindo para o avanço das pesquisas em inteligência artificial aplicada a ambientes visuais complexos.

Palavras-chave: Aprendizado por Reforço Profundo; DQN; PPO; ViZDoom; Inteligência Artificial.

ABSTRACT
This work aims to develop and evaluate a deep reinforcement learning agent applied to the game DOOM, using the ViZDoom platform as a simulation environment. The Dueling Deep Q-Network (Dueling DQN) and Proximal Policy Optimization (PPO) algorithms were compared to analyze their efficiency, stability, and adaptability in different three-dimensional scenarios. The implementation was carried out in Python, using the PyTorch and Stable Baselines3 libraries, and monitoring via TensorBoard. Training took place in the Basic, Defend the Center, and Deadly Corridor scenarios, varying rewards and difficulties to evaluate the agent's learning. The results obtained aim to highlight the behavior and performance of each approach, contributing to the advancement of research in artificial intelligence applied to complex visual environments.

Keywords: Deep Reinforcement Learning; DQN; PPO; ViZDoom; Artificial Intelligence.

1. INTRODUÇÃO

Nas últimas décadas, a Inteligência Artificial (IA) consolidou-se como uma das áreas mais promissoras da ciência da computação, com aplicações que abrangem setores como saúde, finanças, transporte e, de forma marcante, o entretenimento digital. A IA pode ser definida como o campo responsável pelo desenvolvimento de sistemas computacionais capazes de simular comportamentos inteligentes, por meio da percepção

do ambiente, tomada de decisões autônomas e aprendizado contínuo (RUSSELL; NORVIG, 2020), como o desenvolvimento de agentes inteligentes, que se tem se tornado uma área central na pesquisa e inovação no campo dos jogos eletrônicos. Em particular, a criação de bots (robôs) de jogos capazes de simular comportamentos humanos ou até superar habilidades de jogadores reais é um desafio que envolve aspectos complexos de inteligência artificial (IA). Agentes inteligentes em videogames não apenas proporcionam desafios interessantes e dinâmicos para os jogadores, mas também desempenham um papel essencial no avanço das técnicas de aprendizado de máquina, planejamento autônomo e interação homem-máquina.

No domínio dos jogos eletrônicos, a IA desempenha um papel estratégico na criação de experiências mais imersivas e desafiadoras. Um componente central desse processo são os bots, ou personagens não jogáveis (NPCs), que interagem com jogadores humanos dentro dos ambientes virtuais. Tais entidades são implementadas como agentes inteligentes, sistemas autônomos que percebem o ambiente e tomam decisões racionais com base em objetivos definidos (WOOLDRIDGE, 2009). Com o avanço das tecnologias de IA, há uma demanda crescente por bots mais realistas, adaptáveis e capazes de competir de forma semelhante a um jogador humano.

Nesse contexto, o mercado global de jogos eletrônicos demonstra a importância econômica e tecnológica desse tema. Segundo relatório da Newzoo (2023), a indústria ultrapassou os 180 bilhões de dólares em receita anual, e grande parte desse crescimento está associada ao uso de IA para melhorar a jogabilidade e o comportamento dos NPCs. Estudos recentes (YANNIS et al., 2022) destacam a necessidade de se desenvolver bots com maior capacidade de adaptação e aprendizado, especialmente em jogos multiplayer, nos quais a previsibilidade de comportamentos compromete a qualidade da experiência do jogador.

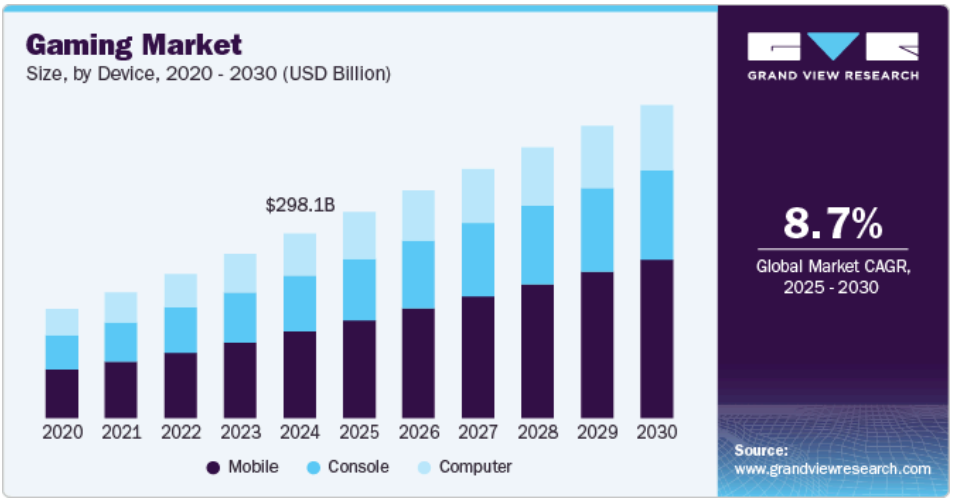

FIGURA 1: CRESCIMENTO DO MERCADO DE JOGOS E A PREVISÃO PARA OS PRÓXIMOS ANOS.
FONTE: GRAND VIEW RESEARCH, 2025

Desta forma, nos últimos anos, o mercado de jogos tem experimentado um crescimento acelerado, com uma taxa de crescimento média de 8.7% ao ano (Grand View Research, 2025), impulsionado pela demanda por experiências mais imersivas e pela busca por inovações tecnológicas que melhorem a interação do jogador com o ambiente digital. Empresas de grande porte, como a Microsoft, a Sony e a Google, têm investido significativamente no desenvolvimento de algoritmos de IA para jogos, não apenas para bots que desafiem os jogadores, mas também para melhorar a experiência geral, com funcionalidades como personalização de comportamentos de NPCs (personagens não jogáveis) e adaptações em tempo real ao estilo de jogo do usuário.

1.1 Problema

O campo do aprendizado por reforço profundo (Deep Reinforcement Learning) tem apresentado resultados expressivos em ambientes controlados de baixa ou média complexidade, tais como jogos 2D ou simulações com entradas de baixa dimensão. No entanto, quando aplicado a ambientes tridimensionais, com observações visuais complexas, alta variabilidade e necessidade de decisões em tempo real, persistem desafios significativos. Conforme observado em pesquisas recentes, tais ambientes impõem problemas como a alta dimensionalidade do espaço de estados, recompensas esparsas, observabilidade parcial e eficiência amostral reduzida (Yang et al., 2021).

Particularmente, ambientes de jogo em primeira pessoa, onde o agente recebe entradas de pixels em primeira pessoa e deve navegar, mirar e agir enquanto adversários ou obstáculos se movem, representam uma barreira para as abordagens convencionais

de aprendizado por reforço profundo. Por exemplo, a extração automática de características relevantes a partir de imagens em 3D, a necessidade de explorar de forma eficiente e a estabilização do aprendizado em face de transições rápidas e imprevisíveis são ainda temas de investigação (Barron et al., 2016).

Além disso, existe uma distinção crítica entre algoritmos *value-based* (como o *DQN*) e *policy-based* ou híbridos, como o PPO, quanto à sua aptidão para lidar com tais cenários: métodos *value-based* frequentemente enfrentam instabilidade ou exigem grande número de interações para convergir, enquanto métodos *policy-based* podem ter melhor robustez, mas ainda demandam afinamento de hiperparâmetros e podem sofrer com recompensas esparsas (Han et al., 2023).

Diante desse panorama, o problema que este trabalho se propõe a resolver é:

“Como projetar e implementar um agente de deep learning machine capaz de aprender políticas autônomas eficientes em ambientes tridimensionais visuais complexos especificamente no jogo *DOOM* via *ViZDoom*, comparando o desempenho dos algoritmos *DQN* e *PPO*, de modo a investigar suas vantagens, limitações e sensibilidade às definições de recompensa e estrutura de entrada?”.

Ao responder essa questão, esta pesquisa pretende contribuir para o avanço prático e teórico de aprendizado por reforço profundo em ambientes visuais tridimensionais, oferecendo uma pesquisa e comparação dos dois algoritmos, *DQN* e *PPO*.

1.2 Justificativa

O estudo e desenvolvimento de agentes inteligentes baseados em aprendizado por reforço profundo têm se consolidado como uma das áreas mais promissoras da inteligência artificial contemporânea, especialmente devido à sua capacidade de aprender comportamentos complexos diretamente a partir de percepções sensoriais, sem a necessidade de supervisão explícita (Sutton & Barto, 2018). Desde os primeiros resultados de Mnih et al. (2015) com o *Deep Q-Network (DQN)*, capazes de superar o desempenho humano em jogos do Atari, a comunidade científica passou a explorar novas arquiteturas e algoritmos para lidar com ambientes mais realistas e desafiadoras.

Entretanto, à medida que os ambientes se tornam tridimensionais, dinâmicos e visualmente ricos como ocorre em jogos de tiro em primeira pessoa (*FPS*), surgem novas

dificuldades relacionadas à alta dimensionalidade do espaço de estados, observabilidade parcial, esparsidade de recompensas e necessidade de decisões em tempo real (Kempka et al., 2016; Zhang et al., 2020). Essas características tornam o processo de aprendizado mais instável e exigente computacionalmente, exigindo técnicas mais avançadas de controle, otimização e generalização.

Neste contexto, a plataforma *ViZDoom* destaca-se como um ambiente experimental amplamente utilizado na pesquisa em *DRL*, por oferecer simulações tridimensionais controladas e de alta fidelidade visual, além de permitir a integração direta com bibliotecas modernas de aprendizado de máquina (Kempka et al., 2016). A utilização do *ViZDoom* possibilita a criação de cenários com diferentes níveis de dificuldade, fornecendo uma base robusta para avaliar o desempenho e a adaptabilidade de diferentes algoritmos.

A escolha dos algoritmos *Deep Q-Network (DQN)* e *Proximal Policy Optimization (PPO)* é justificada por representarem duas abordagens complementares dentro do aprendizado por reforço profundo. O *DQN*, proposto por Mnih et al. (2015) e aprimorado por Wang et al. (2016), é uma técnica *value-based* que combina aprendizado por reforço com redes neurais convolucionais, enquanto o PPO, introduzido por Schulman et al. (2017), é um método *policy-based* que aprimora a estabilidade de aprendizado por meio da otimização proximal de políticas. A comparação entre essas abordagens, em um mesmo ambiente tridimensional, permite avaliar diferenças quanto à eficiência amostral, estabilidade e desempenho em tarefas de navegação, combate e sobrevivência.

Além disso, a investigação proposta contribui para o entendimento de como fatores como arquitetura de rede, função de recompensa e complexidade do ambiente influenciam o aprendizado e o comportamento do agente. Estudos recentes, como os de Andrychowicz et al. (2020) e Han et al. (2023), apontam que a escolha adequada de funções de recompensa e estratégias de exploração é determinante para o sucesso do treinamento em ambientes visuais complexos. Do ponto de vista tecnológico, este trabalho possui relevância prática ao contribuir para o avanço de sistemas autônomos de decisão, com potencial aplicação em áreas como robótica, simulação militar, veículos autônomos e sistemas de navegação inteligente (Kiran et al., 2021). Já do ponto de vista científico, ele reforça a importância de experimentos comparativos e reprodutíveis em DRL, utilizando ambientes abertos e bibliotecas consolidadas como PyTorch e Stable Baselines3, o que fortalece o ecossistema de pesquisa e a transparência metodológica.

1.3 Objetivo geral

Desenvolver, implementar e avaliar experimentalmente um agente de aprendizado por reforço profundo capaz de atuar de forma autônoma em ambientes tridimensionais parcialmente observáveis, comparando o desempenho de duas abordagens consagradas a arquitetura *Dueling Deep Q-Network* (Dueling DQN) e o algoritmo *Proximal Policy Optimization* (PPO) no contexto do jogo *DOOM*, utilizando a plataforma *ViZDoom* como ambiente de simulação. O objetivo geral inclui não apenas a construção dos agentes, mas também a caracterização rigorosa de sua eficiência, estabilidade e capacidade de generalização frente a cenários com distintas estruturas de recompensa e desafios perceptivos.

Para tornar esse objetivo mensurável e operacionalizável, busca-se: (I) implementar agentes baseados em *DQN* e em *PPO* com política convolucional (*Cnn Policy*) e treinar cada agente em três cenários representativos do *ViZDoom (BASIC, DEFEND THE CENTER e DEADLY CORRIDOR*); (II) coletar e analisar um conjunto de métricas quantitativas e recompensas médias por episódio, tempo médio de sobrevivência, número de eliminações (kills), precisão de disparos (*hits/shots*), taxa de convergência e variância entre execuções com repetição de execuções (múltiplas *seeds*) para estimar robustez estatística; (III) investigar a influência do *reward shaping* e do pré-processamento visual (por exemplo, normalização de pixels e *frame stacking*) sobre a velocidade de aprendizado e a qualidade das políticas aprendidas; e (IV) aplicar procedimentos estatísticos apropriados (teste de normalidade, testes paramétricos ou não paramétricos e medidas de tamanho de efeito) para determinar diferenças significativas entre os algoritmos estudados.

Espera-se, com essa proposição, produzir evidências empíricas e análises detalhadas que permitam responder perguntas centrais sobre a aplicabilidade prática de Dueling DQN versus PPO em ambientes 3D visuais, tais como: qual método convergirá mais rapidamente sob recompensas densas ou esparsas; qual método exibirá menor variabilidade entre *runs*; e como a definição de recompensas (componentes e magnitudes) altera a estratégia aprendida pelo agente. O trabalho também visa gerar artefatos reprodutíveis código fonte, wrappers de recompensa, notebooks de treinamento e logs do TensorBoard de modo a favorecer a replicação dos experimentos por outros pesquisadores.

Por fim, o objetivo geral contempla uma dimensão avaliativa e interpretativa: além de reportar métricas de desempenho, o estudo pretende analisar qualitativamente os comportamentos emergentes e discutir as implicações desses comportamentos para aplicações práticas em áreas relacionadas, como robótica autônoma, simulações estratégicas e agentes de navegação que dependem exclusivamente de percepção visual (Mnih et al., 2015; Schulman et al., 2017; Kempka et al., 2016; Sutton & Barto, 2018). Dessa forma, o objetivo geral configura-se como a integração entre desenvolvimento técnico, avaliação experimental rigorosa e reflexão crítica sobre as limitações e potenciais aplicações das técnicas estudadas.

1.4 Objetivo Específico

- Investigar os fundamentos teóricos da Inteligência Artificial aplicados à construção de agentes inteligentes em ambientes dinâmicos, com ênfase em aprendizado por reforço profundo, redes neurais e tomada de decisão autônoma;

- Estudar o funcionamento da plataforma *ViZDoom* como ambiente de simulação para agentes inteligentes, analisando sua arquitetura, API e possibilidades de integração com frameworks de IA como *PyTorch*;

- Selecionar e implementar algoritmos de aprendizado por reforço profundo, como *Deep Q-Network (DQN)* e suas variações (por exemplo, *Dueling DQN*), com o objetivo de possibilitar o aprendizado contínuo e a adaptação do agente a diferentes situações do jogo;

- Desenvolver um agente inteligente que percebe o ambiente de jogo visualmente, por meio de processamento de imagens (frames do jogo) e extração de características relevantes utilizando os algoritmos *PPO (Proximal Policy Optimization)* e *DQN (Deep Q-Network)*.

- Avaliar o desempenho dos algoritmos *DQN* e *PPO* com base em métricas como taxa de acerto, tempo de sobrevivência, número de inimigos eliminados e capacidade de adaptação em cenários com diferentes níveis de dificuldade;

- Identificar limitações, desafios técnicos e oportunidades de melhoria no processo de construção de agentes adaptativos em jogos eletrônicos, propondo sugestões para pesquisas futuras e aplicações práticas.

Desta forma, este artigo visa explorar o processo de desenvolvimento e pesquisa de um agente inteligente destinado a *bots* (robôs) de videogame, com foco na implementação de algoritmos e técnicas de IA que possibilitem a criação de um agente inteligente adaptável ao ambiente que está inserido.

Serão abordados os conceitos fundamentais relacionados ao design e treinamento desses agentes, bem como as abordagens que permitem a aprendizagem de estratégias e comportamentos eficientes, alinhados ao dinamismo dos ambientes virtuais. A partir dessa análise, pretende-se demonstrar e estudar como a evolução dos agentes inteligentes interage em um ambiente controlado.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Historiografia da Inteligência Artificial

Desde tempos antigos, o ser humano sempre procurou facilitar e automatizar atividades consideradas tediosas ou perigosas, buscando criar ferramentas que poupassem esforço manual e aumentassem a eficiência no trabalho cotidiano, segundo o projeto *Omaha in the Anthropocene*, na Mesopotâmia, uma das primeiras civilizações organizadas, surgiu a domesticação e o uso de animais de carga, como bois e asnos, para executar tarefas pesadas, permitindo a irrigação de campos e o transporte de materiais sem depender exclusivamente da força humana, bois e asnos eram componentes centrais na economia agrícola, reduzindo significativamente o tempo e o esforço necessários para a produção de alimentos. Essa externalização do trabalho pesado para seres não humanos representou um marco na história da automação primitiva, inaugurando uma tradição de delegar atividades repetitivas e desinteressantes a sistemas mecânicos ou biológicos. Com o decorrer dos séculos, o conceito de automação evoluiu de práticas rudimentares para mecanismos mais sofisticados, como os moinhos na Idade Média, se aproveitando do movimento da água ou do vento, ajudavam na produção de farinha, moendo, triturando e fragmentando grãos, facilitando em muito a vida dos camponeses (Langdon, 1991), a continuidade dessa busca por simplificação, acontece também, já na Revolução Industrial onde trouxe uma nova etapa, com as máquinas a vapor e teares automáticos que transformaram as fibras têxteis em tecidos sem intervenção manual direta, acelerando a produção em larga escala.

A partir da metade do século XX testemunhou o surgimento de modelos computacionais inspirados no funcionamento do cérebro humano. Em 1943, Warren McCulloch e Walter Pitts (1943) propuseram o primeiro modelo matemático de um

neurônio artificial, estabelecendo as bases para as redes neurais artificiais. Esse modelo, conhecido como neurônio de McCulloch-Pitts, simplificava o comportamento dos neurônios biológicos, permitindo a simulação de processos cognitivos básicos. Posteriormente, em 1958, Frank Rosenblatt (1958) desenvolveu o perceptron, uma rede neural capaz de aprender e reconhecer padrões, marcando um avanço significativo na inteligência artificial. Apesar das limitações iniciais, como a incapacidade de resolver problemas não linearmente separáveis, esses modelos pioneiros abriram caminho para pesquisas mais avançadas.

Já na década de 80, o campo das redes neurais artificiais foi revitalizado com a introdução do algoritmo de retropropagação por David Rumelhart, Geoffrey Hinton e Ronald Williams (1986). Esse algoritmo permitiu o treinamento eficiente de redes neurais multicamadas, superando as limitações dos modelos anteriores. Com o aumento do poder computacional e o desenvolvimento de novas técnicas, como os *ReLU* (*Rectified Linear Unit* ou unidades lineares retificadas) e a *Batch Normalization* (normalização em lotes), as redes neurais profundas tornaram-se ferramentas poderosas para tarefas complexas, incluindo reconhecimento de voz, processamento de linguagem natural e visão computacional. Esses avanços culminaram na era do aprendizado profundo (deep learning), onde modelos como as *CNNs* (redes neurais convolucionais) e as *RNNs* (redes neurais recorrentes) alcançaram desempenhos superiores em diversas aplicações.

2.1 Machine learning

De acordo com Rauber (2005), o *machine learning* (aprendizado de máquina) é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos capazes de aprender padrões a partir de dados, permitindo que sistemas computacionais realizem tarefas sem serem explicitamente programados para cada uma delas. Essa abordagem difere da programação tradicional, onde as regras são definidas manualmente, em vez disso, os modelos de *machine learning* ajustam seus parâmetros com base em dados de entrada e saída, aprimorando seu desempenho ao longo do tempo. Essa capacidade de adaptação torna o Machine Learning particularmente eficaz em ambientes dinâmicos e complexos, onde as variáveis podem mudar rapidamente e as soluções não são facilmente codificáveis.

Existem três principais paradigmas de aprendizado de máquina: supervisionado, não supervisionado e por reforço.

2.1.1 Aprendizado supervisionado

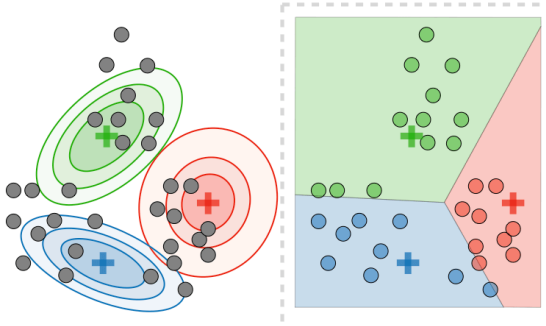

FIGURA 2: EXEMPLIFICAÇÃO DO APRENDIZADO SUPERVISIONADO.
FONTE: STANFORD, 2025.

No aprendizado supervisionado, o modelo é treinado com um conjunto de dados rotulados, aprendendo a mapear entradas para saídas desejadas, é amplamente utilizado em tarefas como classificação de imagens. Como no exemplo da figura 2 acima, que temos um sistema desorganizado de um grupo que contém círculos, dentro deste sistema o modelo tentará compreender qual figura faz parte de qual grupo (verde, vermelho ou azul), o modelo sabe de antemão as formas dos exemplares, pois já foram rotuladas. (Ludermir, 2021)

2.1.2 Aprendizado não supervisionado

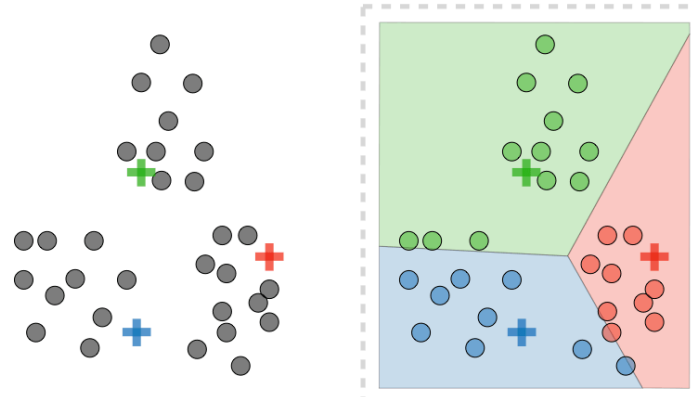

FIGURA 3: EXEMPLIFICAÇÃO DO APRENDIZADO NÃO SUPERVISIONADO.
FONTE: STANFORD, 2025.

O aprendizado não supervisionado, por outro lado, lida com dados não rotulados, buscando identificar estruturas ou padrões ocultos, como agrupamentos ou associações, desta forma, na figura 3 acima o modelo tentará encontrar características em comum em cada uma das amostras, a fim de conseguir catalogar em grupos (azul, verde ou vermelho) cada uma dos círculos. (Ludermir, 2021)

2.1.2 Aprendizado por reforço

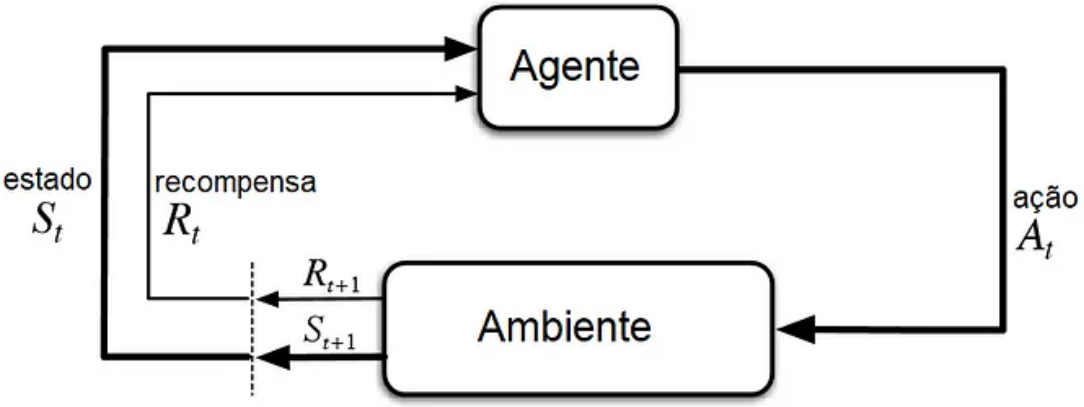

FIGURA 4: EXEMPLIFICAÇÃO DO APRENDIZADO POR REFORÇO.
FONTE: Neves, 2020.

Já o aprendizado por reforço envolve a interação de um agente com um ambiente, onde ele aprende a tomar decisões sequenciais para maximizar uma recompensa cumulativa, desta forma, na figura 4 acima demonstra o ciclo fundamental do aprendizado por reforço, no qual um agente interage com um ambiente ao longo do tempo e se retro-alimenta (*backpropagation*), onde em cada instante t, o agente verifica o estado S_t, e com base nisso executa uma ação A_t, e como resposta o ambiente responde com uma nova situação S_{t+1} e uma recompensa R_{t+1} , com essas novas entradas ele irá reforçar o seu comportamento, a fim de maximizar a sua recompensa. (Ludermir, 2021)

2.1.1.1 Deep Reinforcement learning

De acordo com Li (2017), o *deep reinforcement learning* (Aprendizado por Reforço Profundo) é uma subárea da inteligência artificial que combina o aprendizado por reforço tradicional com redes neurais profundas. Essa abordagem permite que agentes aprendam a tomar decisões em ambientes complexos e de alta dimensionalidade, como jogos de vídeo, robótica e sistemas de recomendação, processando diretamente entradas brutas, como imagens ou sinais sensoriais, sem a necessidade de engenharia manual de características. O *Deep Reinforcement Learning* tem se destacado por sua capacidade de resolver tarefas que anteriormente eram consideradas desafiadoras para máquinas, alcançando desempenhos comparáveis ou superiores aos humanos em diversas aplicações.

A base do *Deep Reinforcement Learning* reside na integração de redes neurais profundas com algoritmos de aprendizado por reforço, permitindo que agentes aprendam políticas de decisão eficazes por meio da interação com o ambiente. Um marco significativo nesse campo foi a introdução do *Deep Q-Network* por Mnih et al. (2013), onde uma rede neural convolucional foi utilizada para estimar a função de valor de ação diretamente a partir de pixels brutos de jogos do *Atari* 2600, alcançando desempenho humano em vários títulos. Esse avanço demonstrou o potencial do *deep reinforcement learning* em aprender representações complexas e tomar decisões eficazes em ambientes com entradas de alta dimensionalidade.

O *Deep Reinforcement Learning* pode ser categorizado em algoritmos baseados em valor, como o *Deep Q-Network*, e algoritmos baseados em política, como o *Proximal Policy Optimizatio*n (PPO) e o T*rust Region Policy Optimization* (TRPO). Algoritmos baseados em valor focam na estimativa da função de valor, enquanto algoritmos baseados em política aprendem diretamente a política de decisão. (Juan, 2025)

2.1.2 Intelligent agent

De acordo com Zhang et al. (2007), o conceito de *intelligent agent* (agente inteligente) é fundamental na área de inteligência artificial (IA), representando sistemas capazes de perceber seu ambiente, tomar decisões autônomas e agir de maneira a alcançar objetivos específicos. Segundo Wooldridge e Jennings (1995), um agente inteligente é caracterizado por propriedades como autonomia, capacidade de aprendizado, reatividade e orientação a objetivos. Essas características permitem que o agente opere de forma independente, adaptando-se a mudanças no ambiente e aprendendo com experiências passadas para melhorar seu desempenho futuro.

A arquitetura de um agente inteligente pode variar conforme sua complexidade e aplicação. Dentre as abordagens mais comuns, destacam-se as arquiteturas reativas, deliberativas e híbridas. As arquiteturas reativas respondem diretamente a estímulos do ambiente, sem a necessidade de representação interna ou planejamento. Já as arquiteturas deliberativas mantêm modelos internos do mundo, permitindo planejamento e tomada de decisões mais complexas. As arquiteturas híbridas combinam elementos das duas anteriores, buscando equilibrar reatividade e deliberação para lidar com ambientes dinâmicos e imprevisíveis. Além disso, agentes inteligentes podem ser classificados com base em suas capacidades cognitivas e comportamentais. Por exemplo, o modelo BDI (*Belief-Desire-Intention*) propõe uma estrutura onde o agente

possui crenças sobre o mundo, desejos que representam seus objetivos e intenções que guiam suas ações para alcançar esses objetivos. Esse modelo tem sido amplamente utilizado no desenvolvimento de agentes que necessitam de raciocínio complexo e tomada de decisões em ambientes incertos. (Fox, 2003; Bonasso, 1995)

A aplicação de agentes inteligentes é vasta, abrangendo desde sistemas de recomendação e assistentes virtuais até robótica autônoma e jogos. No contexto de aprendizado por reforço profundo, agentes inteligentes são treinados para interagir com ambientes complexos, aprendendo políticas de ação que maximizam recompensas cumulativas. Esses agentes são capazes de lidar com tarefas desafiadoras, como jogar videogames em nível humano, controlar robôs em ambientes dinâmicos e otimizar processos em sistemas complexos. (Fox, 2003)

2.2.2 *Backpropagation*

De acordo com Lillicrap et al. (2020), o algoritmo de *backpropagation* (retropropagação) é um método fundamental no treinamento de redes neurais artificiais, especialmente em arquiteturas de múltiplas camadas, como os perceptrons multicamadas. Introduzido por Rumelhart, Hinton e Williams em 1986, o *backpropagation* permite que a rede ajuste seus pesos internos de forma eficiente, minimizando o erro entre a saída prevista e a saída desejada. Esse processo é essencial para que a rede aprenda representações complexas e se generalize bem para novos dados. O funcionamento do *backpropagation* envolve duas fases principais: a *forward pass* (propagação direta) e a *backward pass* (propagação reversa). Na propagação direta, os dados de entrada são processados camada por camada até a obtenção da saída da rede. Em seguida, calcula-se o erro entre a saída obtida e a saída esperada. Na fase de propagação reversa, esse erro é retro-propagado pela rede, ajustando os pesos de cada neurônio com base na contribuição de cada um para o erro total, utilizando o método do gradiente descendente e a regra da cadeia para calcular os gradientes necessários.

Matematicamente, o *backpropagation* calcula o gradiente da função de erro em relação a cada peso da rede, permitindo atualizações precisas que reduzem o erro global. Esse processo iterativo continua até que a rede atinja um nível de erro aceitável ou até que um número máximo de iterações seja alcançado. A eficiência do *backpropagation* tornou possível o treinamento de redes neurais profundas, que são capazes de modelar relações altamente não lineares e complexas nos dados. (Lillicrap et al., 2020)

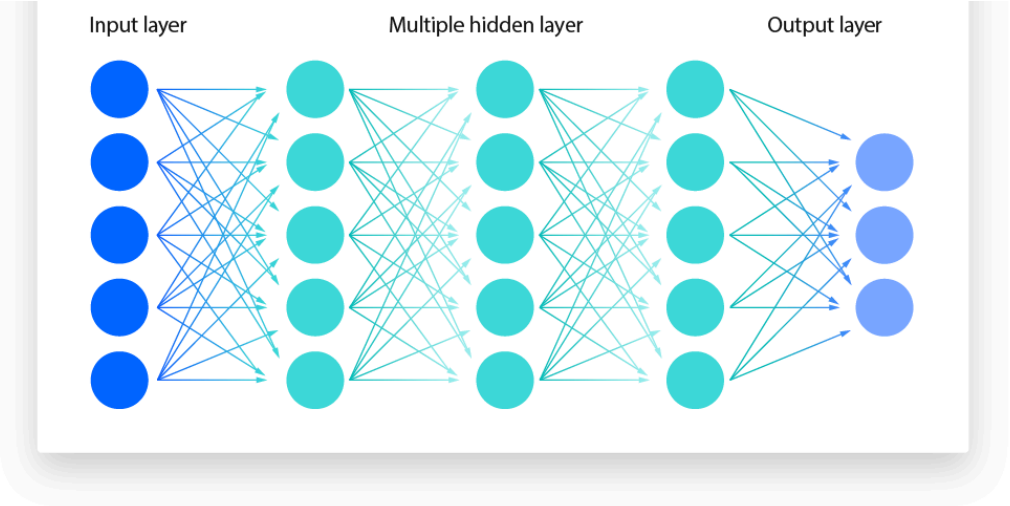

FIGURA 5: Representação visual de uma rede neural básica com três camadas ocultas: camada de entrada, camadas ocultas e camada de saída, respectivamente.

Fonte: IMB, 2025.

As redes neurais artificiais são compostas por camadas organizadas sequencialmente em camada de entrada, camadas ocultas e camada de saída. A primeira camada da rede, responsável por receber os dados brutos do ambiente ou do conjunto de dados. Cada neurônio nesta camada representa uma característica ou atributo específico dos dados de entrada que será utilizada no processamento da próxima camada, a camada oculta é responsável pelo processamento e extração de características dos dados, desta forma, cada neurônio em uma camada oculta recebe entradas ponderadas dos neurônios da camada anterior, aplica uma função de ativação (como *ReLU*, *sigmoid* ou *tanh*) e transmite o resultado para a próxima camada, esta sendo a camada de saída, última camada da rede que gera a resposta final do modelo, ela interpreta as representações apreendidas nas camadas ocultas e produz a previsão ou decisão final. (IBM, 2024)

2.2.3 Rectified Linear Unit

A função de ativação *rectified linear unit* (*ReLU*) é amplamente utilizada em redes neurais profundas devido à sua simplicidade computacional e eficácia na mitigação de problemas como o gradiente desaparecido. Matematicamente, a *ReLU* é definida como $f(x) = max(0, x)$, onde x for maior que 0, o resultado é x caso contrário será zero, o que significa que ela retorna zero para entradas negativas e o próprio valor para entradas

positivas. Essa característica permite que a *ReLU* introduza não linearidades nas redes neurais, essencial para modelar relações complexas nos dados. Além disso, a *ReLU* promove esparsidade nas ativações, pois apenas uma parte dos neurônios é ativada em determinado momento, o que pode levar a modelos mais eficientes e com melhor capacidade de generalização, desta forma a principal vantagem da *ReLU* em relação a funções de ativação anteriores, como a sigmóide ou a tangente hiperbólica, é sua capacidade de reduzir significativamente o problema do gradiente desaparecido. Em funções como a sigmóide, os gradientes podem se tornar extremamente pequenos em regiões de saturação, dificultando o treinamento de redes profundas. Com a *ReLU*, os gradientes permanecem significativos para entradas positivas, facilitando a propagação do erro e acelerando o processo de aprendizado. (He et al., 2015)

2.2.4 Deep Q-Network

O *Deep Q-Network* (DQN) é um algoritmo de aprendizado por reforço profundo que combina o *Q-learning* com redes neurais convolucionais profundas para permitir que agentes aprendam políticas de controle diretamente a partir de entradas sensoriais de alta dimensão, como imagens. Introduzido por Mnih et al. (2013), o *DQN* foi pioneiro ao alcançar desempenho humano em diversos jogos do Atari 2600, utilizando apenas pixels brutos e pontuações como entradas, sem conhecimento prévio sobre as dinâmicas dos jogos . A arquitetura do *DQN* consiste em uma rede neural que estima a função de valor de ação (*Q-function*), mapeando estados para valores de ação, permitindo que o agente selecione ações que maximizem a recompensa esperada. Para estabilizar o treinamento, o DQN introduz duas técnicas fundamentais: o *experience replay* e a *target network*. O *experience replay* armazena transições de experiência (estado, ação, recompensa, próximo estado) em uma memória e a amostra em mini-lotes aleatórios para treinamento, quebrando correlações entre amostras consecutivas e melhorando a eficiência do aprendizado. A *target network* é uma cópia periódica da rede principal usada para calcular os alvos de aprendizado, reduzindo oscilações e instabilidades durante a atualização dos pesos .

2.2.5 Convolutional Neural Network

As redes neurais convolucionais (*Convolutional Neural Network ou CNNs*) constituem uma classe de modelos de aprendizado profundo introduzida originalmente para o reconhecimento de padrões bidimensionais. Estas redes utilizam camadas convolucionais caracterizadas por conexões locais e compartilhamento de pesos, o que

significa que filtros treináveis são aplicados a pequenas regiões da imagem e replicados em diferentes posições. Além disso, camadas de *pooling* (subamostragem) são empregadas para reduzir gradualmente a resolução espacial e introduzir invariância a pequenas translações. Esses mecanismos permitem que as *CNNs* extraiam hierarquicamente características visuais (por exemplo, bordas e texturas) ao longo de múltiplas camadas, o que reduz substancialmente o número de parâmetros em comparação a redes densamente conectadas. Essa organização explora a estrutura local das imagens, assumindo que propriedades estatísticas sejam estacionárias e que pixels próximos apresentem correlações significativas. De fato, comparadas a redes totalmente conectadas de dimensão semelhante, as *CNNs* apresentam muito menos conexões e parâmetros, o que facilita o treinamento. A combinação desses elementos torna as *CNNs* particularmente eficientes na tarefa de reconhecimento de imagens com relativamente poucos parâmetros. (Krizhevsky et al., 2012)

2.2.6 VizDOOM

O ViZDoom é uma plataforma de código aberto baseada no jogo clássico *Doom*, desenvolvida para facilitar a pesquisa em aprendizado por reforço visual que utilizaremos ao decorrer deste artigo para produzir o agente inteligente. O *VizDoom* diferentemente de ambientes tradicionais como o Atari, que oferecem perspectivas bidimensionais e informações estruturadas, o *ViZDoom* proporciona uma perspectiva em primeira pessoa em um ambiente 3D semi-realista. Isso permite que agentes de inteligência artificial aprendam a partir de dados visuais brutos, como pixels da tela, simulando tarefas complexas do mundo real, como navegação, reconhecimento de objetos e tomada de decisões em tempo real. A plataforma é altamente configurável, permitindo a criação de cenários personalizados por meio de editores visuais e linguagens de script, e oferece suporte a múltiplas plataformas, incluindo Linux, macOS e Windows, com APIs para Python, C++ e integração com o *OpenAI Gym*. (Kempka et al., 2016)

2.3 Trabalhos correlatos

Com o avanço da computação e das técnicas de aprendizado por *Deep Reinforcement Learning* (Reforço profundo), tornou-se possível treinar agentes inteligentes capazes de adquirir comportamentos complexos em ambientes visuais desafiadores. No trabalho de pesquisa de Mnih et al. (2013) marcado pelo desenvolvimento de agentes inteligentes capazes de aprender a jogar jogos do Atari marca um ponto de inflexão na pesquisa em inteligência artificial, particularmente no

campo do aprendizado por *Deep Reinforcement Learning*. A ideia central por trás desse avanço consistia em integrar redes neurais profundas com métodos clássicos de aprendizado por reforço para permitir que agentes aprendessem diretamente a partir de entradas sensoriais visuais, como pixels da tela. Essa abordagem permitiu um agente treinado para jogar uma variedade de jogos do Atari 2600 sem qualquer conhecimento prévio das regras ou estratégias. O agente usava apenas imagens dos jogos e os sinais de recompensa como entradas, tornando-se o primeiro modelo a alcançar desempenho comparável ao humano em diversos jogos por meio de aprendizado computacional.

4. DESENVOLVIMENTO

4.1 Visão Geral

O presente trabalho tem como propósito principal o desenvolvimento e a avaliação de um agente de aprendizado por reforço profundo capaz de interagir de forma autônoma com o ambiente tridimensional *ViZDoom*, aprendendo estratégias de combate em primeira pessoa. O objetivo central é comparar o desempenho entre dois algoritmos consagrados na literatura de *DRL*, o *Deep Q-Network* e o *Proximal Policy Optimization*, analisando suas diferenças em termos de estabilidade, eficiência de aprendizado e capacidade de adaptação em ambientes com recompensas parciais e observações visuais complexas.

A implementação foi conduzida de forma experimental, seguindo as boas práticas de reprodutibilidade e documentação científica. Todo o código foi desenvolvido na linguagem Python 3.12.3, utilizando bibliotecas especializadas em aprendizado profundo e controle de agentes autônomos. O processo de desenvolvimento envolveu a criação de uma interface entre os algoritmos e o ambiente *ViZDoom*, com o uso de *wrappers* compatíveis com o *OpenAI Gym*, o que possibilitou padronizar a comunicação entre o agente e o ambiente de simulação.

O desenvolvimento foi dividido em etapas sequenciais: configuração do ambiente computacional, definição das funções de recompensa, implementação dos modelos de aprendizado, treinamento dos agentes e coleta dos resultados. As métricas de desempenho como recompensa média, tempo de sobrevivência, número de eliminações e precisão de disparos foram registradas ao longo do processo de aprendizagem e posteriormente analisadas com o auxílio da ferramenta *TensorBoard*, permitindo o monitoramento contínuo da evolução do treinamento.

De forma geral, a estrutura do experimento foi planejada para que as condições de treinamento entre os algoritmos fossem equivalentes, garantindo uma comparação justa entre o *DQN* e o *PPO*. Nos capítulos seguintes, cada uma das etapas, tecnologias e metodologias empregadas será detalhada, com ênfase na arquitetura dos algoritmos, nos cenários utilizados e na análise dos resultados obtidos.

4.1.1 Ambiente

O ambiente computacional desenvolvido para esta pesquisa integra um conjunto de ferramentas de código aberto amplamente reconhecidas na área de inteligência artificial. O núcleo da implementação foi construído em Python 3.12.3, devido à sua versatilidade e ampla disponibilidade de bibliotecas voltadas ao aprendizado de máquina.

A principal biblioteca utilizada foi o *PyTorch*, uma estrutura de aprendizado profundo que fornece suporte eficiente para o cálculo de gradientes e treinamento de redes neurais convolucionais (Paszke et al., 2019). Para a implementação dos algoritmos de aprendizado por reforço, foi utilizada a biblioteca *Stable-Baselines3* (Raffin et al., 2021), que disponibiliza implementações confiáveis e bem testadas de diversos algoritmos de *DRL*, incluindo o *DQN* e o *PPO*. Essa biblioteca também oferece integração direta com o OpenAI Gym, o que permitiu o uso de *wrappers* padronizados para *comunicação* entre os agentes e o ambiente VizDoom.

O ViZDoom foi escolhido como plataforma de simulação por se tratar de um ambiente tridimensional projetado especificamente para pesquisa em Aprendizado por Reforço Visual, permitindo que o agente aprenda diretamente a partir de pixels, de forma semelhante à percepção humana (Kempka et al., 2016). O VizDoom fornece diferentes cenários predefinidos, como *BASIC, DEADLY CORRIDOR* e *DEFEND THE CENTER*, cada um com níveis crescentes de dificuldade e objetivos distintos. Esses cenários oferecem desafios variados que envolvem reconhecimento visual, coordenação motora e planejamento de ações, o que torna o ambiente ideal para a avaliação de algoritmos de *DRL*.

Além disso, o *Jupyter Notebook* foi utilizado como ambiente de desenvolvimento, devido à sua capacidade de integrar código, visualizações e anotações em um único documento, facilitando o processo de experimentação e documentação científica. Para a análise e monitoramento do desempenho dos agentes durante o treinamento, foi utilizado o *TensorBoard*, uma ferramenta de visualização que permite acompanhar métricas como recompensas, perdas e evolução temporal do aprendizado.

O sistema operacional utilizado foi o Linux Mint 22.1 Xia, escolhido por sua estabilidade, desempenho e compatibilidade com bibliotecas de aprendizado profundo baseadas em *CUDA*. A combinação dessas ferramentas resultou em um ambiente de desenvolvimento robusto, escalável e adequado para a condução de experimentos de aprendizado profundo com uso intensivo de GPU.

4.1.2 Hardware

Os experimentos realizados neste trabalho foram conduzidos em um computador pessoal de alto desempenho, configurado para suportar tarefas de treinamento de redes neurais profundas e execução de simulações complexas. A máquina utilizada possui uma placa de vídeo NVIDIA GeForce GTX 1650, com 4GB de memória VRAM, responsável pelo processamento paralelo das operações matriciais intensivas envolvidas no treinamento das redes neurais convolucionais.

O sistema conta também com um processador Intel Core i5-9300H, com quatro núcleos físicos e oito threads, operando a uma frequência base de 2,4 GHz e podendo atingir até 4,1 GHz em modo *turbo boost*. Essa configuração possibilitou o processamento eficiente de múltiplos threads simultâneos durante o treinamento e a coleta de métricas em tempo real. O computador dispõe de 16 GB de memória RAM DDR4, garantindo capacidade suficiente para armazenar lotes de dados (*batches*), buffers de experiência e parâmetros de rede durante a execução dos algoritmos.

O armazenamento interno é composto por um SSD de 128 GB e o uso de armazenamento sólido foi essencial para evitar gargalos no salvamento contínuo de logs e pesos da rede, que ocorrem durante o treinamento prolongado dos agentes.

A utilização de uma GPU dedicada, associada a uma arquitetura de hardware otimizada para operações paralelas, proporcionou ganhos expressivos no tempo de convergência dos algoritmos e na estabilidade dos experimentos. Conforme apontado por Krizhevsky et al. (2012).

4.2 Algoritmos

Os algoritmos de aprendizagem por reforço profundo empregados neste trabalho combinam princípios clássicos de tomada de decisão sequencial com a capacidade de representação das redes neurais profundas. Em termos gerais, um agente de *DRL* observa um estado do ambiente (no caso, quadros visuais do ViZDoom), executa uma

ação segundo uma política (determinística ou estocástica), recebe uma recompensa e transita para um novo estado; esse ciclo é formalizado pelo modelo teórico dos Processos de Decisão de Markov (*MDP*) e orientado pela Hipótese da Recompensa (Sutton & Barto, 2018). Na prática, políticas e funções-valor são parametrizadas por redes neurais convolucionais que transformam pixels em vetores de características úteis para a tomada de decisão (Krizhevsky et al., 2012). O aprendizado procede via otimização iterativa: os parâmetros da rede são ajustados por algoritmos de gradiente para reduzir uma função de perda que reflete o erro temporal-diferença (TD) ou uma função-objetivo de política. Em experimentos com ambientes visuais, é comum empregar técnicas auxiliares que melhoram a estabilidade e a velocidade de convergência, tais como normalização em lote (*Batch Normalization*), funções de ativação *ReLU* e técnicas de regularização; além disso, práticas de engenharia experimental pré-processamento de frames, empilhamento temporal (frame stacking), clipping de recompensas e wrappers padronizados garantem comparabilidade entre métodos e reprodutibilidade (Ioffe & Szegedy, 2015; Nair & Hinton, 2010).

4.2.1 Algoritmo DQN

O *Deep Q-Network (DQN)* é um método *value-based* que aproxima a função-valor $Q\,(s,\,a)$ por meio de uma rede neural profunda, tipicamente convolucional quando a entrada consiste em imagens (Mnih et al., 2013; Mnih et al., 2015). No DQN, o agente seleciona ações segundo uma política ε-greedy (alternando entre exploração aleatória e exploração pela estimativa atual de Q), e a atualização de parâmetros é feita minimizando-se a discrepância entre o valor estimado $Q(s,\,a;\,\theta)$ e o alvo temporal-diferença $y = r + \gamma max a' Q(s', a'; \theta\, -)$, onde θ^{-} representa os pesos da *target network*, uma cópia amortecida da rede principal que fornece estabilidade para os alvos (Lin, 1992; Mnih et al., 2015). Outro elemento crítico do *DQN* é o *experience replay*: transições (s, a, r, s') são armazenadas em um buffer e amostradas aleatoriamente para compor mini batches de treinamento, quebrando correlações temporais e melhorando a eficiência amostral (Lin, 1992).

```
[14]: env = VizDoomGym(render=True)

[15]: state = env.reset()

[16]: env.step(2) #  step(action)
```

FIGURA 6: Recorte para exemplificação.
Fonte: Autor, 2025.

Na interação com o *ViZDoom*, o *DQN* recebe quadros (frames) renderizados em primeira pessoa; esses frames são pré-processados (ex.: redimensionamento, conversão para escala de cinza, normalização e empilhamento de N frames) para formar o estado de entrada, reduzindo ruído visual e incorporando informação temporal. O espaço de ações é discretizado (movimentação básica, rotação e disparo), de modo compatível com a política *CnnPolicy* empregada. A cada passo de ambiente (env.step(action)), o wrapper personalizado computa a recompensa composta segundo a formulação por cenário (kills, tiros, dano, distância ao objetivo etc.) e a armazena junto ao restante da transição no replay buffer. Periodicamente (por exemplo, a cada train_freq passos), o agente executa atualizações: amostra-se um minibatch do buffer, calcula-se a perda MSE entre Q e y e aplica-se retropropagação para ajustar θ; a *target network* θ^{-} é atualizada a intervalos fixos para manter estabilidade (Mnih et al., 2013). Em cenários com recompensas esparsas ou com dinamismo elevado (como *DEADLY CORRIDOR*), o *DQN* pode apresentar maior sensibilidade ao esquema de exploração (tuning de ε) e requerer um buffer e um número substancial de steps para convergir de forma robusta.

4.2.2 Algoritmo PPO

O algoritmo *PPO (Proximal Policy Optimization)*, este método pertence à família de *policy gradient* e atua diretamente sobre a política parametrizada $\pi\theta(a|s)$. O *PPO* busca otimizar uma função objetivo substituta que penaliza mudanças bruscas na política, tipicamente por meio de um termo *clipped* que limita a razão de probabilidade $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$, levando ao objetivo:

$$L^{CLIP}(\theta) = E_T\left[min\left(r_t(\theta)\hat{A}_t,\ clip\left(rt_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$$

Onde $\hat{A}_t$ é a estimativa de vantagem, frequentemente obtida via *Generalized Advantage Estimation*, e ϵ é um hiperparâmetro que controla o grau de confiança nas atualizações (Schulman et al., 2017). Diferentemente do *DQN*, o *PPO* é um método on-policy: os dados usados para atualizar a política são coletados pela própria política atual durante rollouts de (n_steps) no ambiente; em seguida, realiza-se várias épocas de otimização sobre esses dados, divididos em minibatches, o que aumenta a eficiência amostral sem comprometer a estabilidade.

No contexto do ViZDoom, o PPO interage com o ambiente coletando sequências de transições executadas por π_θ durante um período (por exemplo, n_steps = 2048 ou

4096), computando retornos e vantagens a partir das recompensas fornecidas pelo wrapper de cenário e, então, atualizando os parâmetros da política e da rede de valor. A política é representada por uma *CnnPolicy* que mapeia frames empilhados para distribuições de probabilidade sobre ações discretas; simultaneamente, uma cabeça de valor estima $V(s)$ para calcular $\hat{A}_t$. A natureza on-policy do *PPO* torna-o menos dependente de um grande replay buffer e mais robusto a alterações nas dinâmicas do ambiente; além disso, suas atualizações controladas e a possibilidade de incluir termos de entropia favorecem a exploração contínua durante o treinamento (Schulman et al., 2017). Em ambientes parcialmente observáveis e repletos de ruído visual, como *ViZDoom*, o *PPO* costuma apresentar comportamento de aprendizado mais suave e menos oscilatório que algoritmos value-based puros, embora possa demandar cuidados quanto às escalas de recompensa e ao dimensionamento de n_steps e do número de épocas por batch.

Em ambos os algoritmos, práticas auxiliares foram empregadas: checkpoints regulares, logging detalhado e monitoramento via *TensorBoard*. As diferenças observadas entre *DQN* e *PPO* devem ser interpretadas à luz das suas propriedades algorítmicas, off-policy com *replay/target* no primeiro e *on-policy* com atualização proximal no segundo, e do modo como cada um explora o espaço de estados e responde a sinais de recompensa densos ou esparsos no *ViZDoom* (Mnih et al., 2013; Schulman et al., 2017; Kempka et al., 2016).

4.3 Desenvolvimento do Agente Inteligente

O desenvolvimento do agente inteligente consistiu na implementação e treinamento de dois modelos de aprendizado por reforço profundo, DQN e PPO aplicados aos cenários “*BASIC*”, “*DEADLY CORRIDOR*” e “*DEFEND THE CENTER*” da plataforma *ViZDoom* (Kempka et al., 2016). Cada modelo foi configurado e treinado com hiperparâmetros específicos, utilizando o *framework Stable-Baselines3* em Python 3.12.3, com suporte do *PyTorch* para operações de rede neural e o *TensorBoard* para monitoramento dos resultados.

O agente foi projetado para observar o ambiente em tempo real e aprender a maximizar sua recompensa cumulativa a partir da interação com o ambiente, conforme os princípios da Hipótese da Recompensa (*Reward Hypothesis*) (Sutton & Barto, 2018). Assim, o comportamento do agente não é programado de forma explícita, mas emergente, ele descobre políticas eficazes através da repetição de episódios e do ajuste

contínuo dos parâmetros da rede neural. Cada episódio representa uma execução completa do cenário, iniciando com o agente em um estado inicial e terminando com a morte ou término da fase. Ao longo do processo, o agente coleta pares de experiência $(s_t, a_t, r_t, s_t + 1)$, onde s_t representa o estado atual, no caso a imagem do jogo, a_t a ação escolhida, r_t a recompensa recebida e s_{t+1} o próximo estado.

A função de recompensa varia de acordo com o cenário, moldando o comportamento esperado do agente. No cenário *BASIC*, o agente recebe +106 pontos por eliminar o inimigo, -5 por cada disparo e +1 por cada tic vivo, incentivando precisão e sobrevivência. No *DEADLY CORRIDOR*, as recompensas são mais complexas: o agente recebe recompensas positivas pela proximidade ao colete e pelos abates, e penalidades pela morte, tiros desperdiçados e danos sofridos, tornando o aprendizado mais desafiador e estratégico. Já em *DEFEND THE CENTER*, a recompensa é simplificada +1 por abate e -1 por morte destacando a importância de eficiência ofensiva. Essa modelagem de recompensas desempenha papel fundamental na forma como o agente explora o ambiente e ajusta suas políticas, determinando, em última instância, a natureza do comportamento aprendido (Ng et al., 1999; Mnih et al., 2015).

A estrutura da rede neural adotada é baseada em redes neurais convolucionais (*CNNs*), amplamente utilizadas em ambientes visuais por sua capacidade de extrair características espaciais e temporais relevantes a partir de imagens (Krizhevsky et al., 2012). Cada camada convolucional atua na identificação de padrões como bordas, texturas e formas, que são progressivamente transformados em representações abstratas do ambiente. Essas representações são, então, processadas por camadas totalmente conectadas (*fully connected layers*), que produzem a estimativa da função-valor $Q(s, a)$ no caso do DQN ou os parâmetros da política $\pi(a|s)$ e do valor $V(S)$, no caso do *PPO*. Para melhorar a estabilidade e a eficiência do treinamento, foram aplicadas técnicas de normalização em lote (Ioffe & Szegedy, 2015) e funções de ativação ReLU (Nair & Hinton, 2010), que favorecem o aprendizado não linear e mitigam o problema do gradiente desaparecente.

O treinamento foi realizado ao longo de centenas de milhares de *timesteps* 100.000 para o cenário *BASIC* e *DEFEND THE CENTER*, e 500.000 para o cenário *DEADLY CORRIDOR*, dada sua maior complexidade. Durante esse processo, o agente alternou entre fases de exploração (tentativa de novas ações) e exploração dirigida (execução de ações já aprendidas como eficazes), balanceadas por mecanismos

específicos de cada algoritmo, a política *ε-greedy* no *DQN* e o clipping de política no *PPO* (Schulman et al., 2017). O progresso foi acompanhado por métricas registradas no *TensorBoard*, incluindo recompensa média, perda da rede, tempo de episódio e taxa de sucesso, permitindo uma análise visual e quantitativa da convergência dos modelos.

4.3.1 Pré-processamento das Imagens

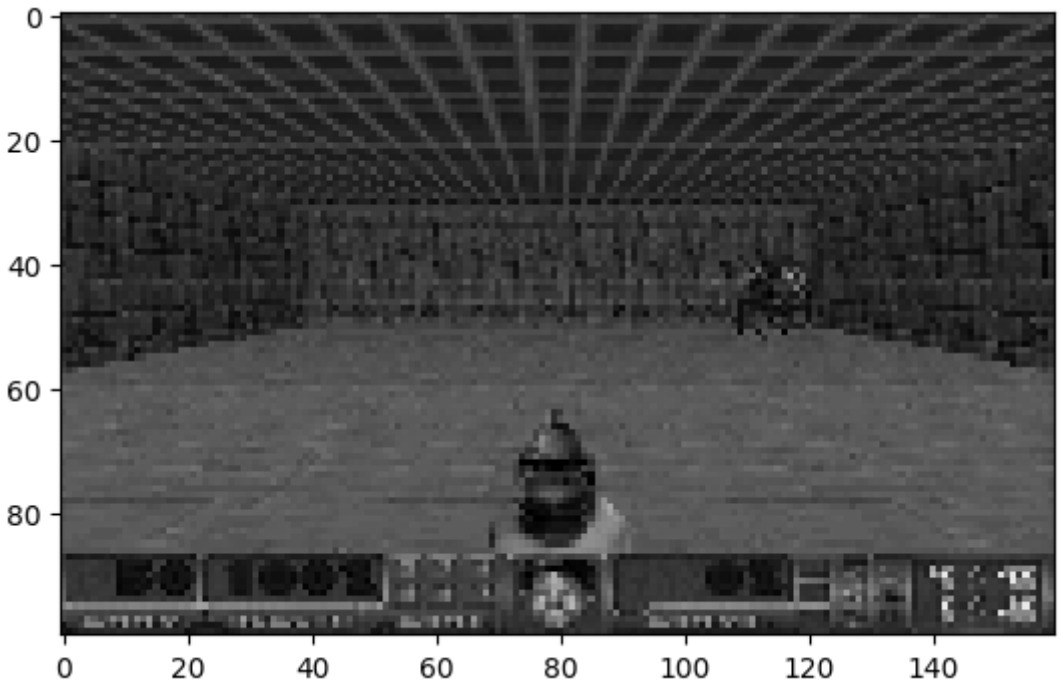

FIGURA 7: Pré-processamento das imagens.
Fonte: Autor, 2025.

O pré-processamento das imagens desempenha papel essencial no desempenho dos agentes em ambientes visuais complexos como o *ViZDoom*. Cada estado do ambiente é representado por uma imagem (frame) capturada diretamente da visão em primeira pessoa do jogo. Entretanto, utilizar essas imagens brutas seria computacionalmente ineficiente e dificultaria o aprendizado, uma vez que os pixels contêm ruídos, informações redundantes e variações irrelevantes (Krizhevsky et al., 2012). Por isso, o pré-processamento busca transformar os dados visuais em representações mais compactas e informativas, facilitando a extração de padrões significativos pela rede neural convolucional. As etapas de pré-processamento seguem o padrão descrito por Mnih et al. (2013) em seu trabalho pioneiro com o *DQN*. Primeiramente, cada frame é redimensionado para uma resolução menor, de 80 x 160 pixels utilizado nesta pesquisa, reduzindo a carga computacional e mantendo apenas informações essenciais da cena. Em seguida, as imagens são convertidas para escala de

cinza, eliminando redundâncias das três dimensões de cor (*RGB*), uma vez que a informação de luminosidade já é suficiente para inferir movimento e formas no ambiente. Posteriormente, aplica-se uma normalização dos valores de pixel (por exemplo, entre 0 e 1), padronizando as entradas da rede e acelerando a convergência durante o treinamento.

Outro passo importante é o empilhamento de frames consecutivos (*frame stacking*), técnica que fornece ao agente uma noção de temporalidade. Como as decisões no *ViZDoom* dependem de movimentos e eventos contínuos (tiros, deslocamentos, aproximações de inimigos), a observação de um único frame não é suficiente para inferir a dinâmica do ambiente. O empilhamento de quatro frames consecutivos, por exemplo, permite que a rede estime a direção e velocidade de objetos, fornecendo uma pseudo-memória temporal (Mnih et al., 2015). Essa abordagem substitui o uso de redes recorrentes em muitos casos, mantendo o processamento eficiente e compatível com arquiteturas convolucionais. Durante a interação com o ambiente, cada frame processado é transformado em um tensor multidimensional, que constitui a entrada da rede neural. A política, no caso do *PPO*, ou a função-valor, no caso do *DQN*, utiliza essas representações para inferir a ação ótima a ser tomada no próximo passo. Essa cadeia de processamento da captura de imagem até a decisão da ação é executada em tempo real durante todo o treinamento, garantindo que o agente aprenda diretamente a partir de sua percepção visual do ambiente.

4.3.2 Cenários e Recompensas

Os cenários utilizados neste trabalho foram selecionados a partir do *ViZDoom* (Kempka et al., 2016), uma plataforma amplamente adotada em pesquisas de Aprendizado por Reforço Profundo voltadas para ambientes tridimensionais e visuais. O *ViZDoom* permite que agentes de IA interajam diretamente com o jogo DOOM, observando o ambiente através de pixels em primeira pessoa e executando ações em tempo real. Cada cenário apresenta níveis distintos de dificuldade e objetivos específicos, possibilitando a análise comparativa do desempenho dos algoritmos *DQN* e *PPO* em contextos de complexidade crescente.

A definição das recompensas associadas a cada cenário é um elemento central no aprendizado por reforço, pois constitui o sinal que orienta o comportamento do agente conforme a Hipótese da Recompensa (*Reward Hypothesis*) proposta por Sutton e Barto (2018). Essa hipótese afirma que “todo objetivo pode ser formulado como a maximização

da recompensa esperada acumulada ao longo do tempo", de modo que a modelagem das recompensas determina o tipo de estratégia que o agente aprenderá. Assim, os esquemas de recompensa adotados neste trabalho foram cuidadosamente ajustados para incentivar comportamentos desejáveis como precisão, sobrevivência e eficiência e punir ações contraproducentes, como disparos aleatórios e exposição excessiva a inimigos.

A seguir, são descritos os três cenários implementados, juntamente com suas respectivas funções de recompensa.

4.3.2.1 Cenário BASIC

FIGURA 8: Cenário "BASIC".
Fonte: Vizdoom, 2025.

O cenário *BASIC* é o mais simples dentre os três utilizados e serve como ponto de partida para o treinamento inicial e calibração dos algoritmos. Nesse ambiente, o agente é colocado em uma sala retangular com um único inimigo estacionário. O objetivo é eliminar o inimigo o mais rapidamente possível utilizando uma arma de fogo, ao mesmo tempo em que se mantém vivo durante o episódio.

Este cenário tem como principal propósito avaliar a capacidade básica de percepção, pontaria e decisão de ataque do agente, sem a necessidade de navegação complexa ou múltiplos alvos. A interação é direta: o agente observa o inimigo, toma a decisão de atirar e recebe recompensas ou punições com base em suas ações.

O esquema de recompensas aplicado ao cenário *BASIC* é o seguinte:

- +106 pontos por eliminar o inimigo (recompensa principal e positiva, reforçando o comportamento de ataque bem-sucedido);

- -5 pontos por cada disparo executado (penalidade que desestimula tiros aleatórios e incentiva precisão);
- +1 ponto por cada *tic* (passo de tempo) em que o agente permanece vivo (recompensa de sobrevivência, que estimula estratégias mais cautelosas).

Esse arranjo de recompensas cria um balanceamento entre agressividade e prudência, forçando o agente a buscar eficiência: atacar de forma assertiva e com economia de recursos. Por ser um ambiente controlado e de baixa complexidade, o *BASIC* é ideal para verificar se o agente aprendeu corretamente a relação entre ação e resultado uma etapa fundamental para o desenvolvimento de estratégias mais avançadas (Kempka et al., 2016; Mnih et al., 2013).

4.3.2.2 Cenário DEFEND THE CENTER

FIGURA 9: Cenário “DEFEND THE CENTER”.
Fonte: Vizdoom, 2025.

O cenário *DEFEND THE CENTER* apresenta um aumento considerável na complexidade e tem como principal objetivo avaliar a capacidade do agente em reagir a múltiplos inimigos e manter-se vivo pelo maior tempo possível. Nesse ambiente, o agente é posicionado no centro de uma arena circular, cercado por inimigos que se aproximam de todas as direções. O campo de visão é amplo, mas o número crescente de inimigos impõe a necessidade de decisões rápidas e precisas.

O agente deve aprender a detectar alvos, priorizar ameaças e executar disparos de forma eficaz, enquanto tenta sobreviver ao ataque contínuo. Esse cenário demanda um equilíbrio entre mobilidade, foco de ataque e controle do gasto de munição, além de testar a capacidade da rede neural em lidar com múltiplos objetos dinâmicos na tela.

A função de recompensas neste cenário é simplificada, com foco direto no desempenho de combate:

- +1 ponto por cada inimigo eliminado;
- -1 ponto pela morte do agente.

Essa formulação direta incentiva comportamentos ofensivos e eficazes, recompensando o sucesso em eliminar inimigos e punindo a vulnerabilidade ou falha defensiva. O uso de recompensas simples neste contexto é proposital, pois facilita a análise comparativa do desempenho entre *DQN* e *PPO*, evidenciando as diferenças em estabilidade e eficiência de aprendizado entre métodos *value-based* e *policy-based* (Schulman et al., 2017; Mnih et al., 2015).

4.3.2.3 Cenário DEADLY CORRIDOR

FIGURA 10: Cenário “DEADLY CORRIDOR”.
Fonte: Vizdoom, 2025.

O cenário *DEADLY CORRIDOR* é o mais desafiador dos três, projetado para testar a capacidade de planejamento, precisão e tomada de decisão sob risco. Nele, o agente deve atravessar um corredor estreito, evitando ser atingido por inimigos armados enquanto se desloca em direção a um colete de proteção posicionado no final do corredor. Esse ambiente requer habilidades mais complexas, como gerenciamento de recursos (munição e energia vital), controle espacial e antecipação de eventos.

As recompensas neste cenário são mais sofisticadas e refletem múltiplos aspectos do desempenho do agente:

- $+\frac{dX}{100}$ pontos ao se aproximar do colete de proteção (recompensa de progresso, reforçando o movimento estratégico em direção ao objetivo final);

- $-\frac{dX}{100}$ pontos ao se afastar do colete (penalidade por retroceder ou desviar da trajetória ideal);
  -1 pontos em caso de morte (punição severa para desencorajar decisões arriscadas);
- -1 ponto por cada *tic* de sobrevivência (penalidade leve para incentivar eficiência no cumprimento do objetivo);
- -0.2 pontos por cada vez que o agente é atingido (punição que valoriza esquiva e controle de exposição);
- - (vida perdida / 10) proporcional ao dano sofrido;
- +3 pontos por cada inimigo eliminado (reforço forte para comportamento ofensivo eficaz);
- +1 pontos por cada acerto em um inimigo (recompensa intermediária que valoriza precisão de tiro);
- -0.1 pontos por disparo realizado (penalização por desperdício de munição).

Essa estrutura complexa de recompensas cria um cenário de conflito estratégico entre ataque e defesa: o agente deve aprender a maximizar ganhos de abate e progressão enquanto minimiza riscos e consumo desnecessário de recursos. Essa dinâmica torna o *DEADLY CORRIDOR* um ambiente adequado para analisar a eficiência dos algoritmos em lidar com recompensas densas e conflitantes, um dos grandes desafios do aprendizado por reforço profundo (Wang et al., 2016; Mnih et al., 2015).

5. RESULTADOS E DISCUSSÃO

5.1 Cenário BASIC

O cenário *BASIC* do ViZDoom é o ambiente mais simples dentre os utilizados, consistindo em uma sala retangular onde o agente precisa eliminar um inimigo com o uso de uma arma de fogo. O objetivo principal é atingir o alvo o mais rapidamente possível, maximizando as recompensas positivas e evitando penalizações por erros ou demora excessiva. Esse ambiente foi escolhido como ponto de partida por permitir a observação clara do processo de aprendizado e da estabilidade dos algoritmos.

O treinamento dos agentes foi realizado por meio dos algoritmos *Deep Q-Network* e *Proximal Policy Optimization*, ambos implementados utilizando a biblioteca *Stable*

Baselines3 em *Python* com suporte do *PyTorch*. Os parâmetros empregados no *DQN* foram os seguintes:

- CnnPolicy: define o tipo de política utilizada. Nesse caso, uma rede neural convolucional (*CNN*), adequada para processar imagens do ambiente do Doom.
- learning_rate (0.0001): controla o tamanho dos ajustes nos pesos da rede durante o aprendizado. Taxas muito altas causam instabilidade, e muito baixas tornam o aprendizado lento.
- buffer_size (100000): determina a quantidade de experiências (transições estado-ação-recompensa-próximo estado) armazenadas para o *replay buffer*. Um buffer grande ajuda o agente a aprender com maior diversidade de situações.
- batch_size (64): define quantas amostras do buffer são utilizadas por atualização de gradiente. Lotes maiores tornam o aprendizado mais estável.
- gamma (0.99): representa o fator de desconto que pondera a importância das recompensas futuras. Valores próximos de 1 priorizam ganhos a longo prazo.
- train_freq (4): indica que a rede será atualizada a cada 4 interações com o ambiente, equilibrando aprendizado e estabilidade.

Para o algoritmo *PPO*, foram utilizados os seguintes parâmetros:

- CnnPolicy: assim como no DQN, utiliza uma rede convolucional para extrair características visuais do ambiente.
- learning_rate (0.0001): taxa de aprendizado utilizada para o otimizador da política e da função de valor, garantindo uma convergência suave.
- n_steps (2048): número de passos de ambiente coletados antes de cada atualização da política. Esse valor influencia diretamente a estabilidade, quanto maior, mais dados são usados por atualização, reduzindo a variância das estimativas de gradiente.

O treinamento total teve duração aproximada de 16 minutos para o *DQN* e 14 minutos para o *PPO*, considerando 100.000 *steps* de interação com o ambiente.

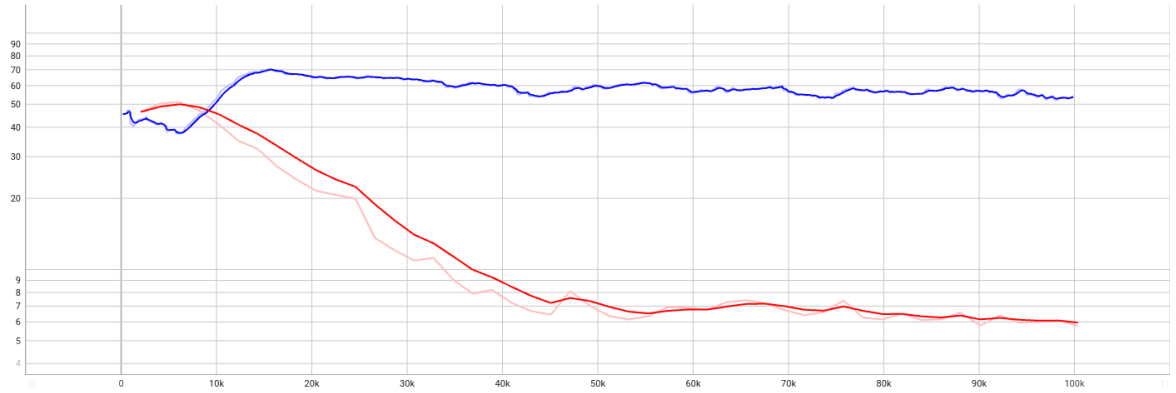

FIGURA 11: Cenário “BASIC”, duração média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

O primeiro gráfico (Figura 11) apresenta a duração média dos episódios para ambos os algoritmos. A curva em azul representa o *DQN* e a curva em vermelho o *PPO*. Observa-se que, neste cenário, episódios mais longos indicam pior desempenho, pois o agente é penalizado por cada disparo errado e pelo tempo adicional gasto antes de eliminar o inimigo. Assim, quanto menor a duração média dos episódios, mais eficiente é o agente em cumprir o objetivo. O PPO apresentou uma redução significativa na duração dos episódios, estabilizando-se rapidamente, enquanto o *DQN* mostrou variação irregular, indicando dificuldade em adaptar-se à posição variável do alvo.

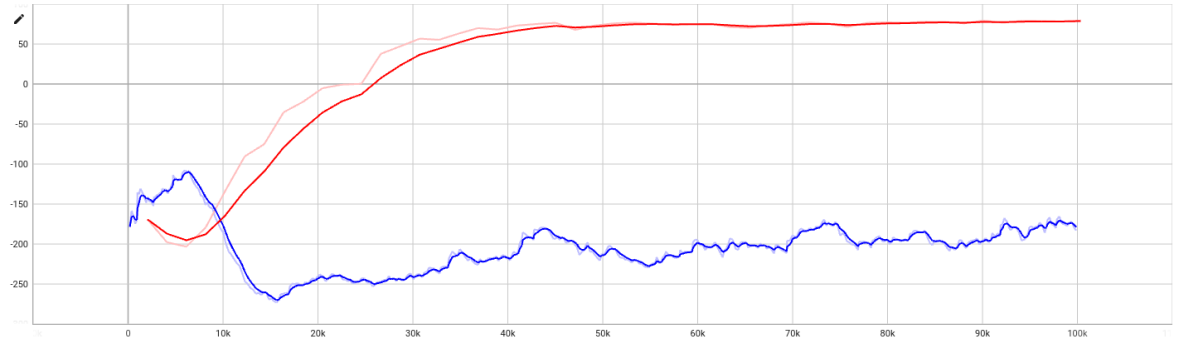

FIGURA 12: Cenário “BASIC”, recompensa média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

O segundo gráfico (Figura 12) mostra a recompensa média por episódio. Nota-se que o *PPO* rapidamente passou a obter recompensas mais elevadas, evidenciando um aprendizado consistente desde os primeiros 10.000 *steps*. Em contraste, o *DQN* apresentou oscilações acentuadas e uma grande queda de aproximadamente 150 pontos em torno dos 15.000 *steps*, seguida de flutuações menores. Esse comportamento pode indicar a ocorrência de *overfitting* local ou vício em determinadas sequências de ações,

um fenômeno comum em redes que atualizam políticas diretamente a partir de experiências armazenadas sem estratégias avançadas de exploração.

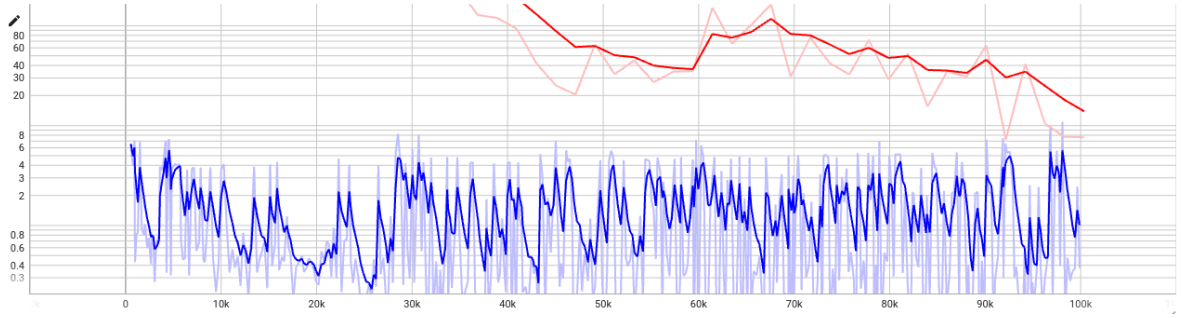

FIGURA 13: Cenário “BASIC”, evolução da função de perda (loss) ao longo do tempo de treinamento. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO, deve-se atentar que o PPO começou em 11000 e foi decaindo conforme o treinamento, pois sendo um número tão elevado, o algoritmo PPO não aparece direito no gráfico.
Fonte: Autor, 2025.

O terceiro gráfico (Figura 13) exibe a evolução da função de perda ao longo do treinamento. Essa métrica é essencial para compreender a estabilidade e a eficiência do processo de aprendizado. O PPO apresentou uma curva de perda estável e gradualmente decrescente, refletindo uma convergência suave e consistente. Já o *DQN* demonstrou uma perda irregular e oscilante, com picos abruptos, o que indica instabilidade e esquecimento de padrões previamente aprendidos (*catastrophic forgetting*). Essa diferença está relacionada à natureza dos algoritmos: o PPO realiza otimizações por aproximação com limites de atualização da política (*clipped surrogate objective*), enquanto o *DQN* depende fortemente da estabilidade da rede-alvo e da amostragem de experiências passadas, sendo mais suscetível a oscilações.

Os resultados sugerem que o *PPO* apresentou um desempenho notavelmente superior neste cenário. Desde os primeiros episódios, o agente treinado com PPO conseguiu compreender rapidamente a dinâmica do ambiente e melhorar seu desempenho de forma contínua. No final do treinamento, alcançou uma média de 6 frames por acerto de alvo, demonstrando um comportamento otimizado e eficiente. O *DQN*, por outro lado, não apresentou aprendizado significativo, o que pode ser atribuído à sua dificuldade em lidar com variações espaciais e temporais do inimigo, bem como à limitação de generalização diante de cenários com ruído visual e posicionamento aleatório. Desta forma, esses achados corroboram a literatura recente, que aponta o *PPO* como um dos algoritmos mais estáveis e eficientes em ambientes contínuos e parcialmente observáveis, como o ViZDoom (Schulman et al., 2017; Andrychowicz et al., 2020). Já o *DQN*, apesar de seu impacto histórico no aprendizado por reforço (Mnih et al.,

2015), tende a apresentar desempenho inferior em ambientes tridimensionais complexos, especialmente quando a percepção visual e a aleatoriedade têm papel central.

5.2 Cenário DEFEND THE CENTER

O cenário *DEFEND THE CENTER* configura-se como uma tarefa contínua de sobrevivência em que o agente, posicionado no centro de uma arena, enfrenta ondas sucessivas de inimigos. Diferentemente do *BASIC*, aqui o objetivo primário é maximizar o tempo de vida e o número de eliminações acumuladas sob restrição de munição e reaparecimento constante de adversários, o que torna a dinâmica de recompensa mais densa e apropriada para a aprendizagem de estratégias defensivas e prioridades de ataque (Kempka et al., 2016). A seguir apresentam-se os parâmetros utilizados para cada algoritmo, os parâmetros empregados no *DQN* foram :

- CnnPolicy: Política baseada em rede convolucional, adequada para entrada visual (frames).
- learning_rate (0.0001):Taxa de aprendizado que determina a magnitude das atualizações dos pesos; valor pequeno favorece estabilidade.
- buffer_size (50000): Capacidade do *replay buffer* (nº de transições armazenadas); maior diversidade de amostras melhora generalização.
  learning_starts(500):
  Número mínimo de passos antes de começar as atualizações; permite acumular experiências suficientes.
- batch_size(32): Número de amostras por atualização; compromete entre variância do gradiente e custo computacional.
- tau (1.0): Indica atualização *hard* da *target network* (substituição completa), em vez de *soft update*.
- gamma (0.95): Fator de desconto para recompensas futuras
- train_freq (4) :Frequência (em passos de ambiente) com que ocorrem as atualizações de rede.
- gradient_steps (1): Número de passos de gradiente por iteração de treinamento.
- target_update_interval(5000):Intervalo de sincronização da *target network*; impacta a estabilidade dos alvos Q.
- exploration_fraction (0.05) / exploration_initial_eps (1.0) / exploration_final_eps (0.05) Parâmetros do esquema ε-greedy: começam com exploração total e decaem até 5% ao longo de 5% do total de passos, equilibrando exploração e exploração.

- max_grad_norm = 10 *Gradient clipping* para evitar explosão de gradientes.

Para o algoritmo *PPO*, foram utilizados os seguintes parâmetros:

- CnnPolicy: Política baseada em rede convolucional,.
- learning_rate (0.0001): taxa de aprendizado utilizada
- n_steps (2048): número de passos de ambiente coletados antes de cada atualização da política.

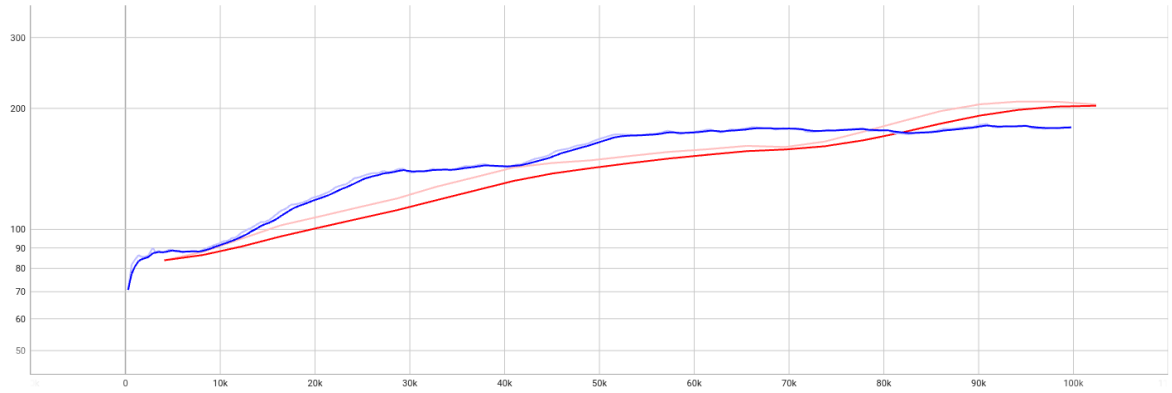

FIGURA 14: Cenário “DEFEND THE CENTER”, duração média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

Neste cenário, uma duração maior de episódio é desejável: episódios longos indicam que o agente conseguiu sobreviver por mais tempo diante das ondas de inimigos. As curvas de ambos os algoritmos mostraram crescimento consistente evidência de aprendizado de estratégias defensivas. O *DQN* e o *PPO* aumentaram progressivamente a capacidade de evitar mortes imediatas (uso de movimentação, focalização de alvos próximos, controle de taxa de fogo). A interpretação cuidadosa indica que ambos captaram sinais úteis do ambiente, embora por mecanismos distintos (replay vs. otimização proximal).

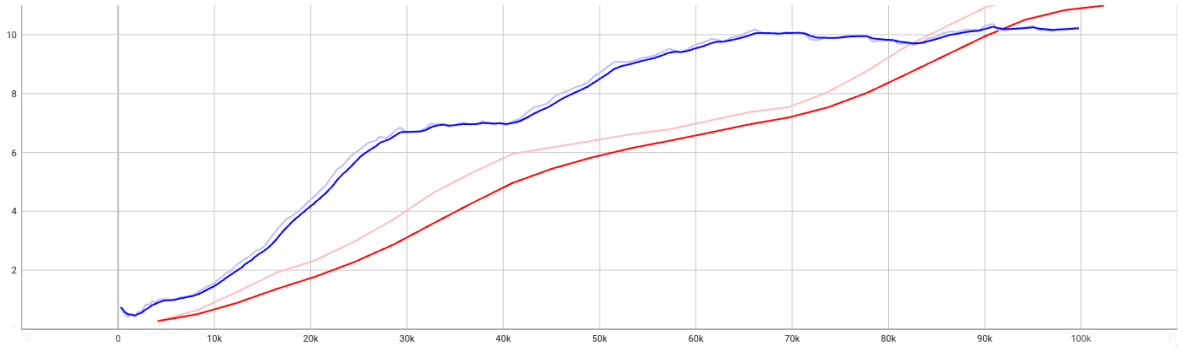

FIGURA 15: Cenário “DEFEND THE CENTER”, recompensa média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

A recompensa média inicial do *DQN* cresceu mais rapidamente, refletindo uma descoberta precoce de ações que geram kills repetíveis. Até cerca de 81.000 *steps* o *DQN* manteve vantagem com um resultado coerente com a propriedade do *experience replay* que reforça transições bem-sucedidas e acelera cristalização de heurísticas locais. Entretanto, ao prosseguir o treinamento, o PPO ultrapassou o DQN, finalizando com média *PPO* = 10,99 versus *DQN* = 10,22. Essa inversão indica que, embora o DQN encontre soluções imediatas com rapidez, o PPO tende a refinar políticas de maior qualidade ao longo do tempo, graças à estabilidade das atualizações por gradiente de política e ao cálculo de vantagem que orienta melhorias de maneira consistente.

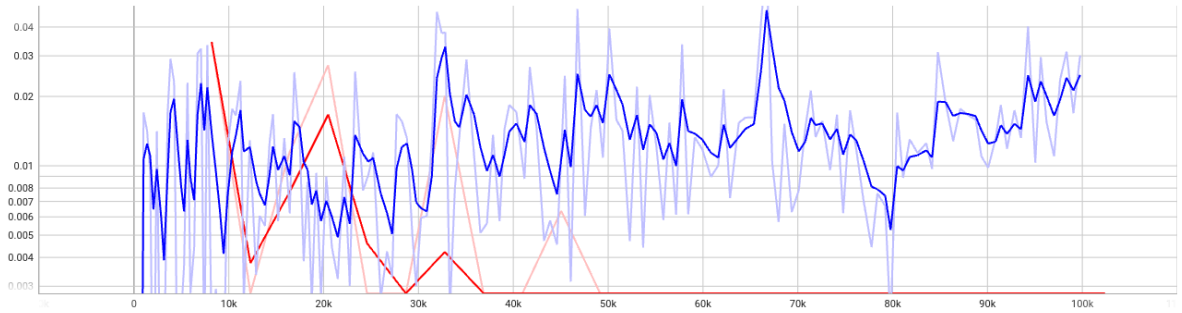

FIGURA 16: Cenário “DEFEND THE CENTER”, evolução da função de perda (loss) ao longo do tempo de treinamento. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.

Ambas as perdas foram menores do que no cenário *BASIC*, condizente com a maior densidade de recompensa que facilita o aprendizado. O PPO apresentou perda estável e decrescente, sinalizando convergência consistente do estimador de vantagem e da política. O *DQN* exibiu perda irregular e oscilante, ainda que com menor ruído relativo do que no *BASIC*; essa oscilação é típica do aprendizado off-policy com replay, quando amostras antigas e recentes geram alvos Q não estacionários, potencialmente provocando "saltos" na superfície de otimização quando a *target network* é sincronizada.

Os resultados obtidos no cenário *Defend the Center* revelam uma distinção clara entre as dinâmicas de aprendizado do DQN e do PPO, tanto em termos de velocidade de convergência quanto na qualidade da política aprendida. O DQN apresentou um desempenho inicial superior, destacando-se por sua rápida identificação das ações que geram recompensas positivas, o que pode ser atribuído à natureza *off-policy* do algoritmo e ao uso do *experience replay*. Esse mecanismo permite que experiências passadas sejam reutilizadas durante o processo de treinamento, acelerando o aprendizado das transições estado-ação que levam a resultados vantajosos. O valor de desconto adotado também contribuiu para esse comportamento, uma vez que privilegia recompensas de curto prazo, o que é vantajoso em um ambiente onde o agente precisa reagir rapidamente a ameaças constantes e próximas. Dessa forma, o *DQN* demonstrou ser eficiente na fase inicial de treinamento, conseguindo identificar estratégias eficazes de sobrevivência e ataque de forma relativamente rápida.

Por outro lado, o *PPO*, embora tenha apresentado uma curva de aprendizado mais lenta nos primeiros episódios, demonstrou maior estabilidade e refinamento ao longo do tempo. Isso se deve à sua característica on-policy, que limita as atualizações a dados coletados pela política atual, reduzindo o risco de aprendizado instável. O uso do mecanismo de *clipping* nas atualizações de política, típico do *PPO*, impediu mudanças abruptas nos parâmetros, garantindo uma convergência mais suave e controlada. Além disso, o uso de trajetórias longas permitiu uma melhor estimação da vantagem e uma redução na variância dos gradientes, o que favoreceu a descoberta de políticas mais sofisticadas e eficazes em situações prolongadas de combate. Como resultado, o *PPO* conseguiu ultrapassar o *DQN* no estágio final do treinamento, obtendo uma recompensa média ligeiramente superior (10,99 contra 10,22), evidenciando uma política mais consistente e adaptada ao ambiente dinâmico e de longo horizonte temporal do cenário *DEFEND THE CENTER*.

A análise crítica dos resultados sugere que o *DQN* é mais eficiente para ambientes com recompensas imediatas e alta frequência de feedback, enquanto o *PPO* demonstra maior competência em contextos onde a estabilidade e a consistência das decisões são cruciais para a maximização do desempenho a longo prazo. A irregularidade observada na função de perda do DQN indica que, embora o algoritmo aprenda rapidamente, ele sofre com flutuações associadas à natureza off-policy e ao uso de amostras não estacionárias no *replay buffer*. Em contraste, a perda decrescente e estável do *PPO* evidencia sua capacidade de ajustar gradualmente a política, minimizando a oscilação e o esquecimento de padrões aprendidos. Esses resultados têm

implicações práticas importantes para a aplicação de aprendizado por reforço profundo em ambientes de simulação complexos, como jogos ou sistemas de controle contínuo. Em tarefas que exigem respostas rápidas e aprendizado inicial acelerado, o *DQN* pode ser preferível devido à sua eficiência exploratória. Já em tarefas que demandam planejamento de longo prazo, adaptação constante e políticas mais suaves, o *PPO* tende a oferecer melhor desempenho final. Um caminho promissor para pesquisas futuras seria a combinação das abordagens utilizando o *DQN* na fase inicial de exploração e o *PPO* no refinamento das políticas, aproveitando a velocidade do aprendizado *off-policy* e a estabilidade das atualizações *on-policy*. Essa integração pode potencializar o desempenho global dos agentes em cenários complexos e dinâmicos como os oferecidos pelo *ViZDoom*.

5.3 Cenário DEADLY CORRIDOR

O cenário *DEADLY CORRIDOR* representa o ambiente mais complexo entre os três utilizados nesta pesquisa, exigindo do agente uma combinação de estratégias ofensivas e defensivas para atingir o objetivo final. Nesse ambiente, o agente é posicionado em um corredor longo dividido em três seções, cada uma contendo pares de inimigos armados que disparam assim que detectam o jogador. O objetivo consiste em sobreviver ao fogo inimigo e alcançar o final do corredor, onde um item concede uma recompensa significativa e encerra o episódio. Essa estrutura impõe um desafio de alto nível, pois o agente precisa equilibrar entre atacar e se esquivar, otimizando o uso de munição e o tempo de sobrevivência, fatores essenciais para maximizar sua pontuação.

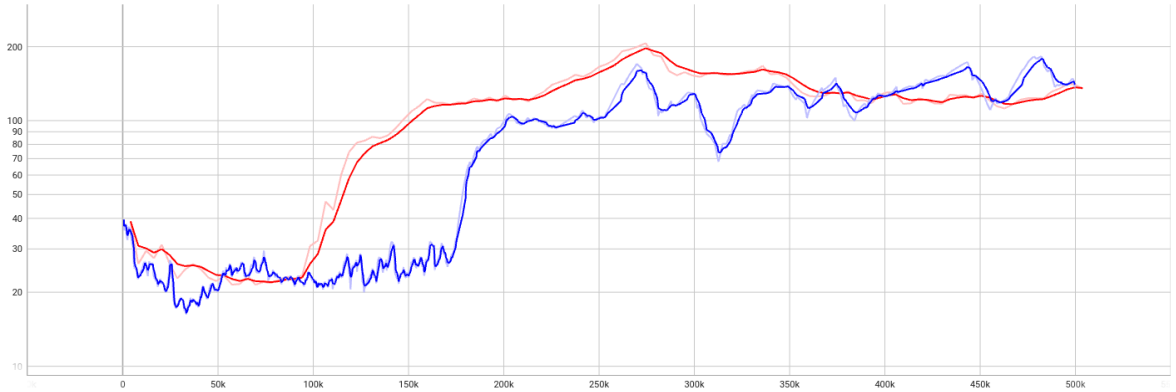

FIGURA 17: Cenário “DEADLY CORRIDOR”, duração média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

A figura 17, que demonstra a duração média dos episódios, revela que tanto o *DQN* quanto o *PPO* obtiveram resultados consistentes e apresentaram aumento progressivo na duração dos episódios ao longo do tempo, o que indica aprendizado e adaptação efetiva ao ambiente. O aumento na duração está diretamente relacionado à capacidade de sobrevivência: agentes que aprendem a evitar disparos e administrar melhor sua posição permanecem mais tempo vivos. Observou-se que o *DQN* teve pequenas quedas de desempenho em determinados intervalos, o que sugere instabilidade na política aprendida. O *PPO*, em contrapartida, apresentou uma evolução mais regular, indicando um aprendizado mais robusto e menos sujeito a oscilações abruptas.

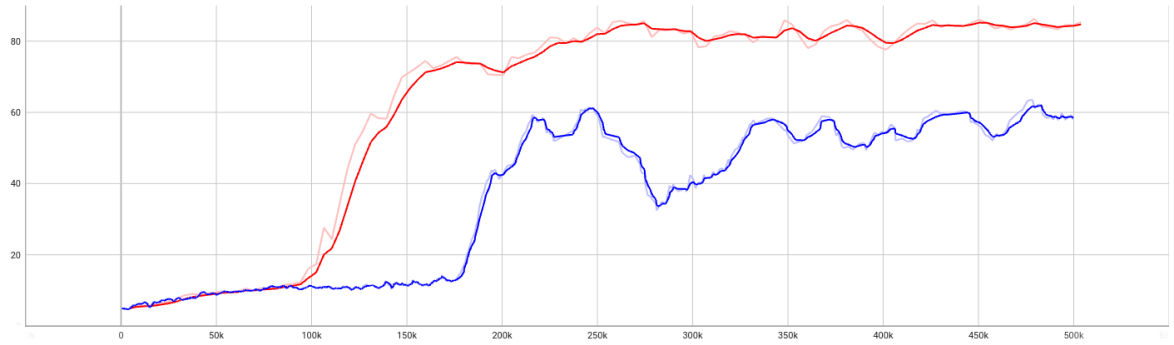

FIGURA 18: Cenário “DEADLY CORRIDOR”, recompensa média dos episódios por step. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO.
Fonte: Autor, 2025.

A figura 18, referente à recompensa média dos episódios, evidencia que ambos os algoritmos compreenderam adequadamente a dinâmica do ambiente, aprendendo a balancear entre o ataque e a defesa. Entretanto, o *PPO* superou o *DQN* de maneira consistente, apresentando de 20% a 40% mais recompensas em média. Essa diferença decorre da natureza on-policy do PPO, que favorece atualizações estáveis e refinadas, permitindo que o agente ajuste gradualmente sua política em relação às mudanças no ambiente. Assim, o *PPO* foi mais eficaz em identificar trajetórias seguras e eficientes até o final do corredor, enquanto o *DQN* demonstrou maior variabilidade em seu desempenho, possivelmente devido à natureza *off-policy* e ao uso de amostras antigas em seu *replay buffer*, que podem não refletir mais o estado atual da política.

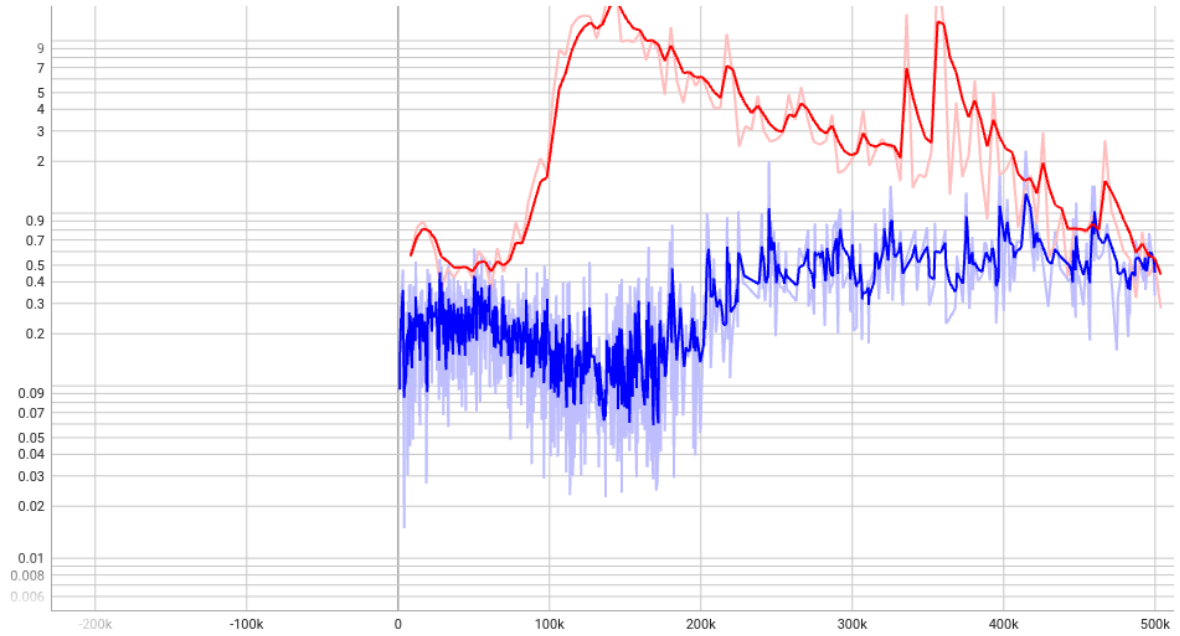

FIGURA 19: Cenário “DEADLY CORRIDOR”, evolução da função de perda (loss) ao longo do tempo de treinamento. Em cor azul é o algoritmo DQN e em cor vermelha é o algoritmo PPO. Fonte: Autor, 2025.

O terceiro gráfico, que mostra a evolução da função de perda, reforça essa distinção entre as abordagens. O *DQN* manteve uma perda irregular e oscilante durante o treinamento, um comportamento esperado devido à variação estocástica das amostras do *replay buffer*. Em contrapartida, o *PPO* apresentou inicialmente uma perda mais elevada, reflexo do ajuste contínuo de sua política, mas que foi decrescendo de maneira estável ao longo dos *steps*. Ao final do treinamento, ambas as curvas convergiram para valores semelhantes, o que demonstra que, apesar da maior complexidade do ambiente, ambos os algoritmos foram capazes de estabilizar seu processo de aprendizado.

O treinamento deste cenário foi significativamente mais longo que os anteriores, totalizando 500.000 *steps*, devido à sua complexidade e à necessidade de aprendizado mais refinado. O tempo médio de treinamento foi de 1 hora e 5 minutos para o *DQN* e 1 hora e 12 minutos para o *PPO*, refletindo a diferença na estrutura interna de cada algoritmo e no processamento das atualizações de política. Ambos foram implementados com os mesmos parâmetros utilizados no cenário anterior (*DEFEND THE CENTER*). Os achados do cenário *DEADLY CORRIDOR* foram bem parecidos com o *DEFEND THE CENTER, o que* indicam que, apesar da complexidade e do aumento significativo no número de interações, ambos os algoritmos foram capazes de aprender políticas eficazes de sobrevivência e progressão no ambiente. O *DQN* mostrou eficiência inicial e aprendizado rápido das relações de curto prazo entre ação e recompensa, mas

apresentou instabilidade e flutuações ocasionais. O *PPO*, embora mais lento no início, demonstrou superioridade em termos de estabilidade e desempenho final, aprendendo estratégias mais consistentes e equilibradas.

Em termos gerais, o *PPO* se destacou por sua capacidade de otimização contínua e eficiente, conseguindo uma política mais adaptada e com maior média de recompensas, enquanto o *DQN* demonstrou maior sensibilidade às variações do ambiente. Esses resultados confirmam a tendência observada nos experimentos anteriores: o *DQN* apresenta aprendizado mais rápido, porém menos estável, enquanto o *PPO* mostra convergência mais lenta, mas com resultados mais consistentes e generalizáveis, especialmente em ambientes de alta complexidade e múltiplas variáveis de decisão, como o *DEADLY CORRIDOR*.

6. CONCLUSÃO

A presente pesquisa teve como objetivo principal responder à seguinte questão: “como desenvolver um agente inteligente capaz de aprender e se adaptar de forma autônoma ao ambiente de um jogo eletrônico?”. Para tanto, foram implementados e comparados dois algoritmos de Aprendizado por Reforço Profundo: *Deep Q-Network* e *Proximal Policy Optimization*, aplicados em três cenários distintos do ambiente *ViZDoom*: *BASIC*, *DEFEND THE CENTER* e *DEADLY CORRIDOR*. Cada um desses cenários apresentou diferentes níveis de complexidade, exigindo do agente habilidades crescentes de percepção, decisão e adaptação.

Com base nos resultados obtidos, pode-se afirmar que o objetivo proposto foi alcançado com êxito. Ambos os algoritmos demonstraram capacidade de aprendizado autônomo e adaptação progressiva aos ambientes simulados, desenvolvendo estratégias coerentes com as recompensas fornecidas. O agente, em todas as instâncias, partiu de um comportamento completamente aleatório e, ao longo do treinamento, passou a tomar decisões orientadas à maximização de recompensas, evidenciando o princípio fundamental da Hipótese de Recompensa.

No cenário *BASIC*, observou-se que o *PPO* teve desempenho superior, conseguindo compreender rapidamente o objetivo do ambiente e maximizar sua pontuação, enquanto o *DQN* apresentou maior instabilidade e dificuldade de convergência. Já no cenário Defend the Center, ambos os algoritmos demonstraram aprendizado sólido, mas o *PPO* novamente se destacou pela consistência e estabilidade de suas recompensas médias. Por fim, no cenário *DEADLY CORRIDOR*, o mais

complexo dos três, ambos os algoritmos conseguiram aprender políticas eficazes, porém o *PPO* manteve um desempenho mais estável e eficiente, apresentando uma média de recompensas até 40% superior à do *DQN* em determinados períodos do treinamento.

Esses resultados permitem concluir que o *PPO* mostrou-se mais robusto, estável e eficiente em ambientes de maior complexidade e com múltiplas variáveis de decisão. Sua estrutura on-policy e o mecanismo de *clipping* na atualização da política contribuíram para evitar oscilações excessivas e aprimorar o desempenho final do agente. O *DQN*, por outro lado, demonstrou aprendizado mais rápido em estágios iniciais e menor custo computacional, mas sua natureza *off-policy* o tornou mais suscetível a instabilidades e flutuações de desempenho. Do ponto de vista técnico, o uso do *ViZDoom* como ambiente experimental mostrou-se extremamente relevante para o estudo de agentes em contextos tridimensionais e dinâmicos. A plataforma permitiu observar comportamentos emergentes complexos, próximos aos desafios encontrados em jogos modernos e aplicações reais de robótica e navegação autônoma. O emprego do *TensorBoard* possibilitou o monitoramento detalhado do processo de aprendizado, fornecendo métricas essenciais para a análise comparativa entre os algoritmos. Entretanto, a pesquisa apresentou algumas limitações. O tempo de treinamento, embora suficiente para observação de tendências claras, poderia ser ampliado para avaliar a convergência em períodos mais longos. Além disso, a ausência de ajustes dinâmicos de hiperparâmetros e de técnicas de *transfer learning* restringiu o potencial máximo de generalização dos agentes.

Como perspectivas futuras, recomenda-se explorar arquiteturas híbridas, como o *Dueling DQN* ou o *PPO2*, que combinam vantagens de múltiplas abordagens para aumentar a estabilidade e eficiência. Também seria promissor investigar o uso de recompensas modeladas ou métodos de aprendizado hierárquico, que permitiriam ao agente aprender objetivos intermediários antes de dominar a tarefa principal. A incorporação de técnicas de aprendizado por curiosidade também poderia tornar o agente mais exploratório e adaptável a ambientes ainda mais complexos. Desta forma este trabalho demonstrou que é plenamente possível desenvolver um agente inteligente capaz de aprender e se adaptar autonomamente ao ambiente de um jogo eletrônico tridimensional, como o *DOOM*, utilizando métodos modernos de aprendizado por reforço profundo. Os resultados reforçam a relevância e o potencial dessas técnicas não apenas para jogos, mas também para aplicações mais amplas em sistemas autônomos, robótica, simulações e ambientes interativos de tomada de decisão.

7. REFERÊNCIAS

ANDRYCHOWICZ, M. et al. **What Matters in On-Policy Reinforcement Learning? A Large-Scale Empirical Study**. *arXiv:2006.05990*, 2020.

BARRON, T.; WHITEHEAD, M.; YEUNG, A. **Deep Reinforcement Learning in a 3-D Blockworld Environment**, 2016.

BONASSO, R. P. **Experiences with an Architecture for Intelligent Reactive Agents.** ***Journal***, 1995. Disponível em: https://www.researchgate.net/publication/221454949_Experiences_with_an_Architecture_for_Intelligent_Reactive_Agents. Acesso em: 16/10/2025.

CHENGQI, Z.; ZILI, Z. **Analysis and Synthesis. In:** ***Encyclopedia of Complexity and Systems Science***. 2007. DOI: 10.1002/9780470050118.ecse194.

FOX, J.; BEVERIDGE, M.; GLASSPOOL, D. **Understanding Intelligent Agents: Analysis and Synthesis**. 2003. Disponível em: https://www.researchgate.net/publication/220309098_Understanding_Intelligent_Agents_Analysis_And_Synthesis. Acesso em: 11/10/2025.

GRAND VIEW RESEARCH. **Gaming Industry Report**. 2025. Disponível em: https://www.grandviewresearch.com/industry-analysis/gaming-industry. Acesso em: 23/06/2025.

HAN, D.; MULYANA, B.; STANKOVIC, V.; CHENG, S. **A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation**. *Sensors*, v. 23, n. 7, p. 3762, 2023.

HE, K.; ZHANG, X.; REN, S.; SUN, J. **Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification**. 2015.

IBM. **Backpropagation – Entenda o Algoritmo**. 2025. Disponível em: https://www.ibm.com/br-pt/think/topics/backpropagation. Acesso em: 10/11/2025.

IOFFE, S.; SZEGEDY, C. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. *arXiv:1502.03167*, 2015.

KEMPKA, M. et al. **ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning.** ***IEEE Conference on Computational Intelligence and Games (CIG)***, 2016. Disponível em: https://arxiv.org/abs/1605.02097.

KIRAN, B. R. et al. **Deep Reinforcement Learning for Autonomous Driving: A Survey**. *IEEE Transactions on Intelligent Transportation Systems*, v. 23, n. 6, p. 4909–4926, 2021.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **ImageNet Classification with Deep Convolutional Neural Networks**. *NIPS*, 2012.

LANGDON, J. **Water-mills and Windmills in the West Midlands**, 1086–1500. *The Economic History Review*, v. 44, n. 3, p. 424–444, 1991.

LARNER, R. **Efficient BackProp. 1998**. Disponível em: https://cseweb.ucsd.edu/classes/wi08/cse253/Handouts/lecun-98b.pdf.

LIN, L. **Reinforcement Learning for Robots Using Neural Networks**. *Experience Replay*. 1992. Disponível em: https://dl.acm.org/doi/10.1007/bf00992699.

LI, Y. **Deep Reinforcement Learning: An Overview**. *arXiv:1701.07274*, 2017.

LUDERMIR, T. B. Redes Neurais Artificiais. 2021. DOI: 10.1590/s0103-4014.2021.35101.007.

MC CULLOCH, W. S.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity.** ***Bulletin of Mathematical Biophysics***, v. 5, p. 115–133, 1943.

MNIH, V. et al. **Playing Atari with Deep Reinforcement Learning**. *arXiv:1312.5602*, 2013.

MNIH, V. et al. **Human-Level Control through Deep Reinforcement Learning**. *Nature*, v. 518, p. 529–533, 2015.

NAIR, V.; HINTON, G. E. **Rectified Linear Units Improve Restricted Boltzmann Machines**. 2010.

NEWZOO. **Global Games Market Report. 2023**. Disponível em: https://newzoo.com. Acessado em: 15/06/2025

NEVES, R. **Introdução ao Aprendizado por Reforço. 2020**. Disponível em: https://medium.com/turing-talks/aprendizado-por-refor%C3%A7o-1-introdu%C3%A7%C3%A3o-7382ebb641ab. Acessado em: 18/09/2025

PASZKE, A. et al. **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. *NeurIPS*, 2019.

PUTERMAN, M. L. **Markov Decision Processes: Discrete Stochastic Dynamic Programming**. Wiley, 1994.

RAFFIN, A. et al. **Stable Baselines3 — Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research***, v. 22, p. 1–8, 2021.

RAUBER, P. Redes Neurais Artificiais. 2005. Disponível em: https://www.researchgate.net/publication/228686464_Redes_neurais_artificiais.

ROSENBLATT, F. **The perceptron: A probabilistic model for information storage and organization in the brain**. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D.; HINTON, G.; WILLIAMS, **R. Learning representations by back-propagating errors**. 1986.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. Pearson, 2020.

SCHULMAN, J. et al. **Proximal Policy Optimization Algorithms**. *arXiv:1707.06347*, 2017.

SILVER, D. et al. **Reward is Enough. *Artificial Intelligence*,** v. 299, 103535, 2021.

STANFORD. **Dicas de Aprendizado Não Supervisionado**. 2025. Disponível em: https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-aprendizado-nao-supervisionado. Acessado em: 01/11/2025.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction.** MIT Press, 2018.

TERVEN, J. **Deep Reinforcement Learning: A Chronological Overview and Methods.** *AI*, v. 6, n. 3, p. 46, 2025. DOI: 10.3390/ai6030046.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. Wiley, 2009.

YANG, T. et al. **Exploration in Deep Reinforcement Learning: A Comprehensive Survey**, 2021.

YANNIS, M.; CHONG, U.; SARKAR, A. et al. **Adaptive AI for Non-Player Characters in Multiplayer Games: A Review.** ***Entertainment Computing***, v. 42, 2022.

YUXI LI. **Deep Reinforcement Learning: An Overview**. *arXiv:1701.07274*, 2017.

ZHANG, Y. et al. **Deep Reinforcement Learning with Successor Features for Navigation Across Similar Environments.** ***IEEE Transactions on Cognitive and Developmental Systems***, v. 12, n. 1, p. 16–28, 2020.

ZUSAMMEN, B. R. et al. **Deep Reinforcement Learning for Autonomous Driving: A Survey**. 2021.

ANEXO I

Código da pesquisa, disponibilizado em:
https://github.com/DayonCzykailo/Intelligent_Agent_DOOM-TCC