Investigação de Métodos de Aprendizagem e de Representação de Ambiente na Construção de Agentes Jogadores: Aplicação ao Jogo FIFA

Matheus Prado Prandini Faria

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Matheus Prado Prandini Faria

Investigação de Métodos de Aprendizagem e de Representação de Ambiente na Construção de Agentes Jogadores: Aplicação ao Jogo FIFA

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Profa. Dra. Rita Maria da Silva Julia
Coorientador: Profa. Dra. Lídia Bononi Paiva Tomaz

Uberlândia
2020

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

F224
2020

Faria, Matheus Prado Prandini, 1996-
Investigação de Métodos de Aprendizagem e de Representação de Ambiente na Construção de Agentes Jogadores [recurso eletrônico] : Aplicação ao Jogo FIFA / Matheus Prado Prandini Faria. - 2020.

Orientadora: Rita Maria da Silva Julia.
Coorientadora: Lídia Bononi Paiva Tomaz.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2020.320
Inclui bibliografia.

1. Computação. I. Julia, Rita Maria da Silva,1960-, (Orient.). II. Tomaz, Lídia Bononi Paiva,1988-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado Acadêmico, 11/2020, PPGCO				
Data:	05 de março de 2020	Hora de início:	**09h00min**	Hora de encerramento:	**12h05min**
Matrícula do Discente:	11812CCP024				
Nome do Discente:	Matheus Prado Prandini Faria				
Título do Trabalho:	Investigação de Métodos de Aprendizagem e de Representação de Ambiente na Construção de Agentes Jogadores: Aplicação ao Jogo FIFA				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se na sala 1B132, Bloco 1B, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Murillo Guimarães Carneiro- FACOM/UFU, George Darmiton da Cunha Cavalcanti - CIN/UFPE, Lídia Bononi Paiva Tomaz - CGEPE/IFTM (Coorientadora) e Rita Maria da Silva Julia - FACOM/UFU, orientadora do candidato.

Ressalta-se que o Prof. Dr. George Darmiton da Cunha Cavalcanti participou da defesa por meio de videoconferência desde a cidade de Recife-PE. Os outros membros da banca e o aluno participaram in loco.

Iniciando os trabalhos a presidente da mesa, Prof.ª Dr.ª Rita Maria da Silva Julia, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Lídia Bononi Paiva Tomaz**, **Usuário Externo**, em 06/03/2020, às 13:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rita Maria da Silva Julia**, **Professor(a) do Magistério Superior**, em 06/03/2020, às 14:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **George Darmiton da Cunha Cavalcanti**, **Usuário Externo**, em 08/03/2020, às 11:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Murillo Guimarães Carneiro**, **Professor(a) do Magistério Superior**, em 09/03/2020, às 10:31, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1907941** e o código CRC **C61A795E**.

Referência: Processo nº 23117.015784/2020-11 SEI nº 1907941

Dedico este trabalho aos meus pais, Cássia e Marcelo,
e também à minha irmã, Luísa, por todo o apoio e ensinamentos
ao longo de minha jornada terrestre.

Agradecimentos

Primeiramente, agradeço a Deus pelo dom da vida e por todas as oportunidades a mim concedidas.

Agradeço aos meus pais e à minha irmã por estarem sempre presentes e serem fundamentais em todos os aspectos de minha vida. Gostaria de agradecer também a todos os animais que já estiveram presentes em nossa família, sempre muito especiais para nós.

À minha namorada Karen por compreender a importância que essa pesquisa teve para mim e, principalmente, por estar ao meu lado. Agradeço também aos seus pais, à sua irmã, à sua avó e a todos os seus familiares pelo apoio e motivação ao longo desse período.

À Professora Rita pela orientação, amizade, gentileza e compreensão durante o desenvolvimento desse trabalho. Obrigado por todas as discussões construtivas e por ser uma segunda mãe para mim.

À Professora Lídia pela coorientação, amizade e por estar sempre disposta a me auxiliar a todo momento apesar de todas as dificuldades que teve ao longo da jornada de meu mestrado: a gravidez e nascimento de seu filho Gabriel.

Aos Professores da Pós Graduação da FACOM/UFU pelos ensinamentos, conselhos e conhecimentos compartilhados. Aos secretários de pós-graduação, Sônia e Erisvaldo, pela boa vontade e competência com que sempre me auxiliaram. Agradeço a CAPES pelo apoio financeiro durante todo esse período.

Aos meus amigos da graduação que são especiais para mim até hoje. Aos amigos de pós-graduação que tive a oportunidade de compartilhar um pouco de minhas ideias e pensamentos. Aos meus amigos do futebol americano e da nossa tradição da *NFL* ano após ano. Ao Projeto Amar para Educar e todas as pessoas ali presentes.

Enfim, agradeço a todos aqueles que sempre me apoiaram e emanaram energias positivas.

"A vida é uma peça de teatro que não permite ensaios. Por isso, cante, chore, dance, ria e viva intensamente, antes que a cortina se feche e a peça termine sem aplausos."

(Charles Chaplin)

Resumo

O objetivo deste trabalho é investigar técnicas apropriadas de Aprendizado de Máquina (AM) - mais especificamente, aprendizagem profunda (AP) - combinadas a técnicas de representação de estado por imagem provenientes da Visão Computacional (VC), para produzir agentes capazes de resolver problemas, em tempo real, em ambientes com propriedades complexas. Tais dificuldades requerem que os agentes apresentem elevada eficiência nos seus processos de aprendizagem (e, consequentemente, de tomada de decisão) e de percepção do ambiente, sem os quais não obterão êxito. O jogo eletrônico *FIFA* - simulador de futebol - é utilizado como estudo de caso por representar um ambiente realístico e desafiador. As técnicas de AM são investigadas no contexto da AP propiciado pelas Redes Neurais Convolucionais (RNCs), sendo elas: aprendizado por imitação, na qual os agentes são dotados da capacidade de resolver problemas de maneira mais próxima à humana; por reforço profundo, na qual o agente é treinado de forma a tentar abstrair, autonomamente, uma política ótima de tomada de decisões. Com relação à percepção de ambiente, investigam-se aqui as seguintes abordagens de representação por imagem: imagens cruas - com e sem informação de cor - e Técnicas de Detecção de Objetos (TDO). A título de melhorar ainda mais o desempenho dos agentes desenvolvidos, são exploradas técnicas de algoritmos genéticos para definir, automaticamente, uma arquitetura otimizada da RNC a ser utilizada como módulo de tomada de decisão de agentes jogadores. Além de corroborar os excelentes resultados que a AP combinada à VC vem produzindo no contexto da AM (particularmente, nos jogos), o presente trabalho mostra o grande potencial da aplicação de TDO no processo de enriquecimento da percepção de ambiente, o que conta como contrapartida relevante ao fato de TDO demandar recursos computacionais de mais alto custo em relação às representações baseadas em imagens cruas.

Palavras-chave: Aprendizado de Máquina. Aprendizagem Profunda. Visão Computacional. Técnicas de Detecção de Objetos. Aprendizado por Imitação. Aprendizado por Reforço. Redes Neurais Convolucionais. Algoritmos Genéticos.

Abstract

The objective behind this study is to investigate appropriate Machine Learning (ML) techniques - more specifically, Deep Learning (DL) - combined with methods from Computer Vision (CV) for state representation by images, to produce agents capable of solving problems, in real time, in environments with complex properties. Such difficulties require agents to be highly efficient in their learning (and, consequently, decision-making) and environmental perception processes, without which they will not be successful. The digital game FIFA - soccer simulator - is used as a case study because it represents a realistic and challenging environment. The ML techniques are investigated in the context of the DL provided by Convolutional Neural Networks (CNNs), being: imitation learning, used here with the purpose of endowing the agent with the ability to solve problems in a way closer to human; by deep reinforcement, in which the agent is trained in an attempt to autonomously abstract an optimal decision-making policy. Regarding the environmental perception, the following state representations approaches are investigated in this study: raw images - with and without color information - and through Object Detection Techniques (ODT). In order to further improve the performance of the agents produced, genetic algorithm techniques are explored to automatically define a CNN architecture to be used as the player agents decision-making module. In addition to corroborating the excellent results that DL combined with CV has been producing in the context of ML (particularly in games), the present work shows the great potential of the application of ODT in the process of echancing the environmental perception, which counts as a relevant counterpart to the fact that ODT demands computational procedures with a higher cost in relation to the representations based on raw images.

Keywords: Machine Learning. Deep Learning. Computer vision. Object Detection Techniques. Imitation Learning. Deep Reinforcement Learning. Convolutional Neural Networks. Genetic Algorithms.

Lista de ilustrações

Lista de tabelas

Lista de siglas

A3C *Asynchronous Advantage Actor-Critic*

AG Algoritmo Genético

AG-RNC Algoritmo Genético para Rede Neural Convolucional

AG-RNC-M Algoritmo Genético para obtenção de Redes Neurais Convolucionais Mínimas

AI Aprendizado por Imitação

AM Aprendizado de Máquina

AP Aprendizado Profundo

AR Aprendizagem por Reforço

ARP Aprendizado por Reforço Profundo

DQN *Deep Q-Networks*

ID Imitação Direta

IA Inteligência Artificial

IAP Imitação Ativa Profunda

PDM Processo de Decisão de Markov

PMC Perceptron de Múltiplas Camadas

PLN Processamento de Línguas Naturais

RGB *Red Green Blue*

RNA Rede Neural Artificial

RNC Rede Neural Convolucional

RNPs Redes Neurais Profundas

TDOs Técnicas de Detecção de Objetos

VC Visão Computacional

Sumário

Capítulo 1

Introdução

1.1 Contextualização

O Aprendizado de Máquina (AM) consolidou-se como um dos pilares para o avanço da Inteligência Artificial (IA) pelo seu sucesso na investigação de abordagens eficazes para dotar agentes automáticos com habilidades essenciais que os tornem aptos a resolver problemas da vida moderna com a máxima autonomia possível (RUSSELL; NORVIG, 2013). Dentre esses problemas, destacam-se, por exemplo, os sistemas autônomos, os sistemas preditivos em geral, o reconhecimento de imagens e os jogos digitais (LIU et al., 2017). Por meio do AM é possível projetar e construir agentes que, ao lidar com os problemas para os quais foram concebidos, conseguem tomar decisões, realizar previsões e aprender a partir de dados coletados ou de experiências vivenciadas sem que essas habilidades tenham sido previamente programadas. Para tanto, é fundamental que os agentes contem com técnicas robustas de percepção do ambiente em que operam e de métodos adequados que processam eficientemente tal percepção a fim de prover uma tomada de decisão com boa qualidade. Em tal cenário de AM, destacam-se os resultados que vêm sendo obtidos pelo Aprendizado Profundo (AP) (LECUN; BENGIO; HINTON, 2015).

O AP é uma abordagem que permite aos agentes a compreensão do mundo (ou ambiente) em termos de uma hierarquia de conceitos (GOODFELLOW; BENGIO; COURVILLE, 2016). Neste sentido, as Redes Neurais Profundas (RNPs) - destacando-se a Rede Neural Convolucional (RNC) - têm apresentado um excelente nível de desempenho em tarefas envolvendo dados visuais, especialmente no campo de reconhecimento de imagem (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; HE et al., 2016). Com isso, é possível obter um avanço significativo na utilização de Visão Computacional (VC) para aprimorar a capacidade que os agentes possuem de perceber e entender o ambiente em que atuam. A VC é uma ciência que estuda as informações de interesse em uma imagem (LULIO, 2011). Particularmente, as Técnicas de Detecção de Objetos (TDOs) são um subconjunto proeminente de VC que estimam os conceitos e localizações dos elementos contidos em

cada imagem para obter informações valiosas e assim fornecer uma melhor compreensão sobre os dados visuais (ZHAO et al., 2018). A VC permite abstrair um alto nível de entendimento das informações de baixo nível - comumente conhecidas como imagens cruas ou *pixels* brutos - produzidas pelas imagens digitais. Tais informações de baixo nível são amplamente utilizadas como representação de ambiente em agentes automáticos pela sua simplicidade (por exemplo, a captura de tela corrente do jogo, elemento enxergado por agentes humanos no contexto de jogos digitais) e por serem consideravelmente apropriadas pelas arquiteturas tomadoras de decisão comumente baseadas em RNPs. Dessa forma, a combinação de técnicas de VC e de métodos de AP tem resultado em extraordinários avanços em habilidades de máquina tais como segmentação semântica e detecção de objetos (ZHAO et al., 2018).

Neste sentido, os jogos digitais destacam-se como um estudo de caso no contexto das pesquisas de AM pelas seguintes razões:

- ❑ Apresentam um grande impacto no cenário econômico atual. A indústria de jogos é considerada nova - estabelecida por volta de 1970 a partir do lançamento do Atari - em comparação a outras indústrias tradicionais de entretenimento, tais como as indústrias de cinema, música, livros, entre outras (SEBASTIAN, 2017). No entanto, o seu mercado global foi estimado no valor de 152 bilhões de dólares em 2019 (WIJMAN, 2019) e é esperado que esse valor continue crescendo, impulsionado pelo crescente avanço tecnológico e pelo crescimento no público geral de jogos (abrangendo cerca de 2,5 bilhões de pessoas em todo o mundo, especialmente em sites de redes sociais *on-line* e dispositivos móveis (WIJMAN, 2019)). Destaca-se que tamanha popularidade foi fundamental para o surgimento de uma nova modalidade de torneios envolvendo jogos eletrônicos, o chamado (*eSports*), o qual consolidou-se como um aspecto importante do consumo no mundo virtual, propiciando premiações milionárias aos competidores (SEO, 2013). A categoria de *eSports* é muito fragmentada no sentido de prover um número diversificado de gêneros que podem ser utilizados competitivamente, desde jogos de corrida até jogos de tiro em primeira pessoa. Como exemplos, pode-se mencionar os jogos *Dota*, *League of Legends*, *Starcraft*, *HearthStone*, *CS:Go*, *Rocket League* e *FIFA*. A popularidade e o surgimento de serviços de *streaming on-line* - destacando-se a plataforma *Twitch*, pertencente à gigante empresa *Amazon* - também ajudaram no crescimento da indústria e são o método mais comum de assistir a torneios de *eSports* (GAMING, 2018).

- ❑ Envolvem dificuldades técnicas muito semelhantes às enfrentadas pelos agentes que lidam com problemas da vida real (TOMAZ, 2019). No entanto, os jogos permitem a investigação de técnicas de AM de um modo mais controlado, sem o risco de se ter consequências negativas caso fossem aplicadas diretamente ao mundo real. Assim sendo, as mesmas técnicas aplicadas a jogos podem também ser utilizadas

com êxito, por exemplo, para tratar problemas práticos tais como os seguintes: 1) *Interação com humanos por meio de um diálogo.* Cada vez mais, a vida moderna demanda agentes que dialoguem com humanos (como, por exemplo, os assistentes virtuais eletrônicos, também conhecidos como *chatbots*, em empresas de prestação de serviços). Estes assistentes virtuais tornaram-se muito populares pela capacidade de simular um comportamento humano por meio de comunicação de texto e voz a partir do uso de técnicas de AP (SHETH et al., 2019). Um dos principais desafios na construção de agentes que atacam esse problema é a questão envolvendo o Processamento de Línguas Naturais (PLN) (CHOWDHURY, 2003). Este desafio envolve prover um agente com a habilidade de compreender a linguagem não só no âmbito sintático, mas principalmente no campo semântico. Neste contexto, há uma importância muito grande de jogos baseados em texto, os quais são amplamente utilizados para os avanços das pesquisas na área de PLN (PICCA; JACCARD; EBERLÉ, 2015; YAO et al., 2019); 2) *Sistemas de navegação autônomos.* Neste caso, a tarefa de aprendizagem parte de um ponto de referência inicial em que um sistema (por exemplo, robô ou carro) deve aprender uma trajetória de navegação de modo a atingir um ponto alvo e, ao mesmo tempo, desviar dos obstáculos do ambiente. Neste sentido, diversos trabalhos utilizam jogos de corrida e ambientes virtuais realísticos de navegação com a capacidade de simular tanto as características físicas do mundo real quanto os aspectos relativos aos veículos automotivos como laboratório de pesquisa para o desenvolvimento e validação de técnicas de AM antes de aplicar, de fato, em ambientes reais (WYMANN et al., 2015; DOSOVITSKIY et al., 2017). Por exemplo, um dos métodos de aprendizado utilizados no presente trabalho de Mestrado - denominado Imitação Ativa Profunda (HUSSEIN et al., 2018) - foi proposto e validado em um ambiente de navegação 3D (ABBET, 2014); 3) *Análise de Imagens Médicas.* Neste contexto, as técnicas de aprendizagem têm tido fundamental importância no aprimoramento da qualidade e da precisão nas tarefas de diagnóstico de doenças por meio de classificações efetuadas automaticamente em imagens e em exames, além da detecção de objetos e lesões, entre outros (LITJENS et al., 2017). Um boa parte desse sucesso consiste na utilização de RNPs que contam com habilidades relacionadas à VC. Tais habilidades permitem com que a identificação dos elementos e das características relevantes às tarefas de classificação de imagens sejam mais rápidas e mais precisas em comparação ao trabalho que teria que ser feito manualmente por especialistas humanos, apesar de ainda apresentarem desafios quanto a interpretabilidade clínica (KERMANY et al., 2018). Assim, os jogos exercem um grande papel na exploração da VC acoplada ao processo de aprendizagem pelo fato de apresentarem um rico domínio para se trabalhar com dados visuais. Além disso, há também uma categoria de pesquisas que utilizam jogos aplicados na área da medicina denominada Jogos para a Saúde (do

inglês, *Games for Health*) (WATTANASOONTORN et al., 2013; BARANOWSKI et al., 2016).

O presente trabalho de Mestrado, dentre as principais abordagens de aprendizado, tem o foco na investigação de duas vertentes: Aprendizagem Supervisionada, quando a experiência dos agentes é adquirida pela influência direta de um humano ou sistema especialista (por exemplo, por uma base de dados construída por humanos), e Aprendizagem por Reforço (AR), quando os agentes aprendem de maneira autônoma no ambiente (neste caso, a experiência é adquirida a partir da interação com tal ambiente). Independente do tipo de aprendizado, as experiências dependem da percepção de como tais agentes estão localizados no ambiente (estado). Esta percepção reflete as principais características do ambiente e a sua representação exerce impacto direto na qualidade do aprendizado de um agente. A partir dela, os agentes devem ser capazes de detectar o conjunto de ações legais que eles podem considerar em suas tomadas de decisão, bem como estar cientes, o máximo que for possível, das alterações que a execução de cada uma delas irá causar ao ambiente. Nesta situação, os agentes devem ser treinados a fim de aprender como escolher as ações apropriadas que os levem a alcançar seus objetivos. Assim sendo, em geral, os principais desafios envolvendo a construção de agentes inteligentes consistem em propiciar uma percepção adequada do ambiente baseada nas características relevantes aos problemas abordados; contar com um método de tomada de decisão apropriado que, com base na percepção de ambiente, efetue ações que os levem a atingir os seus objetivos; e da escolha adequada do tipo de aprendizado com o objetivo de aprimorar o tomador de decisão.

Destaca-se que uma parcela das pesquisas de IA aplicada a jogos, especialmente aquelas referentes à implementação de agentes jogadores inteligentes, têm tido como foco o estudo de técnicas de AM em jogos de tabuleiro clássicos, como Damas, Xadrez e Go. Tais jogos proporcionam um rico e didático domínio para o estudo de métodos de aprendizado pelo fato de prover ambientes completamente observáveis, determinísticos, estáticos e conhecidos (RUSSELL; NORVIG, 2013). Estas características permitem um controle maior sobre o ambiente e, consequentemente, proveem uma melhor e mais fácil compreensão das técnicas estudadas. Apesar de tais fatos, esses jogos englobam aspectos desafiadores, tais como: a complexidade do espaço de estados - definida pelo número de posições legais possíveis no jogo, que podem ser alcançadas a partir do seu estado inicial; a complexidade da árvore de jogo - definida pelo número de folhas na árvore de busca para a solução da posição atual (ou estado). Neste contexto, diversos agentes jogadores têm sido propostos pela comunidade científica internacional tanto no contexto da aprendizagem supervisionada quanto no contexto da AR (RUSSELL; NORVIG, 2013). Por exemplo, os atuais campeões mundiais de Xadrez e de Go em campeonatos homem-máquina são o *AlphaZero* (SILVER et al., 2017a) e o *AlphaGo Zero* (SILVER et al., 2017b). Saliente-se que a equipe na qual o presente trabalho de Mestrado é desenvolvido também vem produzindo diversas

pesquisas no campo de AR relacionadas aos jogos de Damas e de Go (AGUIAR; JULIA, 2013; JUNIOR; JULIA, 2014; SANTOS; JULIA, 2016; TOMAZ; FARIA; JULIA, 2017; FARIA; JULIA; TOMAZ, 2018; TOMAZ; JULIA; DUARTE, 2018; NETO; JULIA, 2018; TOMAZ, 2019).

No entanto, problemas do mundo real geralmente envolvem propriedades estocásticas, dinâmicas e parcialmente observáveis - aspectos mais complexos não alcançados pelos referidos jogos de tabuleiro. Assim, maiores desafios no campo da IA são investigados por meio de pesquisas envolvendo outros tipos de jogos - tais como *Dota 2*, *Starcraft* e *FIFA* - ao longo da última década e meia (YANNAKAKIS; TOGELIUS, 2018). Ressalta-se que uma parte das pesquisas considerando este domínio se concentra no desenvolvimento de agentes inteligentes com a capacidade de abstrair/aprender uma política ótima ou próxima dela nos jogos para os quais são projetados. Em contraste, uma outra parte tem a preocupação de projetar agentes com um comportamento mais humano (agentes que atuam de uma maneira mais próxima do modo de agir humano). Porém, independentemente do propósito com que eles são construídos, todas essas pesquisas recentes investigam o uso de técnicas de AP+VC acopladas ao processo de aprendizagem para aprimorar as habilidades de jogo de tais agentes (MNIH et al., 2015; MNIH et al., 2016; OPENAI, 2018; TRIVEDI, 2018a; HARMER et al., 2018; VINYALS et al., 2019).

Neste sentido, a inclusão de métodos de AR no contexto de AP produziu um novo ramo de pesquisa de sucesso denominado Aprendizado por Reforço Profundo (ARP) - do inglês *Deep Reinforcement Learning* (MOUSAVI; SCHUKAT; HOWLEY, 2018a). No ARP, o agente aprende de acordo com sua própria experiência, sendo guiado por uma função de avaliação que estima o quão bom é seu nível de atuação (representada por recompensas). Assim, agentes baseados em ARP são treinados sem contar com supervisão humana. Tal independência de aprendizado com respeito ao conhecimento humano tem permitido a construção de agentes que alcançam e até superam o nível de jogo humano em diversas tarefas dentro do domínio de jogos. Contudo, o treinamento de agentes por meio de métodos de ARP pode ser muito oneroso em termos de recursos computacionais (tais como tempo e memória), especialmente em problemas complexos que requerem a realização de longas sequências de ações para se alcançar o objetivo (HUSSEIN et al., 2017). Muitos problemas envolvendo jogos se enquadram nesse tipo de complexidade. É por isso que muitos agentes de sucesso existentes baseados em ARP foram primeiramente concebidos de acordo com uma abordagem supervisionada - destacando-se o Aprendizado por Imitação (AI) - e, em seguida, aprimorados por meio de AR. Um famoso exemplo é o atual campeão mundial do jogo *Go* citado anteriormente - o agente jogador *AlphaGo Zero* (SILVER et al., 2017b) - o qual foi treinado somente por meio de *self-play*, teve como versão preliminar o agente *AlphaGo* (SILVER et al., 2016), que, por sua vez, foi treinado por meio de supervisão humana (combinado com AR).

O AI é uma estratégia de treinamento supervisionada na qual os agentes aprendem

observando o comportamento de humanos ou sistemas especialistas que exercem o papel de supervisores (ao invés de ser guiado por recompensas produzidas por uma função de avaliação). Neste caso, a política assimilada pelo agente é altamente relacionada às demonstrações que representam como o especialista lidaria com a mesma tarefa abordada por tal agente (HUSSEIN et al., 2017). Consequentemente, um agente treinado por AI tende a resolver o seu problema de uma maneira mais próxima ao modo de agir humano. Isto implica que o agente é limitado ao desempenho do especialista, isto é, supõe-se que a performance do agente será no máximo tão boa quanto a do seu supervisor. Todavia, o caráter supervisionado do AI aumenta a eficiência do treinamento de agentes inteligentes em termos de tempo computacional.

Assim, incorporando os aspectos e as habilidades mencionadas, foi possível projetar agentes automáticos que obtiveram extraordinários desempenhos e grande respeito no cenário mundial competitivo de jogos. Dentre eles, destacam-se os renomados *OpenAI Five* (OPENAI, 2018) e o *AlphaStar* (VINYALS et al., 2019), implementados para atuar no *Dota 2* e no *Starcraft* respectivamente. Estes jogos são substancialmente populares abrangendo competições mundiais com premiações milionárias. Além disso, eles exigem que os jogadores tenham um ótimo nível de estratégia de jogo para serem bem-sucedidos em suas tarefas, uma vez que os referidos jogos envolvem outro agentes que atuam de modo a minimizar o impacto de suas ações. Desta forma, eles têm atraído uma enorme gama de pesquisas no contexto de AM, especialmente nos campos de AI e de ARP.

1.2 Motivação

Inspirados nas pesquisas citadas do estado da arte, o presente trabalho de Mestrado tem como motivação principal estudar e investigar ferramentas apropriadas de AM (incluindo técnicas de AP+VC) no processo de construção de agentes jogadores que lidam com ambientes complexos, os quais envolvem as seguintes propriedades: estocásticos, dinâmicos, multiagentes e desconhecidos. Para tanto, o jogo eletrônico *FIFA* - simulador de futebol, desenvolvido e publicado pela empresa *Eletronic Arts (EA Games)* - é utilizado como estudo de caso. Tal jogo, assim como o *Dota 2* e o *Starcraft*, compreende um laboratório muito desafiador e realista para o estudo de técnicas de AM e de representação de estado.

O *FIFA*, apesar de ser um jogo mundialmente popular e apresentar qualidades desafiadoras ao processo de construção de agentes jogadores, é ainda um estudo de caso pouco explorado em termos de pesquisas no contexto de AM comparado a outros títulos de sucesso, tais quais os já mencionados *Dota 2* e *Starcraft*. As principais razões para isso são: 1) o fato de estes jogos possuírem interfaces amigáveis específicas para o desenvolvimento de pesquisas científicas, resultado da parceria das grandes empresas *Google* e *OpenAI* com as produtoras dos referidos jogos (*Blizzard* - associada ao *Starcraft* - e *Valve* - *Dota*

2). Tais interfaces disponibilizam um fácil acesso à utilização de diversas representações de ambiente, tais como as imagens cruas dos cenários de jogo e também outras simplificadas que correspondem a um conjunto de características extremamente atreladas aos problemas e manualmente desenvolvidas por especialistas. Além disso, estas interfaces são projetadas para que os agentes desenvolvidos sejam facilmente conectados e acoplados à plataforma dos jogos. Assim, elas facilitam e agilizam o avanço das pesquisas. Destaca-se que não há uma interface similar existente integrada ao *FIFA*; 2) por não existir tal interface para o referido jogo, a principal fonte de representação de ambiente são as observações produzidas por meio da manipulação e processamento das imagens do jogo. Isto é justificado pelo fato dos estados reais de jogo - representados pelas imagens cruas - não serem acessíveis, uma vez que informações relevantes sobre eles são subjacentes ao mecanismo interno do jogo (por exemplo, a posição real de um jogador está localizada em alguma posição de memória apesar de ser possível identificar tal jogador na imagem de jogo).

Em contrapartida aos fatos mencionados acima, o *FIFA* apresenta uma variedade muito maior de cenários interessantes com um grande potencial de serem explorados por pesquisas relacionadas a agentes automáticos em relação ao *Dota 2* e ao *Starcraft*. No caso do *Starcraft* - jogo de estratégia em tempo real (do inglês, *Real-Time Strategy*) - o ambiente estudado é constituído por dois agentes que atuam competitivamente, cujo principal objetivo é o gerenciamento de recursos e o controle de exército para destruir a base de seu oponente (VINYALS et al., 2019). No *Dota 2* - classificado como arena de batalha multijogador online (do inglês, *Multiplayer online battle arena*) - o ambiente estudado é constituído por um grupo de cinco agentes que interagem cooperativamente entre si e competitivamente contra um outro grupo composto por cinco agentes, cujo principal objetivo é também a destruição da base inimiga (OPENAI, 2018). Destaca-se que, em ambos os jogos, o ambiente pode sofrer algumas leves modificações aleatórias de configuração para cada partida, porém as suas principais características não se alteram. Já o *FIFA* possui um modo principal de jogo constituído por um ambiente competitivo entre dois agentes em que cada um deles controla um time de futebol composto por 11 jogadores. Além desse modo, o referido jogo conta com uma grande quantidade de ambientes distintos direcionados ao aprimoramento das habilidades de jogo dos usuários. Eles envolvem tarefas relacionadas à troca de passes entre jogadores, finalização ao gol, cobrança de faltas, dribles com obstáculos estáticos e dinâmicos, entre outros. É importante destacar que estes ambientes proveem diferentes aspectos, objetivos e níveis de complexidade a serem levados em consideração no processo de construção de agentes inteligentes.

Trivedi explora dois desses ambientes *FIFA* para o estudo de técnicas de ARP e de AI. Em (TRIVEDI, 2018b), o referido autor combina o uso de TDO com ARP - especificamente, o método *Deep Q-Learning* ou *Deep Q-Networks* (DQN) (MNIH et al., 2015) - para desenvolver um agente capaz de efetuar cobrança de faltas em um cenário estático

que não envolve a presença de um goleiro. A TDO foi utilizada para realizar a detecção dos principais elementos particulares a esse problema (dentre eles, o gol e a bola) a fim de gerar uma percepção de ambiente processada pelo módulo de tomada de decisão do agente, constituído por uma RNP. Desta forma, a tomada de decisão foi aprimorada pelo método de aprendizagem DQN a partir da interação do agente com o ambiente, isto é, tal agente foi aperfeiçoando as suas habilidades conforme mais faltas ele cobrava. Já em (TRIVEDI, 2018a), Trivedi combina o uso de TDO com o aprendizado supervisionado (por meio de técnicas de AI) para projetar um agente com a habilidade atuar no modo principal do *FIFA*. Destaca-se que o agente resultante foi capaz de derrotar o nível mais fácil da máquina do referido jogo.

Assim sendo, a presente pesquisa de Mestrado expande as pesquisas no contexto de AM realizadas por Trivedi no jogo *FIFA* utilizando como estudo de caso os seguintes cenários de jogo: 1) tarefa de cobrança de faltas a qual não envolve a presença de um goleiro (cobrança de faltas sem goleiro). Destaca-se que este cenário é estático e agente único; 2) tarefa de cobrança de faltas a qual envolve a presença de um goleiro (cobrança de faltas com goleiro). Este cenário é estático e multiagente; 3) modo de confronto, o qual representa um ambiente constituído por dois agentes que atuam de modo competitivo (cada agente controla uma equipe composta por dois jogadores), cujo principal objetivo é marcar o maior número de gols e sofrer o menor número de gols possível em um determinado tempo. Ressalta-se que este cenário é dinâmico e multiagente. É importante notar que os cenários estudados apresentam graus de complexidade que aumentam gradativamente, ou seja, o primeiro deles (cobrança de faltas sem goleiro) é o que apresenta características e dificuldades mais simples, enquanto que o último cenário (modo de confronto) é aquele que engloba maiores desafios técnicos. No entanto, em todos eles a propriedade de ambiente completamente observável, desconhecido e o caráter estocástico do *FIFA* (aleatoriedades intrínsecas ao simulador de jogo) estão amplamente presentes. Apesar da primeira constituir uma propriedade simples, as outras duas são muito desafiadoras na construção de agentes inteligentes. Dessa forma, utilizam-se os referidos cenários de jogo para realizar as seguintes investigações: 1) técnicas eficientes de aprendizagem de acordo com as abordagens de AI e de ARP; 2) métodos de tomada de decisão baseados em RNP apropriados aos problemas abordados; 3) representações de ambiente baseadas em imagens cruas e em TDOs, além da análise de seus impactos na qualidade da tomada de decisão ao longo do processo de aprendizagem de um agente. Tais investigações são resumidas na sequência.

Com relação às técnicas eficientes de aprendizagem investigadas, destaca-se o algoritmo DQN, um dos precursores do ramo ARP, o qual atingiu um desempenho sobre-humano em diversos jogos do console *Atari*. Pelas limitações já discutidas referentes à plataforma *FIFA*, há uma grande dificuldade de se obter ou modelar uma função de avaliação (função de recompensa) adequada e precisa para problemas de AR (especialmente

em ambientes dinâmicos e desconhecidos, como o modo de confronto). Por exemplo, um jogador executa a ação de passe para seu companheiro de time, contudo o *FIFA* não oferece um meio de se verificar exatamente o resultado de tal ação pelo mecanismo interno da plataforma, apenas por meio das imagens de jogo. Dessa forma, seria necessário utilizar algoritmos de processamento de imagem ou algum outro modo para realizar esse tipo de verificação. Isto resultaria em uma provável perda de desempenho por parte do agente jogador, uma vez que os recursos computacionais (processamento e memória) teriam que ser compartilhados entre ele o método de verificação, pois ambos executariam simultaneamente em tempo real. Por esta razão, o DQN é estudado apenas nas duas tarefas de cobrança de faltas, uma vez que, mesmo com as limitações do *FIFA*, não é inviabilizada a construção e treinamento de agentes automáticos baseados em ARP. Por isso, métodos supervisionados são explorados neste trabalho, destacando-se a Imitação Direta (ID) e a Imitação Ativa Profunda (IAP). A ID é um dos métodos mais difundidos de AI, também conhecido como clonagem comportamental (do inglês, *behavioral cloning*) (LIU et al., 2018), pela sua simplicidade (semelhante ao aprendizado supervisionado tradicional, no qual o objetivo é encontrar uma função válida que mapeia diretamente entradas em suas respectivas saídas). No entanto, a ID apresenta grandes problemas de generalização em cenários complexos. Dentre os métodos que tentam minimizar esta fragilidade, destaca-se a IAP, que combina a estratégia supervisionada usada na ID com aprendizado ativo (HUSSEIN et al., 2018).

Com relação aos métodos de tomada de decisão, o presente trabalho utiliza exclusivamente RNPs nos agentes aqui projetados - especificamente, Perceptron de Múltiplas Camadas (PMC) e RNC (a qual é uma variação de PMC). Elas compõem um grupo de ferramentas com a habilidade de reconhecer padrões a partir da implementação de métodos estatísticos (CAMPOS; GARCIA, 2019). Considerando o contexto supervisionado, as RNPs podem aprender a partir de um conjunto de demonstrações/exemplos de modo a identificar padrões de comportamento implícitos em tal conjunto. Além disso, elas podem ser muito eficazes em situações as quais não é possível definir os passos ou as regras que levam à solução de um problema. Sob a ótica do AR, elas possuem uma grande capacidade de extrair conhecimento a partir da interação com o ambiente, principalmente levando em consideração as técnicas de VC no aprimoramento da qualidade de percepção de tal ambiente (relacionadas às RNCs). Devido à sua alta capacidade de generalização, a escolha adequada da arquitetura de uma RNP depende das características da tarefa em que ela irá atuar. Assim, a presente pesquisa também explora técnicas evolutivas - no caso, Algoritmo Genético (AG) - no processo de construção de arquiteturas otimizadas de RNC. No caso, essa arquitetura se refere às quantidades de camadas e de neurônios em cada uma delas. Tal estratégia utiliza uma função de avaliação que faz um balanceamento entre a acurácia obtida com uma determinada configuração de RNC e a sua dimensão em termos do número de parâmetros (vetor de pesos e bias).

Por fim, é estudado o impacto da qualidade da percepção do ambiente ao longo do processo de aprendizagem e tomada de decisão em um agente. Neste contexto, são analisadas os seguintes tipos de representação: 1) dados visuais brutos (imagens cruas), considerando informação de cor *Red Green Blue* (RGB) ou sem levar em conta tal informação (escala de cinza); 2) oriundas de TDO. Neste caso, a representação proposta por Trivedi no *FIFA* (TRIVEDI, 2018a) é bastante investigada em termos teóricos e práticos. A partir das limitações encontradas em tal investigação, este trabalho também utiliza uma representação de TDO alternativa (VENKATESH, 2018) que tem como principais características a simplicidade e melhor interpretabilidade.

1.3 Objetivos da Pesquisa

Considerando os pontos apresentados na seção 1.2, o objetivo geral deste trabalho é investigar técnicas apropriadas de AM - mais especificamente, AP - combinadas a técnicas de representação do ambiente provenientes da VC, para produzir agentes capazes de resolver problemas, em tempo real, em ambientes completamente observáveis, estocásticos e desconhecidos que variam em função das seguintes propriedades: ambiente envolvendo agente único ou multiagente, estático ou dinâmico. Para tanto, o presente trabalho utiliza três cenários distintos do jogo *FIFA* como laboratório de estudo de tais técnicas para o desenvolvimento de agentes automáticos. Assim, cada um dos cenários abordados por este trabalho serão investigados com base em duas óticas: 1) Análise da estratégia de aprendizagem mais apropriada (supervisionada - por meio do AI; ARP); 2) Investigação do modo mais adequado de se representar o ambiente (representação por imagens cruas do cenário de jogo; representação oriunda de TDOs).

Desta forma, a fim de perfazer o objetivo geral proposto aqui, os seguintes objetivos específicos devem ser obtidos:

1. Desenvolvimento de uma interface de conexão para integrar os agentes implementados baseados tanto na ótica de ARP quanto de AI à plataforma de jogo *FIFA*.

2. Investigação das seguintes abordagens considerando os cenários *FIFA* de cobrança de faltas sem e com goleiro: 1) agentes automáticos implementados por meio do método DQN e por representações de ambiente baseadas nos dados visuais brutos (considerando informação de cor e sem considerar tal informação); 2) agentes automáticos baseados no DQN e com percepção de ambiente proveniente do uso de TDOs.

3. Investigação das seguintes abordagens para o estudo de caso considerando os cenários *FIFA* de cobrança de faltas sem e com goleiro: 1) agentes automáticos implementados por meio do IAP (método de AI) e por representações de ambiente

baseadas nos dados visuais brutos (considerando informação de cor e sem considerar tal informação); 2) agentes automáticos baseados no IAP e com percepção de ambiente proveniente do uso de TDOs.

4. Investigação da seguinte abordagem de aprendizado considerando o cenário *FIFA* denominado Modo Confronto: agentes automáticos implementados por meio dos métodos ID e IAP (referentes ao AI) e por representação de ambiente baseada nos dados visuais brutos sem considerar informação de cor.

5. Investigação de uma abordagem evolutiva baseada em AG com o propósito de definir arquiteturas otimizadas de RNCs utilizadas como módulo de tomada de decisão de agentes inteligentes. Como o cenário *FIFA* modo de confronto é o mais complexo dentre todos os analisados até o momento (pelo fato de englobar um ambiente dinâmico, multiagente e desconhecido), isso motiva a utilização de técnicas de IA para aprimorar a tomada de decisão baseada na representação do ambiente por meio de um mecanismo com a capacidade de encontrar, automaticamente, uma arquitetura de rede neural apropriada para o problema. Neste contexto, as representações baseadas em imagens cruas (sem e com cor) são investigadas.

1.4 Hipóteses

As hipóteses desta pesquisa estão relacionadas, principalmente, com o aprimoramento do processo de escolha de ações dos agentes que aqui serão implementados por meio de melhorias nos mecanismos de aprendizagem e nas representações de estado (percepção dos agentes). Assim, são hipóteses deste trabalho:

1. A forma mais utilizada de se representar os estados em jogos digitais é o dado visual bruto (imagem crua ou *pixels* brutos). Esta forma de representação pode considerar informação de cor - imagem colorida - ou não considerar tal informação - imagem em escala de cinza. Esta última representação exige menos poder computacional para ser processada. Uma forma alternativa é prover uma representação por meio de TDOs, a qual exige um maior poder computacional de processamento. Assim, é esperado que as imagens brutas com informação de cor e as representações baseadas em TDO provenham uma percepção de estado com maior qualidade em relação àquelas que não levam em consideração a informação de cor (imagens em escala de cinza).

2. A qualidade da tomada de decisão de um agente jogador está bastante relacionada à representação utilizada do estado de jogo. Neste sentido, é muito importante que um agente conte com uma arquitetura (no caso deste trabalho, baseada em RNP) apropriada para o problema o qual está resolvendo. O método IAP (descrito na

subseção 2.3.2) compreende uma arquitetura de RNC fixa como módulo tomador de decisão. No entanto, dependendo da complexidade e das particularidades do problema a ser resolvido, tal arquitetura deve ser adaptada de modo a tornar-se mais adequada e eficiente ao problema. Assim, é esperado que a definição automática de arquitetura proposta neste trabalho obtenha resultados competitivos e até melhores em relação à arquitetura fixa utilizada pelos métodos supervisionados aqui estudados no contexto de agentes jogadores.

1.5 Contribuições

As principais contribuições do presente trabalho são:

- Criação de uma interface de conexão que permite a conexão dos agentes implementados com a plataforma de jogo *FIFA*. Destaca-se que ela tem um caráter genérico no sentido de poder ser utilizada não só para o *FIFA*, mas para auxiliar futuros trabalhos que lidam com jogos comerciais que não possuem uma interface adequada integrada para estudos científicos.

- Criação de agentes jogadores para os cenários de cobrança de faltas baseados no método DQN e implementados considerando distintas formas de representação (baseadas tanto em imagens cruas quanto em TDO).

- Criação de agentes jogadores para os cenários de cobrança de faltas baseados no método IAP e implementados considerando distintas formas de representação (baseadas tanto em imagens cruas quanto em TDO).

- Criação de agentes jogadores para o modo de confronto baseados nos métodos ID e IAP, implementados considerando distintas formas de representação (baseadas em imagens cruas).

- Obtenção de um método automático com a capacidade de gerar arquiteturas de RNC otimizadas utilizadas como módulo de tomada de decisão em agentes jogadores.

1.6 Produções Científicas

Os principais resultados desta pesquisa em termos de publicação científica são:

- Artigo *A Deep Reinforcement Learn-Based FIFA Agent with Naive State Representations and Portable Connection Interfaces* publicado na conferência internacional *The 32nd International FLAIRS Conference* (FARIA; JULIA; TOMAZ, 2019b);

- Artigo *Deep Active Imitation Learning in FIFA Free-Kicks Player Platforms based on Raw Image and Object Detection State Representation* publicado na conferência internacional *31st International Conference on Tools with Artificial Intelligence* (FARIA; JULIA; TOMAZ, 2019a);
- Artigo *Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents* publicado na conferência internacional *18th IEEE International Conference on Machine Learning and Applications* (FARIA; JULIA; TOMAZ, 2019c);
- Resumo intitulado "Aprimorando o Desempenho na Cobrança de Faltas de Agentes Jogadores de FIFA por meio de Técnicas de Detecção de Objetos" publicado nos anais do XIII Workshop de Teses e Dissertações em Ciência da Computação (WTDCC 2019), premiado como segundo melhor trabalho de mestrado.

1.7 Etimologia dos Nomes dos Agentes Propostos neste Trabalho

A etimologia adotada para nomear os agentes implementados pelo presente trabalho tem o seguinte padrão: M_i-C_j-*Agent*, onde M_i corresponde ao método i e C_j representa o cenário j. Neste sentido, considera-se o conjunto de métodos {DQN (i=1), ID (i=2) e IAP (i=3)} e o conjunto de cenários {Faltas sem Goleiro representado pela sigla SG (j=1), Faltas com Goleiro representado pela sigla CG (j=2) e Modo de Confronto representado pela sigla MC (j=3)}. Alguns exemplos de agentes:

1. *DQN-SG-Agent*: corresponde ao agente implementado com DQN para o cenário de faltas sem goleiro.
2. IAP-*CG-Agent*: corresponde ao agente implementado com IAP para o cenário de faltas com goleiro.
3. IAP-*MC-Agent*: corresponde ao agente implementado com IAP para o cenário modo de confronto.

1.8 Organização da Dissertação

Os próximos capítulos deste trabalho estão organizados conforme disposto a seguir:

Capítulo 2 apresenta todo o referencial teórico necessário para o entendimento da presente pesquisa.

Capítulo 3 apresenta o estado da arte dos trabalhos correlatos.

Capítulo 4 apresenta o desenvolvimento da interface de conexão com o jogo *FIFA* aqui construída.

Capítulo 5 descreve as técnicas e estruturas utilizadas para a construção dos agentes baseados em ARP nos cenários de cobrança de faltas.

Capítulo 6 descreve as técnicas e estruturas utilizadas para a construção dos agentes baseados em AI nos cenários de cobrança de faltas.

Capítulo 7 descreve as técnicas e estruturas utilizadas para a construção dos agentes baseados em AI no cenário modo de confronto.

Capítulo 8 descreve o sistema de definição automática de arquitetura otimizada de RNC baseado em AG aplicado ao cenário modo de confronto.

Capítulo 9 apresenta a conclusão e as perspectivas de trabalhos futuros.

Capítulo 2

Referencial Teórico

Este capítulo apresenta os principais conceitos das técnicas de IA utilizadas na construção do presente trabalho.

2.1 Agentes Inteligentes

Segundo (RUSSELL; NORVIG, 2013), um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. A Figura 1 ilustra a interação entre um agente e seu ambiente de atuação.

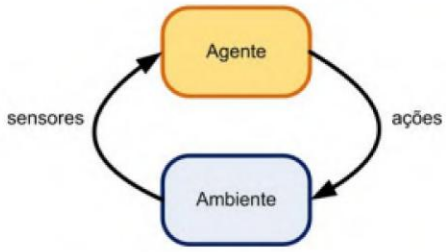


Figura 1 – Agentes interagem com ambientes por meio de sensores e atuadores (TOMAZ, 2013)

Em particular, conforme apresentado na Figura 2, este trabalho considera que um agente inteligente é composto por três módulos principais: Representação; Tomada de Decisão; e Aprendizagem. O primeiro módulo é estritamente relacionado com a percepção do agente sobre o ambiente, processando as informações capturadas pelos sensores de modo a gerar uma representação adequada. O segundo módulo é responsável por decidir as ações a serem realizadas no ambiente baseando-se nas representações geradas pelo primeiro módulo. Na presente pesquisa, tal módulo é representado por RNPs. Por fim, o último módulo tem a função de permitir que o agente tenha a habilidade de aprimorar a sua tomada de decisão baseado em exemplos fornecidos por um supervisor ou a partir de sua própria experiência.

Figura 2 – Arquitetura geral dos agentes inteligentes considerados neste trabalho

Destaca-se que a detecção das principais características do ambiente é uma tarefa fundamental para projetar agentes inteligentes eficientes. Desta forma, (RUSSELL; NORVIG, 2013) sugere as seguintes propriedades para a categorização de ambientes:

Completamente Observável *versus* Parcialmente Observável: um ambiente é completamente observável se o agente consegue obter informações completas e precisas sobre os estados do ambiente relevantes para a tomada de decisão. Caso contrário, diz-se que o ambiente é parcialmente observável. Destaca-se que todos os ambientes estudados pelo presente trabalho de Mestrado são completamente observáveis.

Agente Único *versus* Multiagente: um ambiente é considerado agente único se não há outros agentes interagindo nele. Caso contrário, o ambiente é considerado multiagente. Neste caso, os agentes podem atuar de forma cooperativa (isto é, eles trabalham em conjunto para se chegar a um objetivo maior) ou de modo competitivo (isto é, as tomadas de decisão de cada um visam minimizar o resultados das ações dos outros agentes).

Determinístico *versus* Estocástico: um ambiente é determinístico se o próximo estado é completamente determinado pelo estado corrente e pela ação executada pelo agente. Caso contrário, ele é estocástico. Destaca-se que todos os ambientes estudados pelo presente trabalho de Mestrado são estocásticos.

Episódico *versus* Sequencial: em um ambiente episódico, a experiência do agente é dividida em episódios atômicos. Cada episódio consiste na percepção do agente, seguida da execução de uma única ação. É crucial que o episódio seguinte não dependa das ações executadas em episódios anteriores. Por outro lado, em ambientes sequenciais, a decisão atual pode afetar todas as decisões futuras. No presente trabalho, todos os cenários são sequenciais.

Estático *versus* Dinâmico: se enquanto o agente estiver deliberando o ambiente puder sofrer alterações, este é considerado dinâmico. Caso contrário, ele é considerado estático para o agente.

Discreto *versus* Contínuo: um ambiente é discreto se existir um número fixo e finito de estados e as ações. Por exemplo, as representações de estado utilizadas pela presente pesquisa (imagens brutas ou TDO) são consideradas discretas. No caso das imagens brutas, há um imenso espaço de estados a ser considerado por conta da distribuição dos *pixels*, todavia, tal espaço é finito. Além disso, as ações também são um número finito, portanto, discreto. Caso contrário, quando o agente deve lidar com grandezas contínuas sejam elas ligadas aos estados do ambiente, às percepções ou às ações, o ambiente é dito contínuo.

Conhecido *versus* Desconhecido: essa distinção se refere ao conhecimento do agente sobre as "leis da física" do ambiente. Em um ambiente conhecido, os resultados (ou probabilidades de resultados, se o ambiente for estocástico) são fornecidos para todas as ações. Caso contrário, em um ambiente desconhecido, o agente precisará aprender como a dinâmica funciona para tomar boas decisões. No contexto deste trabalho, a propriedade de um ambiente ser desconhecido está extremamente atrelada às características de direção e movimento da bola e dos jogadores (isto é, relacionada à cinética do ambiente). Todos os cenários aqui estudados são desconhecidos na ótica dos agentes automáticos.

2.2 Redes Neurais Artificiais

Rede Neural Artificial (RNA) constitui uma técnica da IA que se inspira no modelo biológico do cérebro humano para a inteligência (ARBIB, 2003). Ela é formada por um número individual de elementos simples (denominados de neurônios) que se interconectam uns aos outros, formando redes capazes de armazenar e transmitir informações vindas do exterior.

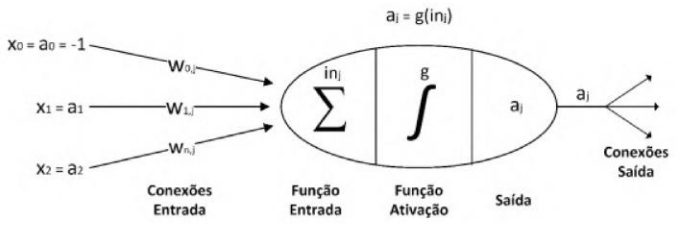


Figura 3 – Célula neural artificial proposta em (MCCULLOCH; PITTS, 1988)

O primeiro modelo matemático de um neurônio foi proposto em (MCCULLOCH; PITTS, 1988), ilustrado na Figura 3, composto pelos seguintes elementos:

1. Sinais de Entrada $\{x_1, x_2, ..., x_n\}$: sinais que representam os dados a serem computados (altamente relacionados ao problema a ser abordado);

2. Pesos Sinápticos $\{w_1, w_2, ..., w_n\}$: valores usados para ponderar cada entrada do neurônio (permite quantificar a relevância de cada entrada em relação à funcionalidade do neurônio);

3. Combinador linear $\sum$ (Função Entrada): efetua a operação de soma dos sinais de entrada ponderados pelos respectivos pesos sinápticos;

4. Limiar de Ativação θ: especifica o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção à saída do neurônio (sinal excitatório);

5. Potencial de ativação ou Sinal de Ativação u: diferença entre o resultado produzido pelo combinador linear e o limiar de ativação. Se $u \geq \theta$, o neurônio produz um potencial excitatório; caso contrário, o potencial será inibitório;

6. Função de Ativação ou Função de Transferência g: seu objetivo é limitar o sinal de saída do neurônio a valores adequados ao problema tratado pelo sistema neural (exemplos de funções: *sigmoide*, *ReLU* (*Rectified Linear Unit*) (NAIR; HINTON, 2010), *softmax*);

7. Sinal de Saída y ou a_j: valor final produzido pelo neurônio em relação ao conjunto de entradas, podendo ser utilizado por outros neurônios sequencialmente ligados ao neurônio gerador do sinal de saída.

Dessa forma, o conhecimento adquirido e mantido por uma RNA é inerente às conexões entre os diversos neurônios que a compõe (HAYKIN, 1999). Assim, as RNAs possuem uma alta capacidade de aprender e modelar problemas não-lineares e complexos, o que as permitem serem úteis nas tarefas de reconhecimento/classificação de padrões, no agrupamento de dados, em sistemas de previsão, na otimização de sistemas, entre outras.

O objetivo original de tal abordagem era criar um sistema computacional capaz de simular a modelagem do cérebro humano (sistema de processamento de informação altamente complexo) para a resolução de problemas (HEMANTH; ESTRELA, 2017). No entanto, pela dificuldade de se efetuar esta simulação de uma forma altamente fiel, o foco dos pesquisadores passou a ser a utilização de RNAs para resolver tarefas específicas, afastando-se de uma abordagem estritamente biológica. Assim sendo, elas possuem uma enorme importância nos avanços realizados em diversos campos científicos, tais como reconhecimento de fala, PLN, VC, imagens médicas e jogos digitais. Destaca-se que, em virtude da grande diversidade de arquiteturas de RNAs existentes na literatura, apenas aquelas de maior relevância para o trabalho proposto são abordadas, tais como: as Redes Neurais Multicamadas e as RNCs.

2.2.1 Redes Neurais Multicamadas

As Perceptron de Múltiplas Camadas (PMCs) são RNAs com arquitetura *feedforward* (isto é, os sinais de entrada se propagam em um único sentido: da camada de entrada para a camada de saída) que contém mais de uma camada de neurônios processadores (SILVA; SPATTI; FLAUZINO, 2010). Neste caso, as camadas processadoras dispostas entre a camada de entrada e a camada de saída são denominadas camadas ocultas, conforme ilustrado na Figura 4.

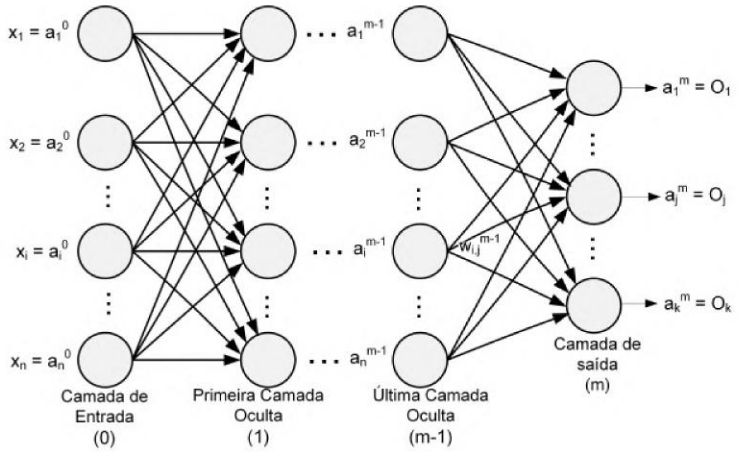


Figura 4 – Perceptron Multicamadas (TOMAZ, 2019)

Resumidamente, o treinamento de um PMC ocorre da seguinte maneira: as respostas produzidas pelas saídas da rede são comparadas com as respectivas respostas desejadas que estejam disponíveis. Mais especificamente, os desvios (erros) entre as respostas produzidas e as respostas desejadas produzidos pelos m neurônios de saída, durante cada época, nortearão o processo de ajuste de pesos sinápticos. Neste processo, o reajuste ocorrerá sob a estratégia de *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986), ou seja, os erros produzidos nos neurônios de saída para cada amostra desencadearão o seguinte fluxo: inicialmente, eles serão utilizados para reajustar os pesos das conexões entre a última camada oculta e a camada de saída; na sequência, serão ajustados os pesos das conexões entre a penúltima e a última camada oculta. Os ajustes continuam nessa dinâmica de retropropagação até que se tenham reajustado os pesos entre a camada de entrada e a primeira camada oculta.

2.2.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (RNCs), assim como as PMCs, são RNAs com arquitetura *feedforward*. Elas são muito apropriadas para lidar com tarefas envolvendo dados visuais (imagens) devido à grande capacidade de reconhecer padrões diretamente das imagens originais a partir de *pixels* brutos com pouco ou até nenhum pré-processamento (GU

et al., 2018). Dessa forma, elas são usadas (aplicadas) em classificação de imagens, bem como no reconhecimento de objetos como faces, placas e células cancerígenas em exames médicos. As RNCs têm proporcionado grandes contribuições na área do aprendizado profundo devido ao rápido crescimento na quantidade de dados anotados/rotulados e as melhorias nas eficiência do uso de unidades de processadores gráficos (PONTI; COSTA, 2018), com especial destaque para os avanços produzidos no enfrentamento de problemas relacionados a sistemas autônomos de navegação, robótica, drones e tratamentos para deficientes visuais.

Uma RNC tradicional é formada, a partir da entrada, por sucessivos pares de camadas neuronais de processamento, sendo que a primeira camada de cada um desses pares é denominada camada convolucional, ao passo que a segunda é chamada de camada de *pooling* (LECUN; BENGIO; HINTON, 2015). Tais pares possuem a habilidade de extrair características relevantes das imagens originais. A rede termina com uma sequência de camadas conhecidas como camadas totalmente conectadas (ou, completamente conectadas), a qual utiliza as características extraídas pelo último par (camada convolucional, camada de *pooling*) para efetuar algum tipo de tomada de decisão (como a classificação de uma imagem), conforme ilustrado na Figura 5. A quantidade ideal de camadas em uma RNC também depende das especificidades do problema com o qual ela lida. As camadas convolucionais, as camadas de *pooling* e as camadas totalmente conectadas são descritas na sequência.

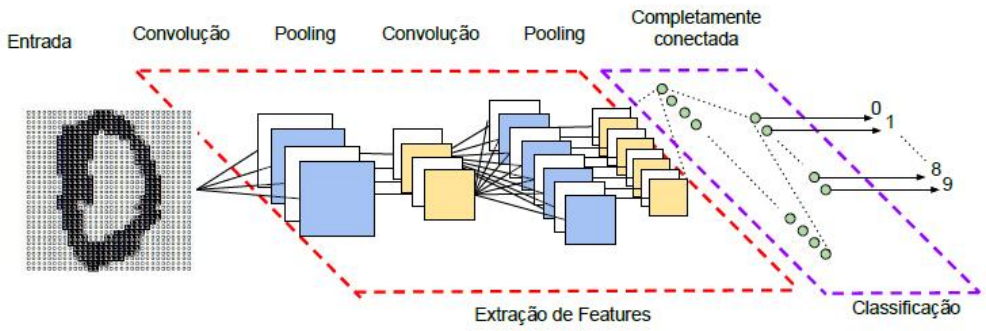


Figura 5 – Exemplo do processamento de uma imagem bruta em uma RNC (PENHA, 2018)

Camada Convolucional

A camada convolucional consiste, basicamente, da operação de convolução (função linear baseada em multiplicação de matrizes), a qual é responsável por varrer toda a imagem de entrada e aplicar um conjunto de filtros (*kernel*) com o objetivo de extrair mapas de características (*feature maps*), conforme ilustrado na Figura 6. Nela, um filtro (representando um conjunto de pesos) é aplicado como uma janela deslizante da esquerda para a direita e de cima para baixo sobre a imagem de entrada efetuando

a operação de convolução. Ao final desse processo, um mapa de características é produzido. É importante destacar que existem alguns hiperparâmetros relacionados à camada convolucional, descritos na Tabela 1.

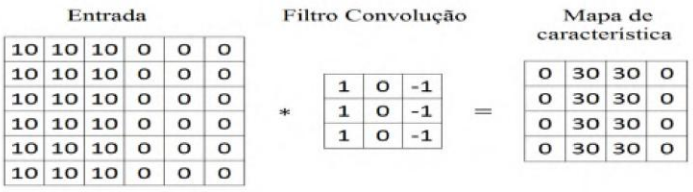


Figura 6 – Exemplo da operação de convolução (VARGAS; PAES; VASCONCELOS, 2016)

Tabela 1 – Hiperparâmetros de uma camada convolucional

Hiperparâmetro	**Significado**
Número de Filtros	Representa a quantidade de filtros utilizados na operação de convolução
Tamanho dos Filtros	Corresponde ao tamanho de cada filtro utilizado na operação de convolução
Stride	Representa o tamanho do salto (ou passo) dos filtros ao percorrer a imagem de entrada
Padding	Define como a borda da imagem de entrada é tratada
Função de Ativação	Função não-linear aplicada sobre os mapas de características resultantes

Baseado nas arquiteturas recentes de RNC, o número de filtros normalmente corresponde a um valor em potência de dois (16, 32, 64, ...). O tamanho dos filtros é comumente atribuído à dimensão 3x3 ou 5x5. Já o *stride* utilizado é igual a um. Por fim, o *padding* pode assumir dois valores: *valid*, a imagem entrada não sofre nenhuma alteração. No entanto, o mapa de características de saída terá uma dimensão menor em relação a entrada; *same*, a imagem de entrada sofre alterações em sua borda (normalmente, adiciona-se o valor zero em toda a sua borda). Contudo, o mapa de características de saída terá a mesma dimensão da imagem de entrada. A função de ativação mais famosa considerando RNCs é a *ReLu*. No exemplo da Figura 6, o filtro tem tamanho 3x3, o valor de *stride* é igual a um e o *padding* usado corresponde ao *valid*. Nela, não foi usada nenhuma função de ativação.

Camada de *Pooling*

O principal propósito dessa camada é reduzir a dimensionalidade de uma mapa de características, aplicando uma operação de *pooling* que compacta esse mapa de características de modo que as suas principais características ainda fiquem presentes (isto é, tal operação tenta resumir um mapa de características (HIJAZI; KUMAR; ROWEN, 2015)). Há várias formas de efetuar a operação de *pooling* em um mapa de características, sendo as principais utilizadas pelo presente trabalho o *max pooling* (no qual o valor máximo de uma janela é extraído) e o *average pooling* (no qual o valor médio dos valores de uma janela é extraído). A Figura 7 ilustra o processo da

operação de *pooling*, usando um filtro de tamanho 2x2 e um valor de *stride* igual a dois (valores comumente utilizados na literatura (SUN et al., 2018)).

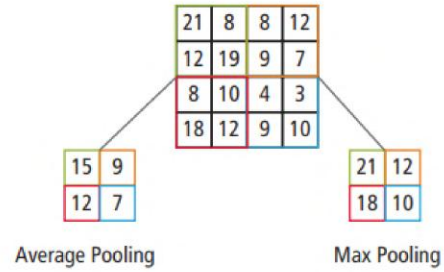


Figura 7 – Exemplo das operações de *max pooling* e *average pooling* (HIJAZI; KUMAR; ROWEN, 2015)

Camada Totalmente Conectada

Os mapas de características gerados pelo último par de camada convolucional e de *pooling* são todos concatenados para serem processados por uma sequência de camadas totalmente conectadas. Estas camadas são caracterizadas pela conexão de todos os neurônios da camada atual com todos da camada anterior, conforme ilustrado na Figura 4. Elas combinam todas as características aprendidas em camadas anteriores, gerando descritores de características da imagem que podem ser mais facilmente classificados pela camada de saída, a qual normalmente usa a função *softmax* (PENHA, 2018).

2.3 Abordagens de Aprendizado

Esta seção apresenta as principais técnicas de aprendizagem que embasam o presente trabalho.

2.3.1 Aprendizado por Reforço Profundo: *Deep Q-Networks*

Aprendizado por Reforço (AR) refere-se a uma área de AM na qual um agente aprende a tomar suas decisões baseando-se no resultado de suas ações por meio da interação com um determinado ambiente (RUSSELL; NORVIG, 2013). Neste sentido, um agente irá deliberar no ambiente percebendo o seu estado atual e selecionando ações segundo uma política. O efeito de cada ação gera um valor, denominado recompensa, calculado por uma função de recompensa que retorna um sinal de reforço ou penalidade de acordo com o resultado da ação executada. Ressalta-se que não é dado ao agente a informação de qual ação ele deve realizar, isto é, ele deve descobrir as ações que levam às maiores recompensas por meio de tentativa e erro. Assim, o agente é completamente responsável por selecionar

as ações que serão aplicadas sobre o estado atual, por esse motivo os agentes da AR são caracterizados como autônomos (MARTINS et al., 2007).

Um sistema típico de aprendizagem por reforço constitui-se, basicamente, de um agente interagindo em um ambiente via percepção e ação (RUSSELL; NORVIG, 2013). O agente percebe as situações dadas no ambiente (pelo menos parcialmente) e seleciona uma ação a ser executada em consequência de sua percepção. A ação executada muda, de alguma forma, o ambiente; e as mudanças são comunicadas ao agente por um sinal de reforço (NETO, 2007). A Figura 8 ilustra esta interação.

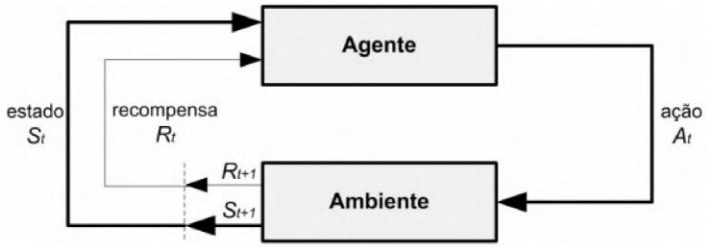


Figura 8 – Ciclo de interação entre o agente e o ambiente na aprendizagem por reforço (SUTTON; BARTO, 1998)

Conforme pode ser observado na Figura 8, um sistema de AR é constituído por dois elementos principais: *agente* e *ambiente*. Todavia, este tipo de sistema pode contar com quatro subelementos:

Uma política: define a forma de o agente aprender a se comportar em determinado momento. Em outras palavras, uma política π é um mapeamento de estados $s \in S$ percebidos em ações $a \in A$ a serem tomadas naquele momento. Assim, uma política pode ser definida em um valor $\pi(s,a)$. Se um agente muda a sua política, então as probabilidades de seleção de ações sofrem mudanças e, consequentemente, o comportamento do sistema apresenta variações à medida que o agente vai acumulando experiência a partir das interações com o ambiente. Portanto, o processo de aprendizado no sistema AR pode ser expresso em termos da convergência até uma política ótima ($\pi^*(s,a)$) que conduza à solução do problema de forma ótima;

Função de recompensa: mapeia cada estado do ambiente (ou par estado-ação) em um valor de modo a direcionar o aprendizado do agente. Por esta razão, o objetivo da AR é maximizar as recompensas recebidas ao longo do tempo. Além disso, essa função pode servir como base para ajustar a política utilizada. Por exemplo, se a escolha de uma ação resulta em uma penalidade, a política pode ser alterada para que outra ação possa ser tomada em tal situação no futuro de modo a melhorar o reforço do agente;

Função valor: especifica a recompensa que o agente espera acumular sobre o futuro, partindo de um certo estado. Enquanto a função de recompensa avalia um estado em sentido imediato, a função valor avalia o estado a longo prazo. Isso permite que um certo estado receba uma baixa recompensa, mas um alto valor, visto que ele pode ser seguido de outros estados com alta recompensa. O inverso também pode ocorrer. O objetivo é procurar ações que tragam melhores valores do que recompensas, visto que a longo prazo o efeito tende a ser mais positivo em relação às recompensas. Contudo, recompensas são mais fáceis de calcular, pois são dadas diretamente pelo ambiente, ao passo que valores devem ser estimados a partir da sequência de observações que o agente faz durante sua atuação no problema. Assim, em um algoritmo de AR, o componente de maior importância é aquele que estima valores de modo eficiente (SUTTON; BARTO, 1998);

Modelo do ambiente: constitui uma forma de simular o comportamento do ambiente. Por exemplo, dado um par estado-ação, o modelo pode predizer o próximo estado e a recompensa a ser retornada. Modelos são utilizados para planejar, isto é, tem por meta auxiliar na decisão de ações considerando situações possíveis antes delas ocorrerem. No caso do presente trabalho, não é possível obter um modelo do ambiente, pelo fato do jogo *FIFA* não ser disponibilizado à comunidade em código aberto.

Assim, para um agente ser bem-sucedido em uma tarefa de AR, ele necessita das seguintes habilidades: aprender sobre a interação entre estados, ações e recompensas subsequentes; determinar qual a melhor ação a ser escolhida a partir do seu aprendizado sobre a interação com o ambiente.

Existem dois conceitos que devem ser conhecidos para facilitar a modelagem de um problema como um sistema de AR:

- **Propriedade de Markov**: trata-se de quando o sistema possui a característica de a propriedade de transição de um estado s para um estado s' depender apenas do estado s e da ação a adotada em s. Isto significa que o futuro independe do passado dado o presente (LIMNIOS; OPRIŞAN, 2001). A propriedade de Markov é fundamental na AR, uma vez que tanto as decisões como os valores são funções apenas do estado atual, abrindo a possibilidade de métodos de soluções incrementais, os quais podem obter soluções a partir do estado atual e dos estados futuros, como é feito nos Métodos das Diferenças Temporais (SUTTON; BARTO, 1998).

- **Processo de Decisão de Markov (PDM)**: descreve, formalmente, um ambiente de AR como um processo estocástico no qual o estado no próximo passo de tempo depende apenas do estado no passo de tempo atual e da decisão escolhida no presente - Propriedade de Markov. Segundo (SUTTON; BARTO, 1998), um PDM é definido como um conjunto de estados S, um conjunto de ações A(s), $\forall s \in S$, um conjunto de

transições T entre estados associadas com as ações, um conjunto de probabilidades P sobre o conjunto S que representa uma modelagem das transições entre os estados e um conjunto de recompensas R associadas a cada estado ou a cada par estado-ação.

Por exemplo, no jogo *Snake* (ilustrado na Figura 9), o agente controla a cobra e o seu objetivo é chegar até a posição onde se encontra a maçã. Alcançar tal objetivo representa vitória, no entanto, morrer (bater em alguma das paredes que limitam o tamanho da tela), representa derrota. A maçã é representada pelo *grid* vermelho. A cabeça da cobra, a qual guia o seu movimento, é representada pelo *grid* verde.

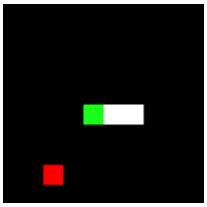


Figura 9 – Jogo (*Snake*).

Dessa forma, uma possível formulação do jogo *Snake* utilizando o conceito de PDM é a seguinte:

- **Estados**: representados pelos *pixels* brutos das imagens do jogo.
- **Ações**: movimentos para cima, para baixo, para a direita e para a esquerda.
- **Recompensas**: recompensa positiva (1) em caso de vitória; recompensa negativa (-1) em caso de derrota; recompensa neutra (0) em caso de movimentos intermediários, os quais não levam nem à vitória nem à derrota.

Nota-se que a função de transição não foi especificada pelo fato do jogo *Snake* ter uma característica determinística, ou seja, ao se realizar uma ação a em um estado s, o resultado sempre é conhecido e garantido. Não existe incerteza/aleatoriedade sobre o estado final do ambiente após a execução da ação. Por exemplo, na situação de jogo apresentada na Figura 9, caso seja efetuada a ação de movimento para baixo, é garantido que a cobra irá seguir o comando para baixo. Dada a ação para baixo, não existe uma probabilidade de efetuar, por exemplo, o movimento para baixo de 50% e não efetuar a ação de 50%, o que configuraria uma característica estocástica ao ambiente.

Assim, para grande parte dos problemas de AR é suposto que o ambiente tenha a forma de um PDM, desde que seja satisfeita a Propriedade de Markov no ambiente. Nem todos os métodos de AR necessitam de uma modelagem PDM inteira do ambiente, mas é necessário ter-se pelo menos a visão do ambiente como um conjunto de estados e ações (SUTTON; BARTO, 1998).

Recentemente, as técnicas de AP incorporaram os métodos de AR, produzindo um novo ramo de pesquisa chamado ARP. O ARP provou ser muito bem-sucedido em dominar políticas de controle em nível humano em uma ampla variedade de tarefas, incluindo o domínio de jogos digitais (ARULKUMARAN et al., 2017). Exemplos de aplicações bem-sucedidas incluem a arquitetura de aprendizado denominada DQN, a qual alcançou desempenho sobre-humano em vários jogos do console Atari (MNIH et al., 2015). Além disso, destaca-se o AlphaGo-Zero (SILVER et al., 2017b) e o AlphaZero (SILVER et al., 2017a), jogadores automáticos de Go e Xadrez respectivamente, compostos por RNPs treinadas por meio de uma variação do método de Monte Carlo. Em particular ao presente trabalho, o método DQN foi investigado nas tarefas de cobrança de faltas.

Nota-se que ele foi um dos primeiros algoritmos da abordagem de ARP a obter um excelente desempenho em jogos digitais. O DQN combina as vantagens da aprendizagem profunda usando RNC para representação abstrata com o método *Q-learning* (WATKINS, 1989) - daí a denominação de *Q-Network* para tal RNC - de modo a aprender uma política ótima baseada, exclusivamente, nas imagens que representam os estados do jogo. Em outras palavras, tal método permite um agente a aprender a agir de tal maneira que maximize o valor esperado da soma cumulativa de recompensas (MNIH et al., 2015), conforme equação 1 (na qual T representa o fim de jogo, e $\gamma \in [0, 1]$ é o fator de desconto que determina a importância das recompensas futuras (SUTTON; BARTO, 1998)).

$$R_t = \sum_{t'=t}^{T} (\gamma^{(t'-t)} r_{t'}) \tag{1}$$

A função Q de uma determinada política π é definida como o retorno esperado da execução de uma ação a em um estado s (equação 2):

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a] \tag{2}$$

Assim, a *Q-Network* do método DQN é parametrizada por θ (correspondente aos seus vetores de pesos e bias), a qual efetua uma estimativa da função Q da política atual próxima da função ideal Q^* (definida como o maior retorno que podemos esperar obter seguindo qualquer estratégia), conforme a equação 3:

$$Q^{\pi}(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a] = \max_{\pi} Q^{\pi}(s, a) \tag{3}$$

O objetivo é encontrar θ tal que Q_{θ}(s, a) $\approx$ Q^*(s, a). A função Q ideal verifica a equação de otimização de *Bellman*, conforme a equação 4:

$$Q^*(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a') | s, a] \tag{4}$$

Dessa forma, se $Q_{\theta} \approx Q^*$, é natural pensar que Q_{θ} deveria estar perto de também verificar a equação de *Bellman* (LAMPLE; CHAPLOT, 2017). Isso leva à função de perda definida na equação 5 (onde t é a etapa de tempo atual e $y_t = r_j + \gamma \max_{a'} Q_{\theta_t}(s, a)$). Tal

função - denominada *Huber Loss* (HUBER; RONCHETTI, 2009) - é essencial no método DQN.

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'} \left[\ (y_t - Q_{\theta_t}(s,a))^2 \ \right] \tag{5}$$

A combinação de algoritmos de AR *model-free* (que não usam um modelo do ambiente) (SUTTON; BARTO, 1998), como o algoritmo *Q-learning*, com redes neurais pode acarretar em instabilidades e fazer com que o treinamento do modelo não convirja (MOUSAVI; SCHUKAT; HOWLEY, 2018b). As duas principais razões para tais problemas são: 1) a variação das atualizações de aprendizado, porque há uma alta correlação entre os estados subsequentes; 2) pequenas mudanças nos valores de Q alteram a política frequentemente, o que pode causar o esquecimento de experiências passadas.

O grande sucesso do DQN reside no fato de que ele solucionou tais problemas de instabilidade da seguinte forma: 1) utiliza a abordagem *experience replay* (LIN, 1992) para reduzir o problema de correlação, permitindo que o agente baseado em AR use atualizações *mini-batch* estocásticas com amostragem uniformemente aleatória em dados previamente observados *offline*; 2) ele usa uma *target network* (MNIH et al., 2015) para o problema da instabilidade da política Q. A *target network* é obtida, inicialmente, pela mesma arquitetura da *Q-Network* (ademais, com os mesmos parâmetros) e é essencial no cálculo do componente y_j na função perda. Ela é mantida congelada por um período de tempo e atualizada periodicamente com os parâmetros da *Q-Network*, o que auxilia na convergência do DQN.

O Algoritmo 1 descreve todo o processamento de treinamento efetuado pelo método DQN. Primeiramente, um *buffer* de experiências D com tamanho N é inicializado a partir transições efetuadas no ambiente por uma política completamente aleatória (linha 1). Em seguida, a *Q-Network* (RNC que representa a função Q neste caso) é inicializada com parâmetros aleatórios (linha 2). No início de cada episódio e, alguma técnica de pré-processamento é aplicada ao estado inicial s_1, formando ϕ_1 (linha 4). Para cada transição t em e, uma ação a_t é selecionada por meio da estratégia de exploração *decaying epsilon-greedy* (SUTTON; BARTO, 1998) (linhas 6 e 7). Na sequência, a_t é executada no ambiente, resultando em uma recompensa r_t e no próximo estado pré-processado ϕ_{t+1} (linhas 8 e 9). Esta transição (ϕ_t, a_t, r_t, $\phi(s_{t+1})$) é, então, armazenada no *buffer* D (linha 10). Após isso, um mini-lote de transições é recuperado de D e a saída correspondente a cada exemplo j é calculada e armazenada em y_j (linhas 11 e 12). Em estados não terminais, o valor Q é computado pela *target network* (parâmetros θ^-). Por fim, é efetuado o treinamento da *Q-Network* usando a função de perda $(y_j - Q(\phi_j, a_j; \theta))^2$ (linha 13).

A *Q-Network* proposta em (MNIH et al., 2015) é composta por três camadas convolucionais (*Conv1*, *Conv2* e *Conv3*) e duas camadas totalmente conectadas (*TC1* e *TC2*). As configurações utilizadas nas camadas convolucionais são apresentadas na Tabela 2. As configurações das camadas totalmente conectadas são exibidas na Tabela 3. Assim, tal

Algoritmo 1 DQN com *experience replay*

Inicialize o *buffer* de experiências D com capacidade N
Inicialize a função Q (representada pela *Q-Network*) com parâmetros aleatórios
for episódio $e = 1, ..., M$ **do**
  Inicialize estado $s_1 = \{x_1\}$ e estado pré-processado $\phi_1 = \phi(s_1)$
  for transição $t = 1, ..., T$ **do**
    Com probabilidade ϵ selecione uma ação aleatória a_t
    Caso contrário, selecione $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute ação a_t e observe recompensa r_t e imagem $x_t + 1$
    Crie estado $s_{t+1} = st$ e estado pré-processado $\phi_{t+1} = \phi(s_{t+1})$
    Armazene a transição (ϕ_t, a_t, r_t, $\phi(s_{t+1})$) em D
    Recupere um mini-lote de transições (ϕ_t, a_t, r_t, $\phi(s_{t+1})$) de D
    Crie $y_j = \begin{cases} \mathrm{r}_j, & \text{para estados } \phi_{j+1} \text{ terminais} \\ \mathrm{r}_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta^-), & \text{para estados } \phi_{j+1} \text{ não terminais} \end{cases}$
    Execute uma etapa de descida de gradiente $(y_j - Q(\phi_j, a_j; \theta))^2$
  end for
end for

RNC retorna como saída o valor Q associado a cada ação possível (um valor N dependente do problema). Como a saída da RNC é um valor real, isto significa que o DQN atua, essencialmente, em um contexto de regressão.

Tabela 2 – Configurações das camadas convolucionais de acordo com (MNIH et al., 2015)

Camada	**Número de Filtros**	**Tamanho do Filtro**	***Stride***	***Padding***	**Função de Ativação**
Conv1	32	8x8	4x4	*same*	*ReLu*
Conv2	64	4x4	2x2	*same*	*ReLu*
Conv3	64	3x3	1x1	*same*	*ReLu*

Tabela 3 – Configurações das camadas totalmente Conectadas de acordo com (MNIH et al., 2015)

Camada	**Número de Neurônios**	**Função de Ativação**
TC1	512	*ReLu*
Saída (*TC2*)	N	Nenhuma (Linear)

É importante destacar que existem alguns hiperparâmetros relacionados ao método DQN, descritos na Tabela 4.

Tabela 4 – Hiperparâmetros associados ao DQN

Hiperparâmetro	**Significado**
Tamanho do Mini-Lote	Número de transições utilizadas para efetuar a atualização da *Q-Network*
Tamanho do *buffer* de experiências	Número de transições que podem ser armazenadas no *buffer*
Atualização da *Target Network*	Quantidade de transições necessárias para efetuar a atualização da *Target Network* pela *Q-Network* atual
Fator de desconto (*Gamma*)	Indica o quão importante o futuro é levado em consideração na atualização da *Q-Network*
Taxa de aprendizagem	Representa o quanto será aprendido pelo agente com relação ao ajuste calculado
Valor inicial de ϵ	Representa o valor de exploração que o método adota no início do treinamento
Valor final de ϵ	Representa o valor mínimo de exploração que o método ao longo do treinamento

2.3.2 Aprendizado por Imitação: Imitação Direta e Imitação Ativa Profunda

O AI é uma abordagem de aprendizado inspirada na neurociência e é considerada uma importante parte da área de interação humano-computador, sendo visto como um relevante ramo para os futuros avanços da robótica (SCHAAL, 1999). Ao contrário do AR, na qual um agente aprende pelo método de tentativa e erro, o AI está interessado em fornecer bons exemplos para que o agente obtenha uma boa política de ações de modo mais rápido, além de minimizar as chances de levar a mínimos locais - os quais resultam em uma política de ações sub-ótimas (as tomadas de decisão do agente podem levar a um desempenho razoável, no entanto, há melhores ações que levam a um melhor desempenho).

De modo análogo ao aprendizado supervisionado tradicional, no qual os exemplos representam pares de entrada e o rótulo desejado, no AI os exemplos correspondem a pares de estado - também chamado de observação em muitas ocasiões - e ação (noção de AR), onde o estado representa as características relevantes da posição corrente em que o agente se encontra e a ação realizada por tal agente neste estado. Tipicamente, um algoritmo de AI se inicia a partir de um processo de coleta de exemplos - referentes à interação de um especialista com o problema abordado. Em seguida, tais exemplos são utilizados para treinar uma política inicial de um agente. Ressalta-se que esses dois passos (coleta de exemplos e treinamento de uma política inicial a partir de tais exemplos) compõem a forma mais trivial de AI, denominada ID. No entanto, utilizar apenas estes dois passos geralmente não é suficiente para alcançar um comportamento desejado devido a pobre generalização do comportamento a novas situações não vistas nas demonstrações. Isto pode ser causado por motivos tais como: dificuldade em coletar as demonstrações no ambiente; ou ruídos podem prejudicar a qualidade dos exemplos; ou a necessidade de coletar uma grande quantidade de demonstrações pode ser um gargalo para o processo de aprendizagem. Portanto, técnicas de AI frequentemente envolvem uma etapa adicional que requer que o agente treinado (política inicial) desempenhe o comportamento aprendido no ambiente e otimize a política aprendida de acordo com o desempenho da tarefa (HUSSEIN et al., 2017). Normalmente, esta otimização é realizada por meio de mais exemplos, como, por exemplo, a partir da utilização de aprendizado ativo. No entanto, tal otimização também pode ser feita via AR, o que resulta em uma abordagem híbrida de aprendizado. Destaca-se que a política é formada por um método tomador de decisão arbitrário.

Há uma grande variedade de pesquisas sendo empregadas a partir da aplicação de técnicas de AI. Alguns exemplos incluem problemas de navegação, os quais empregam robôs que funcionam como veículos voadores (SAMMUT et al., 1992) ou veículos terrestres (SILVER; BAGNELL; STENTZ, 2008). Outros exemplos são relacionados a robôs humanoides, os quais podem aprender a levantar e também a caminhar (BERGER et al.,

2008). Além disso, AI também é utilizado em tarefas envolvendo ambientes simulados, como jogos digitais em (GORMAN, 2009) e (ROSS; BAGNELL, 2010). Os métodos de AI investigados pelo presente trabalho são a ID e a IAP, os quais serão explicadas na sequência.

A ID via clonagem de comportamento (BAIN; SAMMUT, 1995) é um dos métodos mais triviais para lidar com um problema de AI. Este método geralmente assume que um agente recebe demonstrações consistindo em pares de estado (observação) e ação (LIU et al., 2018). O objetivo é produzir uma política que mapeie diretamente observações em ações a partir das demonstrações nos dados de treinamento. O grande problema dessa técnica, conforme já mencionado anteriormente, é o fato de que a política produzida geralmente não possui uma boa generalização a novas situações não representadas nos dados de treinamento.

Assim, dentre as técnicas que tentam minimizar tal fragilidade da ID, a IAP - proposta em (HUSSEIN et al., 2018) - se destaca pelo fato de combinar a utilização de aprendizado ativo e RNP. Essa técnica estende a abordagem de ID por meio do uso de aprendizado ativo para refinar a política aprendida inicialmente, tornando a tomada de decisão do agente mais robusta a situações para as quais a política inicial não foi treinada (atacando a grande fragilidade da ID). Desta forma, a IAP consiste das seguintes etapa: 1) coleta de demonstrações do especialista. Nesta etapa, o especialista interage com o ambiente e cada par constituído de observação e ação é armazenado como uma demonstração, formando um conjunto de dados de treinamento; 2) realização do treinamento supervisionado de uma RNC a partir do conjunto de dados formado na etapa anterior (etapa 1), gerando uma política inicial; 3) aplicação de aprendizado ativo para aprimorar a RNC treinada na etapa 2 (política inicial). É necessário enfatizar que as duas primeiras etapas da IAP representam o método ID. Assim, a terceira etapa é fundamental para a construção de um agente que tenha um comportamento ainda melhor em relação a um agente treinado por ID.

O Algoritmo 2 apresenta o processo de aprendizado de uma agente implementado pelo método IAP e a Figura 10 ilustra a aplicação do aprendizado ativo (linhas 3 a 17), descritos na sequência.

Após a efetuação das etapas 1 e 2 do treinamento do método IAP (correspondente a ID), a política inicial é formada. Ao longo da etapa 3, tal política interage com o ambiente. Para cada observação processada, a entropia total das probabilidades das ações geradas pela predição da política inicial é calculada (representada pelo valor $H(X)$ na linha 7). Se a entropia for maior que um determinado limite definido empiricamente pelo hiperparâmetro β (linha 8), é solicitada a intervenção do agente humano, o que configura um *feedback* corretivo. Neste caso, a política inicial não executa nenhuma ação e deixa que o supervisor humano realize a melhor ação para corrigir o comportamento do agente (linhas 9 e 10). Sempre que a política inicial requisita o supervisor humano, a demons-

Algoritmo 2 Método IAP

1: Coleta de um conjunto de dados D representando pares (observação, ação) pelo agente supervisor
2: Treinamento da RNC (política inicial π) sobre D
3: $Dados\_Ativos = []$
4: **while** tamanho($Dados\_Ativos$) < $Percentual\ Dados\ Ativos$ **do**
5: $x = observação\_atual$
6: $u = \pi(x)$
7: $H(X) = -\sum_i P(u_i) \log P(u_i)$
8: **if** $H(X) > \beta$ **then**
9: $y = Requisita\_Ação\_Supervisor(x)$
10: executa ação y
11: adiciona o par (x, y) em $Dados\_Ativos$
12: **else**
13: executa $max(u)$
14: **end if**
15: **end while**
16: $Conjunto\_Dados\_Estendidos = D + Dados\_Ativos$
17: Atualiza π usando $Conjunto\_Dados\_Estendidos$

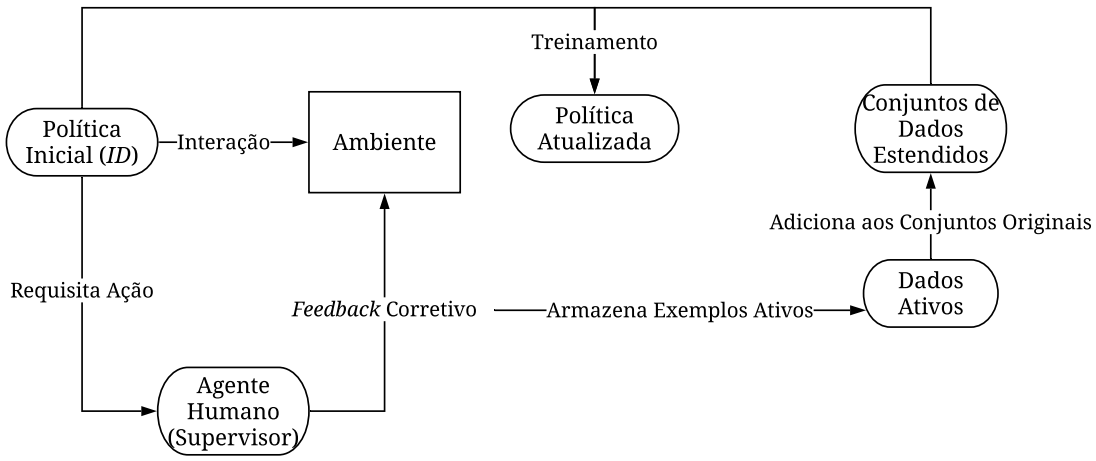


Figura 10 – Processo de aprendizagem de um agente pelo Método IAP

tração fornecida pelo supervisor - cuja entropia ultrapassou o limite - é adicionada a um novo conjunto de dados denominado *Dados Ativos* (linha 12). É interessante destacar que um alto valor de entropia sugere um baixo nível de confiança associado à capacidade de tomada de decisão do agente, o que ocorre em situações de jogo nas quais a política inicial produz valores de probabilidade muito semelhantes para todas as ações legais (devido a cenários nunca vistos pelo referido agente). Caso contrário, se a entropia não exceder o valor limite (β), a política inicial executa a ação com a maior probabilidade calculada (melhor ação encontrada para a observação atual, conforme linha 13). Esse processo de treinamento continua até o ponto em que o número de observações no referido conjunto chegue a um determinado percentual definido pelo hiperparâmetro *Percentual Dados Ati-*

vos. Quando isso ocorrer, o conjunto *Dados Ativos* é adicionado ao conjunto de dados original (D coletado na etapa 1), formando o *Conjunto de Dados Estendido* (linha 16). Por fim, a política inicial é treinada novamente a partir dos novos exemplos e também dos exemplos originais (*Conjunto de Dados Estendidos*), o que gera uma nova política atualizada (linha 17). Assim, espera-se que esta nova política tenha um melhor desempenho em relação à política inicial.

Destaca-se que a política π proposta em (HUSSEIN et al., 2018) corresponde a uma RNC composta por três camadas convolucionais (*Conv1*, *Conv2* e *Conv3*) e duas totalmente conectadas (*TC1* e *TC2*). As configurações utilizadas nas camadas convolucionais são apresentadas na Tabela 5. Destaca-se que a operação de normalização em lote seguida pela função de ativação *ReLu* e por uma operação de *max pooling* são adicionadas à saída de cada camada convolucional, conforme ilustrado na Figura 11. As configurações das camadas totalmente conectadas são exibidas na Tabela 6. Assim, tal RNC retorna como saída uma probabilidade (calculada pela função de ativação *Softmax* da camada de saída) associada a cada ação possível (conjunto de ações de tamanho N dependente do problema). Isto significa que os métodos de AI investigados no presente trabalho lidam, essencialmente, com tarefas de classificação.

Tabela 5 – Configurações das camadas convolucionais de acordo com (HUSSEIN et al., 2018)

Camada	**Número de Filtros**	**Tamanho do Filtro**	***Stride***	***Padding***	**Função de Ativação**
Conv1	20	7x9	1x1	*valid*	*ReLu*
Conv2	50	5x5	1x1	*valid*	*ReLu*
Conv3	70	4x5	1x1	*valid*	*ReLu*

Tabela 6 – Configurações das Camadas Totalmente Conectadas de acordo com (HUSSEIN et al., 2018)

Camada	**Número de Neurônios**	**Função de Ativação**
TC1	500	*ReLu*
Saída (*TC2*)	N	*Softmax*

2.4 Avaliação de Métodos de AI

Como os métodos de AI lidam com tarefas de classificação de uma maneira supervisionada, eles induzem modelos (ou hipóteses) a partir de conjuntos de dados. Assim, eles possuem um viés indutivo relacionado à propensão de considerar uma hipótese específica (ou conjunto de hipóteses) (GAMA et al., 2015). No caso dos métodos de AI utilizados no presente trabalho, a forma de representação que eles possuem para descrever a hipótese induzida é por meio dos parâmetros de uma RNP treinada a partir de um determinado conjunto de dados. Dessa forma, torna-se necessário avaliá-los quanto à qualidade do

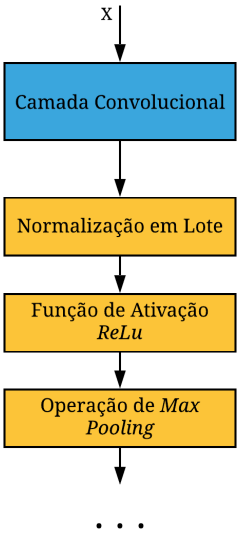


Figura 11 – Normalização em Lote, Função *ReLu* e Operação de *Max Pooling* adicionadas após uma camada convolucional.

aprendizado obtida (isto é, a capacidade de efetuar boas inferências após treinados para exemplos nunca vistos). No presente trabalho, os agentes baseados nos métodos de AI são avaliados diretamente nos cenários de jogo para os quais foram implementados (a partir de predições em tempo real). Particularmente ao caso da abordagem evolutiva utilizada nesse trabalho, a técnica *Cross Validation* foi explorada no processo de avaliação dos indivíduos e é descrita na sequência.

2.4.1 *Cross-Validation*

A validação cruzada (*cross-validation*) divide os dados em dois segmentos: um usado para treinar um modelo e o outro usado para validá-lo. Uma das formas básicas desse método é denominada *k-fold cross-validation*, na qual os conjuntos de treinamento e validação devem cruzar em rodadas sucessivas, de modo que cada exemplo contido no conjunto de dados tenha a chance de ser utilizado na etapa de validação (REFAEILZADEH; TANG; LIU, 2009). A Figura 12 ilustra o processo de particionamento efetuado pelo *k-fold cross-validation* (com $k = 5$).

Conforme observado, no *k-fold cross-validation* (com $k = 5$), os dados são divididos em cinco partições mutuamente exclusivas. Em seguida, cinco iterações de treinamento e validação são executadas de modo que, em cada iteração, uma partição diferente dos dados é utilizada para validação, enquanto as quatro partições restantes são usadas para o treinamento.

Uma variação denominada *k-fold cross-validation* estratificada é utilizada pelo presente trabalho, a qual mantém nas partições as proporções de exemplos das classes presentes

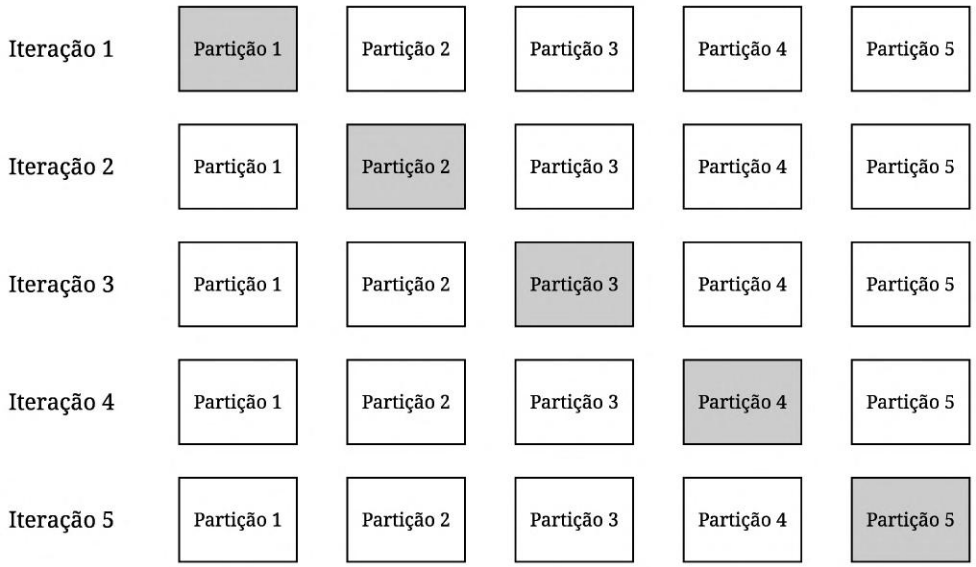


Figura 12 – Exemplo do Particionamento dos Dados Efetuado pelo *k-fold cross-validation* (com $k = 5$)

no conjunto total de dados.

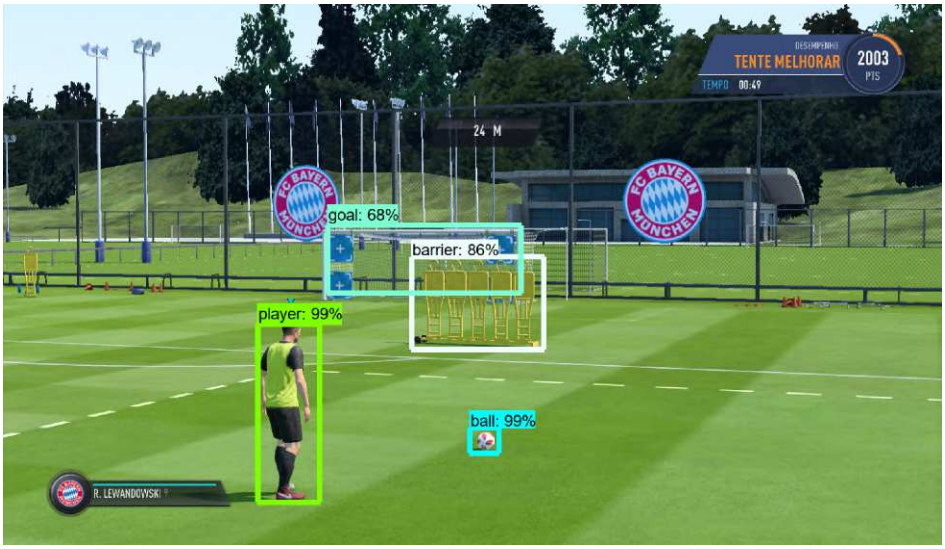
2.5 Técnicas de Detecção de Objetos

Nesta seção, é apresentado um breve resumo da TDO utilizada no presente trabalho, denominada *Single-Shot Multibox Detector MobileNet* (*SSD MobileNet*).

Single-Shot MultiBox Detector (*SSD*) é um método de detecção de objetos em imagens que utiliza uma única RNP e é mais rápido e significativamente mais preciso do que o estado-da-arte anterior para detectores denominados *single-shot*, sendo o principal o algoritmo YOLO (LIU et al., 2016). *Single-shot* significa que as tarefas de localização e classificação de objetos são realizadas apenas em uma única passagem da rede neural. *Multibox* refere-se à técnica de regressão de *bounding boxes* desenvolvida no trabalho do *SSD*. Resumidamente, ela começa com *bounding boxes* pré-calculadas e de tamanho fixo como predições que tentam se aproximar da distribuição das *bounding boxes* originais - rotuladas por um humano. Por fim, *Detector* pelo fato de a rede, além de executar a localização dos objetos, também efetua a classificação desses objetos. A Figura 13a) mostra a imagem original e a Figura 13b) ilustra a detecção de objetos realizada pelo *SSD* (identificação das *bounding boxes*). O presente trabalho usa uma versão do *SSD* com o *MobileNetV2* como um extrator de características para o modelo de TDO.

Brevemente, o *MobileNetV2* (SANDLER et al., 2018) é uma RNC profunda e leve, projetada para dispositivos móveis e máquinas com baixo poder computacional. Ele se baseia em uma versão anterior, denominada *MobileNetV1* (HOWARD et al., 2017), usando a técnica de convolução separável em profundidade para reduzir o custo de com-

(a)



(b)

Figura 13 – Exemplo de detecção de objetos executada pelo *SSD MobileNet*

plexidade da rede. Além disso, o *MobileNetV2* aprimora seu antecessor com dois novos recursos: blocos residuais invertidos e módulos lineares entre as camadas da rede. Com o *MobileNetV2* como peça fundamental para extração de características, foram alcançados desempenhos excepcionais em tarefas de classificação, detecção de objetos e segmentação semântica (SANDLER et al., 2018).

2.6 Técnicas de Representação de Ambiente

Esta seção apresenta as técnicas de representação de ambiente utilizadas no presente trabalho. A subseção 2.6.1 exibe a representação por meio de imagens cruas com informação de cor, enquanto a subseção 2.6.2 apresenta a representação do ambiente por meio de

imagens cruas sem informação de cor. Por fim, a subseção 2.6.3 descreve as representações baseadas em TDO.

2.6.1 Imagens Cruas Com Informação de Cor

Esta representação, também denominada de Imagens Coloridas, é baseada em imagens brutas no formato RGB (*Red* - vermelho, *Green* - verde e *Blue* - azul). Isto é, ela considera as informações de cores contidas nas imagens (três componentes de cores do RGB), conforme ilustrado na Figura 14. Nela, é possível verificar que cada *pixel* da imagem contém os valores correspondentes a cada componente do formato RGB. Tais valores estão no intervalo de zero a 255.

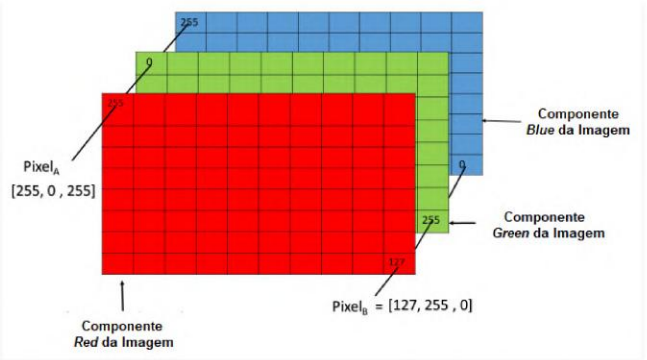


Figura 14 – Exemplo da representação por imagem colorida (IHRITIK, 2020)

2.6.2 Imagens Cruas Sem Informação de Cor

Esta representação é baseada em imagens brutas que não levam em consideração as informações de cores contidas nas imagens (no caso, as três componentes de cores do RGB), conforme ilustrado na Figura 15. Nela, é possível verificar que cada *pixel* da imagem contém apenas um valor, o qual é o resultado da combinação dos valores correspondentes aos componentes RGB. Destaca-se que o valor convertido está no intervalo entre zero e 255. A fórmula de conversão de um *pixel* em RGB para a escala de cinza utilizada neste trabalho é a seguinte (conforme (BRADSKI, 2000)):

$$Pixel \leftarrow 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{6}$$

Desta forma, conforme mostrado na Figura 15, o $Pixel_A = [255, 0, 255]$ ($R = 255$, $G = 0$, $B = 255$) é convertido para escala de cinza com valor igual a 115. Analogamente, o $Pixel_B = [127, 255, 0]$ é convertido para escala de cinza com valor igual a 188. Assim, apesar de descartar as informações de cor que podem ser úteis (pois os três valores do RGB são mais representativos do que um único valor da escala de cinza), tal representação reduz a complexidade dos dados.

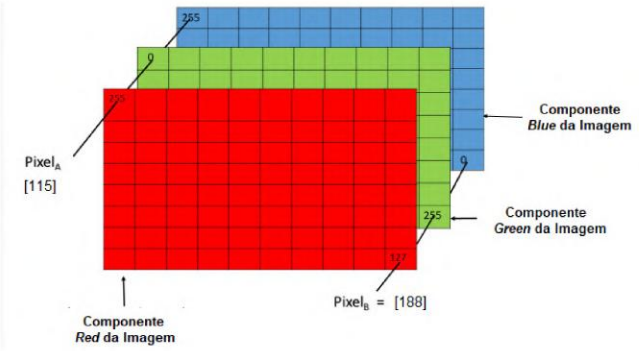


Figura 15 – Exemplo da representação por Imagem em Escala de Cinza. Adaptada de (IHRITIK, 2020)

2.6.3 Oriundas de TDOs

O processo de geração das representações baseadas em TDO é efetuado por meio da arquitetura ilustrada na Figura 16. Ela possui o seguinte ciclo de funcionamento: a Imagem Original (*frame*[1] atual.), a qual representa a situação corrente de jogo, é processada pelo *MobileNet* de modo a gerar uma representação menor com as principais características presentes na Imagem Original, denominada *Mapa de Características de Alto Nível*, utilizada por um modelo *SSD* que efetua a detecção das *bounding boxes* que descrevem os objetos desejados para detecção. Destaca-se que tal arquitetura é treinada a partir de um conjunto de imagens anotadas/rotuladas[2] com tais objetos desejados. Os detalhes dos módulos *Extrator de Características MobileNet* e *SSD* são descritos na sequência.

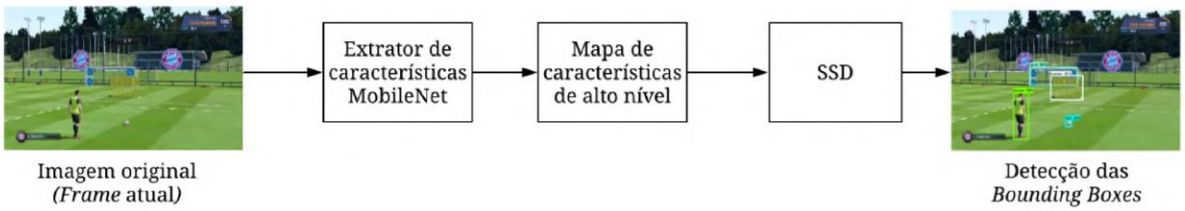


Figura 16 – Arquitetura de TDO utilizada neste trabalho

Extrator de Caraterísticas MobileNet (MobileNet): esse módulo é utilizado para converter a representação original do estado de jogo (Imagem Original) em uma mais simplificada que corresponde às principais características extraídas dos objetos dispostos em tal estado (*Mapa de Características de Alto Nível*). Elas incluem propriedades e padrões de disposição dos *pixels* que permitem efetuar a detecção

[1] Um *frame* corresponde a uma imagem fixa de um produto audiovisual (AUMONT; MARIE, 2006)

[2] Neste contexto, anotar ou rotular corresponde à associação das *bounding boxes* a cada elemento de interesse em cada imagem da base.

dos objetos desejados. No entanto, elas também podem incluir informações sobre outros elementos dispostos na imagem que não se deseja identificar (por exemplo, informações sobre as árvores no fundo do cenário da Imagem Original na Figura 16, as quais não possuem relevância para a escolha de uma boa ação).

SSD: esse módulo, com base no *Mapa de Características de Alto Nível*, realiza a detecção dos objetos por meio das tarefas de classificação e de localização, as quais geram as chamadas *bounding boxes*. Tais *bounding boxes* carregam informações importantes sobre os objetos, como a classe (o rótulo do objeto, indicada por uma probabilidade) e a localização (representada por pontos no plano cartesiano).

Aqui, é necessário enfatizar que o processo de treinamento das TDOs utilizadas nesse trabalho foi realizado até que a medida denominada perda total (do inglês, *total loss*) alcançasse um valor entre 0,5 e 1 através da plataforma *Tensorflow Object Detection API* (HUANG et al., 2017). Dessa forma, os modelos apresentaram uma precisão satisfatória na tarefa de detecção de objetos dentro desse intervalo de *total loss*.

Assim, o presente trabalho explora duas representações de estado oriundas de tal arquitetura de TDO:

Representação TDO 1 : corresponde ao *Mapa de Características de Alto Nível* gerada pelo módulo *Extrator de Características MobileNet*. Ela foi proposta em (TRIVEDI, 2018b). No referido trabalho, a versão *MobileNetV1* foi empregada, enquanto que o presente trabalho de Mestrado utiliza a versão mais recente e melhorada *MobileNetV2*. Essa representação (*Mapa de Características de Alto Nível*) corresponde a um vetor compreendendo 128 valores relativos às saídas da última camada convolucional do *MobileNet*.

Representação TDO 2 : corresponde às *bounding boxes* dos objetos desejados geradas pelo módulo *SSD*. Esta representação é baseada em (VENKATESH, 2018). Desta forma, ela considera apenas as informações referentes à localização dos objetos relevantes e é formada por um vetor que contém os seguintes quatro valores correspondentes aos eixos X (abscissa) e Y (ordenada) da *bounding box* associada para cada objeto: X_{min}, correspondente ao valor mínimo da abscissa; X_{max}, correspondente ao valor máximo da abscissa; Y_{min}, correspondente ao valor mínimo da ordenada; e Y_{max}, correspondente ao valor máximo da ordenada. Para exemplificá-la, considera-se a Figura 17. Nela, os pontos A, B, C e D correspondem, respectivamente, aos valores X_{min}, X_{max}, Y_{min} e Y_{max} extraídos do elemento barreira (considerando que ele é um dos objetos desejados para detecção). Tal processo é efetuado para todos os outros objetos que se deseja identificar.

Figura 17 – Exemplo da **Representação TDO 2** produzida pelo módulo *SSD*.

2.7 Algoritmo Genético

Algoritmo Genético (AG) é uma técnica de busca heurística inspirada na genética e na evolução das espécies por seleção natural, incorporando tais conceitos de modo a simular a evolução natural de sistemas biológicos (DOMINGOS, 2018), na qual os indivíduos com características favoráveis tem mais chances de sobreviver e se reproduzir do que aqueles com características menos favoráveis. Assim, tal técnica é baseada em um processo coletivo de aprendizagem dentro de uma população de indivíduos, cada um dos quais representando um ponto no espaço de busca de solução para um dado problema. A população é inicializada e evolui através de gerações com o uso de operadores de seleção, reprodução e mutação.

Inicialmente, é gerada uma população de indivíduos que representam as possíveis soluções para um determinado problema. Essa população será submetida a alguns operadores, conhecidos como operadores genéticos. Na ordem que é descrita, a população inicial passará por uma função de avaliação (aptidão), cujo objetivo é avaliar a qualidade daqueles indivíduos (soluções) e identificar os mais aptos para resolver o problema. Após isso, outro operador é encarregado de selecionar os indivíduos que serão os genitores, ou seja, aqueles responsáveis por produzir novos indivíduos. Esse processo de reprodução é realizado por outro operador que executa a recombinação do código genético dos genitores. Os novos indivíduos que foram gerados através da reprodução podem ainda sofrer um processo de mutação, o qual modifica o conteúdo de algum dos seus genes e produz uma variação na população. Por fim, um subconjunto de todos esses indivíduos é selecionado para compor a nova população.

No caso do presente trabalho, um método baseado em AG foi utilizado para evoluir diversas arquiteturas de RNCs de modo a encontrar aquela mais adequada para o cenário

modo de confronto. Ele é discutido em detalhes no capítulo 8, assim como todas as operações envolvidas.

2.8 Testes Estatísticos

Os testes estatísticos são procedimentos que utilizam os dados observados em um determinado experimento para avaliar se determinadas hipóteses (hipótese nula - referida como H_0 - ou hipótese alternativa - referida como H_1) são verdadeiras ou não (GRAYBILL; IYER; BURDICK, 1998). A hipótese nula (também chamada de hipótese de igualdade) consiste em afirmar que os parâmetros ou características matemáticas de duas ou mais populações são idênticas (FARIA, 2017). Por outro lado, a hipótese alternativa, necessariamente, difere de H_0. Dessa forma, os testes estatísticos têm a habilidade de determinar quais resultados de um experimento podem levar à rejeição da hipótese nula H_0 a um nível de significância pré-estabelecido. Este nível de significância representa um limiar de confiança que indica se a hipótese nula será rejeitada ou não. No presente trabalho de Mestrado, os testes estatísticos *Teste T* (COLEMAN, 2009), *ANOVA de uma via* (HEIBERGER; NEUWIRTH, 2009), *Wilcoxon* (WILCOXON, 1992) e *Kruskal-Wallis* (KRUSKAL; WALLIS, 1952) foram utilizados para validar as comparações efetuadas com relação ao desempenho dos diversos agentes projetados por meio do *software IBM SPSS* (IBMCORP, 2011).

2.8.1 *Teste T de Amostras Independentes*

O *Teste T de Amostras Independentes* ou, simplesmente, *Teste T*, é um teste estatístico inferencial que determina se há uma diferença significativa entre as médias entre dois grupos não relacionados. Ele calcula os seguintes valores: *valor da estatística t* (*valor-t*), *graus de liberdade* e *valor-p*. O primeiro indica o resultado real do teste estatístico e a direção da diferença (se houver). O segundo serve para ajustar o cálculo com dados amostrais de modo a tornar o resultado válido para a população. Por fim, o *valor-p* representa o valor de significância do teste.

2.8.2 Análise de Variância (*ANOVA de uma via*)

O *ANOVA de uma via* é um teste estatístico inferencial que determina se há uma diferença estatisticamente significativa entre as médias de dois ou mais grupos não relacionados. Para ser confiável, ele emprega as seguintes suposições:

- **Grupos não relacionados**: também chamados de grupos não pareados ou grupos independentes, são grupos nos quais os casos (por exemplo, participantes) em cada grupo são diferentes. No presente trabalho, cada grupo corresponde a um agente e os casos correspondem às execuções dos agentes em uma determinada tarefa;

- **Normalidade da variável dependente**: a variável dependente se aproxima de uma distribuição normal dentro de cada grupo;
- **Homogeneidade de variância**: as variâncias dos dois grupos que estão sendo comparados devem ser aproximadamente iguais. Neste trabalho, essa suposição foi validada usando o *Teste de Levene* (GASTWIRTH; GEL; MIAO, 2009).

No caso deste trabalho, o uso desse método permite verificar se há uma diferença significativa no desempenho apresentado por agentes jogadores quando o número de grupos é maior do que dois. No entanto, através do *ANOVA de uma via*, não é possível identificar os grupos discrepantes quando a hipótese nula é rejeitada. Assim, ele é frequentemente seguido por um teste *post-hoc* no qual cada par de grupos é submetido a um processo de comparação para que seja efetuada uma análise do padrão de diferença entre sua médias. Para esse fim, o presente trabalho utiliza o popular *Teste de Tukey* (SALKIND, 2010).

Dentre os valores calculados pelo *ANOVA de uma via*, os mais importantes são a estatística F e o *valor-p*. O primeiro tem como objetivo representar a relação existente entre as médias das amostras de cada grupo e a variabilidade dentro de cada amostra. Já o segundo representa o valor de significância do teste.

2.8.3 *Wilcoxon*

O *Wilcoxon* é um método não paramétrico empregado em situações de teste de hipóteses envolvendo dois grupos. Da mesma forma que o *Teste T*, o método *Wilcoxon* também adota uma estratégia de tentar verificar a hipótese nula. Nesse sentido, ele é útil na comparação de grupos em que o resultado de cada um deles em cada amostra é dependente (isto é, possui alguma relação).

Seja T a menor soma de classificações positivas e negativas entre as diferenças de pares de dados (considerando dois grupos: *Grupo 1* e *Grupo 2*), ou seja, $T = Min\{R^+, R^-\}$. R^+ representa a soma das classificações em que o *Grupo 1* superou o *Grupo 2*, enquanto que R^- representa a soma das classificações em que o *Grupo 1* foi superado pelo *Grupo 2*. Para determinar a significância dos resultados obtidos, um *valor-p* é também calculado pelo método.

2.8.4 *Kruskal-Wallis*

O *Kruskal-Wallis* é um método não paramétrico empregado em situações de teste de hipóteses, envolvendo uma situação com dois ou mais grupos. Este método é equivalente ao *ANOVA de uma via*, sendo mais relaxado do que este por não assumir as mesmas suposições feitas sobre os dados (por exemplo, ele não assume uma distribuição normal dos dados). Por esse motivo, a significância estatística é efetuada sobre as medianas dos grupos (ao contrário do *ANOVA de uma via*, o qual efetua a avaliação sobre as médias dos

grupos) Ele retorna um *valor-p* que permite verificar a diferença entre os grupos. Caso houver (neste caso, a hipótese nula é rejeitada), um *Teste de Dunn-Bonferroni* (DUNN, 1964) é realizado entre cada par de grupos (funciona como o *Teste de Tukey* no *ANOVA de uma via*).

2.9 Ambiente do Jogo FIFA

Esta seção apresenta os cenários de jogo *FIFA* explorados pelo presente trabalho. As seguintes propriedades de ambiente (descritas na seção 2.1) são generalizadas para todos eles:

Completamente Observável: destaca-se que o estado real do jogo não é totalmente representado em sua imagem correspondente, pois informações relevantes sobre ele são subjacentes ao mecanismo do jogo, o que não é acessível. No entanto, os dados fornecidos pelas imagens (capturas de tela) que representam uma determinada situação de jogo apresentam todos os elementos relevantes para se tomar uma decisão (escolher uma ação).

Estocástico: há algumas aleatoriedades intrínsecas ao simulador de jogo que podem afetar os resultados de algumas ações efetuadas.

Desconhecido: em todos os cenários, a física do jogo exerce impacto direto na deliberação de um agente. Por exemplo, a ação de chute pode ter um efeito diferente dependendo da direção em que a bola está se movimentando antes de ser chutada, por conta da resultante das forças entre os elementos.

Além delas, cada cenário é descrito em termos de suas propriedades específicas, conforme visto na sequência.

2.9.1 Cenário 1: Cobrança de Faltas sem Goleiro

Este cenário explora o modo de cobrança de faltas que envolve situações com e sem barreiras, as quais simulam jogadores bloqueando o gol. Apesar da dificuldade de marcar gols na presença dessas barreiras, este cenário não inclui um goleiro. As Figuras 18 e 19 ilustram algumas situações possíveis dentro desse cenário.

Este cenário é considerado o mais simples dentre os estudados por este trabalho devido às suas propriedades de ambiente:

Estático: devido ao fato de lidar apenas com obstáculos fixos, como a barreira e as traves do gol. Além disso, o tempo não é um fator determinante para a tomada de decisão, pois mesmo que o agente automático demore a escolher uma ação, isso não afetará em mudanças no estado de jogo.

Figura 18 – Situação de falta sem barreira

Figura 19 – Situação de falta com barreira

Agente Único: não há nenhum outro agente interagindo no ambiente (como, por exemplo, outro jogador de linha ou goleiro).

Além disso, é necessário enfatizar que há uma característica aleatória neste modo de jogo, no sentido de que em um certo estado, nem sempre a mesma ação resulta no mesmo resultado. Por exemplo, em uma situação com uma barreira de jogadores, um chute direcionado a ir por cima da barreira com a mesma força e a mesma direção pode resultar em gol em alguns momentos e, em outros, em tentativa frustrada que não resulta em gol. Portanto, o ambiente é inerentemente estocástico, o que também é um grande desafio no processo de aprendizagem de um agente, mesmo o cenário sendo simples.

2.9.2 Cenário 2: Cobrança de Faltas com Goleiro

Este cenário explora o modo de cobrança de faltas que envolve situações com barreiras e com a presença de um goleiro. Ele é mais complexo do que o cenário sem goleiro e, consequentemente, mais desafiador no processo de treinamento de agentes produzidos por meio de técnicas de AM. A Figura 20 ilustra possíveis situações dentro desse cenário. Observa-se que a barreira agora pode assumir diversos formatos, variando em termos da altura e da largura, o que corresponde a um desafio ainda maior no processo de aprendizagem de um agente nesse cenário (além do goleiro).

Este cenário possui as seguintes propriedades:

Estático: apesar de haver um outro agente no ambiente (goleiro), o ambiente não é modificado enquanto o agente "raciocina". Além disso, assim como no cenário de faltas sem goleiro, o tempo não é um fator determinante para a tomada de decisão, pois mesmo que o agente automático demore a escolher uma ação, isso não afetará em mudanças no estado de jogo.

Multiagente: neste caso há um outro agente interagindo no ambiente, o goleiro. Nesse contexto, ele representa um adversário que tenta minimizar as ações do agente cobrador de faltas.

Figura 20 – Exemplos de situações de falta com a presença de goleiro

2.9.3 Cenário 3: Modo de Confronto

Este cenário explora a situação envolvendo dois jogadores de futebol controlados pelo agente (em uniforme amarelo) contra dois jogadores manipulados pelo próprio jogo *FIFA* (em uniforme laranja), conforme ilustrado na Figura 21. Se o agente marcar um gol, a pontuação de jogo é aumentada em 1000 pontos. Por outro lado, se a máquina do *FIFA* marcar um gol, a pontuação de jogo é reduzida em 100 pontos. A execução de uma partida dura 45 segundos.

É necessário enfatizar que esse modo fornece um excelente estudo de caso por conta das suas propriedades:

Dinâmico: neste cenário, há outros agentes que modificam o ambiente. Além disso, o tempo é um fator determinante para a tomada de decisão, pois o agente automático deve efetuar uma ação mais rapidamente por conta dessa constante modificação do ambiente.

Multiagente: pela presença dos dois jogadores manipulados pela máquina do *FIFA*. Nesse contexto, eles representam adversários que tentam minimizar as ações do agente jogador.

Neste cenário, a propriedade de desconhecido é mais acentuada pelo fato de diversos jogadores se movimentarem no campo e também pelo fato da bola estar em constante

Figura 21 – Modo de confronto no jogo *FIFA*

movimento. Assim, é necessário levar em consideração as informações de direção e de movimento dos elementos que compõem o ambiente. No entanto, apenas um *frame* não é suficiente para se obter tais informações, conforme ilustrado na Figura 21. Apesar de ser possível identificar os elementos relevantes no ambiente de jogo (por exemplo, a bola, os jogadores de cada equipe, a delimitação do campo), somente esse *frame* não permite conhecer qual a direção da bola (no caso, ela pode ter sido passada de cima para baixo ou de baixo para cima). Assim, o presente trabalho utiliza o AI para minimizar esse problema em tal cenário por meio do agente supervisor humano que possui conhecimento sobre a direção e movimentação dos elementos.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta os principais trabalhos correlatos nos quais a presente pesquisa se inspira. São abordados trabalhos referentes à construção de agentes jogadores baseados em ARP e em AI (focando naqueles envolvendo o jogo *FIFA*), às interfaces de conexão entre agentes e ambientes existentes, além da técnica evolutiva que serve de base para o método proposto no capítulo 8.

3.1 Agentes Jogadores

Uma notável conquista em jogos digitais foi alcançada em (MNIH et al., 2015), desenvolvida pelo *Google Deepmind*, por meio de agentes treinados baseados em técnicas de ARP - agentes treinados sem contar com supervisão humana por meio da combinação de redes neurais profundas e técnicas de AR - a partir das entradas representadas pelas imagens cruas (pixels brutos) de jogo. Tal independência de aprendizagem em relação ao conhecimento humano permitiu que os agentes implementados de acordo com o DQN ultrapassassem a expertise humana em vários jogos do clássico *videogame Atari* 2600.

Em 2017, a empresa *OpenAI* desenvolveu um sistema inteligente capaz de derrotar os melhores jogadores mundiais no jogo *Dota 2* na categoria 1v1, na qual um jogador atua contra um outro jogador (OPENAI, 2017). Mais recentemente, um notório resultado foi alcançado pela mesma empresa na principal categoria do referido jogo, 5v5, ao conseguir desenvolver um sistema que derrotou a campeã de 2018 do principal campeonato de *Dota 2* (SYNCED, 2019). Tal sistema consiste de cinco redes neurais cooperativas, cada uma representando um jogador diferente dentro do jogo. Denominado *OpenAI Five*, o modelo utiliza técnicas de AR de última geração, como o *Proximal Policy Optimization* (PPO) (SCHULMAN et al., 2017) para dominar os detalhes de um jogo. Tal método provou-se incrivelmente valioso para resolver desafios relativos à exploração do ambiente. Além disso, o treinamento do *OpenAI Five* é essencialmente baseado na técnica *self-play*, na qual o modelo joga aproximadamente 180 anos de jogos contra si mesmo todos os dias, rodando em 256 GPUs e 128.000 núcleos de CPU (OPENAI, 2018).

Em (VINYALS et al., 2019) é apresentado o notável agente jogador *AlphaStar* para atuar no *Starcraft 2*. Tal agente é baseado em técnicas de redes neurais e foi capaz de derrotar um jogador profissional no fim de 2018. O *AlphaStar* foi treinado inicialmente usando uma política baseada em aprendizado por imitação (na qual o objetivo é prover ao agente a capacidade de jogar de maneira mais humana) e, em seguida, aprimorado por meio da combinação entre aprendizado por reforço e a estratégia *self-play*. Neste sentido, é utilizado um treinamento baseado em população, no qual vários agentes treinam uns contra os outros e, ao final do processo, o agente final é constituído da combinação ou mistura das características dos melhores agentes da fase de treinamento. Tal abordagem difere da presente proposta no sentido de que não há um módulo que modela o perfil de oponente no *AlphaStar*, apesar da sua incrível performance contra diversos adversários com estratégias diferentes. Assim, pretende-se desenvolver um módulo de identificação da estratégia do oponente de forma a modelar o seu perfil de estilo de jogo por meio de suas ações, fato que será incorporado no processo de tomada de decisão de um agente jogador.

Seguindo uma linha diferente de construção de agentes dotados de IA, diversos trabalhos utilizam técnicas de AM para a criação de jogos com um caráter mais amigável de modo a aprimorar a experiência de jogo dos usuários (ZHAO et al., 2019). Neste sentido, técnicas de Aprendizado por Imitação (AI) são usadas para processar eficientemente dados coletados a fim de realizar a extração de modelos de conhecimento e comportamento humano, permitindo a geração de agentes com a capacidade de exibir e reproduzir um comportamento autônomo semelhante ao de um jogador humano. Um exemplo de aplicação é a implementação de personagens não jogáveis (do inglês, *non-playable characters*) inteligentes. Por exemplo, (Buche; Even; Soler, 2018) provê um *framework* para a coleta e processamento de dados para a criação de *bots* para o jogo *Unreal Tournament 3*.

Convém salientar os esforços no desenvolvimento de técnicas de AI visando a criação de jogos com um caráter mais amigável a fim de proporcionar uma experiência de jogo mais rica aos jogadores (ZHAO et al., 2019). Neste sentido, uma das preocupações consiste em implementar personagens não jogáveis (do inglês, *non-playable characters* NPCs) - os quais atuam como adversários ou companheiros de equipe do jogador - que tornem o jogo mais divertido e/ou desafiador para ele (ORTEGA et al., 2013). Ressalta-se que nem toda IA projetada para atuar em um jogo (como os NPCs) o faz de maneira humana, frequentemente apresentando comportamentos considerados não naturais aos humanos. Assim, as técnicas de AI possuem um grande foco no processamento eficiente (em termos de quantidade e de rapidez) de dados coletados relativos à interação de jogadores humanos com o jogo a fim de se efetuar uma extração de modelos sobre os seus comportamentos. Tais modelos permitem aprimorar o processo de construção de NPCs inteligentes em ambientes virtuais complexos. Os trabalhos (ORTEGA et al., 2013) e (Buche; Even; Soler, 2018) são exemplos de investigação de AI aplicada, respectivamente, aos jogos

Super Mario Bros e *Unreal Tournament 3*.

O principal trabalho correlato com relação aos métodos de AI utilizados na presente pesquisa é (HUSSEIN et al., 2018). Nele, é apresentado um novo método que combina aprendizado profundo supervisionado e aprendizado ativo, denominado IAP. Os autores do referido trabalho compararam o método proposto com a ID e também com dois algoritmos estado-da-arte de ARP: DQN e *Asynchronous Advantage Actor-Critic* (A3C) (MNIH et al., 2016). Essa comparação foi realizada em uma tarefa simples de navegação em *grid* 2D e no simulador MASH (ABBET, 2014), que contém ambientes 3D estáticos e de agente único projetados para navegação. O agente implementado (constituído por uma RNC) treinado com IAP provou ser a melhor versão entre todos os outros agentes, construídos de acordo com os outros métodos comparados. Por esse motivo, a IAP é explorada no presente trabalho em todos os cenários de jogo *FIFA* descritos em 2.9.

3.2 FIFA

Em (TRIVEDI, 2018a), o jogo *FIFA* é utilizado como estudo de caso para a criação de um agente que atua no modo principal do jogo, no qual o agente controla um time formado por 11 jogadores e compete com o motor de jogo que controla outro time formado também por 11 jogadores, de maneira próxima a humana a partir da ótica de ID. Trivedi gravou várias de suas partidas jogando contra a máquina do *FIFA* de modo a construir um conjunto de dados composto por diversas demonstrações - representadas por pares estado (ou observação) e ação. Em seguida, utilizou o conjunto de dados construído para treinar um modelo de AP baseado em uma Rede Neural Recorrente para servir como módulo de tomada de decisão do agente projetado. Destaca-se que a representação dos estados de jogo é proveniente da **Representação TDO 1** (descrita na subseção 2.6.3). Tal modelo de detecção de objetos é constituído por uma RNC com a habilidade de identificar os elementos mais relevantes do problema - em (TRIVEDI, 2018a), os objetos tratados pela RNC foram a bola, o gol e os jogadores. A fim de verificar o nível de desempenho do agente construído por meio da abordagem ID, tal agente foi colocado à prova para atuar contra a máquina do *FIFA* no nível mais fácil (modo iniciante) e conseguiu marcar quatro gols em cerca de seis partidas. O autor do trabalho mencionado afirmou que o resultado pode ser melhorado ainda mais a partir da investigação de técnicas mais elaboradas de AI. IAP é uma técnica mais complexa em relação a ID e será utilizada nos variados cenários *FIFA* abordados no presente trabalho.

Em (TRIVEDI, 2018b) é apresentada uma breve descrição do problema de cobrança de faltas no *FIFA* e como o algoritmo DQN foi aplicado para resolver esta tarefa. O cenário de faltas sem goleiro - descrito na seção 2.9 - foi considerado pelo referido trabalho. Nele, um modelo de detecção de objetos foi treinado usando a técnica *MobileNetV1* (HOWARD et al., 2017) para identificar os seguintes elementos: a bola, o gol e jogador que cobra a

falta. Desta forma, o *MobileNetV1* processa as capturas de tela do jogo, fornecendo um mapa de características de 128 dimensões (vetor de 128 elementos) como representação de estado para o modelo DQN - conforme descrito na seção 2.6, considerando os elementos identificados. Neste caso, a *Q-Network* considerada corresponde a uma PMC composta por três camadas totalmente conectadas (incluindo a camada de saída), conforme apresentado na Tabela 7. A camada de saída é constituída por quatro neurônios, cada um representado uma ação que o agente poderia realizar (*mover para a direita*, *mover para a esquerda*, *chute alto* e *chute baixo*). O agente resultante, referido aqui como *Trivedi*, alcançou cerca de 50% de taxa de chutes bem sucedidos (taxa de gols) ao longo de 1000 épocas de treinamento em uma GPU GTX-1070. É importante notar que o modelo não é treinado para detectar outros objetos relevantes para a tarefa de faltas, como barreira, o que poderia melhorar o desempenho do modelo de aprendizado. Isto é explorado pelo presente trabalho, além de expandir Trivedi (2018b) ao investigar o cenário de faltas com goleiro a fim de aumentar a complexidade do problema abordado.

Tabela 7 – Arquitetura da Rede Neural de acordo com (TRIVEDI, 2018b)

Camada	**Número de Neurônios**	**Função de Ativação**
TC1	512	*ReLu*
TC2	512	*ReLu*
Saída (*TC3*)	4	Nenhuma (Linear)

3.3 Interfaces de Conexão

Diversas plataformas para testar e desenvolver algoritmos de IA em jogos digitais foram concebidas. Essas plataformas fornecem interfaces que permitem a conexão entre o módulo de aprendizado dos agentes e o ambiente do jogo. Entre essas plataformas, destacam-se o *OpenAI Gym* (BROCKMAN et al., 2016) e o *OpenAI Universe* (MOUSAVI; SCHUKAT; HOWLEY, 2018b) (ambas código aberto para a comunidade).

O *OpenAI Gym* disponibiliza várias interfaces para conectar o módulo de aprendizado de um agente jogador a vários ambientes de jogos, como o *Atari 2600* usando a ferramenta *Arcade Learning Environment* (ALE), jogos clássicos, mecanismo de física *MuJoCo*, entre outros. A equipe do *OpenAI Gym* incentiva novas pesquisas que incluem novos ambientes em seu repositório. Como alternativa a tal plataforma, o *OpenAI Universe* suporta ambientes mais complexos para jogos em *Flash*, aplicativos de navegador e também jogos realísticos como o *Grand Theft Auto IV* (MOUSAVI; SCHUKAT; HOWLEY, 2018b). Entretanto, diferentemente do *OpenAI Gym*, o *OpenAI Universe* não permite à comunidade de pesquisa incluir novos ambientes em seu repositório, inviabilizando o uso dessa plataforma para jogos que não pertencem a seus ambientes internos. Outros exemplos de plataformas conhecidas e que não permitem a inserção de novos ambientes comple-

xos são: *FlashRL* (ANDERSEN; OLSEN; GRANMO, 2018) (concebida em um cenário de jogos *Flash*), GVGAI (TORRADO; BONTRAGER; TOGELIUS, 2018) (construída com o objetivo de estimular pesquisas referentes ao problema de se criar agentes que conseguem atuar com um bom nível de comportamento para jogos em geral) e VizDoom (KEMPKA; WYDMUCH; RUNC, 2016) (produzida exclusivamente para o jogo *Doom*). Como nenhuma dessas interfaces viabiliza a conexão com o jogo *FIFA*, o presente trabalhou desenvolveu uma versão própria baseadas nas explicadas anteriormente.

3.4 Técnica Evolutiva para Definição Automática de RNC baseada em AG

Em projetos envolvendo redes neurais (abrangendo o AP), uma das tarefas mais importantes é a definição de suas arquiteturas (por exemplo, o número de camadas, o número de neurônios em cada uma delas, hiperparâmetros). Dessa forma, pelo enorme número de possibilidades de combinações (grande espaço de busca), a tarefa de se definir uma boa arquitetura é também muito complicada. Nesse sentido, é muito comum que elas sejam projetadas por especialistas que possuem um ótimo conhecimento com relação aos dados do problema e ao domínio de redes neurais (SUN et al., 2018). No entanto, mesmo com tais conhecimentos, é inevitável que o tempo gasto até que se encontre uma arquitetura adequada seja alto, por todo o estudo teórico e prático que tal tarefa demanda. Assim, diversas propostas de automatização desse processo de definição de arquiteturas de redes neurais vêm sendo feitas na literatura, divididas em duas categorias principais: 1) baseadas em algoritmos evolutivos (Xie; Yuille, 2017; REAL et al., 2017; LIU et al., 2018; SUN et al., 2018); 2) baseadas em AR (ZOPH; LE, 2017; BAKER et al., 2017).

Nesse contexto, dentre os algoritmos de automatização baseados em técnicas evolutivas, destaca-se o (SUN et al., 2018). Ele propõe um método que automatiza o processo de construção de arquiteturas de RNC consideravelmente apropriadas para problemas de classificação de imagens utilizando AG. Tal proposta é referenciada pelo presente trabalho como Algoritmo Genético para Rede Neural Convolucional (AG-RNC). Basicamente, o AG-RNC procura evoluir uma população de indivíduos (cada um representa uma arquitetura arbitrária de RNC) compostos por blocos denominados de *Skip* e de *Pooling*, onde a aptidão deles corresponde à acurácia na base de teste designada pelo conjunto de dados fornecido como entrada para o método. O objetivo é maximizar tal acurácia. Ao fim do processo evolutivo, a melhor arquitetura de RNC encontrada é retornada pelo AG-RNC. Destaca-se que ele obteve resultados científicos expressivos em bases de dados *benchmark* conhecidas, como a CIFAR10 e a CIFAR100. Recentemente, o referido trabalho foi ainda mais incrementado, gerando Sun et al. (2019), o qual apresenta novidades quanto à construção e definição dos blocos utilizados pelas arquiteturas de RNC. Conforme exibido na Tabela 8, a presente pesquisa de Mestrado efetuou algumas adaptações sobre o AG-RNC

com relação à codificação e avaliação dos indivíduos, limitando o tamanho da arquitetura de um indivíduo e criando uma função de aptidão que combina acurácia e número de parâmetros da arquitetura de RNC. Todas as características expostas na referida tabela serão detalhadas no capítulo 8.

Tabela 8 – Principais diferenças entre o AG-RNC e a abordagem adotada pela presente pesquisa de Mestrado

Abordagem	AG-RNC (SUN et al., 2018)	Presente Pesquisa de Mestrado
Indivíduo	Tamanho ilimitado	Tamanho limitado (máximo de 10 blocos)
Blocos	*Skip* e *Pooling*	*Skip* e *Pooling*
Avaliação dos Indivíduos	Acurácia	Acurácia e Número de Parâmetros
Seleção dos Pais	Torneio Binário	Torneio Binário
Cruzamento	*Crossover* de um ponto	*Crossover* de um ponto
Mutação	Adicionar, remover ou modificar um bloco	Adicionar, remover ou modificar um bloco
Atualização da População	Torneio Binário + Elitismo	Torneio Binário + Elitismo

Capítulo 4

Desenvolvimento de uma Interface de Conexão entre Agentes Jogadores e o Ambiente de Jogo *FIFA*

Este capítulo apresenta a contribuição referente à proposta de uma interface para conectar agentes jogadores (baseados em AM) ao ambiente complexo e desafiador do jogo *FIFA*. Tal interface - inspirada nas plataformas *OpenAI Gym* e *OpenAI Universe* (seção 3.3) - possui um caráter genérico nos sentidos de flexibilidade e de portabilidade, uma vez que é possível ser utilizada para diversas tarefas diferentes dentro do *FIFA*, além de poder ser reaproveitada em outros jogos distintos. Convém salientar que a interface também apresenta a vantagem de ser amigável em termos de usabilidade por ter sido implementada de acordo com o paradigma orientado a objetos. A fim de detalhar como foi realizada a implementação da interface de conexão proposta, as seções 4.1 e 4.2 apresentam, respectivamente, as visões em alto e baixo nível da conexão entre um agente e seu ambiente por meio da interface.

4.1 Arquitetura da Interface de Conexão

A Figura 22 ilustra o fluxo de interação em alto nível entre um agente e o seu ambiente de atuação por meio da interface de conexão. Basicamente, ela é composta por dois módulos principais, denominados Visão e Controle. O primeiro deles é responsável por intermediar a obtenção de informações do ambiente pelo agente. Seu principal objetivo é implementar um procedimento de captura das imagens do jogo e fornecê-las ao agente. Desta forma, torna-se possível que o agente perceba o ambiente e processe o estado atual de modo a tomar suas decisões. O segundo módulo visa intermediar a tomada de decisão do agente e o ambiente. Para tanto, foi implementado um método capaz de simular ações do teclado diretamente no ambiente de jogo, permitindo com que o agente atue sobre ele. Assim, por meio dos módulos de Visão e Controle, é possível que o agente jogador

interprete o estado do ambiente, efetue suas decisões (realizando ações no meio) e meça o impacto de tais decisões de modo a aprimorar o seu aprendizado.

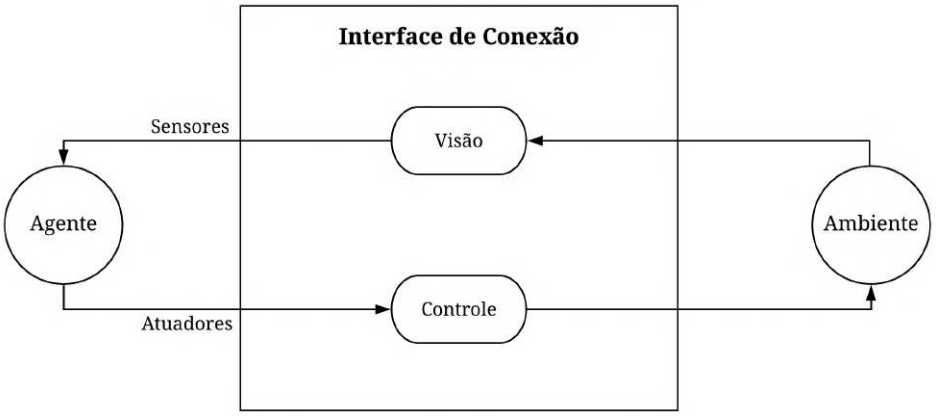


Figura 22 – Fluxo de interação em alto nível entre um agente jogador e um ambiente por meio da interface proposta

4.2 Detalhamento dos Módulos da Arquitetura

A interface é constituída, principalmente, por três métodos: *Observar*, *Resetar* e *Passo*. A Figura 23 ilustra o fluxo de interação entre um agente e o seu ambiente evidenciando tais métodos. O primeiro deles - pertencente ao Módulo de Visão - retorna o estado atual do ambiente. Basicamente, ele efetua uma captura de tela do momento atual de jogo e retorna uma observação *o* aos sensores do agente. É importante notar que a imagem da captura de tela é representada por *pixels* coloridos no formato RGB. Os métodos *Resetar* e *Passo* pertencem ao Módulo de Controle. *Resetar* apenas reinicia o ambiente de jogo para o estado inicial. Já o método *Passo* tem como objetivo executar a ação selecionada pelo agente baseado no estado atual (observação *o*) do ambiente e retornar a observação *o'* gerada por consequência de tal tomada de decisão. Este método processa a ação da seguinte forma:

1. Executa a ação no ambiente e verifica se o jogo acabou. Por exemplo, no caso de cobranças de falta, o final do jogo está relacionado à ação de chutar a bola;

2. Utiliza o método *Observar* para perceber o novo estado gerado, ou seja, o impacto (resultado) da ação escolhida pelo agente;

3. Em cenários de AR, a partir do novo estado gerado por essa ação, é calculada uma recompensa. Ele retorna ao agente uma tupla formada pela nova observação gerada, pela recompensa associada e por um valor *booleano* que indica o final do jogo (o valor *True* indica que o episódio terminou).

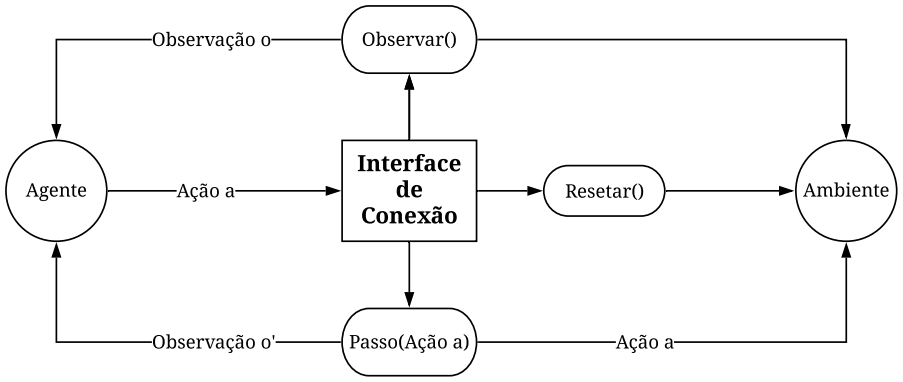


Figura 23 – Fluxo de interação em baixo nível entre um agente jogador e um ambiente por meio da interface proposta

É importante enfatizar a semelhança dos métodos implementados pela interface proposta em relação àqueles implementados pelas plataformas *OpenAI Gym* e *OpenAI Universe*. Tais plataformas também utilizam o paradigma orientado a objetos, no entanto, podem ser utilizadas apenas para alguns jogos específicos (dentre os quais o *FIFA* não está contido), conforme especificado na seção 3.3. Nelas, existem duas funções principais também denominadas *Resetar* e *Passo*:

***Resetar*:** apenas reinicia o jogo para o seu estado inicial original e retorna a imagem corrente de observação ao agente.

***Passo*:** possui o mesmo objetivo do método *Passo* implementado na interface proposta. A única diferença é que, além de retornar os mesmos aspectos, tal função implementada pelas referidas plataformas também fornece informações de diagnóstico úteis para depuração (as quais não exercem impacto no processo de treinamento).

Portanto, como a interface proposta aqui é altamente baseada nas plataformas *OpenAI*, amplamente utilizadas em inúmeras pesquisas relacionadas a jogos e ao ARP, ela se torna mais amigável e compreensível tanto para novos usuários quanto para usuários experientes. Além disso, a utilização do paradigma de programação orientado a objetos a torna mais flexível e fácil de ser aprimorada com novas funcionalidades. Ela foi documentada e disponibilizada à comunidade[1].

Por ter sido primeiramente utilizada e validada em um contexto de ARP (capítulo 5), a presente interface foi uma das contribuições do artigo *A Deep Reinforcement Learn-Based FIFA Agent with Naive State Representations and Portable Connection Interfaces* publicado na conferência *The 32nd International FLAIRS Conference* (FARIA; JULIA; TOMAZ, 2019b). No entanto, ela também foi usada em um contexto supervisionado, sendo muito importante no processo de coleta de demonstrações e na conexão de agentes baseados em AI ao ambiente *FIFA* (capítulos 6, 7 e 8).

[1] <https://github.com/matheusprandini/FifaFreeKickLearning2019>

Capítulo 5

Aprendizado por Reforço DQN aplicado a Agentes Cobradores de Faltas no *FIFA*

Este capítulo apresenta a contribuição referente à investigação da abordagem de ARP - especificamente, o método DQN - nos cenários de cobrança de faltas sem e com goleiro descritos na seção 2.9. A fim de lidar com tais cenários, a seção 5.1 apresenta a formulação dos mesmos como um problema de AR, usando o conceito de PDM (descrito na seção 2.3.1). Tal formulação é usada para avaliar o desempenho de agentes com as seguintes formas de representação: baseadas em imagens cruas e em TDO. As representações são investigadas na seção 5.2 e a arquitetura dos agentes é detalhada na seção 5.3. Por fim, os experimentos e a análise dos resultados de ambos os cenários são apresentados na seção 5.4.

Essa investigação é inspirada na proposta apresentada em (TRIVEDI, 2018b) - explicada na seção 3.2 - cujo objetivo era explorar a combinação do método DQN com TDO no domínio do cenário de faltas sem goleiro (isto é, envolve um ambiente estático e agente único), produzindo o agente *Trivedi*. Nesse sentido, a presente pesquisa expande tal proposta por investigar um ambiente multiagente que inclui um goleiro.

5.1 Formulação da Tarefa de Cobrar Faltas como um Problema de AR

Com base nas descrições dos cenários de faltas simplificado e complexo na seção 2.9, é possível formular a tarefa de cobrança de faltas como um problema de AR usando o conceito de PDM da seguinte maneira:

- **Estados**: correspondem ao cenário de jogo corrente. No caso deste trabalho, podem ser representados pelas imagens cruas do jogo ou por meio de TDO, conforme

descrito na seção 5.2.

- **Ações**: existem quatro possíveis ações legais que podem ser executadas pelos agentes: mover para a esquerda, mover para a direita, chute rasteiro e chute alto em uma altura preestabelecida (que geralmente varia entre a altura da barreira e a altura do gol). É importante ressaltar que a força do chute é considerada constante nas duas possibilidades de chute, sendo empiricamente definida pelos autores do trabalho. Ela corresponde ao apertar e segurar as teclas que realizam tais ações de chute no jogo por aproximadamente 0,25 segundos. Detalhes podem ser encontrados no repositório disponibilizado à comunidade[1].

- **Recompensas**: após a execução de uma ação de chute, se o agente marcar um gol, a recompensa é positiva (igual a 1). Caso contrário, a recompensa dada ao agente é negativa (igual a -1). Dessa forma, ações de chute levam ao fim de um episódio. Para ações intermediárias (movimentos para esquerda e direita), a recompensa é um valor negativo próximo de zero (no caso, -0,05). A finalidade dessa recompensa ligeiramente negativa é evitar que o agente execute *loops* indesejáveis de movimentos para a esquerda e/ou direita, conforme identificado em (TRIVEDI, 2018b), o qual utilizou um valor de recompensa igual a zero para as ações de movimento.

  Após alguma ação de chute ser efetuada, é aplicada a técnica de Reconhecimento Ótico de Caracteres (do inglês, *Optical Character Recognition*) para verificar se houve alteração na pontuação de jogo, conforme ilustrado na Figura 24. Caso tal pontuação aumentar em 500 pontos ou mais, significa que o agente marcou um gol (em ambos os cenários de falta abordados). Caso contrário, ele não marcou um gol. Para as ações intermediárias, esta técnica não precisa ser aplicada.

Figura 24 – Reconhecimento Ótico de Caracteres aplicado para verificar mudanças na pontuação de jogo

[1] <https://github.com/matheusprandini/FifaFreeKickLearning2019>

5.2 Especificação das Representações de Estados

Esta seção apresenta as representações de estados utilizadas pelos agentes implementados no presente capítulo. A subseção 5.2.1 apresenta a representação do ambiente por meio de imagens cruas sem informação de cor, enquanto a subseção 5.2.2 exibe a representação por meio de imagens cruas com informação de cor. Por fim, a subseção 5.2.3 apresenta as representações oriundas de TDO.

5.2.1 Escala de Cinza 4 frames

Esta representação é semelhante à explorada pelo artigo original do DQN em que se criam agentes automáticos para atuar nos jogos do console Atari (MNIH et al., 2015). Nesse sentido, as entradas brutas são pré-processadas de modo a reduzir a dimensão da observação original (*frame* correspondente à situação de jogo corrente) para 84x84, convertendo-a para escala de cinza (conforme descrito na subseção 2.6.2) e efetuando uma etapa de normalização dos *pixels* considerando o intervalo entre zero e um. Além disso, os quatro *frames* mais recentes são empilhados, formando um estado.

A título de exemplo, a Figura 26 mostra a representação correspondente aos últimos quatro *frames* vistos por um agente na Figura 25 (a sequência é vista da esquerda para a direta e de cima para baixo). No caso, cada estado é representado por quatro imagens de tamanho 84x84 (isto é, o estado é uma matriz 84x84x4) produzidas pela etapa de pré-processamento. Ressalta-se que essa representação foi utilizada com o propósito de estudar e replicar o artigo original do DQN que a leva em consideração.

Figura 25 – Exemplo dos últimos quatro *frames* vistos por um agente

Figura 26 – Exemplo da Representação **Escala de Cinza 4 frames**

5.2.2 RGB

Esta representação é baseada em imagens brutas coloridas, conforme apresentada na subseção 2.6.1. Ela tem em comum com a representação anterior (**Escala de Cinza 4 frames**) os seguintes aspectos: etapa de pré-processamento que reduz a dimensão da imagem original para 84x84 igual a representação anterior e a etapa de normalização dos *pixels* para o intervalo entre zero e um. Por outro lado, a representação **RGB** difere da anterior nas seguintes questões: por manter as informações de cor (isto é, não converte os *pixels* para escala de cinza); por considerar apenas a observação correspondente ao cenário de jogo corrente como percepção do agente, isto é, apenas o último *frame*. Tal representação é retratada na Figura 27. Portanto, cada estado é representado pela imagem corrente de jogo de tamanho 84x84x3 (3 componentes de cores do RGB) produzida pela etapa de pré-processamento.

Figura 27 – Exemplo da Representação **RGB**

5.2.3 TDOs

As TDOs aqui desenvolvidas refletem aprimoramentos efetuados sobre a proposta relativa ao agente *Trivedi* (descrita na seção 3.2). Dessa forma, a TDO utilizada para gerar as representações para o cenário de faltas sem goleiro foi treinada para identificar os seguintes elementos: barreira, bola, gol e o jogador que cobra faltas. Ela expande a TDO utilizada pelo agente *Trivedi* por aprimorar a percepção do agente, o que é conseguido pela adição do elemento barreira - extremamente relevante no contexto de faltas - aos objetos considerados na tarefa de detecção. Já a TDO utilizada para gerar as representações para o cenário de faltas com goleiro, estende aquela projetada para o cenário sem goleiro ao

adicionar o elemento goleiro dentre os objetos a serem considerados na tarefa de detecção. Dessa forma, os detalhes das representações oriundas das TDOs treinadas para lidar com o cenário de faltas sem goleiro e com goleiro são exibidas na sequência.

Representação TDO 1: 128 valores representando o *Mapa de Características de Alto Nível* (conforme apresentado na subseção 2.6.3).

Representação TDO 2 : representação de estado proveniente do módulo *SSD* da arquitetura de TDO (conforme explicado na subseção 2.6.3). A Tabela 9 exibe a representação utilizada no cenário de faltas sem goleiro, formada por um vetor de 16 valores (quatro valores correspondentes a cada um dos elementos). A Tabela 10 exibe a representação utilizada no cenário de faltas com goleiro, formada por um vetor de 20 valores (quatro valores correspondentes a cada um dos elementos).

Tabela 9 – **Representação de TDO 2** para o Cenário de Faltas sem Goleiro

Posição	**Característica**
0	X_{min} jogador
1	X_{max} jogador
2	Y_{min} jogador
3	Y_{max} jogador
4	X_{min} bola
5	X_{max} bola
6	Y_{min} bola
7	Y_{max} bola
8	X_{min} barreira
9	X_{max} barreira
10	Y_{min} barreira
11	Y_{max} barreira
12	X_{min} gol
13	X_{max} gol
14	Y_{min} gol
15	Y_{max} gol

5.3 Arquitetura dos Agentes Jogadores Baseados em Distintas Representações de Estado

Esta seção apresenta os agentes propostos para atuar nos dois cenários de faltas - com e sem goleiro - abordados no presente trabalho. O renomado método de ARP denominado DQN foi aplicado com o objetivo de treinar agentes a marcar gols. Tais agentes são baseados em distintas percepções do ambiente descritas na seção 5.2.

A arquitetura geral dos agentes é ilustrada na Figura 28. Ela é composta por três módulos principais: Representação de Estado; Tomada de Decisão; e DQN. Ressalta-se que, como os agentes são baseados em distintas representações de estado, os modelos/arquiteturas de redes neurais utilizadas como módulo de tomada de decisão são inerentes a tais representações.

Tabela 10 – **Representação de TDO 2** para o Cenário de Faltas com Goleiro

Posição	Característica
0	X_{min} jogador
1	X_{max} jogador
2	Y_{min} jogador
3	Y_{max} jogador
4	X_{min} bola
5	X_{max} bola
6	Y_{min} bola
7	Y_{max} bola
8	X_{min} barreira
9	X_{max} barreira
10	Y_{min} barreira
11	Y_{max} barreira
12	X_{min} gol
13	X_{max} gol
14	Y_{min} gol
15	Y_{max} gol
16	X_{min} goleiro
17	X_{max} goleiro
18	Y_{min} goleiro
19	Y_{max} goleiro

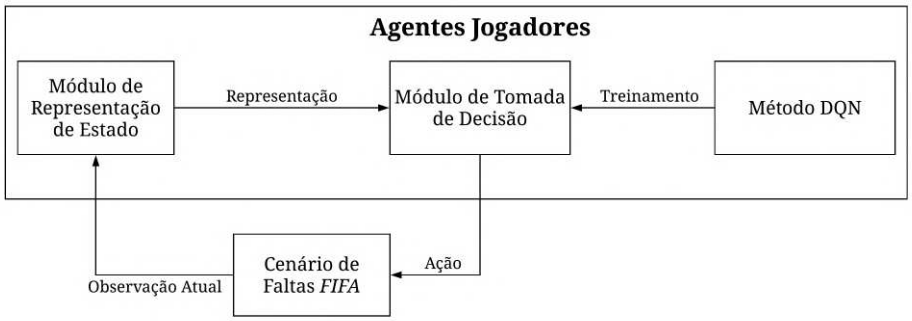


Figura 28 – Arquitetura geral dos agentes implementados por meio da abordagem de aprendizado por reforço

A Tabela 11 resume as principais características dos agentes propostos para os cenários de faltas aqui estudados. Eles possuem algumas variações relativas às representações de estados, as quais são apresentadas na sequência.

Tabela 11 – Agentes Implementados Baseados no Método DQN

Agente	Cenário de Faltas	Representação de Estado	Ações Possíveis (Saída da RNA)
DQN-SG-Agent	Sem Goleiro (*SG*)	Variável	4
DQN-CG-Agent	Com Goleiro (*CG*)	Variável	4

- *DQN-SG-Agent* e *DQN-CG-Agent* com **Escala de Cinza 4 frames**: considera a representação por imagens brutas sem informação de cor (descrita na subseção 5.2.1). O módulo de tomada de decisão corresponde a uma RNC com mesma arquitetura da *Q-Network* original (apresentada na subseção 2.3.1).

- *DQN-SG-Agent* e *DQN-CG-Agent* com **RGB**: considera a representação por imagens brutas com informação de cor (descrita na subseção 5.2.2). O módulo de tomada de decisão corresponde a uma RNC com mesma arquitetura da *Q-Network* original.

- *DQN-SG-Agent* e *DQN-CG-Agent* com **Representação TDO 1**: considera a **Representação TDO 1** (descrita na subseção 5.2.3). O módulo de tomada de decisão é representado por uma PMC com mesma arquitetura daquela utilizada pelo agente *Trivedi* (apresentada na Tabela 7 na seção 3.2).

- *DQN-SG-Agent* e *DQN-CG-Agent* com **Representação TDO 2**: considera a **Representação TDO 2** (descrita na subseção 5.2.3). O módulo de tomada de decisão é representado por uma PMC com mesma arquitetura daquela utilizada pelo agente *Trivedi*.

5.4 Experimentos e Resultados

Os experimentos aqui têm como objetivo geral avaliar a qualidade do processo de aprendizado dos agentes - baseados no método DQN - levando em consideração as representações de estado baseadas em imagens cruas (também referidas como representações ingênuas) e as representações baseadas em TDO descritas na subseção 5.2. Para tanto, foi efetuado um experimento para cada cenário por meio dos agentes *DQN-SG-Agent* (Experimento 1) e *DQN-CG-Agent* (Experimento 2). Tais experimentos foram conduzidos por meio de testes comparativos com base nos seguintes parâmetros: taxa de chutes bem-sucedidos (taxa de gols) e tempo de treinamento. Saliente-se que, exclusivamente nesta seção, o termo época tem o mesmo significado de episódio. Os experimentos foram executados em uma máquina com uma GPU Nvidia GeForce GTX-745 e 16 GB de RAM.

5.4.1 Experimento 1: Análise do Agente *DQN-SG-Agent* baseado em Distintas Representações

O objetivo aqui é avaliar as representações de estados descritas na seção 5.2 no cenário de faltas sem goleiro por meio do agente *DQN-SG-Agent*. Para tanto, de modo a facilitar a análise do impacto de cada representação, ele foi decomposto em três fases que serão explanadas a seguir.

Fase 1 - Comparação entre o *DQN-SG-Agent* com Representações Ingênuas e o agente *Trivedi*:

Essa fase efetua uma análise comparativa entre as representações ingênuas propostas neste capítulo e a representação utilizada em (TRIVEDI, 2018b), por meio das

variações *DQN-SG-Agent* com **Escala de Cinza 4 frames**, *DQN-SG-Agent* com **RGB** e do agente *Trivedi*. Além disso, ela valida a utilização da interface de conexão descrita no capítulo 4. Os resultados aqui obtidos permitiram a publicação do artigo *A Deep Reinforcement Learn-Based FIFA Agent with Naive State Representations and Portable Connection Interfaces* na conferência *The 32nd International FLAIRS Conference* (FARIA; JULIA; TOMAZ, 2019b).

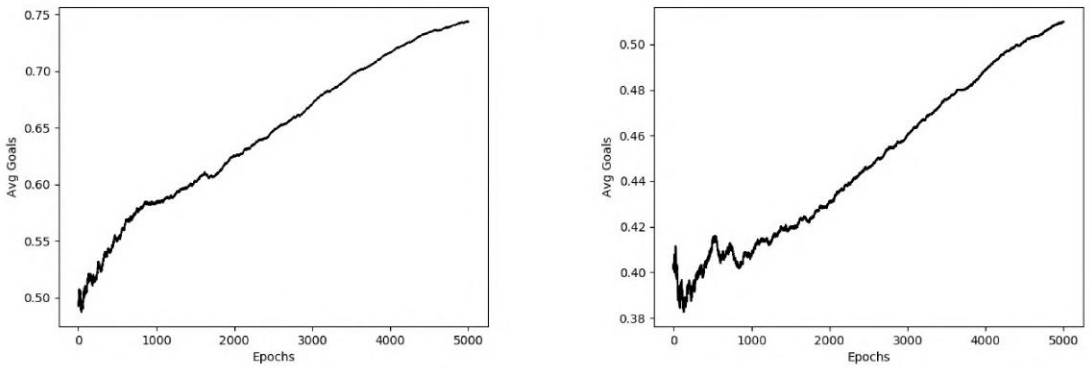
Nesse contexto, tal análise foi alcançada por meio de dois casos de teste: o primeiro (I) avalia a qualidade dos agentes considerando as quatro ações descritas na subseção 5.1 (mover para a esquerda, mover para a direita, chute rasteiro e chute alto). O segundo (II) avalia a qualidade dos agentes de acordo com apenas três ações (mover para a esquerda, mover para a direita e chute rasteiro). Em ambos, cada agente foi treinado no decorrer de 5000 épocas (uma época termina quando uma ação de chute é executada). Deve-se notar que, antes do início do treinamento, são coletados 500 exemplos (transições) a partir de um comportamento totalmente exploratório por parte dos agentes que são armazenados no *buffer* de experiências (eles agem de forma aleatória no ambiente para aumentar a diversidade dos exemplos que compõem este *buffer*). Os hiperparâmetros utilizados para o treinamento são apresentados na Tabela 12, extraídos do estado da arte (TRIVEDI, 2018b). Por fim, destaca-se que foram efetuadas cinco execuções de treinamento para cada variação e os resultados expostos são com relação à melhor execução (em termos da taxa de gols acumulada) de cada uma.

Tabela 12 – Hiperparâmetros do DQN Utilizados na Fase 1

Hiperparâmetro	**Valor**
Tamanho do Mini-Lote	32
Tamanho do *buffer* de experiências	10000
Atualização da *Target Network*	1
Fator de desconto (*Gamma*)	0,9
Taxa de aprendizagem	0,0001
Valor inicial de exploração (*Epsilon*)	1
Valor final de exploração (*Epsilon*)	0,1

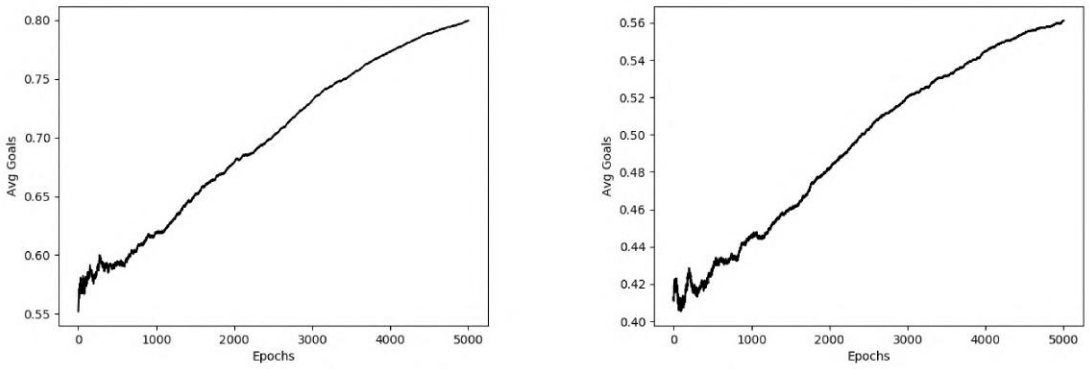
As Figuras 29 e 30 ilustram o processo de aprendizado, respectivamente, de *DQN-SG-Agent* com **Escala de Cinza 4 frames** e de *DQN-SG-Agent* com **RGB** em termos da taxa média de gols ao longo das 5000 épocas utilizadas no treinamento para os dois casos de teste considerados.

Conforme é possível notar, a taxa média de gols é uma grandeza diretamente proporcional ao número de épocas durante o treinamento de tais agentes. Além disso, a taxa associada à época zero é correspondente à taxa obtida na coleta das 500 transições utilizadas para popular o *buffer* de experiências antes de se iniciar o treinamento de fato (ou seja, representa o comportamento totalmente aleatório). A



(a) (b)

Figura 29 – Gráficos de aprendizado de *DQN-SG-Agent* com **Escala de Cinza 4 frames** (eixo das coordenadas corresponde ao número de épocas e o eixo das abscissas representa a taxa média acumulada de gols): (a) Caso de Teste I e (b) Caso de Teste II



(a) (b)

Figura 30 – Gráficos de aprendizado de *DQN-SG-Agent* com **RGB** (eixo das coordenadas corresponde ao número de épocas e o eixo das abscissas representa a taxa média acumulada de gols): (a) Caso de Teste I e (b) Caso de Teste II

Tabela 13 apresenta as taxas médias de gols obtidas pelo *DQN-SG-Agent* considerando cada representação ingênua ao final do treinamento em ambos os casos de teste, mostrando uma superioridade da variação baseada em imagem crua com informação de cor em ambas as situações. As taxas mais baixas apresentadas no caso de teste II em relação ao caso I podem ser explicadas pelo seguinte fato: a maior dificuldade de se marcar gols sem contar com a ação de chute alto, especialmente em situações que envolvem barreiras.

Com relação ao tempo de treinamento, *DQN-SG-Agent* com **Escala de Cinza 4 frames** gastou 16 horas, enquanto *DQN-SG-Agent* com **RGB** levou 14 horas para treinar no caso de teste I. No caso II, os tempos de treinamento de *DQN-*

Tabela 13 – Taxas médias acumuladas de gols de *DQN-SG-Agent* em cada caso de teste com 5000 épocas de treinamento

	DQN-SG-Agent com **Escala de Cinza 4 frames**	*DQN-SG-Agent* com **RGB**
Caso de Teste I	75%	80%
Caso de Teste II	50%	56%

SG-Agent com **Escala de Cinza 4 frames** e de *DQN-SG-Agent* com **RGB** foram, respectivamente, 22 horas e 18 horas. O tempo foi consideravelmente maior no último cenário devido ao fato de que os agentes necessitaram de uma maior interação com o ambiente em termos de número de transições (executando ações e recebendo *feedbacks* com relação a elas) de modo a descobrir como evitar chutes na barreira e convergir a um comportamento apropriado, uma vez que apenas o chute rasteiro estava disponível a eles.

A Tabela 14 mostra a comparação entre os resultados obtidos pelas variações do *DQN-SG-Agent* implementadas neste trabalho e pelo agente *Trivedi*. Destaca-se que este agente não foi executado nesta fase, sendo que o resultado de sua performance foi extraído de (TRIVEDI, 2018b). É importante observar que apenas o caso de teste I com 1000 épocas de treinamento é avaliado, uma vez que somente ele foi considerado no referido trabalho. Os resultados mostram uma superioridade de *DQN-SG-Agent* com **RGB**, mesmo usando uma máquina menos poderosa do que a utilizada e descrita em (TRIVEDI, 2018b). Dessa forma, os autores do presente trabalho efetuaram um estudo sobre as técnicas utilizadas para a construção do agente *Trivedi* e, baseado nas fragilidades encontradas em tal abordagem relativas à TDO, alguns aprimoramentos sobre ela foram propostos, conforme já apresentado na subseção 5.2.3. Isto é explorado na próxima fase.

Tabela 14 – Taxas médias de gols das variações de *DQN-SG-Agent* e de *Trivedi* no caso de teste I com 1000 épocas de treinamento

	DQN-SG-Agent com **Escala de Cinza 4 frames**	*DQN-SG-Agent* com **RGB**	*Trivedi*
Caso de Teste I	58%	62%	50%

Os autores acreditam que o melhor desempenho obtido pelo *DQN-SG-Agent* com **RGB** se deve à adição de informações sobre cores ao processo de tomada de decisão, sendo que as informações sobre a relação de *frames* subsequentes utilizada pelo *DQN-SG-Agent* com **Escala de Cinza 4 frames**, nesse contexto de faltas, não são tão interessantes. De fato, reconhecer o elemento barreira a partir das informações de cor é mais intuitivo do que sem tais informações para um agente humano. No entanto, não é possível afirmar com certeza que isso foi o fator principal para gerar um agente com melhor desempenho, pois a representação baseada em imagens em escala de cinza também considera mais *frames* (e não apenas um, como na

representação **RGB**). Por fim, a interface proposta neste trabalho (apresentada no capítulo 4) foi validada, pois ela serviu como ferramenta auxiliar na melhoria dos agentes treinados pelo método DQN ao intermediar a interação de tais agentes com um ambiente complexo.

Fase 2 - Avaliação dos aprimoramentos realizados sobre o agente *Trivedi*:

Essa fase avalia os aprimoramentos efetuados no presente trabalho sobre a TDO proposta em (TRIVEDI, 2018b) por meio do desempenho dos agentes *Trivedi*, *Trivedi* com *MobileNetV2* e *DQN-SG-Agent* com **Representação TDO 1**. É importante lembrar que o agente *Trivedi* é composto pela versão *MobileNetV1* em sua arquitetura de TDO, enquanto os outros dois foram construídos de acordo com a versão *MobileNetV2*. Assim, essa fase realiza dois estudos: 1) avaliação do impacto da atualização da TDO com base no *MobileNetV1* para a TDO baseada no *MobileNetV2* usada no presente trabalho; 2) avaliação do impacto na qualidade do processo de aprendizagem dos agentes considerando a representação de estado usada em (TRIVEDI, 2018b) e a proposta no presente trabalho que estende essa última adicionando a barreira como um elemento relevante à tarefa de detecção. Ressalta-se que os resultados aqui obtidos geraram um resumo intitulado "Aprimorando o Desempenho na Cobrança de Faltas de Agentes Jogadores de FIFA por meio de Técnicas de Detecção de Objetos" publicado nos anais do XIII Workshop de Teses e Dissertações em Ciência da Computação (WTDCC 2019).

Assim sendo, os agentes foram treinados com as quatro ações descritas na seção 5.1 (mover para a esquerda, mover para a direita, chute rasteiro e chute alto). É importante destacar que cada agente foi treinado no decorrer de 1000 épocas e com os mesmos hiperparâmetros utilizados na fase anterior (apresentados na Tabela 12). 10 sessões foram realizadas com os agentes já treinados, cada uma composta por 100 épocas, para verificar a existência de diferença de desempenho por meio do *Teste T* (descrito na seção 2.8). Por fim, destaca-se que, assim como na **Fase 1**, foram efetuadas cinco execuções de treinamento para cada agente considerado e os resultados expostos são com relação à melhor execução de cada uma (também em termos da taxa de gols acumulada).

Para lidar com o primeiro objetivo de estudo, o agente *Trivedi* (composto pelo *MobileNetV1*) é comparado com uma nova versão implementada com um extrator de características mais atualizado, agente *Trivedi* com *MobileNetV2*. É importante ressaltar que os resultados apresentados por (TRIVEDI, 2018b) e mencionados na seção 3.2 foram obtidos de uma máquina composta com uma GPU GTX-1070 (mais recente e, portanto, com desempenho superior à máquina usada pelo presente trabalho). Com o propósito de fornecer coerência a esse estudo, ambas as versões foram

executadas e treinadas na mesma máquina. Isso foi possível, pois o referido trabalho está disponibilizado à comunidade[2]. Portanto, a principal contribuição desse primeiro estudo é investigar o impacto no processo de aprendizado causado pela utilização do *MobileNetV2* como extrator de características.

A Tabela 15 mostra que o agente *Trivedi* com *MobileNetV2* apresentou um nível de jogo semelhante em termos da taxa média de gols em comparação ao agente *Trivedi* na fase de treinamento. No entanto, obteve um desempenho muito melhor em relação ao tempo de treinamento. De fato, aquele exigiu 90 minutos a menos (25% menos tempo) do que o agente *Trivedi* para concluir o treinamento e com uma taxa de gols ligeiramente maior. Além disso, é necessário enfatizar que o resultado com relação à taxa média de gols do agente *Trivedi* aqui obtido foi diferente daquele exposto na **Fase 1** - extraído de (TRIVEDI, 2018b). Os autores da presente pesquisa encontraram evidências de erro nos cálculos da taxa média de gols em (TRIVEDI, 2018b), o que foi relatado ao autor de tal trabalho.

Tabela 15 – Desempenho dos agentes *Trivedi* e *Trivedi* com *MobilenetV2* com relação à etapa de treinamento considerando 1000 épocas (em termos da taxa média acumulada de gols e do tempo de treinamento)

Agente	**Taxa média acumulada de gols (em %)**	**Tempo de Treinamento (em minutos)**
Trivedi	62,1	360
Trivedi com *MobileNetV2*	63,4	270

Assim, foi realizado um teste final a fim de comparar o desempenho desses agentes já treinados, executando 10 sessões compostas de 100 épocas cada. A Tabela 16 mostra, respectivamente, a taxa de gols (em %), a média total e o desvio padrão de ambos os agentes ao longo das sessões. Os resultados obtidos indicam um desempenho muito semelhante entre os agentes.

Tabela 16 – Desempenho dos agentes *Trivedi* e *Trivedi* com *MobileNetV2* em relação à etapa de teste considerando 10 sessões (em termos da taxa de gols)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Média**	**Desvio Padrão**
Trivedi	76	77	77	81	72	77	72	71	78	82	76,3	3,713
Trivedi com *MobileNetV2*	77	76	81	79	82	74	85	75	73	77	77,9	3,814

Nesse sentido, um *Teste T* com um *nível de significância* ($\alpha = 0,05$) foi efetuado para verificar a existência ou não de diferença de desempenho entre eles. Dessa maneira, a seguinte hipótese nula H_0 foi criada: o agente *Trivedi* mantém o mesmo nível de desempenho que o agente *Trivedi* com *MobileNetV2*.

Como indicado na Tabela 17, o *valor-t* negativo (t(18) = -0,951) sugere que a taxa média de gols obtida pelo agente composto pelo *MobileNetV2* é maior do que a média do agente composto pelo *MobileNetV1* com *graus de liberdade* igual a 18.

[2] <https://github.com/ChintanTrivedi/DeepGamingAI_FIFARL>

No entanto, como *valor-p* $= 0,354$ não é menor do que o *nível de significância* ($\alpha = 0,05$), a hipótese nula não pode ser rejeitada. Assim, esses resultados indicam que a alteração do extrator de características não afeta significativamente a taxa de gols de um agente implementado no cenário de faltas simplificado. Portanto, conclui-se que a grande vantagem do agente *Trivedi* com *MobileNetV2* sobre o agente *Trivedi* é a maior rapidez na fase de treinamento.

Tabela 17 – *Teste T* aplicado aos resultados obtidos pelos agentes *Trivedi* e *Trivedi* com *MobileNetV2* na etapa de teste

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
Trivedi x *Trivedi* com *MobileNetV2*	-0,951	18	0,354

Na segunda avaliação de estudo efetuado nessa fase, o agente *Trivedi* com *MobileNetV2* é comparado a uma nova versão - *DQN-SG-Agent* com **Representação TDO 1** - implementada com a mesma TDO (também composta pelo *MobileNetV2*). No entanto, esta última incorpora, adicionalmente, o elemento barreira em sua representação (conforme apresentado na subseção 5.2.3). Assim, a principal contribuição desse experimento é investigar o impacto no processo de aprendizagem causado pela adição da barreira aos elementos identificados pela TDO.

A Tabela 18 mostra que o *DQN-SG-Agent* com **Representação TDO 1** apresentou um nível de jogo mais alto em termos da taxa média gols em comparação ao agente *Trivedi* com *MobileNetV2* na fase de treinamento. De fato, o primeiro obteve uma taxa de 3,3% a mais em relação ao segundo. No entanto, em termos de tempo de treinamento, o desempenho foi semelhante.

Tabela 18 – Desempenho dos agentes *Trivedi* com *MobileNetV2* e *DQN-SG-Agent* com **Representação TDO 1** com relação à etapa de treinamento considerando 1000 épocas (em termos da taxa média acumulada de gols e do tempo de treinamento)

Agente	**Taxa média acumulada de gols (em %)**	**Tempo de Treinamento (em minutos)**
Trivedi com *MobileNetV2*	63.4	270
DQN-SG-Agent com **Representação TDO 1**	66.7	290

Assim, foi realizado um teste final a fim de comparar o desempenho desses agentes já treinados, executando 10 sessões compostas de 100 épocas cada. A Tabela 19 mostra, respectivamente, a taxa de gols (em %), a média total e o desvio padrão de ambos os agentes ao longo das sessões. Um *Teste T* com um *nível de significância* ($\alpha = 0,01$) foi efetuado para verificar a existência ou não de diferença de desempenho entre eles. Dessa maneira, a seguinte hipótese nula H_0 foi criada: o agente *Trivedi* com *MobileNetV2* mantém o mesmo nível de desempenho que o *DQN-SG-Agent* com **Representação TDO 1**.

Tabela 19 – Desempenho dos agentes *Trivedi* com *MobileNetV2* e *DQN-SG-Agent* com **Representação TDO 1** em relação à etapa de teste considerando 10 sessões (em termos da taxa de gols)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Média**	**Desvio Padrão**
Trivedi com *MobileNetV2*	77	76	81	79	82	74	85	75	73	77	77.9	3.814
DQN-SG-Agent com **Representação TDO 1**	85	81	87	83	85	83	87	82	83	80	83.6	2.366

Como apresentado na Tabela 20, o *valor-t* negativo (t(18) = -4,016) indica que a taxa média de gols obtida pelo *DQN-SG-Agent* com **Representação TDO 1** é significativamente maior do que a a média obtida pelo agente *Trivedi* com *MobileNetV2* com *graus de liberdade* igual a 18. Como o *valor-p* $= 0,001$ é menor que o *nível de significância* ($\alpha = 0,01$), a hipótese nula pode ser rejeitada. Esses resultados sugerem que a nova representação de estado (incorporando a barreira) tem um efeito significativo na taxa de gols obtidas por um agente jogador considerando um intervalo de confiança de 99% para a diferença média. Portanto, conclui-se que a barreira é realmente um elemento relevante no contexto de faltas e, consequentemente, o agente que a incorpora em sua representação obteve um desempenho significativamente melhor que aquele que não a considera no cenário de cobrança de faltas sem goleiro.

Tabela 20 – *Teste T* aplicado aos resultados obtidos pelos agentes *Trivedi* com *MobileNetV2* e *DQN-SG-Agent* com **Representação TDO 1** na etapa de teste

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
Trivedi com *MobileNetV2* x *DQN-SG-Agent* com **Representação TDO 1**	-4.016	18	0.001

Os resultados obtidos atestam a eficácia do *MobileNetV2* sobre o *MobileNetV1* como extrator de características para gerar a representação de estado usada por agentes inteligentes (no caso, agentes baseados em ARP) com relação ao tempo de treinamento. Deve-se notar que esses resultados foram consistentes com os apresentados em (SANDLER et al., 2018), o qual compara o *MobileNetV1* e o *MobileNetV2* em um renomado conjunto de dados, pois ambos obtiveram desempenho semelhante, diferindo apenas no tempo de processamento. A versão de TDO baseada no *MobileNetV2* foi melhor (em termos de tempo de processamento) nos cenários avaliados em (SANDLER et al., 2018) e no cenário avaliado no presente trabalho. O tempo de processamento de cada imagem não foi calculado diretamente neste último cenário, sendo medido em termos do tempo de treinamento dos agentes. Destaca-se que esse tempo de processamento será utilizado como uma medida de avaliação na **Fase 3**.

Além disso, destaca-se que a inclusão da barreira como elemento a ser identificado e processado pela TDO foi fundamental para melhorar a percepção do ambiente e, consequentemente, resultou no aprimoramento no processo de aprendizado do agente. Isto já era esperado pela importância de tal elemento na tomada de decisão de um agente. Dessa forma, o presente trabalho evidencia a importância da

qualidade da representação dos dados no processo de aprendizagem de um agente baseado em ARP.

Fase 3 - Avaliação Detalhada de *DQN-SG-Agent* e suas Variações

Essa última fase efetua uma análise comparativa detalhada entre as representações geradas por imagens cruas (**Escala de Cinza 4 frames** e **RGB**) e por TDO (**Representação TDO 1** e **Representação TDO 2**) por meio do *DQN-SG-Agent*. Para tanto, os seguintes parâmetros avaliativos são considerados: tempo de convergência; tempo de processamento; tempo de tomada de decisão. Ao contrário das fases anteriores as quais consideravam um número fixo de épocas na etapa de treinamento e efetuavam a medição do tempo gasto em tal etapa em minutos ou horas, esta fase mensura o tempo de treinamento pelo número de episódios necessários para que um agente convirja a uma taxa de gols de 85% nos últimos 100 episódios (tal taxa foi escolhida com base nos resultados obtidos nas fases anteriores), por isso o nome tempo de convergência. O tempo de processamento corresponde ao processo de geração da representação de estado. O tempo de tomada de decisão corresponde ao processo de escolha de ação do agente.

Nesse contexto, os agentes foram treinados com as quatro ações descritas na seção 5.1 (mover para a esquerda, mover para a direita, chute rasteiro e chute alto). Deve-se notar que, antes do início do treinamento, foram coletados 500 exemplos a partir de um comportamento totalmente exploratório por parte dos agentes que são armazenados no *buffer* de experiências. Os hiperparâmetros utilizados para o treinamento são apresentados na Tabela 21. Assim, foi executado o processo de treinamento 10 vezes em cada agente de modo independente e os resultados são apresentados a seguir.

Tabela 21 – Hiperparâmetros do DQN Utilizados no Experimento 3

Hiperparâmetro	**Valor**
Tamanho do Mini-Lote	32
Tamanho do *buffer* de experiências	10000
Atualização da *Target Network*	500
Fator de desconto (*Gamma*)	0.9
Taxa de aprendizagem	0.0001
Valor inicial de exploração (*Epsilon*)	1.0
Valor final de exploração (*Epsilon*)	0.1

A Tabela 22 mostra, respectivamente, o número de amostras (N), a média total, a mediana e o desvio padrão com relação ao tempo de convergência (representando o número médio de episódios necessários para a convergência) de tais agentes considerando 10 execuções de treinamento. Os resultados indicam um melhor desempenho da variação de *DQN-SG-Agent* produzida de acordo com a **Representação de**

TDO 2, a qual obteve uma média de 782 episódios aproximadamente para convergir a uma taxa de gols de 85%. Além disso, é possível verificar que as taxas obtidas pela variação que utiliza a representação de **Escala de Cinza 4 frames** são muito superiores em relação às outras, o que sugere um comportamento substancialmente pior dela.

Tabela 22 – Resultados de *DQN-SG-Agent* e suas variações em termos do tempo de convergência (em número de episódios)

	N	Média	Mediana	Desvio Padrão
DQN-SG-Agent com **Escala de Cinza 4 frames**	10	1577,10	1319,00	833,878
DQN-SG-Agent com **RGB**	10	912,40	897,50	220,336
DQN-SG-Agent com **Representação TDO 1**	10	883,10	861,50	129,071
DQN-SG-Agent com **Representação TDO 2**	10	781,80	777,50	143,056

A Figura 31 mostra o histograma relativo ao tempo de convergência (medido em termos do número de episódios). Nela, é possível observar que a variável *Episódios* está em um intervalo de, aproximadamente, 500 a 3500. Além disso, ela não é normalmente distribuída conforme verificado pela curva normal no histograma. De fato, há uma assimetria positiva na qual a média (1038,60) é maior do que a moda (representada por algum valor entre 500 a 1000). É bem provável que as altas taxas obtidas pela variação *DQN-SG-Agent* com **Escala de Cinza 4 frames** tenham sido responsáveis por afetar drasticamente essa distribuição de dados (o desvio padrão principalmente). Isso geralmente não é um problema para amostras maiores (por exemplo, de pelo menos 30 execuções em cada grupo). No entanto, para a pequena amostra disponível (10 execuções para cada grupo), isso representa um problema real na análise comparativa efetuada por alguns métodos estatísticos.

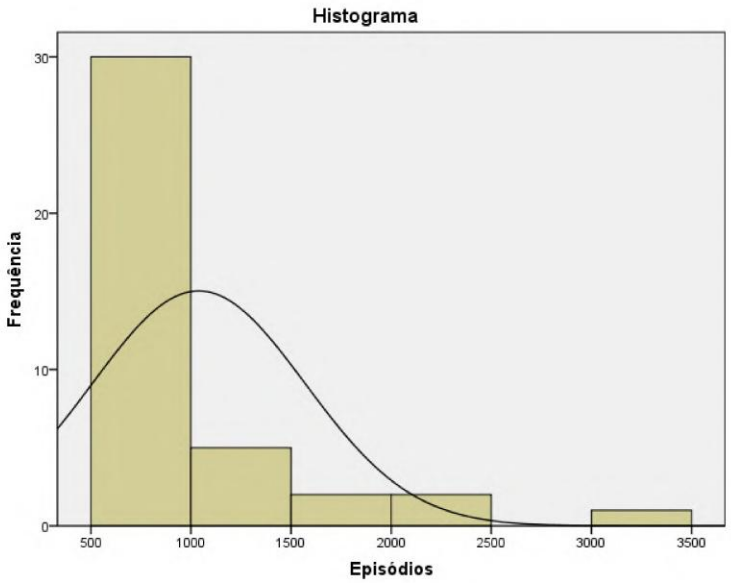


Figura 31 – Histograma relativo ao Tempo de Convergência medido em Episódios

Desta forma, de modo a estimar a significância de tais resultados, o método *Kruskal-Wallis* (descrito na seção 2.8) com um *nível de significância* ($\alpha = 0,05$) foi efetuado, conforme mostrado na sequência. Assim, foi criada a seguinte hipótese nula H_0: as variações de *DQN-SG-Agent* possuem o mesmo nível de desempenho em termos do tempo de convergência.

O *valor-p* $= 0,003$ encontrado pelo *Kruskal-Wallis* foi menor do que o *nível de significância* ($\alpha = 0{,}01$). Portanto, há evidências muito fortes para sugerir uma diferença entre pelo menos um par de grupos (representados pelos agentes envolvidos). Assim, H_0 foi rejeitada e o *Teste de Dunn-Bonferroni* foi realizado entre os seis pares de grupos possíveis.

Conforme indicado na Tabela 23, existe uma diferença estatisticamente significativa entre o *DQN-SG-Agent* com **Escala de Cinza 4 frames** e o *DQN-SG-Agent* com **Representação TDO 2** (*valor-p* = 0,002). O tempo médio (neste caso, representado pela mediana devido aos motivos discutidos no método *Kruskal-Wallis* na seção 2.8) de convergência para *DQN-SG-Agent com* ***Escala de Cinza 4 frames*** foi de 1319 episódios em comparação com 778 episódios de *DQN-SG-Agent com* ***Representação TDO 2***. No entanto, não houve evidência de diferença de desempenho entre os outros pares de agentes.

Tabela 23 – *Teste de Dunn-Bonferroni* aplicado aos resultados de *DQN-SG-Agent* e suas variações. O símbolo '*' indica que a diferença média é significativa considerando um *nível de significância* de 0,05

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
Representação TDO 2 - Escala de Cinza 4 frames	19,100	3,653	,002*
Representação TDO 1 - Escala de Cinza 4 frames	13,200	2,525	,069
RGB - Escala de Cinza 4 frames	12,900	2,467	,082
Representação TDO 2 - RGB	6,200	1,186	1,000
Representação TDO 2 - Representação TDO 1	5,900	1,129	1,000
Representação TDO 1 - RGB	,300	,057	1,000

As Tabelas 24 e 25 apresentam algumas estatísticas descritivas (média e desvio padrão) relativas, respectivamente, ao tempo de processamento (mensurado pelo segundos gastos para gerar a representação de estado) e ao tempo de tomada de decisão (também medido em segundos) de *DQN-SG-Agent* e suas variações considerando as 10 execuções de treinamento realizadas.

Quanto ao tempo de processamento, os resultados sugerem que a representação de estado mais rápida de ser gerada é aquela baseada em **Escala de Cinza 4 frames** ($0{,}01 \pm 0{,}002$ segundos) e a mais lenta é a oriunda da **Representação de TDO 2** ($0{,}17 \pm 0{,}03$ segundos). Neste caso, é interessante observar que o tempo para gerar as representações de TDO são maiores. Isso é devido ao fato delas serem provenientes de modelos compostos por RNPs (ou seja, de procedimentos computacionais de maior custo do que os utilizados para gerar as representações ingênuas).

Já o tempo de tomada de decisão é semelhante entre os agentes, sendo mais rápido nos agentes baseados em representação de TDO por serem compostos por uma arquitetura de rede neural com menos parâmetros em relação às arquiteturas de RNC utilizadas nos agentes baseados em imagem crua. Porém, em termos práticos, considera-se que os agentes possuem mesmo desempenho em relação a essa medida.

Tabela 24 – Resultados de *DQN-SG-Agent* e suas variações em termos do tempo de processamento (em segundos)

Agente	N	Média	Desvio Padrão
DQN-SG-Agent com ***Escala de Cinza 4 frames***	10	0,010	0,002
DQN-SG-Agent com ***RGB***	10	0,023	0,001
DQN-SG-Agent com ***Representação TDO 1***	10	0,103	0,001
DQN-SG-Agent com ***Representação TDO 2***	10	0,17	0,03

Tabela 25 – Resultados de *DQN-SG-Agent* e suas variações em termos do tempo de tomada de decisão (em segundos)

Agente	N	Média	Desvio Padrão
DQN-SG-Agent com ***Escala de Cinza 4 frames***	10	0,006	0,002
DQN-SG-Agent com ***RGB***	10	0,013	0,004
DQN-SG-Agent com ***Representação TDO 1***	10	0,002	0,001
DQN-SG-Agent com ***Representação TDO 2***	10	0,002	0,001

Os resultados indicam que a representação **Escala de Cinza 4 frames** é a menos adequada para esse cenário, sendo que as informações de cores relacionadas aos elementos relevantes a serem considerados pela percepção do agente e, consequentemente, em sua tomada de decisão, é algo fundamental para que ele obtenha sucesso mais rapidamente (isto é, com menos interações com o ambiente). Além disso, eles reforçam que a utilização de representações de ambiente oriundas de TDOs em um contexto de ARP propiciam uma boa qualidade dos dados no processo de tomada de decisão de agentes jogadores. Dentre as duas utilizadas - **Representação TDO 1** e **Representação TDO 2** - destaca-se esta última, uma vez que ela proporciona uma interpretabilidade dos dados gerados (relativa às *bounding boxes* criadas pelo módulo *SSD*) muito maior em relação à **Representação TDO 1** (na qual os dados gerados são relativos à ativação da última camada da RNC *MobileNetV2*). Por outro lado, essas representações oriundas de TDO demandam um tempo de processamento muito maior em relação àquelas baseadas em imagens cruas. Ressalta-se que a representação *RGB* mostrou-se também bastante apropriada (por conta das informações de cores), possuindo um desempenho bastante competitivo com relação às representações oriundas de TDO em termos do tempo de convergência. Contudo, apresentou um tempo de processamento muito melhor (bem mais veloz) com relação a elas.

5.4.2 Experimento 2: Análise das Representações de Estado no Cenário de Faltas com Goleiro

Este experimento tem como objetivo avaliar as representações de estados descritas na seção 5.2 no cenário de faltas com goleiro por meio do agente *DQN-CG-Agent*. Nesse contexto, as variações de *DQN-CG-Agent* foram treinadas com as quatro ações descritas na seção 5.1 (mover para a esquerda, mover para a direita, chute rasteiro e chute alto) em 5000 épocas. Deve-se notar que, antes do início do treinamento, foram coletados 500 exemplos a partir de um comportamento totalmente exploratório por parte dos agentes que são armazenados no *buffer* de experiências. Os hiperparâmetros utilizados para o treinamento são os mesmos usados na **Fase 3** do primeiro experimento, apresentados na Tabela 21. Assim, foi executado o processo de treinamento cinco vezes em cada variação de modo independente e os resultados expostos são com relação à melhor execução de cada uma (em termos da taxa de gols acumulada).

A Figura 32 ilustra o processo de aprendizado das variações do *DQN-CG-Agent* em termos da taxa média de gols ao longo das 5000 épocas utilizadas no treinamento. Conforme é possível observar, todas as variações não obtiveram boas taxas médias de gols em relação ao número de épocas (ao contrário do que foi visto no cenário de faltas sem goleiro, na qual a taxa de gols estabeleceu-se como uma grandeza proporcional ao número de épocas). Isto é, o agente *DQN-CG-Agent* falhou em aprender independentemente da percepção do ambiente. De fato, as taxas obtidas por ele ao final da etapa de treinamento com cada representação foram piores do que as obtidas no processo de população do *buffer* de experiências (o que estabelece o início do treinamento), na qual o agente atua de modo completamente aleatório no ambiente.

Os autores acreditam que esse baixo desempenho do *DQN-CG-Agent* se deve às seguintes razões: o fato do método DQN ter uma grande dificuldade em lidar com ambientes complexos onde a função Q é muito complicada para ser aprendida. Nesse cenário, o goleiro apresentou-se como um elemento primordial para que o referido método não obtivesse sucesso. Combinado a isso, a função de recompensa não representou muito bem o conhecimento (em forma de *feedback*) que o agente pode obter do ambiente, pois o resultado de um chute mal-sucedido é sempre o mesmo. Ou seja, um chute errado na trave ou muito próximo ao gol tem o mesmo valor de um chute errado direcionado para bem longe do gol (isto é, a recompensa é a mesma, correspondente ao valor de -1), conforme ilustrado na Figura 33 (o círculo vermelho indica a posição da bola). A consequência disso é que o agente tenta explorar o máximo possível o ambiente para tentar encontrar situações em que consegue um bom chute (situações que resultam em gol). Contudo, são mais raras as transições que resultam nisso em relação ao cenário de faltas sem goleiro, pois mesmo que o agente direcione corretamente o seu chute em direção a uma posição que resulte em gol, há ainda o caráter estocástico do cenário que pode atrapalhar e que é extremamente relevante para o péssimo resultado obtido em termos da taxa de gols.

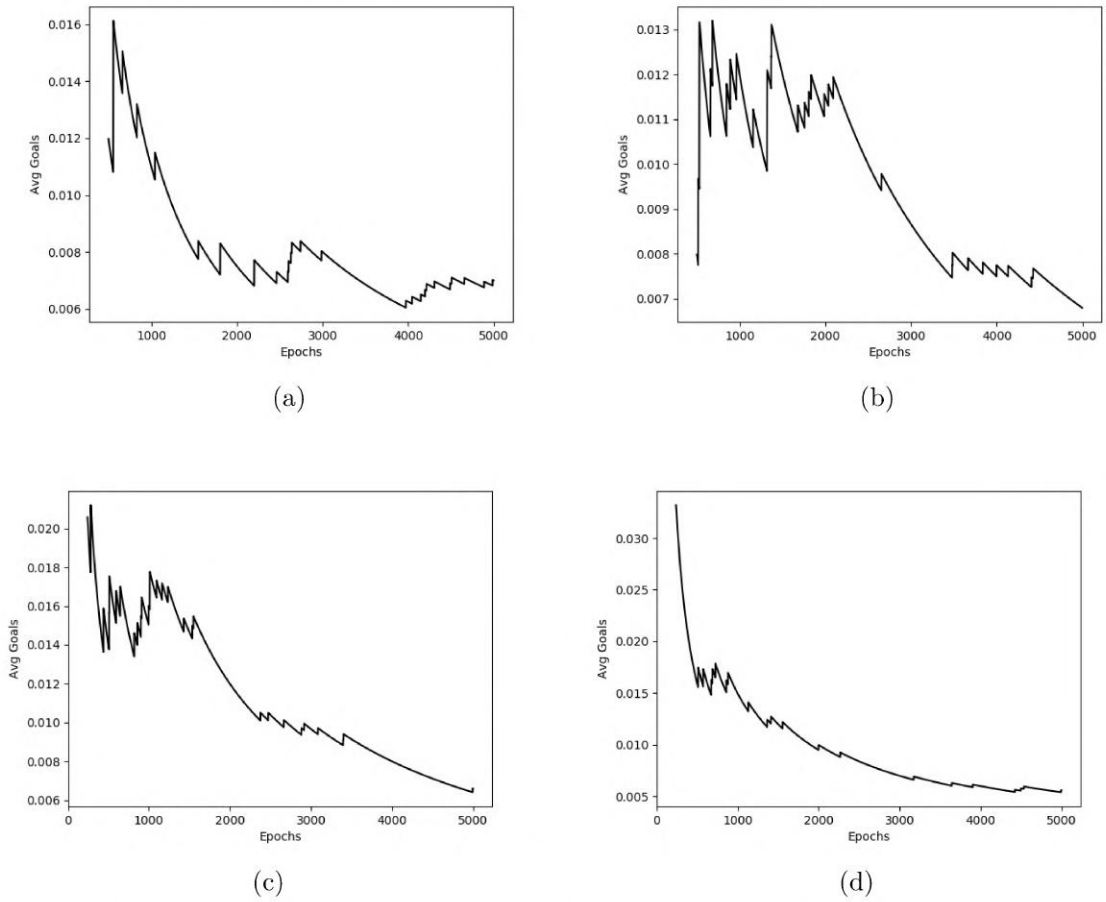


Figura 32 – Gráficos de aprendizado de *DQN-CG-Agent* (eixo das coordenadas corresponde ao número de épocas e o eixo das abscissas representa a taxa média acumulada de gols): (a) **Escala de Cinza 4 frames** (b) **RGB** (c) **Representação de TDO 1** (d) **Representação de TDO 2**

Assim sendo, é possível melhorar a performance de agentes baseados em ARP no contexto de faltas com goleiro a partir da utilização de uma função de recompensa mais detalhada que permita mensurar o erro do agente baseado na distância entre a posição que a bola atinge após o chute e a posição do gol. Para tanto, uma solução plausível seria utilizar técnicas avançadas de visão computacional para efetuar o rastreamento da posição da bola durante as cobranças de faltas (pois só é possível acessar tal posição por meio das imagens do jogo). Também seria interessante estudar métodos de ARP mais recentes, tais como: *Rainbow* (HESSEL et al., 2018), o qual efetua um compilado de várias modificações importantes e que são essenciais para a melhora de desempenho de uma agente baseado em DQN); ou o *A3C*, projetado a partir de uma abordagem diferente da utilizada pelo DQN, denominada gradiente de política, a qual mostrou-se mais robusta para lidar com problemas em que é difícil definir uma função de recompensa detalhada, como o caso do presente cenário (por conta da plataforma fechada *FIFA*). Por fim, a solução adotada pelo presente trabalho foi utilizar uma abordagem supervisionada (por meio de métodos de

(a)

(b)

Figura 33 – Exemplos de chutes mal-sucedidos: (a) Chute Perto (b) Chute Longe

AI) para contornar tais problemas, o que será visto no capítulo 6.

Capítulo 6

Aprendizado por Imitação IAP em cenários de cobrança de faltas no *FIFA*

Este capítulo apresenta a contribuição referente à investigação da abordagem de AI - especificamente, o método IAP - nos cenários de cobrança de faltas sem goleiro (isto é, envolve um ambiente estático e agente único) e com goleiro (engloba um ambiente estático e multiagente), descritos na seção 2.9. O referido método é avaliado por meio do desempenho de agentes projetados de acordo com representações baseadas em imagens cruas e em TDO. Assim, tais representações são descritas na seção 6.1 e as arquiteturas dos agentes são detalhadas na seção 6.2. Por fim, os experimentos e a análise dos resultados de ambos os cenários são apresentados na seção 6.3.

Esta investigação é inspirada na proposta apresentada em (HUSSEIN et al., 2018), a qual validou o método IAP em ambientes de navegação estáticos e agente único. Nesse sentido, a presente contribuição expande tal proposta por investigar o comportamento da IAP em um ambiente multiagente que inclui um goleiro.

6.1 Especificação das Representações de Estados

Esta seção apresenta as representações de estados utilizadas pelos agentes implementados neste capítulo. A subseção 6.1.1 apresenta a representação do ambiente por meio de imagens cruas sem informação de cor, enquanto a subseção 6.1.2 exibe a representação por meio de imagens cruas com informação de cor. Por fim, as representações baseadas em TDO aqui utilizadas são descritas na subseção 6.1.3.

6.1.1 Escala de Cinza

Esta representação transforma a imagem original (representando o *frame* atual) em uma imagem de tamanho menor (dimensões 120x90) e a converte para escala de cinza (removendo as informações de cores), conforme mostrado na Figura 34. Nota-se que a

representação formada é baseada apenas no *frame* atual (diferentemente da representação **Escala de Cinza 4 *frames*** considerada no capítulo 5, a qual é baseada nos últimos quatro *frames* mais recentes, conforme descrito na seção 5.2). Portanto, cada estado é representado por uma imagem de tamanho 120x90 (isto é, o estado é uma matriz 120x90x1) produzidas por esta etapa de pré-processamento. Ressalta-se que a diferença do tamanho da imagem com relação ao capítulo anterior (de tamanho 84x84) tem como propósito a adaptação à arquitetura original proposta em (HUSSEIN et al., 2018).

Figura 34 – Exemplo da Representação por Imagem em Escala de Cinza.

6.1.2 RGB

Esta representação é baseada em imagens brutas coloridas no formato RGB. A etapa de pré-processamento reduz a dimensão da imagem original para 120x90 e mantém as informações de cores contidas nas imagens, conforme ilustrado na Figura 35. Portanto, cada estado é representado pela imagem corrente de jogo de tamanho 120x90x3 (3 canais de cores da RGB) produzida pela etapa de pré-processamento.

Figura 35 – Exemplo da Representação por Imagem em RGB.

6.1.3 TDOs

As representações oriundas de TDO aqui utilizadas são provenientes daquelas discutidas na subseção 5.2.3 do capítulo anterior. Ou seja, no cenário de faltas sem goleiro, os elementos relevantes considerados na tarefa de detecção são a bola, a barreira, o gol e o jogador que cobra faltas. No cenário de faltas com goleiro, o elemento goleiro é adicionado dentre aqueles levados em consideração pela TDO. Considerando este último cenário, é proposta aqui uma modificação com relação aos objetos relevantes para a tarefa de detecção, sendo agora considerados os elementos bola, barreira, goleiro e os quatro alvos

ao redor do gol, conforme ilustrado na Figura 36. Destaca-se que os elementos gol e jogador cobrador de faltas foram removidos do conjunto considerado (tais remoções serão discutidas nos experimentos).

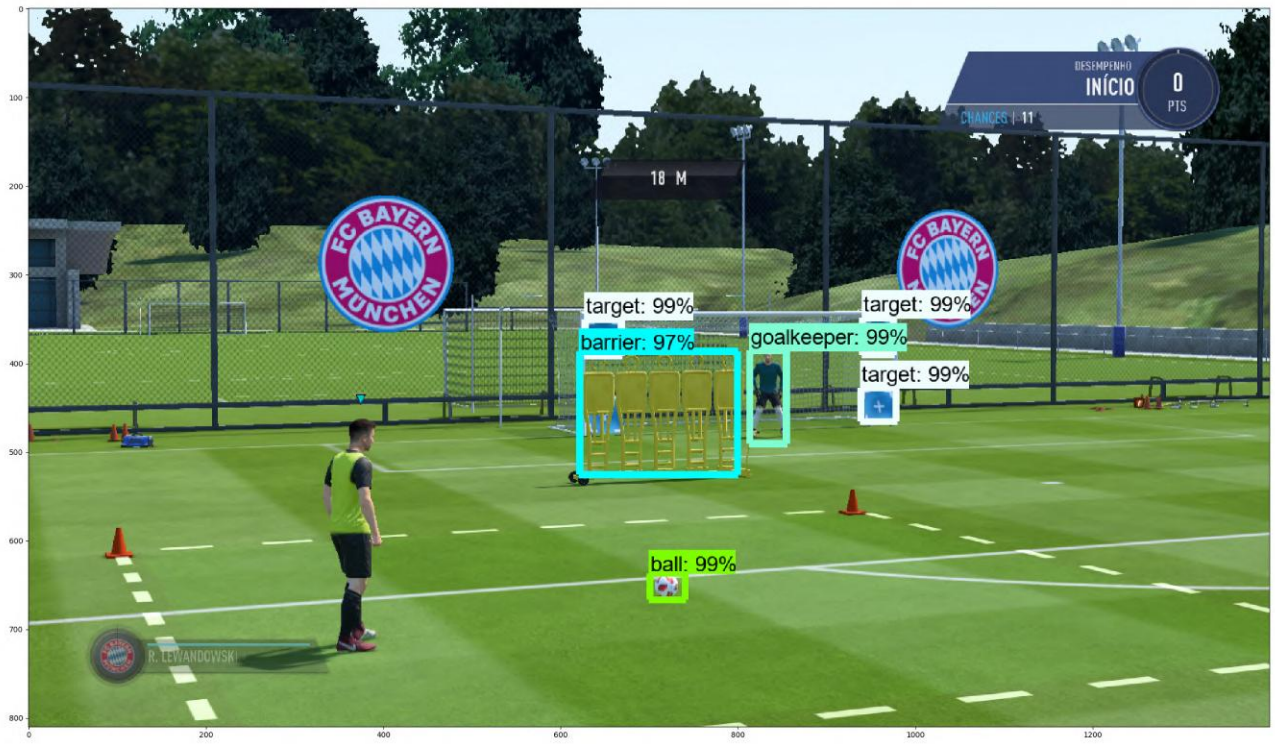


Figura 36 – Exemplo da modificação da TDO utilizada no cenário de faltas com goleiro.

Assim sendo, as representações baseadas em TDO propostas no capítulo 5 (correspondentes à **Representação TDO 1** e à **Representação TDO 2**) são mantidas e utilizadas no presente capítulo, além das seguintes representações oriundas da TDO modificada para lidar com o cenário de faltas com goleiro:

- **Representação TDO 1 Modificada**: 128 valores representando o *Mapa de Características de Alto Nível* (conforme apresentado na subseção 2.6.3).
- **Representação TDO 2 Modificada**: representação de estado proveniente do módulo *SSD* da arquitetura de TDO (conforme explicado na subseção 2.6.3). A Tabela

26 exibe a representação utilizada no cenário de faltas com goleiro, formada por um vetor de 28 valores (quatro valores correspondentes a cada um dos elementos).

Tabela 26 – **Representação TDO 2 Modificada** para o Cenário de Faltas com Goleiro

Posição	**Característica**
0	X_{min} bola
1	X_{max} bola
2	Y_{min} bola
3	Y_{max} bola
4	X_{min} alvo 1
5	X_{max} alvo 1
6	Y_{min} alvo 1
7	Y_{max} alvo 1
8	X_{min} alvo 2
9	X_{max} alvo 2
10	Y_{min} alvo 2
11	Y_{max} alvo 2
12	X_{min} alvo 3
13	X_{max} alvo 3
14	Y_{min} alvo 3
15	Y_{max} alvo 3
16	X_{min} alvo 4
17	X_{max} alvo 4
18	Y_{min} alvo 4
19	Y_{max} alvo 4
20	X_{min} barreira
21	X_{max} barreira
22	Y_{min} barreira
23	Y_{max} barreira
24	X_{min} goleiro
25	X_{max} goleiro
26	Y_{min} goleiro
27	Y_{max} goleiro

6.2 Arquitetura dos Agentes Jogadores Baseados em Distintas Representações de Estado

Esta seção apresenta a implementação dos agentes aqui propostos baseados em RNPs e no método IAP para lidar com os dois cenários de cobrança de faltas. Nesses cenários, o conjunto de ações legais disponíveis sempre que os agentes devem selecionar uma ação no estado atual do jogo é: mover para a esquerda, mover para a direita, chute rasteiro e chute alto. É importante ressaltar que tais ações foram especificadas na seção 5.1. A arquitetura geral desses agentes é ilustrada na Figura 37.

Conforme mencionado acima, em termos de estratégia de aprendizagem, os agentes aqui são treinados de acordo com o método IAP, ou seja, são inicialmente treinados por uma abordagem supervisionada tradicional (treinamento de um modelo a partir de um conjunto de dados representando pares entrada e saída) e, em seguida, a política de tomada de decisão produzida com esse treinamento é refinada a partir do aprendizado ativo. É importante ressaltar que os dados de treinamento (exemplos) usados nas duas

Figura 37 – Arquitetura Geral dos Agentes Baseados em IAP para os Cenários de Faltas.

etapas de treinamento (supervisionada tradicional e aprendizado ativo, conforme descrito na subseção 2.3.2) são fornecidos pelo mesmo supervisor humano. Cada exemplo é um par composto pelos seguintes elementos:

- **Observação**: esse elemento é a representação relativa a uma determinada situação de jogo. Neste contexto, distintas formas de representação de estados são exploradas, como mostrado na seção 6.1.

- **Ação**: esse elemento representa a ação selecionada por um determinado agente humano (considerando o conjunto possível de ações) na situação do jogo apresentada na sua respectiva observação (primeiro elemento do par).

Neste sentido, os exemplos foram recuperados - por meio da interface descrita no capítulo 4 - de jogos nos quais o agente supervisor humano atuou nos dois cenários de cobrança de faltas considerados (sem e com goleiro). Nesse sentido, o humano é representado por um dos autores que possui uma boa expertise nos cenários de faltas (nível avançado). Para cada cenário, um conjunto de dados composto por 400 demonstrações foi coletado. Tais conjuntos são desbalanceados, no sentido de não possuir a mesma proporção de exemplos correspondentes a cada classe (por exemplo, há um maior número de exemplos relativos à saída de chute alto do que chute rasteiro). Eles foram utilizados para gerar a política inicial de cada agente, que pode ser distinta para cada um deles pelo fato de ser altamente baseada na representação de estado considerada e na inicialização aleatória dos parâmetros iniciais. Em seguida, o número de demonstrações em cada conjunto de dados foi incrementado em 5% por meio do aprendizado ativo, totalizando 420 exemplos. É importante observar que, nesta etapa, as demonstrações foram coletadas pelos agentes a partir de suas respectivas políticas iniciais. O novo conjunto formado, então, foi utilizado para efetuar a segunda etapa de treinamento, gerando a versão final de cada agente. Destaca-se que todos os exemplos do novo conjunto foram considerados

na segunda etapa de treinamento. A Tabela 27 resume as principais características dos agentes propostos para os cenários de faltas aqui estudados.

Tabela 27 – Agentes Implementados Baseados no Método IAP

Agente	Cenário de Faltas	Representação de Estado	Ações Possíveis (Saída da RNA)
IAP-SG-Agent	Sem Goleiro (*SG*)	Variável	4
IAP-CG-Agent	Com Goleiro (*CG*)	Variável	4

Os agentes apresentados na Tabela 27 apresentam as seguintes variações em relação às representações de estados:

- *IAP-SG-Agent* e *IAP-CG-Agent* com **Escala de Cinza**: considera a representação por imagens brutas sem informação de cor (descrita na subseção 6.1.1). O módulo de tomada de decisão corresponde a uma RNC com mesma arquitetura da *Q-Network* original (apresentada na subseção 2.3.1).
- *IAP-SG-Agent* e *IAP-CG-Agent* com **RGB**: considera a representação por imagens brutas com informação de cor (descrita na subseção 6.1.2). O módulo de tomada de decisão corresponde a uma RNC com mesma arquitetura da *Q-Network* original.
- *IAP-SG-Agent* e *IAP-CG-Agent* com **Representação TDO 1**: considera a **Representação TDO 1** (descrita na subseção 5.2.3). O módulo de tomada de decisão é representado pela PMC apresentada na Tabela 28.
- *IAP-SG-Agent* e *IAP-CG-Agent* com **Representação TDO 2**: considera a **Representação TDO 2** (descrita na subseção 5.2.3). O módulo de tomada de decisão é representado pela PMC apresentada na Tabela 28.
- *IAP-CG-Agent* com **Representação TDO 1 Modificada**: considera a **Representação TDO 1 Modificada** (descrita na subseção 6.1.3). O módulo de tomada de decisão é representado pela PMC apresentada na Tabela 28.
- *IAP-CG-Agent* com **Representação TDO 2 Modificada**: considera a **Representação TDO 2 Modificada** (descrita na subseção 6.1.3). O módulo de tomada de decisão é representado pela PMC apresentada na Tabela 28.

6.3 Experimentos e Resultados

Os experimentos aqui têm como objetivo geral avaliar a qualidade do processo de aprendizado dos agentes - embasados no método IAP - levando em consideração as representações de estado baseadas em imagens cruas (representações ingênuas) e em TDO descritas na subseção 6.1. Nesse sentido, foi efetuado um experimento para cada cenário

Tabela 28 – Arquitetura da PMC utilizada pelas Variações com Representação baseada em TDO (similar àquela usada em (TRIVEDI, 2018b), alterando apenas a função de ativação da camada de saída)

Camada	**Número de Neurônios**	**Função de Ativação**
TC1	512	*ReLu*
TC2	512	*ReLu*
Saída (*TC3*)	4	*Softmax*

de faltas por meio dos agentes *IAP-SG-Agent* (Experimento 1) e *IAP-CG-Agent* (Experimento 2) e suas variações (descritas na seção 6.2). A métrica utilizada foi taxa de gols. Os experimentos foram executados em uma máquina com uma GPU Nvidia GeForce GTX-745 e 16 GB de RAM.

Por fim, destaca-se que parte dos resultados aqui obtidos foram apresentados no artigo *Deep Active Imitation Learning in FIFA Free-Kicks Player Platforms based on Raw Image and Object Detection State Representation* publicado na conferência internacional *31st International Conference on Tools with Artificial Intelligence* (FARIA; JULIA; TOMAZ, 2019a). A implementação efetuada foi disponibilizada à comunidade[1].

6.3.1 Experimento 1: Análise do Agente *IAP-SG-Agent* baseado em Distintas Representações de Estado

O objetivo aqui é avaliar a qualidade da tomada de decisão de *IAP-SG-Agent* e suas variações em termos de representação de estado. Tais variações foram treinadas a partir de demonstrações recuperadas de tomadas de decisão nas quais um agente supervisor humano alcançou uma taxa de gols aproximadamente igual a 95%. Em seguida, cada variação treinada foi submetida a 10 sessões avaliativas, cada uma composta por 100 execuções (cada execução termina assim que uma ação de chute é executada). Assim, ao todo cada variação efetua 1000 chutes.

A Tabela 29 apresenta, respectivamente, o número de sessões (N), a média total e o desvio padrão de *IAP-SG-Agent* e suas variações em termos da taxa de gols (em %). Conforme indicado, a variação baseada na **Representação TDO 2** obteve a maior média dentre o conjunto de agentes (92,10%). Além disso, nota-se que aquelas baseadas em imagem bruta (**Escala de Cinza** e **RGB**) tiveram um desempenho médio melhor do que a oriunda da **Representação TDO 1**. Para estimar a significância de tais resultados, foi efetuado o método *ANOVA de uma via* com um *nível de significância* ($\alpha = 0,05$). Neste contexto, a seguinte hipótese nula H_0 foi criada: as variações de *IAP-SG-Agent* possuem o mesmo nível de desempenho em termos da taxa de gols.

Conforme o resultado obtido pelo *ANOVA de uma via* indicado na Tabela 30 (F(3,36) = 6,294, *valor-p* = 0,002), foi possível concluir que tais variações não apresentam o mesmo

[1] https://github.com/matheusprandini/FifaFreeKicksImitationLearning

Tabela 29 – Resultados de *IAP-SG-Agent* e suas variações em termos de taxa de gols

Agente	N	Média	Desvio Padrão
IAP-SG-Agent com **Escala de Cinza**	10	90,50	2,173
IAP-SG-Agent com **RGB**	10	91,60	1,776
IAP-SG-Agent com **Representação TDO 1**	10	87,30	3,622
IAP-SG-Agent com **Representação TDO 2**	10	92,10	2,923

nível de desempenho em termos da taxa de gols - pois o *valor-p* = 0,002 é menor do que o *nível de significância* (α = 0,05). Assim, H_0 foi rejeitada e um *Teste de Tukey* foi realizado para detectar quais delas apresentaram taxas de gols discrepantes.

Tabela 30 – *ANOVA de uma via* aplicado aos resultados de *IAP-SG-Agent* e suas variações

Taxa de Gols	Soma dos Quadrados	gl	Quadrado Médio	F	valor-p
Entre Grupos	139,475	3	46,492	6,294	0,002
Dentro dos Grupos	265,900	36	7,386		
Total	405,375	39			

É possível visualizar na Tabela 31 que existe uma diferença estatisticamente significativa entre as variações **RGB** e **Representação TDO 1** (*significância* = 0,006) e entre os agentes **Representação TDO 1** e **Representação TDO 2** (*significância* = 0,002) com relação a taxa de gols. No entanto, não houve diferença significativa de desempenho entre os outros pares de variações. Tais resultados foram obtidos a partir da taxa média de gols relacionada a cada variação apresentada na Tabela 29, isto é: **Escala de Cinza** que não considera informações de cores (90,50 ± 2,2 %), **RGB** que leva em consideração informações de cores (91,60 ± 1,8 %), **Representação TDO 1** (87,30 ± 3,6 %) e, finalmente, **Representação TDO 2** (92,10 ± 2,923 %).

Tabela 31 – *Teste de Tukey* aplicado aos resultados de *IAP-SG-Agent* e suas variações. O símbolo '*' indica que a diferença média é significativa considerando um *nível de significância* de 0,05

(I) Grupo 1	**(J) Grupo 2**	**Diferença Média (I - J)**	**Significância**
Escala de Cinza	**RGB**	-1,100	,802
Escala de Cinza	**Representação TDO 1**	3,200	,057
Escala de Cinza	**Representação TDO 2**	-1,600	,559
RGB	**Representação TDO 1**	4,300*	,006
RGB	**Representação TDO 2**	-,500	,976
Representação TDO 1	**Representação TDO 2**	-4,800*	,002

O teste comparativo efetuado entre as representações de estado baseadas em TDO e imagens brutas operando no cenário de falta sem goleiro mostrou que as variações de *IAP-SG-Agent* que consideram imagens brutas foram muito competitivas com relação àquela oriunda da **Representação TDO 2** em termos de taxa de gols. Ressalta-se

que a variação baseada na **Representação TDO 1** obteve pior desempenho, enquanto que aquela implementada de acordo com a **Representação TDO 2** obteve a melhor taxa de gols (porém, sem significância estatística em comparação aos agentes baseados em imagens cruas). É importante notar que, diferentemente dos resultados obtidos pela imagem crua sem informação de cor no contexto de ARP (**Escala de Cinza 4 frames**), a representação **Escala de Cinza** utilizada no contexto do AI apresentou um ótimo desempenho. Isso sugere que apenas um único *frame* é suficiente para uma boa tomada de decisão neste cenário de faltas. A discussão sobre tais representações é detalhada no próximo experimento, após as conclusões obtidas nos testes avaliativos no cenário de faltas com goleiro.

6.3.2 Experimento 2: Análise do Agente *IAP-CG-Agent* baseado em Distintas Representações de Estado

Analogamente ao primeiro experimento, o objetivo aqui é avaliar as representações ingênuas e aquelas baseadas em TDO. Além disso, ele também tem o propósito de validar a eficácia do método IAP em um cenário estático e multiagente (cenário de faltas com goleiro). Dessa forma, este experimento foi decomposto em duas fases: 1) realização de uma análise comparativa entre as variações de *IAP-CG-Agent* (considerando as mesmas representações avaliadas no primeiro experimento); 2) avaliação das modificações efetuadas na TDO por meio da **Representação TDO 1 Modificada** e da **Representação TDO 2 Modificada**. Destaca-se que todas essas variações foram treinadas a partir de demonstrações recuperadas de tomadas de decisão nas quais um agente supervisor humano alcançou uma taxa de gols aproximadamente igual a 7,5%. Em seguida, cada variação treinada foi submetida a 10 sessões avaliativas, cada uma composta por 100 execuções.

Fase 1 - Avaliação Detalhada de *IAP-CG-Agent* e suas Variações

A Tabela 32 mostra, respectivamente, o número de sessões, a média total e o desvio padrão de *IAP-SG-Agent* e suas variações em termos da taxa de gols (em %). Assim como ocorreu no Experimento 1, é possível observar na tabela que as variações baseadas em imagem bruta obtiveram um desempenho médio melhor do que aquela baseada na **Representação TDO 1**. No entanto, o sucesso da **Representação TDO 2** obtido no Experimento 1 não foi replicado nesse cenário mais complexo. De fato, a variação que considera tal representação obteve a pior média de taxa de gols (1,7%) em relação às demais.

De modo a estimar a significância dos resultados, foi aplicado o método *ANOVA de uma via* com um *nível de significância* ($\alpha = 0,05$), conforme mostrado na sequência. Assim, foi criada a seguinte hipótese nula H_0: as variações consideradas de *IAP-*

Tabela 32 – Resultados de *IAP-CG-Agent* e suas variações em termos de taxa de gols

Agente	N	Média	Desvio Padrão
IAP-CG-Agent com **Escala de Cinza**	10	5,80	2,150
IAP-CG-Agent com **RGB**	10	7,00	1,944
IAP-CG-Agent com **Representação TDO 1**	10	4,50	1,841
IAP-CG-Agent com **Representação TDO 2**	10	1,70	1,337

CG-Agent nessa fase possuem o mesmo nível de desempenho em termos da taxa de gols.

Conforme o resultado obtido pelo *ANOVA de uma via* indicado na Tabela 33 (F(3,36) = 15,336, *valor-p* = 0,000), foi possível concluir que tais variações não apresentam o mesmo nível de desempenho em termos da taxa de gols - pois o *valor-p* = 0,000 é menor do que o *nível de significância* ($\alpha = 0{,}05$). Assim, H_0 foi rejeitada e um *Teste de Tukey* foi realizado para detectar quais agentes apresentaram taxas de gols discrepantes.

Tabela 33 – *ANOVA de uma via* aplicado aos resultados de *IAP-CG-Agent* e suas variações

Taxa de Gols	Soma dos Quadrados	gl	Quadrado Médio	F	valor-p
Entre Grupos	159,875	3	53,292	15,336	,000
Dentro dos Grupos	125,100	36	3,475		
Total	284,975	39			

É possível observar na Tabela 34 que existe diferença estatisticamente significativa, relativa à taxa de gols, entre os pares de variações **Escala de Cinza** e **Representação TDO 2** (*significância* = 0,000), **RGB** e **Representação TDO 1** (*significância* = 0,018), **RGB** e **Representação TDO 2** (*significância* = 0,000), **Representação TDO 1** e **Representação TDO 2** (*significância* = 0,010). No entanto, não houve diferenças entre os outros pares. Tais resultados foram obtidos a partir da taxa média de gols relacionada a cada variação apresentada na Tabela 32, isto é: **Escala de Cinza** sem informações de cores (5,8 ± 2,2 %), **RGB** com informações de cores (7,0 ± 1,9 %), **Representação TDO 1** (4,5 ± 1,8 %) e, finalmente, **Representação TDO 2** (1,7 ± 1,337 %).

Os resultados obtidos nesta fase atestam a eficácia da IAP como método de aprendizado para agentes jogadores que operam em cenários de cobrança de faltas com goleiro (multiagente). De fato, a variação **RGB** de *IAP-CG-Agent*, a qual percebe o ambiente por meio de imagens brutas que levam em consideração informações de cores, provou ser o mais eficiente, atingindo uma taxa de gols de 7% aproximadamente (muito próxima à taxa obtida pelo agente humano, 7,5%).

Tabela 34 – *Teste de Tukey* aplicado aos resultados de *IAP-CG-Agent* e suas variações. O símbolo '*' indica que a diferença média é significativa considerando um *nível de significância* de 0,05

(I) Tipo Variação	(J) Tipo Variação	Diferença Média (I - J)	Significância
Escala de Cinza	**RGB**	-1,300	,414
Escala de Cinza	**Representação TDO 1**	1,300	,414
Escala de Cinza	**Representação TDO 2**	-4,100*	,000
RGB	**Representação TDO 1**	2,600*	,018
RGB	**Representação TDO 2**	5,400*	,000
Representação TDO 1	**Representação TDO 2**	2,800*	,010

Com relação à análise comparativa entre as representações de estado baseadas em TDO e imagens brutas, as variações baseadas em imagem bruta (destacando-se a **RGB**), em geral, foram amplamente superiores em relação àquelas oriundas de TDO em termos de taxa de gols neste cenário.

Assim como observado no cenário de faltas sem goleiro, a **Representação TDO 1** teve um desempenho fraco em relação às variações de imagem bruta (principalmente, em relação a **RGB**, na qual a diferença foi estatisticamente significativa). Os autores acreditam que o baixo desempenho apresentado pela variação **Representação TDO 1** nos dois cenários de faltas deve-se ao seguinte fato: a forma usada para representar o cenário do jogo (correspondente ao *Mapa de características de alto nível*), ao invés de levar em conta apenas os objetos selecionados para serem detectados, pode também representar informações de todos os elementos irrelevantes restantes da imagem, os quais resultam em uma espécie de ruído nos dados de entrada processados pelo módulo de tomada de decisão. Tais ruídos podem comprometer o desempenho da capacidade de aprendizado dos agentes.

Com relação à **Representação TDO 2**, ela mostrou-se extremamente adequada ao cenário de faltas sem goleiro. Considerando o cenário multiagente, ela foi superada facilmente por todas as outras representações. Os autores acreditam que isso se deve ao fato de que os objetos utilizados pela TDO não são suficientes para a efetuação de uma boa tomada de decisão neste último cenário. Por exemplo, reconhecer o elemento gol produziu um excelente resultado no cenário sem goleiro, pois o agente apenas precisava efetuar uma ação de chute em direção ao gol para pontuar (justamente por não haver a presença de um goleiro). Já no cenário com goleiro, essa informação mostrou-se incompleta, pela dificuldade de se marcar gols. Assim, os resultados mostraram que é necessário mais informações para aprimorar a qualidade da representação e, consequentemente, a tomada de decisão de um agente. Por isso, os alvos ao redor do gol foram estudados como possíveis elementos com grande potencial para melhorar os resultados obtidos por meio do *IAP-CG-Agent*, o que é avaliado na próxima fase.

Fase 2 - Investigação das Modificações nas Representações Baseadas em TDO no *IAP-CG-Agent*

Esta fase é fruto das discussões realizadas sobre as fragilidades das variações de *IAP-CG-Agent* implementadas de acordo com as representações baseadas em TDO. Dessa forma, ela investiga o impacto que os aprimoramentos realizados na representação de estado por meio da adição dos alvos ao redor gol exerce no desempenho dos agentes. Para tanto, foram efetuados dois casos de teste: o primeiro (I) efetua uma análise comparativa entre todas as representações baseadas em TDO no cenário de faltas com goleiro abordando as modificações efetuadas (adição dos alvos ao redor do gol e as remoções dos elementos gol e jogador que cobra faltas); o segundo (II) compara as representações de TDO modificadas com aquelas embasadas em imagens brutas. Em ambos os casos, a significância estatística relativa ao desempenho dos agentes foi calculada por meio do método *ANOVA de uma via* ao longo de 10 sessões avaliativas, cada uma composta por 100 execuções.

Com relação ao primeiro caso de teste, a Tabela 35 mostra, respectivamente, o número de sessões, a média total e o desvio padrão das variações modificadas de TDO de *IAP-CG-Agent* em relação a taxa média de gols (em %). Observa-se que a **Representação TDO 2 Modificada** obteve uma taxa de gols maior do que a **Representação TDO 1 Modificada**, bem como em relação àquelas baseadas na **Representação TDO 1** e **Representação TDO 2** (extraídas da Tabela 32).

Tabela 35 – Resultados de *IAP-CG-Agent* e suas variações baseadas na TDO modificada em termos de taxa de gols

Agente	N	Média	Desvio Padrão
IAP-CG-Agent com **Representação TDO 1**	10	4,50	1,841
IAP-CG-Agent com **Representação TDO 2**	10	1,70	1,337
IAP-CG-Agent com **Representação TDO 1 Modificada**	10	4,60	1,578
IAP-CG-Agent com **Representação TDO 2 Modificada**	10	5,20	1,135

De modo a estimar a significância de tais resultados, o método *ANOVA de uma via* com um *nível de significância* ($\alpha = 0,05$) foi conduzido. Assim, foi criada a seguinte hipótese nula H_0: as variações **Representação TDO 1**, **Representação TDO 2**, **Representação TDO 1 Modificada** e **Representação TDO 2 Modificada** de *IAP-CG-Agent* possuem o mesmo nível de desempenho.

Conforme o resultado obtido pelo *ANOVA de uma via* indicado na Tabela 36 ($F(3,36) = 10,928$, *valor-p* = 0,000), foi possível concluir que tais variações não apresentam o mesmo nível de desempenho em termos da taxa de gols - pois o *valor-p* = 0,000 é menor do que o *nível de significância* ($\alpha = 0,05$). Assim, H_0 foi rejeitada e um *Teste de Tukey* foi realizado para detectar quais delas apresentaram taxas de gols discrepantes.

Tabela 36 – *ANOVA de uma via* aplicado aos resultados de *IAP-CG-Agent* e das variações baseadas em TDO

Taxa de Gols	Soma dos Quadrados	gl	Quadrado Médio	F	valor-p
Entre Grupos	73,400	3	24,467	10,928	,000
Dentro dos Grupos	80,600	36	2,239		
Total	154,000	39			

É possível observar na Tabela 37 que existe diferença estatisticamente significativa, relativa à taxa de gols, entre os pares de variações **Representação TDO 1** e **Representação TDO 2** (*significância* = 0,001), **Representação TDO 2** e **Representação TDO 1 Modificada** (*significância* = 0,001), **Representação TDO 2** e **Representação TDO 2 Modificada** (*significância* = 0,000). No entanto, não houve diferenças entre os outros pares de variações. Tais resultados foram obtidos a partir da taxa média de gols relacionada a cada variação apresentada nas Tabelas 29 e 35, isto é: **Representação TDO 1** (4,5 ± 2,8 %), **Representação TDO 2** (1,7 ± 1,3 %), **Representação TDO 1 Modificada** (4,6 ± 1,6 %) e, finalmente, **Representação TDO 2 Modificada** (5,2 ± 1,1 %).

Tabela 37 – *Teste de Tukey* aplicado aos resultados de *IAP-CG-Agent* e das variações baseadas em TDO. O símbolo '*' indica que a diferença média é significativa considerando um *nível de significância* de 0,05

(I) Tipo Variação	**(J) Tipo Variação**	**Diferença Média (I - J)**	**Significância**
Representação TDO 1	**Representação TDO 2**	2,800*	,001
Representação TDO 1	**Representação TDO 1 Modificada**	-,100	,999
Representação TDO 1	**Representação TDO 2 Modificada**	-,700	,724
Representação TDO 2	**Representação TDO 1 Modificada**	-2,900*	,001
Representação TDO 2	**Representação TDO 2 Modificada**	-3,500*	,000
Representação TDO 1 Modificada	**Representação TDO 2 Modificada**	-,600	,807

Com relação ao segundo caso de teste (comparação entre as representações de TDO modificadas e aquelas baseadas em imagens brutas), a Tabela 38 compacta a média total e o desvio padrão em relação a taxa média de gols (em %) já mencionadas em experimentos passados das variações aqui consideradas. Observa-se a superioridade da taxa obtida pelo *IAP-CG-Agent* com **RGB** (7%) sobre as outras variações.

Tabela 38 – Resultados de *IAP-CG-Agent* e suas variações baseadas em imagens e nas modificações da TDO em termos de taxa de gols

Agente	N	Média	Desvio Padrão
IAP-CG-Agent com **Escala de Cinza**	10	5,80	2,150
IAP-CG-Agent com **RGB**	10	7,00	1,944
IAP-CG-Agent com **Representação TDO 1 Modificada**	10	4,60	1,578
IAP-CG-Agent com **Representação TDO 2 Modificada**	10	5,20	1,135

De modo a estimar a significância de tais resultados, o método *ANOVA de uma via* com um *nível de significância* ($\alpha = 0,05$) foi conduzido. Assim, foi criada a seguinte

hipótese nula H_0: as variações **Escala de Cinza**, **RGB**, **Representação TDO 1 Modificada** e **Representação TDO 2 Modificada** de *IAP-CG-Agent* possuem o mesmo nível de desempenho.

Conforme o resultado obtido pelo *ANOVA de uma via* indicado na Tabela 39 ($F(3,36) = 3{,}656$, *valor-p* = 0,021), foi possível concluir que tais variações não apresentam o mesmo nível de desempenho em termos da taxa de gols - pois o *valor-p* = 0,021 é menor do que o *nível de significância* ($\alpha = 0{,}05$). Assim, H_0 foi rejeitada e um *Teste de Tukey* foi realizado para detectar quais delas apresentaram taxas de gols discrepantes.

Tabela 39 – *ANOVA de uma via* aplicado aos resultados de *IAP-CG-Agent* e suas variações baseadas em imagens brutas e nas modificações da TDO

Taxa de Gols	Soma dos Quadrados	gl	Quadrado Médio	F	valor-p
Entre Grupos	34,275	3	11,425	3,656	,021
Dentro dos Grupos	112,500	36	3,125		
Total	146,775	39			

É possível observar na Tabela 40 que existe diferença estatisticamente significativa, relativa à taxa de gols, apenas entre as variações **RGB** e **Representação TDO 1 Modificada** (*significância* = 0,016). No entanto, não houve diferenças entre os demais pares de variações.

Tabela 40 – *Teste de Tukey* aplicado aos resultados de *IAP-CG-Agent* e suas variações baseadas em imagens e nas modificações da TDO. O símbolo '*' indica que a diferença média é significativa considerando um *nível de significância* de 0,05

(I) Tipo Variação	**(J) Tipo Variação**	**Diferença Média (I - J)**	**Significância**
Escala de Cinza	**RGB**	-1,300	,367
Escala de Cinza	**Representação TDO 1 Modificada**	1,200	,438
Escala de Cinza	**Representação TDO 2 Modificada**	,600	,872
RGB	**Representação TDO 1 Modificada**	2,500*	,016
RGB	**Representação TDO 2 Modificada**	1,900	,095
Representação TDO 1 Modificada	**Representação TDO 2 Modificada**	-,600	,872

Assim sendo, os resultados obtidos no primeiro caso de teste permitem verificar o ganho de desempenho obtido pela nova representação de TDO (modificada) com a adição dos alvos ao redor do gol como elementos desejados para detecção e com a remoção dos elementos gol e jogador que cobra falta. Destaca-se que a remoção do gol se deu por conta da adição dos novos elementos. Os alvos ao redor do gol permitem identificar as localizações em que o agente terá maior chances de marcar um gol (pois o goleiro terá maior dificuldade de bloquear o chute), contudo, as barreiras neste cenário são um grande dificultador (conforme descrito na seção 2.9). Com relação à remoção do jogador que cobra faltas, ela foi uma opção dos autores visando um aprendizado com caráter mais geral no sentido de que, caso o agente fosse

treinado apenas com demonstrações provindas de um jogador cobrador de faltas destro, por exemplo, a representação final não seria válida para ser utilizada com jogadores canhotos e vice-versa. Caso o agente fosse treinado com demonstrações provindas tanto de jogadores que chutam com a perna direita quanto com a perna esquerda, pode ser que o processo de aprendizado seja dificultado ou, então, que mais exemplos sejam necessários para abstrair tais informações. Assim, optou-se por retirar esta informação, tornando a representação final independente dela.

Por fim, os resultados obtidos no segundo caso de teste permitem observar que ainda há uma diferença significativa de desempenho entre a representação baseada em imagens cruas com informação de cor (**RGB**) e aquela embasada na **Representação TDO 1 Modificada**, mesmo com a adição dos alvos ao redor do gol. Isso corrobora os motivos já discutidos anteriormente sobre a fragilidade da forma de representação proveniente do *Mapa de Características de Alto Nível*. No entanto, verifica-se que a variação **Representação TDO 2 Modificada** obteve um desempenho mais próximo aos dos agentes implementados de acordo com imagens brutas (pelo fato de não haver diferença significativa entre eles). Assim, por todos os pontos mencionados nessa discussão, é possível afirmar que o aprimoramento efetuado sobre as representações de TDO surtiu um efeito positivo no desempenho de um agente jogador, apesar de haver margens de estudo para incrementar a sua qualidade. Por exemplo, considerar apenas o alvo mais promissor em uma determinada situação de falta, isto é, os autores sugerem remover os dois alvos mais próximos ao goleiro, além de desconsiderar o alvo rasteiro que fica atrás da barreira (nos casos em que ela é fixa ao chão) ou desconsiderar o alvo superior (nos casos em que a barreira somente fornece uma abertura para um chute rasteiro).

Capítulo 7

Aprendizado por Imitação ID e IAP em Agentes Atuantes no Modo de Confronto

Este capítulo apresenta a contribuição referente à investigação da abordagem de AI - especificamente, os métodos ID e IAP - no modo de confronto, descrito na seção 2.9. Os referidos métodos são avaliados comparativamente de acordo com a representação de estado baseada em imagem crua sem informação de cor. Tal representação é explanada na seção 7.1 e as arquiteturas dos agentes são detalhadas na seção 7.2. Por fim, os experimentos e a análise dos resultados são apresentados na seção 7.3.

A presente investigação estende a abordagem do capítulo anterior em dois aspectos: 1) por utilizar os métodos de imitação (ID e IAP) como recurso essencial para lidar com a propriedade de ambiente desconhecido (detalhado na seção 2.9) inerente ao modo de confronto. Conforme apresentado ao longo desse capítulo, tal "imitação" atenua o impacto causado pela referida propriedade ao contar com um humano supervisor que possui conhecimento sobre a dinâmica do ambiente e que, consequentemente, efetua boas tomadas de decisão no mesmo. Dessa forma, o objetivo aqui é construir agentes com a capacidade de replicar o comportamento humano; 2) por lidar com um ambiente que possui as seguintes propriedades: dinâmico e multiagente.

Além disso, destaca-se que esta investigação é inspirada na proposta apresentada em (HUSSEIN et al., 2018), a qual validou o método IAP em ambientes de navegação estáticos e de agente único a partir de uma análise comparativa com relação ao método ID. Nesse sentido, esta pesquisa expande os trabalhos de Hussein ao efetuar uma comparação de desempenho entre a IAP e a ID em um estudo de caso mais complexo (cenário modo de confronto).

7.1 Especificação da Representação de Estado

A única representação utilizada neste capítulo é baseada em imagens cruas sem informação de cor, idêntica à utilizada em (HUSSEIN et al., 2018) e descrita na subseção 6.1.1 (referida como **Escala de Cinza**), conforme ilustrada na Figura 38. Portanto, cada estado é representado por uma imagem de tamanho 120x90 (isto é, o estado é uma matriz 120x90x1).

Figura 38 – Exemplo da representação **Escala de Cinza** no modo de confronto

7.2 Arquitetura dos Agentes Jogadores

Esta seção apresenta a implementação dos agentes baseados em AI para o Modo de confronto. Em tal cenário, a cada vez que o agente deve selecionar uma ação na situação corrente de jogo, o processo de tomada de decisão deve definir um par de ações cujo primeiro elemento aponta a direção do movimento (*Mover para a esquerda*, *Mover para baixo*, *Mover para a direita* ou *Mover para cima*) e o segundo controla as interações envolvendo o agente, a bola e os adversários (*Defender*, *Passar*, *Chutar* ou *Nenhuma Ação*). É importante ressaltar que as forças das ações *Passar* e *Chutar* são consideradas constantes, sendo empiricamente definidas pelos autores (conforme especificado no repositório disponibilizado à comunidade[1]). A força de *Passar* corresponde a apertar e segurar a tecla que efetua tal ação por 0,1 segundos. Da mesma forma, a força de *Chutar* corresponde a apertar e segurar a tecla que efetua tal ação por 0,2 segundos.

Dessa forma, os agentes aqui propostos consistem de duas RNCs - nomeadas como *RNC de Movimentação* e *RNC de Controle* - treinadas para poder apontar as direções de movimento apropriadas (primeiro elemento do par correspondente à tomada de decisão) e as interações apropriadas do jogo (segundo elemento desse par), respectivamente, conforme mostrado na Figura 39. Então, neste cenário, as ações legais são divididas em dois grupos: *ações de movimento* e *ações de controle*, dos quais são recuperados, respectivamente, o primeiro e o segundo elemento do par de ações relacionadas à tomada de decisão. Por exemplo, uma tomada de decisão em que a *RNC de Controle* efetua a ação

[1] <https://github.com/matheusprandini/IL-Fifa2X2>

de chute e a *RNC de Movimentação* efetua a direção de tal chute para a esquerda, o par formado corresponde a (*Mover para a Esquerda*, *Chutar*).

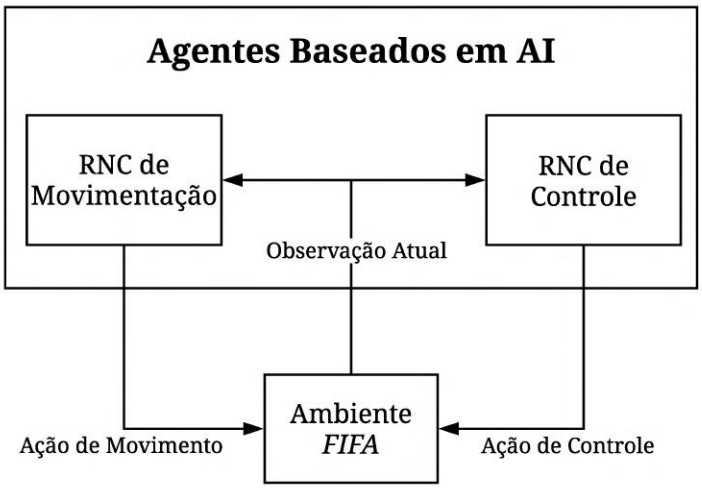


Figura 39 – Arquitetura Geral dos Agentes Jogadores baseados em AI no Modo de Confronto.

As duas RNCs apresentam a mesma arquitetura proposta em (HUSSEIN et al., 2018) (descrita na subseção 2.3.2). Cada RNC recebe como entrada um *frame* de tamanho 120x90 representando o estado atual do jogo e retorna a probabilidade de execução de cada ação possível (por meio da função de ativação *Softmax*).

A Tabela 41 apresenta os dois agentes baseados na representação **Escala de Cinza** propostos para o modo de confronto, os quais são descritos na sequência.

Tabela 41 – Agentes Implementados Baseados em AI

Agente	Representação de Estado	Ações Possíveis (Saída da ***RNC de Movimentação***)	Ações Possíveis (Saída da ***RNC de Controle***)
ID-MC-Agent	**Escala de Cinza**	4	4
IAP-MC-Agent	**Escala de Cinza**	4	4

7.2.1 Descrição do *ID-MC-Agent*

Este agente é baseado no método *ID*, isto é, ele aprende conforme a estratégia supervisionada da segunda etapa do método *IAP* a partir de demonstrações coletadas de tomadas de decisão de um agente humano (representado por um dos autores, o qual possui uma boa expertise no modo de jogo aqui abordado). Considerando o Modo de Confronto, cada demonstração é um par constituído pelos seguintes elementos:

- **Observação**: esse elemento é um *frame* que representa um determinado cenário de jogo. A representação utilizada é descrita na seção 7.1.

- ❑ **Ação**: esse elemento representa a ação selecionada por um determinado agente humano na situação do jogo apresentada na observação (primeiro elemento do par). É interessante ressaltar que as RNCs de Movimento e de Controle são treinadas a partir de demonstrações cujo elemento *Ação* pertence às *ações de movimento* ou às *ações de controle*, respectivamente.

Analogamente ao realizado no cenário de faltas, as demonstrações deste cenário foram recuperadas a partir de jogos reais nos quais o agente humano enfrentou a máquina do *FIFA*. Mais especificamente, os dois conjuntos de pares de demonstração (correspondentes às *ações de movimento* e às *ações de controle*) utilizados para treinar as RNCs foram capturados da entrada do teclado juntamente com as imagens do jogo por meio da interface proposta no capítulo 4.

Neste sentido, destaca-se que a "imitação" proveniente do humano supervisor é inerente aos conjuntos de dados fornecidos. Neles, as observações não contém as informações relativas à dinâmica de jogo (referente ao constante movimento dos elementos no cenário), pois são representadas por apenas um *frame*. Contudo, as tomadas de decisão em cada uma delas (referentes às *ações de movimento* e às *ações de controle*) foram efetuadas por um agente humano o qual leva em consideração a sequência de movimentos do ambiente (isto é, ele compreende o funcionamento do mesmo). Dessa forma, apesar do ambiente ser desconhecido para o *ID-MC-Agent*, espera-se que ele atue no mesmo de modo razoável pelo fato de ter sido treinado com exemplos provenientes de um agente para o qual o ambiente é conhecido (humano). Destaca-se que o supervisor humano é representado por um dos autores da presente pesquisa de Mestrado, o qual possui um nível de jogo entre intermediário e avançado no modo de confronto.

O *ID-MC-Agent* conta com demonstrações extraídas de 20 jogos do supervisor humano, produzindo um conjunto de dados composto por 2000 demonstrações correspondentes às *ações de movimento* (500 exemplos de cada ação possível) e outro conjunto de dados composto por 500 demonstrações relativas às *ações de controle* (125 exemplos de cada uma delas). Por fim, tais conjuntos foram balanceados e todos os dados foram utilizados para o treinamento de suas respectivas RNCs. A Tabela 42 apresenta os hiperparâmetros utilizados no treinamento das RNCs do *ID-MC-Agent*.

Tabela 42 – Hiperparâmetros Utilizados no Treinamento das RNCs do *ID-MC-Agent*

Tamanho do Mini-Lote	64
Número de Épocas	10
Otimizador	*Adam* (KINGMA; BA, 2015)
Taxa de Aprendizagem	0,001
Função de Perda	*Categorical Cross Entropy* (MANNOR; PELEG; RUBINSTEIN, 2005)

7.2.2 Descrição do *IAP-MC-Agent*

Este agente corresponde a uma versão aprimorada do *ID-MC-Agent*. Esse aprimoramento é obtido por meio da terceira etapa introduzida pelo método IAP (descrita na subseção 2.3.2), ou seja, o aprendizado ativo é usado para adicionar demonstrações relacionadas a situações não vistas pelo *ID-MC-Agent* (as quais ele não foi treinado), a fim de aprimorar a política aprendida pelo referido agente. Como ele é formado por duas *RNCs*, o aprendizado ativo é aplicado de modo independente em cada uma delas. A Figura 40 ilustra o processo de coleta dos exemplos ativos aplicado sobre a *RNC de Movimentação*, no qual o agente supervisor humano atua apenas sobre as *ações de movimento* (pelo *feedback* corretivo), enquanto a *RNC de Controle* não sofre intervenção humana. Do mesmo modo é efetuado para a *RNC de Controle* (neste caso, apenas ela é supervisionada pelo agente humano), conforme ilustrado na Figura 41. Esse processo de coleta é conduzido até o ponto em que o número de demonstrações nos dois conjuntos de dados (correspondentes às *ações de movimento* e às *ações de controle*) aumenta em 5% devido à inclusão de exemplos ativos. Destaca-se que o supervisor humano monitora constantemente a interação do agente automático com o ambiente.

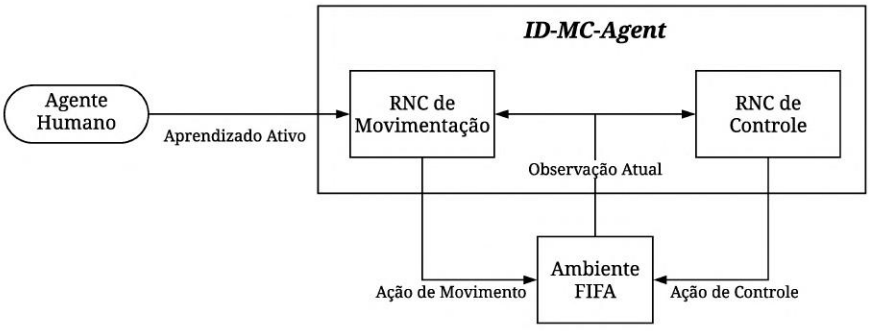


Figura 40 – Coleta dos Exemplos Ativos Relativa à *RNC de Movimentação*.

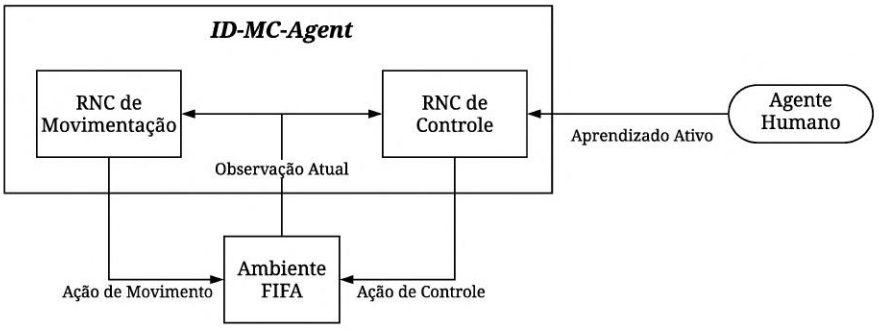


Figura 41 – Coleta dos Exemplos Ativos Relativa à *RNC de Controle*.

Nesse sentido, é importante observar que esta fase de interação entre o humano e o agente automático (*ID-MC-Agent*) proporcionada pelo aprendizado ativo é fundamental

para auxiliar este agente quanto ao seu desconhecimento sobre o ambiente. Nas situações de jogo em que ele não tem certeza de qual ação efetuar - normalmente representadas por observações que o *ID-MC-Agent* não foi treinado para reconhecer e, portanto, ele não consegue decidir por conta da questão da dinâmica dos elementos do ambiente - o agente humano efetua boas ações em seu lugar, uma vez que este monitora o comportamento do *ID-MC-Agent* a todo momento e, consequentemente, tem consciência de tudo o que ocorre no ambiente (relativo ao deslocamento e movimentação dos jogadores e da bola). Em seguida, tais intervenções humanas são utilizadas para aprimorar a política do *ID-MC-Agent*.

Assim, 100 exemplos ativos relacionados às *ações de movimento* e 25 amostras ativas correspondentes às *ações de controle* são adicionadas aos seus respectivos conjuntos de dados. Por fim, a política de *ID-MC-Agent* é refinada considerando também estes conjuntos de dados, o que gera a nova versão denominada aqui como *IAP-MC-Agent*. Ressalta-se que esta nova versão torna-se mais capaz de lidar com as mesmas situações as quais *ID-MC-Agent* mostrou-se ineficiente, conforme validado nos experimentos apresentados na próxima seção. A Tabela 43 apresenta os hiperparâmetros utilizados no treinamento das RNCs do *IAP-MC-Agent*.

Tabela 43 – Hiperparâmetros Utilizados no Treinamento das RNCs do *IAP-MC-Agent*

Tamanho do Mini-Lote	64
Número de Épocas	10
Otimizador	*Adam*
Taxa de Aprendizagem	0,001
Função de Perda	*Categorical Cross Entropy*
Percentual Dados Ativos	5%
β **da *RNC de Movimentação***	1,5
β **da *RNC de Controle***	1,9

7.3 Experimentos e Resultados

Os experimentos aqui realizados possuem como objetivo estudar o impacto exercido pela etapa de aprendizado ativo presente no método IAP (e não presente na ID) no desempenho de um agente inteligente em um ambiente dinâmico e multiagente. Assim sendo, uma avaliação comparativa foi realizada essencialmente de acordo com os parâmetros pontuação de jogo (conforme explicado na subseção 2.9.3) e taxa de coincidência entre as escolhas de jogadas realizadas por cada agente e as escolhas das jogadas que o agente humano supervisor que forneceu os exemplos ativos executou nas situações representadas por tais exemplos (por meio do *ID-MC-Agent* e do *IAP-MC-Agent*). Os experimentos foram executados em uma arquitetura composta por uma máquina com uma GPU Nvidia GeForce 940MX e 8 GB de RAM.

Destaca-se que os resultados aqui obtidos foram apresentados no artigo *Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents* publicado na conferência internacional *18th IEEE International Conference on Machine Learning and Applications* (FARIA; JULIA; TOMAZ, 2019c). Por fim, os detalhes de implementação de ambos os agentes e todo o processo de treinamento deles foram devidamente documentados e disponibilizados à comunidade[2].

7.3.1 Experimento 1: Avaliação dos Agentes em Termos da pontuação de jogo

Um torneio composto por 20 partidas foi executado entre os agentes *ID-MC-Agent* e *IAP-MC-Agent* contra a máquina da *FIFA* (controlado automaticamente). Enfatiza-se que o motor de jogo usado contra ambos agentes possuía o mesmo nível de jogo (ou melhor, mesma dificuldade) de maneira a não beneficiar nenhum agente em particular.

A Tabela 44 mostra, respectivamente, a média total (Média) e o desvio padrão (Desvio Padrão) de ambos os agentes no torneio efetuado em termos da pontuação de jogo. Os resultados indicam um melhor desempenho do agente produzido de acordo com o método IAP sobre o agente baseado na técnica ID com uma pontuação média mais alta (1625 pontos). Um *Teste T* com um *nível de significância* ($\alpha = 0,05$) foi executado sobre tais resultados para validar a comparação de desempenho entre os referidos agentes. Neste sentido, a seguinte H_0 foi criada: *ID-MC-Agent* e *IAP-MC-Agent* possuem o mesmo nível de desempenho sobre a pontuação de jogo.

Tabela 44 – Resultados de *ID-MC-Agent* e de *IAP-MC-Agent* no torneio em termos da pontuação de jogo

	Agente	Média	Desvio Padrão
Pontuação	*ID-MC-Agent*	905,00	858,686
	IAP-MC-Agent	1625,00	1086,702

Como indica a Tabela 45, o *valor-t* negativo (t(38) = -2,325) indica que a média da pontuação relativa ao *IAP-MC-Agent* é significativamente maior do que a média obtida pelo *ID-MC-Agent* com *graus de liberdade* igual a 38. Como *valor-p* $= 0,026$ é menor do que o *nível de significância* ($\alpha = 0,05$), a hipótese nula pode ser rejeitada.

Tabela 45 – *Teste T* aplicados aos resultados de *ID-MC-Agent* e de *IAP-MC-Agent*

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
ID-MC-Agent x *IAP-MC-Agent*	-2.325	38	0,026

Esses resultados sugerem que os exemplos ativos obtidos por meio do aprendizado ativo no método IAP têm um efeito significativo no desempenho (em termos de pontuação de

[2] https://github.com/matheusprandini/IL-Fifa2X2

jogo) com um intervalo de confiança de 95% com relação a diferença média. Portanto, conclui-se que o *IAP-MC-Agent* superou o agente *ID-MC-Agent* no modo de confronto.

7.3.2 Experimento 2: Avaliação da Escolha de Movimentos nos Exemplos Ativos em Relação ao Supervisor Humano

Essa avaliação tem como objetivo avaliar o nível de qualidade do processo de tomada de decisão de *ID-MC-Agent* e de *IAP-MC-Agent*. Ela é baseada na taxa de coincidência entre a escolha dos movimentos realizados por cada agente nos exemplos ativos (conjunto *Dados Ativos* descritos na subseção 2.3.2) e a escolha dos movimentos realizados pelo agente humano que forneceu esses exemplos. Eles consistem de 100 demonstrações correspondentes às *ações de movimento* e 25 demonstrações relacionadas às *ações de controle*, totalizando 125 demonstrações rotuladas pelo agente humano durante a fase de aprendizado ativo do método IAP (conforme descrito na subseção 7.2.2).

A Tabela 46 apresenta os resultados comparativos de *ID-MC-Agent* e de *IAP-MC-Agent*. De fato, o último apresentou uma taxa de coincidência de movimento em comparação com o agente humano igual a 36,8% (46 movimentos, sendo 37 correspondentes às *ações de movimento* e apenas 9 correspondentes às *ações de controle*), enquanto a mesma taxa relacionada ao primeiro agente foi igual a 20,8% (26 movimentos, sendo 19 correspondentes às *ações de movimento* e apenas 7 correspondentes às *ações de controle*) nos mesmos dados. Isso já era esperado pelo fato de *ID-MC-Agent* não ter sido treinado com os exemplos que compõem os dados ativos, enquanto que o agente *IAP-MC-Agent* foi.

Tabela 46 – Taxas de Coincidência de Movimentos com o Supervisor Humano

	ID-MC-Agent	*IAP-MC-Agent*
Taxa de coincidência de movimentos	20,8%	36,8%

Tal superioridade também pode ser demonstrada por meio do método estatístico *Wilcoxon*. Neste contexto, a seguinte hipótese nula H_0 foi criada: o *ID-MC-Agent* mantém o mesmo nível de desempenho (em termos de taxa de coincidência de movimentos com o supervisor humano) que o *IAP-MC-Agent*. R^+ representa a soma das classificações em que *IAP-MC-Agent* superou *ID-MC-Agent* - situações em que o *IAP-MC-Agent* selecionou a mesma ação rotulada pelo agente humano, enquanto que o *ID-MC-Agent* selecionou uma diferente (38 exemplos) - enquanto R^- é a soma das classificações em que *IAP-MC-Agent* foi superado por *ID-MC-Agent* (18 exemplos).

Conforme indicado na Tabela 47, o *IAP-MC-Agent* apresenta uma melhoria significativa em relação ao *ID-MC-Agent* com um alto nível de significância $\alpha = 0,01$ ou 99%, uma vez que o *valor-p* $= 0,008$ é menor do que α. Assim, a hipótese nula pode ser rejeitada, o que confirma que o agente baseado em IAP superou o agente implementado

de acordo com ID. Esses resultados sugerem que o primeiro possui uma maior capacidade de imitar ou reproduzir o comportamento do supervisor humano em situações inesperadas ou desconhecidas por conta da etapa de aprendizado ativo.

Tabela 47 – Teste de *Wilcoxon* Aplicado aos Resultados Relativos às Taxas de Coincidências Obtidas por *ID-MC-Agent* e *IAP-MC-Agent*

Experimento	R^+	R^-	*valor-p*
ID-MC-Agent x *IAP-MC-Agent*	1083	513	0,008

7.3.3 Discussão dos Resultados

Os resultados obtidos atestam a eficácia da IAP como método de aprendizado para agentes jogadores de *FIFA* que operam em cenários dinâmicos e multiagente. Com relação ao Experimento 1 (desempenho da IAP em termos de pontuação de jogo no cenário dinâmico explorado neste trabalho), o *IAP-MC-Agent* alcançou uma pontuação média mais alta do que o *ID-MC-Agent*) devido ao refinamento da política por meio do aprendizado ativo usado pelo primeiro. Este processo de refinamento foi o principal responsável por melhorar o desempenho do agente.

Com relação ao Experimento 2 (escolha de movimentos nos dados ativos em relação ao supervisor humano), de forma coerente com o parâmetro avaliativo pontuação de jogo, o *IAP-MC-Agent* superou novamente o agente *ID-MC-Agent*, o que confirma a superioridade do primeiro em relação ao segundo em um cenário complexo. Essa diferença de desempenho se deve principalmente ao aprendizado ativo usado para coletar demonstrações relevantes, aumentando o conjunto de dados usado para treinar novamente o agente *ID-MC-Agent*, gerando o agente *IAP-MC-Agent*. Apesar dessa diferença ter sido significativa, a taxa de coincidência obtida pelo agente baseado na IAP foi relativamente baixa considerando o fato de ter sido treinado com todos os exemplos ativos. Isto pode ter sido causado pela pequena taxa de aprendizado ativo (5%) e pela ambiguidade inerente à representação de dados utilizada, no sentido de que demonstrações que consistem de observações muito parecidas podem ter diferentes ações associadas a cada uma. Isso é justificado pelo fato da dinâmica de movimentação do ambiente não estar implícito nelas, sendo apenas conhecida pelo agente humano que fornece tais demonstrações. Nesse sentido, é necessário enfatizar que, apesar da taxa de coincidência do *IAP-MC-Agent* ter sido baixa, o seu desempenho melhorou consideravelmente pelo fato de processar vários *frames* por segundo (isto é, efetua várias escolhas de ações nesse período de tempo). Dessa forma, mesmo que não tome a mesma decisão do supervisor em uma determinada situação, ele adquiriu, de certo modo, a capacidade de "consertar" o seu comportamento nas situações subsequentes e de generalizar boas ações que levam ao seu sucesso à longo prazo (mesmo não considerando a dinâmica do ambiente em sua tomada de decisão), uma vez que o cenário envolve a propriedade sequencial (isto é, uma longa sequência de ações

para se chegar ao objetivo). Tais fatos mostram que esse parâmetro não é o ideal para avaliar o desempenho dos agentes neste caso.

Esses resultados já eram esperados, já que o IAP é uma extensão do ID acrescentando uma etapa de refinamento, embora tal etapa tenha permitido um ganho considerável de desempenho para o primeiro. Isso corrobora os resultados obtidos em (HUSSEIN et al., 2018) (o qual validou o método IAP em cenários estáticos), o que mostra que o referido método pode ser usado para resolver problemas complexos. É importante notar que esses resultados estatisticamente significativos foram obtidos com uma baixa taxa de aprendizado ativo (representado pelo hiperparâmetro *Percentual Dados Ativos* igual a 5%) e uma pequena quantidade de dados em comparação aos sistemas baseados em AP, os quais geralmente são treinados com milhões de exemplos.

Em termos práticos, é interessante ressaltar que o *IAP-MC-Agent* apresentou um comportamento razoável nos lances de ataque - nos quais ele está com o controle da bola e tenta marcar um gol - alcançando uma grande semelhança em relação ao agente humano em diversas execuções. Contudo, ele apresentou um péssimo comportamento ao defender os ataques do adversário (a ação *Defender* não foi corretamente aprendida, logo, isso comprometeu os resultados). Além disso, houve uma relativa dificuldade em se definir o valor do hiperparâmetro β do método IAP, pois não há qualquer tipo de referência sobre essa definição no trabalho que o propôs (HUSSEIN et al., 2018). Neste caso, tal valor foi definido empiricamente, observando o comportamento do *ID-MC-Agent* no ambiente, todavia, caso o referido método fosse aplicado diretamente em um problema real (por exemplo, carros autônomos), isto seria uma grande limitação devido aos riscos de se ter consequências negativas (por exemplo, acidentes no caso de carros autônomos).

Baseado em toda a discussão aqui efetuada, o próximo capítulo efetua um estudo no mesmo cenário considerando uma quantidade de dados maior e uma taxa de aprendizado ativo mais alta. Ademais, visando trabalhar melhor com a representação de estado utilizada, o presente trabalho investiga técnicas evolutivas para a definição de módulos de tomada de decisão adequados para se lidar com determinados conjuntos de dados.

Capítulo 8

Aprimoramento das Arquiteturas Tomadoras de Decisão Baseadas em RNC por meio de Técnicas Evolutivas

Nos capítulos anteriores, as RNAs dos agentes construídos mantiveram as mesmas arquiteturas definidas manualmente nos trabalhos correlatos nos quais a presente pesquisa se inspira. Com relação às representações baseadas em imagens brutas (com e sem informação de cor) no contexto de AI, as RNCs utilizadas como módulo de tomada decisão derivaram de (HUSSEIN et al., 2018). No entanto, seria interessante automatizar o processo de construção de uma RNC baseado nas informações provindas de um conjunto de dados a fim de torná-la mais apropriada e mais eficiente com relação ao problema ou tarefa abordada representada em tal conjunto.

Assim, este capítulo apresenta a contribuição referente à investigação de técnicas evolutivas para a automação do processo de definição das arquiteturas das RNCs utilizadas no cenário modo de confronto (mais especificamente, AG). Para tanto, inicialmente, são implementados agentes com arquiteturas construídas manualmente, conforme a abordagem de RNC tratada em (HUSSEIN et al., 2018) e descrita na subseção 2.3.2. Posteriormente, são construídos agentes com arquiteturas definidas automaticamente por meio de AG. Tal abordagem proposta será referenciada aqui como Algoritmo Genético para obtenção de Redes Neurais Convolucionais Mínimas (AG-RNC-M). A validação dessa abordagem é feita a partir de agentes baseados em aprendizagem ID e IAP no modo de confronto, os quais são implementados de acordo com representações de estado baseadas em imagens cruas sem e com informação de cor.

A proposta deste capítulo inspira-se na abordagem apresentada em (SUN et al., 2018) - denominada AG-RNC - descrita na seção 3.4. Desta forma, a presente contribuição estende a referida abordagem nos seguintes aspectos: 1) Elaboração de uma nova função de aptidão com caráter multiobjetivo, a qual combina o número de parâmetros e a acurácia do modelo associada aos exemplos de teste de generalização (comumente designados como

exemplos de validação) como características relevantes a serem levadas em consideração no processo de construção de uma RNC. Nesse sentido, o termo "Mínimas" presente na denominação do AG-RNC-M é devido ao fato da minimização do número de parâmetros de uma arquitetura de RNC considerada na função de aptidão; 2) Utilização da estratégia de treinamento de validação cruzada conhecida como *k-fold cross-validation* estratificada (descrita na subseção 2.4.1) com o objetivo de explorar mais adequadamente a distribuição das demonstrações/exemplos e, consequentemente, aumentar as chances de se encontrar um modelo tomador de decisão com maior qualidade.

Dessa forma, o método AG-RNC-M aqui proposto é explicado e detalhado na seção 8.1. As representações de estado são descritas na seção 8.2. Os agentes aqui projetados (com arquiteturas manual e automaticamente produzidas) são apresentados na seção 8.3. Por fim, os experimentos e a análise dos resultados são apresentados na seção 8.4.

8.1 Método Evolutivo: AG-RNC-M

Esta seção apresenta as principais características da adaptação feita pelo presente trabalho de Mestrado sobre o AG-RNC (SUN et al., 2018), gerando o AG-RNC-M. O Algoritmo 3 exibe a sua estrutura, a qual é idêntica à estrutura do AG-RNC. Basicamente, ambos os métodos encontram a melhor arquitetura de RNC para classificar um conjunto de dados de imagem fornecido como entrada por meio de um processo evolutivo, composto pelos seguintes passos: 1) Inicialização de uma população arbitrária. Neste passo, uma população de indivíduos com tamanho predefinido é inicializada aleatoriamente. Aqui, cada indivíduo representa uma arquitetura de rede neural que pode ser composta somente por blocos denominados *Skip* e *Pooling*; 2) Avaliação da aptidão dos indivíduos. Neste passo, cada indivíduo da população corrente é avaliado em termos de alguma métrica sobre o conjunto de dados fornecido; 3) Geração dos filhos (novos indivíduos). Neste passo, os indivíduos pais são selecionados com base na aptidão e, por meio da operação de *crossover*, geram novos indivíduos filhos. Em seguida, os filhos passam pela operação de mutação. Por fim, eles são avaliados; 4) Seleção dos indivíduos da nova população. Neste último passo, a nova população é selecionada com base na atual população (incluindo os filhos gerados no passo anterior). Tal seleção leva em consideração a aptidão dos indivíduos.

Os passos apresentados e as modificações realizadas para a construção do AG-RNC-M serão descritas em detalhes na sequência.

8.1.1 Passo 1: Inicialização da População

A população é composta por indivíduos que correspondem a arquiteturas de RNC. Neste trabalho, tais arquiteturas são constituídas por dois tipos distintos de blocos: *Skip* e *Pooling*. O bloco de *Skip*, inspirado na construção dos blocos residuais (inicialmente propostos em *ResNet* (HE et al., 2016)) e ilustrado na Figura 42(a), é composto por

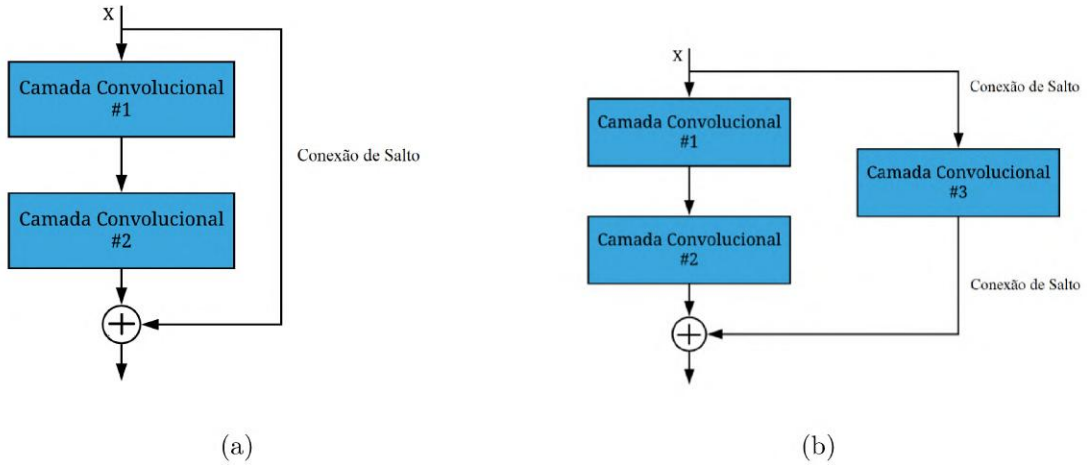
Algoritmo 3 Estrutura dos Métodos AG-RNC e AG-RNC-M

Entrada: O tamanho da população N, o número de gerações T, o conjunto de dados de imagem para classificação.

Saída: A melhor arquitetura de RNC descoberta.

1: $P_0 \leftarrow$ Inicialize uma população com tamanho N (Passo 1)
2: $t \leftarrow 0$
3: **while** $t < T$ **do**
4: Avalie a aptidão de cada indivíduo em P_t (Passo 2)
5: $Q_t \leftarrow$ Filhos gerados pela operação de *Crossover* entre os indivíduos progenitores selecionados (Passo 3)
6: $P_{t+1} \leftarrow$ Seleção ambiental de $P_t \bigcup Q_t$ (Passo 4)
7: $t \leftarrow t + 1$
8: **end while**
9: **return** Indivíduo com melhor aptidão em P_t.

duas camadas convolucionais e uma conexão de salto. Esta conexão é responsável por ligar a entrada X da primeira camada convolucional (referida como *conv1*) à saída da segunda camada convolucional (*conv2*). Caso os tamanhos espaciais da entrada de *conv1* e da saída de *conv2* sejam distintos, uma nova camada convolutiva (*conv3*) é aplicada à entrada X a fim de se obter o mesmo tamanho espacial da saída de *conv2*, conforme ilustrado na Figura 42(b).



(a) (b)

Figura 42 – Exemplos de camadas de *Skip*: (a) Situação com duas camadas convolucionais e (b) Situação com três camadas convolucionais

A Tabela 48 apresenta a configuração utilizada na implementação do bloco de *Skip*. Nota-se que os hiperparâmetros *Tamanho do Filtro*, *Stride* e *Padding* são representados por valores fixos, enquanto que o *Número de Filtros* é o único levado em consideração ao longo do processo evolutivo. Este hiperparâmetro, no caso do presente trabalho, pode assumir os seguintes valores: 8, 16, 32, 64.

Tabela 48 – Configuração dos Hiperparâmetros do Bloco de *Skip*

	Número de Filtros	**Tamanho do Filtro**	***Stride***	***Padding***
conv1	F1	3x3	1x1	*same*
conv2	F2	3x3	1x1	*same*
conv3	F2	1x1	1x1	*same*

O bloco de *Pooling* usado no AG-RNC-M tem os valores dos hiperparâmetros *Tamanho do Filtro* e *Stride* iguais a 2x2. Desse modo, o único hiperparâmetro levado em consideração no processo de codificação é o tipo da operação de *pooling* (P1). Tal operação pode assumir um dos seguintes valores: *max pooling* e *average pooling* (explanados na subseção 2.2.2).

O Algoritmo 4 apresenta o processo de inicialização da população inicial. Nela, cada indivíduo recebe uma arquitetura de RNC representada por uma lista de nodos. O número de nodos varia entre um e dez, sendo definido aleatoriamente. Além disso, cada um deles corresponde exclusivamente a um bloco de *Skip* ou a um bloco de *Pooling*. Durante a configuração de cada nó, um número r é gerado aleatoriamente a partir de uma distribuição de valores entre zero e um. Caso r seja menor do que 0.5, o nó compõe um bloco de *Skip* e os parâmetros F1 e F2, correspondentes ao hiperparâmetro *Número de Filtros* das camadas convolucionais *conv1* e *conv2* respectivamente, são escolhidos aleatoriamente dentre os valores possíveis. Caso contrário, o nó compõe um bloco de *Pooling* com iguais probabilidades de assumir as operações de *max pooling* ou *average pooling* (parâmetro P1). Observa-se que essa lista de nodos criada corresponde às camadas intermediárias da RNC. A camada de saída é adicionada e descrita no Passo 2.

É importante destacar que uma diferença do AG-RNC-M com relação ao AG-RNC é que, no presente método, um indivíduo é composto por no máximo dez nodos, enquanto que não há limite com relação ao tamanho da arquitetura no método desenvolvido em (SUN et al., 2018). Isto é, um indivíduo pode ter uma arquitetura de qualquer tamanho finito no AG-RNC. As principais razões que justificam a escolha feita pelos autores da presente pesquisa de Mestrado de se limitar o tamanho da arquitetura são: 1) as melhores arquiteturas e, consequentemente, melhores soluções encontradas pelo AG-RNC contam com no máximo nove nodos; 2) simplicidade de se implementar uma função de aptidão baseada no número de parâmetros, uma vez que é possível simular a arquitetura com o maior número de parâmetros possível baseado nas configurações sobre os blocos já mencionadas; 3) o fator tempo de execução é uma grande fragilidade de métodos evolutivos. Dessa forma, limitar o tamanho máximo de um indivíduo é uma forma de acelerar a execução do método.

Por fim, ressalta-se que as configurações e hiperparâmetros aqui adotados são embasados em projetos bem-sucedidos de renomadas RNCs e também naqueles explorados pelo AG-RNC.

Algoritmo 4 Inicialização da População Inicial

Entrada: O tamanho da população N.
Saída: A população inicial P_0.

1: $P_0 \leftarrow \emptyset$
2: **while** $|P_0| < N$ **do**
3: $L \leftarrow$ Geração aleatória de um número inteiro entre um e dez
4: *lista* $\leftarrow$ Criação de uma lista que contém L nodos
5: **for** *cada nodo de lista* **do**
6: $r \leftarrow$ Geração de número aleatório uniformemente distribuído entre zero e um
7: **if** $r < 0.5$ **then**
8: *nodo.tipo* $\leftarrow$ bloco *Skip*
9: *nodo.F1* $\leftarrow$ Geração de um número aleatório dentre os valores de *Número de Filtros*
10: *nodo.F2* $\leftarrow$ Geração de um número aleatório dentre os valores de *Número de Filtros*
11: **else**
12: *nodo.tipo* $\leftarrow$ bloco de *Pooling*
13: $q \leftarrow$ Geração de número aleatório uniformemente distribuído entre zero e um
14: **if** $q < 0.5$ **then**
15: *nodo.P1* $\leftarrow$ *max pooling*
16: **else**
17: *nodo.P1* $\leftarrow$ *average pooling*
18: **end if**
19: **end if**
20: **end for**
21: $P_0 \leftarrow P_0 \bigcup$ *lista*
22: **end while**
23: **return** P_0.

8.1.2 Passo 2: Avaliação da Aptidão dos Indivíduos

O Algoritmo 5 apresenta o processo de avaliação dos indivíduos de uma população. Para cada indivíduo, são realizadas duas fases: 1) Construção da arquitetura de RNC (linha 2); 2) Avaliação do seu desempenho (linhas 3 a 18).

8.1.2.1 Fase 1: Construção da Arquitetura

Primeiramente, a arquitetura de RNC do indivíduo p é decodificada a partir de sua respectiva lista de nodos (linha 2). Ao longo do processo de decodificação da RNC, a operação de normalização em lote seguida pela função de ativação *ReLu* é adicionada à saída de cada camada convolucional nos blocos de *Skip*, conforme ilustrado na Figura 43. Após todos os nodos serem decodificados, uma camada totalmente conectada - representando a camada de saída e implementada com o classificador *softmax* - é adicionada ao final de tal arquitetura. O número específico de classes é determinado pelo conjunto de dados fornecido.

Algoritmo 5 Avaliação da População

Entrada: A população P_t, o conjunto de dados para classificação, número máximo de parâmetros de uma RNC *MAX_PARAM*.

Saída: A população P_t avaliada.

1: **for** *cada indivíduo p em* P_t **do**
2: $p.RNC \leftarrow$ arquitetura de RNC construída a partir da lista de nodos em p
3: *partições* $\leftarrow$ Criação de cinco partições por meio do método *k-fold cross-validation* estratificado ($K = 5$)
4: $soma\_acurácias \leftarrow 0$
5: **for** *cada iteração i em partições* **do**
6: Inicialização da $p.RNC$ com parâmetros aleatórios
7: $Dados_{treino} \leftarrow$ dados correspondentes às partições de treinamento na iteração i
8: $Dados_{teste} \leftarrow$ dados correspondentes à partição de teste na iteração i
9: Realização do treinamento da $p.RNC$ com o conjunto $Dados_{treino}$
10: $acurácia_i \leftarrow$ execução da $p.RNC$ sobre o conjunto $Dados_{teste}$
11: $soma\_acurácias \leftarrow soma\_acurácias + acurácia_i$
12: **end for**
13: *dimensão* $\leftarrow$ número total de parâmetros contidos em $p.RNC$
14: $aptidão\_dimensão \leftarrow dimensão \ / \ MAX\_PARAM$
15: $acurácia\_média \leftarrow soma\_acurácias \ / K$ (K=5)
16: $aptidão\_acurácia \leftarrow 1 - acurácia\_média$
17: $p.aptidão \leftarrow aptidão\_dimensão + aptidão\_acurácia$
18: **end for**
19: **return** P_t

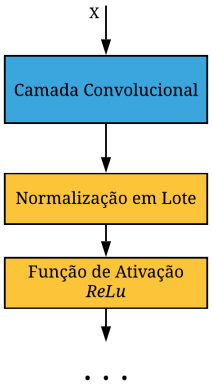


Figura 43 – Normalização em Lote e Função *ReLu* adicionadas após uma camada convolucional.

8.1.2.2 Fase 2: Avaliação do Desempenho da Arquitetura

O primeiro passo dessa fase é a criação das *partições* utilizadas para o treinamento e teste da rede neural por meio do método *k-fold cross-validation* estratificado (descrito na seção 2.4) com valor de K igual a cinco (linha 3). Dessa forma, é garantido que a arquitetura será avaliada uma vez com todos os exemplos do conjunto de dados fornecido.

Além disso, a variável *soma_acurácias* - utilizada para o cálculo da aptidão - é inicializada com o valor zero (linha 4). Para cada iteração efetuada sobre *partições*, a arquitetura é inicializada com parâmetros (vetor de pesos e bias) aleatórios (linha 6), treinada com as partições de treinamento (linha 9) e avaliada com a partição de teste (linha 10). Esta avaliação gera uma medida denominada $acurácia_i$ (acurácia referente à iteração i), a qual é adicionada em *soma_acurácias* (linha 11). O treinamento da RNC em cada iteração é efetuada com base nos hiperparâmetros apresentados na Tabela 49. Ao fim de todas as iterações, *soma_acurácias* corresponde à soma das acurácias obtidas para cada combinação de *partições*. Com isso, o cálculo da aptidão do indivíduo é computado (linhas 13 a 17).

Tabela 49 – Hiperparâmetros utilizados no treinamento da arquitetura de RNC de um indivíduo

Tamanho do Mini-Lote	32
Número de Épocas	10
Otimizador	*Adam*
Taxa de Aprendizagem	0,001
Função de Perda	*Categorical Cross Entropy*

A aptidão de um indivíduo (linha 18) corresponde a uma combinação linear entre dois termos definida pela Equação 7. Os termos TA_1 e TA_2 representam, respectivamente, as aptidões relativas ao número de parâmetros e ao desempenho com relação à acurácia da arquitetura de RNC. Dessa forma, F (função de aptidão) é um problema de minimização em relação aos termos TA_1 e TA_2, os quais são descritos na sequência.

$$F = min(TA_1) + min(TA_2) = min(TA_1 + TA_2) \tag{7}$$

Termo de Aptidão TA_1: o número total de parâmetros que compõem a arquitetura do indivíduo é setado em *dimensão* (linha 13). Em seguida, TA_1 (*aptidão_dimensão*) é definida baseada na divisão de *dimensão* por *MAX_PARAM* (linha 14). *MAX_PARAM* é um valor fixo que representa o número máximo de parâmetros que uma arquitetura de RNC pode ter, isto é, corresponde ao pior caso em termos da dimensão do vetor de pesos e bias. Como uma arquitetura pode ter no máximo dez nodos, o pior caso corresponde ao seguinte cenário: todos os nodos são representados por blocos de *Skip* com números de filtro F_1 e F_2 iguais ao valor máximo dentre aqueles possíveis (no caso, 64), o que corresponde a 3.474.500 parâmetros. Assim, o valor de TA_1 é melhor quanto menor for a *dimensão* de uma arquitetura. Além disso, o valor de TA_1 está normalizado no intervalo entre zero e um.

Termo de Aptidão TA_2: a acurácia média obtida nas *partições* é calculada a partir da *soma_acurácias* e do valor de K (número de *partições*) e setada em *acurácia_média*

(linha 15). Dessa forma, é possível afirmar que a *acurácia_média* terá um bom valor quando as acurácias de teste obtidas em cada iteração sobre as *partições* forem altas, isto é, quando a arquitetura atinge uma ótima habilidade ao lidar com os dados do conjunto fornecido. Destaca-se que o valor de *acurácia_média* está no intervalo entre zero e um. Em seguida, TA_2 (*aptidão_acurácia*) é definida baseado no cálculo do erro da aptidão média (1 - *acurácia_média*) (linha 16). Assim, o valor de TA_2 é melhor quanto menor for o erro.

8.1.3 Passo 3: Geração dos Indivíduos Filhos

O Algoritmo 6 apresenta os detalhes do processo de geração dos filhos. Ele é composto por duas fases: 1) seleção dos indivíduos pais e execução da operação de *crossover* (linhas 1 a 17); 2) realização da operação de mutação nos filhos gerados (linhas 18 a 26).

8.1.3.1 Fase 1: Seleção dos Pais e Execução do *Crossover*

A fase de seleção (linhas 3 a 6) é realizada por meio do popular método torneio binário (MILLER et al., 1995), o qual possui a seguinte dinâmica: dois indivíduos da população P_t são escolhidos aleatoriamente e aquele com melhor aptidão é selecionado como pai. Este processo é efetuado duas vezes para a obtenção de dois indivíduos pais.

Após os pais serem selecionados, um número que determinará a realização ou não da operação *crossover* é gerado aleatoriamente (linha 7). Caso o valor de tal número estiver acima da probabilidade p_c, então o *crossover* não é efetuado e esses dois pais são inseridos como filhos em Q_t (linha 15). Caso contrário, a operação denominada de *Crossover* de um ponto (Srinivas; Patnaik, 1994) é efetuada entre os pais selecionados. Esta operação funciona da seguinte forma: cada indivíduo pai é aleatoriamente dividido em duas partes (linhas 9 e 10), definidas por um ponto de corte (este ponto não é necessariamente o mesmo para cada indivíduo), conforme ilustrado na Figura 44. A primeira parte de um é combinada com a segunda parte do outro de modo a gerar dois filhos (linhas 11 a 15), como ilustrado na Figura 45. Esta operação foi escolhida especificamente pela sua simplicidade e pelo fato de que os indivíduos possuem tamanho variável (consequentemente, os pais podem ter um número distinto de nodos). Com ela, foi possível obter bons resultados, conforme será mostrado nos experimentos (seção 8.4).

Destaca-se que um total de $|P_t|$ filhos são gerados ($|\ .\ |$ indica o tamanho do conjunto) caso o tamanho da população seja um número par. Caso contrário, serão gerados $|P_t| + 1$ filhos. Por exemplo, se a população é composta por dez indivíduos, serão gerados também dez filhos. No entanto, se o tamanho da população for igual a cinco, então seis filhos serão gerados (pelo fato de que dois pais sempre geram dois filhos e pela condição de parada na linha 2).

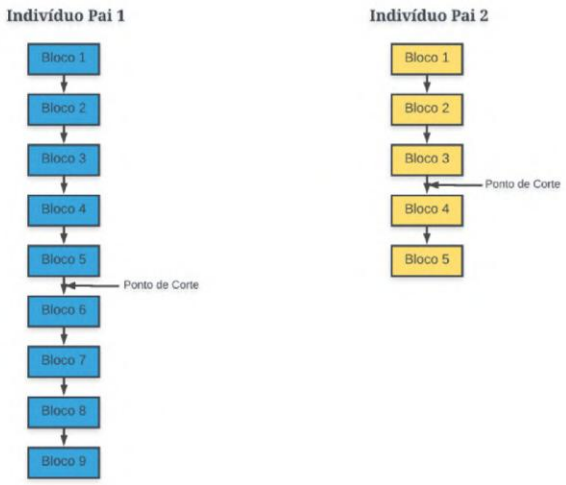


Figura 44 – Exemplo de indivíduos pais e seus respectivos pontos de corte na operação de *crossover*

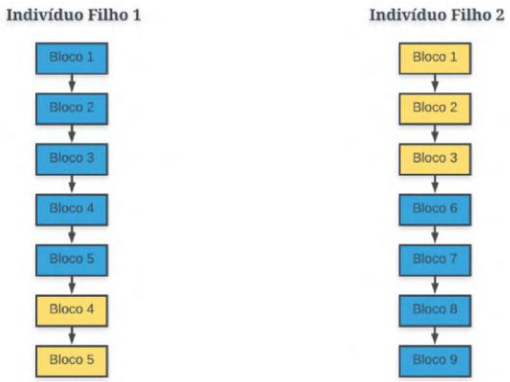


Figura 45 – Exemplo da obtenção dos indivíduos filhos pela operação de *crossover*

Algoritmo 6 Geração dos Filhos

Entrada: População P_t, probabilidade de execução de *crossover* p_c, probabilidade de execução de mutação p_m, lista de operações de mutação l_m, probabilidades associadas a cada operação de mutação p_l.

Saída: Indivíduos filhos Q_t.

1: $Q_0 \leftarrow \emptyset$
2: **while** $|Q_t| < |P_t|$ **do**
3: **while** $p_1 < p_2$ **do**
4: $p_1 \leftarrow$ Seleciona aleatoriamente dois indivíduos de P_t. Escolhe aquele com melhor aptidão dentre os dois selecionados.
5: $p_2 \leftarrow$ Idem p_1.
6: **end while**
7: $r \leftarrow$ Geração de número aleatório entre zero e um
8: **if** $r < p_c$ **then**
9: Escolhe aleatoriamente um ponto em p_1 e divide em duas partes
10: Escolhe aleatoriamente um ponto em p_2 e divide em duas partes
11: $\sigma_1 \leftarrow$ Combina a primeira parte de p_1 e a segunda parte de p_2
12: $\sigma_2 \leftarrow$ Combina a primeira parte de p_2 e a segunda parte de p_1
13: $Q_t \leftarrow Q_t \bigcup \sigma_1 \bigcup \sigma_2$
14: **else**
15: $Q_t \leftarrow Q_t \bigcup p_1 \bigcup p_2$
16: **end if**
17: **end while**
18: **for** *indivíduo* p *em* Q_t **do**
19: $r \leftarrow$ Geração de número aleatório entre zero e um
20: **if** $r < p_m$ **then**
21: $i \leftarrow$ Escolhe aleatoriamente um ponto em p
22: $m \leftarrow$ Seleciona uma operação da lista l_m baseado nas probabilidades em p_l
23: Efetua a mutação m no ponto i de p
24: **end if**
25: **end for**
26: **return** Q_t.

Por fim, a lista de nodos de cada filho pode ultrapassar o tamanho definido de dez nodos. Dessa forma, os filhos também são limitados em dez blocos de modo a manter a coerência do algoritmo.

8.1.3.2 Fase 2: Operação de Mutação nos indivíduos filhos

Com os filhos já gerados ao fim da **Fase 1**, o processo de mutação pode ser efetuado para cada um deles. Ao longo desse processo, um número aleatório é calculado (linha 19) e uma operação de mutação é efetuada no indivíduo filho se este número estiver abaixo de p_m (linhas 20-24). Neste caso, uma posição válida do indivíduo (referida como i), correspondente a um nodo de sua lista de nodos, é selecionada aleatoriamente e uma operação de mutação específica (referida como m) é também escolhida de modo aleatório

- com base nas probabilidades em p_l - da lista de mutações l_m. Por fim, m é aplicado na posição i do filho.

Neste sentido, as operações de mutação disponíveis definidas em l_m são:

Operação 1: Adicionar um bloco de *Skip* com configurações de *F1* e *F2* aleatórias;

Operação 2: Adicionar um bloco de *Pooling* com configurações aleatórias (*max pooling* ou *average pooling*);

Operação 3: Remover o bloco na posição i selecionada;

Operação 4: Alterar, aleatoriamente, as configurações do bloco na posição i selecionada.

Por sua vez, as probabilidades em p_l associadas à cada operação de mutação são:

Probabilidade da Operação 1: 70%;

Probabilidade da Operação 2: 10%;

Probabilidade da Operação 3: 10%;

Probabilidade da Operação 4: 10%.

As operações de mutação fornecidas são selecionadas com diferentes probabilidades. Especificamente, há uma maior chance da operação de adicionar um bloco de *Skip* ocorrer em relação às outras operações. Conforme justificado em (SUN et al., 2018), isso foi realizado pelo fato de que adicionar este tipo de bloco incentiva o aumento da profundidade da RNC, o que pode torná-la mais eficiente (em termos de acurácia) para lidar com os dados de entrada.

Ressalta-se novamente que os indivíduos filhos são limitados ao número de dez nodos (excluindo-se a camada de saída). Dessa forma, caso um indivíduo filho que já possui dez nodos sofra o processo de mutação pela **Operação 1** ou **Operação 2** (as quais adicionam um novo bloco em sua arquitetura), a sua nova lista de nodos será formada pelos dez primeiros blocos de sua respectiva arquitetura. Por exemplo, conforme ilustrado na Figura 46, um filho com dez nodos sofre uma das duas referidas operações de mutação na posição oito. A sua nova arquitetura é formada agora pelos dez primeiros blocos, excluindo o *Bloco 10* que estava presente na arquitetura original. Por fim, os indivíduos filhos são avaliados.

8.1.4 Passo 4: Seleção Ambiental

O algoritmo 7 exibe os detalhes do processo de seleção da nova população P_{t+1}. O tamanho de P_{t+1} é igual ao tamanho de P_t ($|P_{t+1}| = |P_t|$). Dessa forma, $|P_t|$ indivíduos são selecionados do conjunto formado pela população ($Q_t \bigcup P_t$) por meio do torneio binário

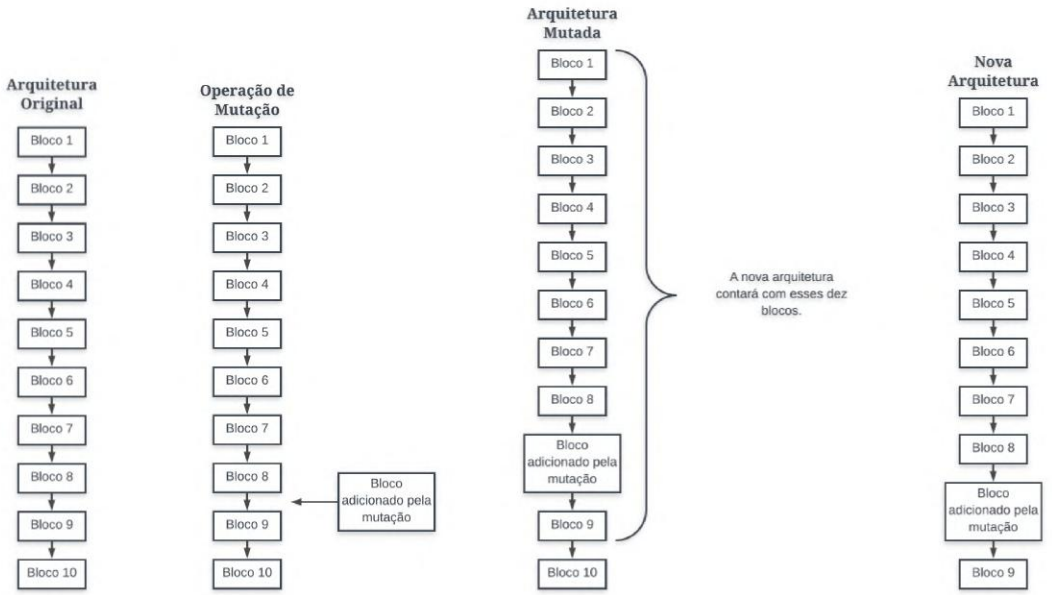


Figura 46 – Exemplo da operação de mutação em um indivíduo

e adicionados em P_{t+1} (linhas 2 a 7). Em seguida, é verificado se o melhor indivíduo dentre $P_t \bigcup Q_t$ está presente em P_{t+1}. Caso não estiver, ele será colocado no lugar do pior indivíduo de P_{t+1} (linhas 8 a 11).

Assim, tal processo equilibra a qualidade das soluções presentes na próxima geração por uma combinação entre o torneio binário e uma estratégia elitista. Uma população desejável contém tanto soluções boas quanto as relativamente ruins de modo a aumentar a sua diversidade e evitar o fenômeno da convergência prematura (MICHALEWICZ, 1996) - no qual o processo evolutivo fica preso a um ótimo local caso apenas os melhores indivíduos fossem selecionados (DAVIS, 1991). Para tanto, o torneio binário é utilizado. Contudo, ele não garante que a melhor solução estará presente na nova população, o que pode prejudicar a convergência do processo evolutivo. Desta forma, o melhor indivíduo é adicionado de modo explícito na nova população, o que é considerado uma estratégia elitista (BHANDARI; MURTHY, 1996).

8.2 Especificação das Representações de Estado

As representações utilizadas nos agentes implementados neste capítulo são baseadas em imagens cruas sem e com informação de cor, as quais são descritas na sequência.

Algoritmo 7 Seleção da Nova População

Entrada: Atual População P_t, Indivíduos Filhos Q_t
Saída: População da nova geração P_{t+1}.

1: $P_{t+1} \leftarrow \emptyset$
2: **while** $|P_{t+1}| < |P_t|$ **do**
3: $p_1 \leftarrow$ Seleciona aleatoriamente um indivíduo do conjunto $P_t \bigcup Q_t$
4: $p_2 \leftarrow$ Idem p_1
5: $p \leftarrow$ Seleciona o indivíduo com melhor aptidão dentre $\{p_1, p_2\}$
6: $P_{t+1} \leftarrow P_{t+1} \bigcup p$
7: **end while**
8: $p_{best} \leftarrow$ Encontra o melhor indivíduo (melhor aptidão) do conjunto $P_t \bigcup Q_t$
9: **if** p_{best} *não está em* P_{t+1} **then**
10: Substitui o indivíduo com pior aptidão em P_{t+1} por p_{best}
11: **end if**
12: **return** P_{t+1}.

8.2.1 Escala de Cinza

Essa representação é idêntica à utilizada no capítulo 7. Portanto, cada estado é representado por uma imagem de tamanho 120x90 (isto é, o estado é uma matriz 120x90x1).

8.2.2 RGB

Esta representação é idêntica àquela descrita na subseção 6.1.2 do capítulo 6, conforme ilustrada na Figura 47. Portanto, cada estado é representado pela imagem corrente de jogo de tamanho 120x90x3 (3 canais de cores da RGB).

Figura 47 – Exemplo da representação **RGB** no modo de confronto

8.3 Arquitetura dos Agentes Jogadores

Esta seção detalha a construção dos agentes baseados em AI construídos para o modo de confronto. Assim como descrito na seção 7.2 do capítulo 7, todos eles são constituídos de duas RNCs (*RNC de Movimentação* e *RNC de Controle*). A Tabela 50 resume as principais características dos agentes aqui propostos.

Tabela 50 – Agentes Baseados em AI Implementados para o Modo de Confronto

Agente	**Representação de Estado**	**Definição das RNCs**
ID-MC-Agent	Variável	Variável
IAP-MC-Agent	Variável	Automática

Os agentes apresentados na Tabela 50 apresentam as seguintes variações em relação à representação de estado e à forma de definição das RNCs:

- *ID-MC-Agent* com **Escala de Cinza** e **Manual**.
- *ID-MC-Agent* com **RGB** e **Manual**.
- *ID-MC-Agent* com **Escala de Cinza** e **Automática**.
- *ID-MC-Agent* com **RGB** e **Automática**.
- *IAP-MC-Agent* com **Escala de Cinza** e **Automática**.
- *IAP-MC-Agent* com **RGB** e **Automática**.

Destaca-se que o *ID-MC-Agent* leva em consideração demonstrações relativas a 100 jogos, produzindo um conjunto de *ações de movimento* formado por 18000 exemplos e um conjunto de *ações de controle* composto por 3000 demonstrações. Ambos os conjuntos são balanceados. Tais demonstrações foram recuperadas de tomadas de decisão nas quais um agente supervisor humano alcançou uma pontuação de jogo média igual a 3000 aproximadamente (tal supervisor é representado por um dos autores do presente trabalho de Mestrado, o qual possui um nível de jogo entre intermediário e avançado na tarefa). Além disso, ele varia em relação às representações de estado e também com relação à definição das RNCs. O *IAP-MC-Agent* varia apenas com relação às representações de estado, sendo construído a partir das versões de *ID-MC-Agent* que utilizam o AG-RNC-M para definição automática das RNCs.

8.3.1 Variações Baseadas nas RNCs definidas Manualmente

O *ID-MC-Agent* com **Escala de Cinza** e **Manual** e o *ID-MC-Agent* com **RGB** e **Manual** são as variações em que tanto a *RNC de Movimentação* quanto a *RNC de Controle* são definidas manualmente de acordo com a arquitetura proposta em (HUSSEIN et al., 2018), conforme descrito na subseção 2.3.2. A primeira variação considera a representação **Escala de Cinza** (descrita na subseção 8.2.1), enquanto a segunda considera a representação **RGB** (descrita na subseção 8.2.2).

8.3.2 Variações Baseadas nas RNCs definidas Automaticamente pelo AG-RNC-M

Essas variações são construídas de acordo com o método AG-RNC-M para definir as RNCs de maneira automática. Os parâmetros de execução do AG-RNC-M são expostos na Tabela 51: tamanho da população igual a 10; número de gerações igual a 10; probabilidade de efetuar a operação *crossover* igual a 0,8; e, probabilidade de efetuar operação de mutação igual a 0,2.

Tabela 51 – Parâmetros de execução do AG-RNC-M

Tamanho da População	10
Número de Gerações	10
Probabilidade de *Crossover*	*0,8*
Probabilidade de Mutação	0,2

A Tabela 52 apresenta a melhor arquitetura para a *RNC de Movimentação* encontrada pelo AG-RNC-M para a representação de estado **Escala de Cinza**. Como pode ser visto, ela é formada por cinco blocos (sendo três do tipo *Skip* e dois do tipo *Pooling*) e, no total, possui 11 camadas que consistem em nove camadas convolucionais e duas camadas de *Pooling*. Além disso, ao todo, ela possui 120.380 parâmetros. Da mesma forma, a Tabela 53 apresenta a melhor arquitetura de *RNC de Controle*. Ela também é formada por cinco blocos (sendo dois do tipo *Skip* e três do tipo *Pooling*) e, no total, possui nove camadas que consistem em seis camadas convolucionais e três camadas de *Pooling*. Ela possui apenas 20.348 parâmetros. Assim, tais RNCs são utilizadas pelas variações *ID-MC-Agent* com **Escala de Cinza** e **Automática** e *IAP-MC-Agent* com **Escala de Cinza** e **Automática** (a qual estende a anterior).

Tabela 52 – *RNC de Movimentação* Otimizada Considerando a Representação de Estado **Escala de Cinza**

Bloco	**F1**	**F2**	**P1**
Bloco 1 - *Skip*	64	8	-
Bloco 2 - *Pooling*	-	-	*max pooling*
Bloco 3 - *Skip*	16	8	-
Bloco 4 - *Pooling*	-	-	*max pooling*
Bloco 5 - *Skip*	4	32	-

Analogamente, a Tabela 54 apresenta a melhor arquitetura para a *RNC de Movimentação* encontrada pelo AG-RNC-M para a representação de estado **RGB**. Ela é formada por dez blocos (sendo sete do tipo *Skip* e três do tipo *Pooling*) e, no total, possui 22 camadas que consistem em 19 camadas convolucionais e três camadas de *Pooling*. Além disso, ao todo, ela possui 155.372 parâmetros. Da mesma forma, a Tabela 53 apresenta a melhor arquitetura de *RNC de Controle*. Ela é formada por sete blocos (sendo quatro

Tabela 53 – *RNC de Controle* Otimizada Considerando a Representação de Estado **Escala de Cinza**

Bloco	**F1**	**F2**	**P1**
Bloco 1 - *Skip*	16	8	-
Bloco 2 - *Pooling*	-	-	*average pooling*
Bloco 3 - *Pooling*	-	-	*max pooling*
Bloco 4 - *Skip*	32	16	-
Bloco 5 - *Pooling*	-	-	*max pooling*

do tipo *Skip* e três do tipo *Pooling*) e, no total, possui 15 camadas que consistem em 12 camadas convolucionais e três camadas de *Pooling*. Ela possui apenas 33.764 parâmetros.

Tabela 54 – *RNC de Movimentação* Otimizada Considerando a Representação de Estado **RGB**

Bloco	**F1**	**F2**	**P1**
Bloco 1 - *Skip*	8	64	-
Bloco 2 - *Pooling*	-	-	*max pooling*
Bloco 3 - *Skip*	32	16	-
Bloco 4 - *Skip*	32	16	-
Bloco 5 - *Skip*	64	16	-
Bloco 6 - *Skip*	64	32	-
Bloco 7 - *Pooling*	-	-	*max pooling*
Bloco 8 - *Pooling*	-	-	*max pooling*
Bloco 9 - *Skip*	32	16	-
Bloco 10 - *Skip*	8	64	-

Tabela 55 – *RNC de Controle* Otimizada Considerando a Representação de Estado **RGB**

Bloco	**F1**	**F2**	**P1**
Bloco 1 - *Skip*	8	16	-
Bloco 2 - *Pooling*	-	-	*average pooling*
Bloco 3 - *Skip*	16	8	-
Bloco 4 - *Skip*	16	64	-
Bloco 5 - *Pooling*	-	-	*max pooling*
Bloco 6 - *Skip*	16	8	-
Bloco 7 - *Pooling*	-	-	*max pooling*

É importante ressaltar que, após encontrar as RNCs por meio do método automático aqui utilizado, todas elas foram treinadas com todos os exemplos contidos em seus respectivos conjuntos fornecidos como entrada. Por fim, as variações de *IAP-MC-Agent* foram implementadas com os hiperparâmetros apresentados nas Tabelas 56 e 57.

8.4 Experimentos e Resultados

Os experimentos aqui realizados têm como objetivo principal validar o método evolutivo AG-RNC-M no processo de construção automática de arquiteturas de RNC otimizadas por meio dos agentes *ID-MC-Agent* e *IAP-MC-Agent* no modo de confronto.

Tabela 56 – Hiperparâmetros Utilizados no Treinamento do *IAP-MC-Agent* com **Escala de Cinza**

Tamanho do Mini-Lote	32
Número de Épocas	10
Otimizador	*Adam*
Taxa de Aprendizagem	0,001
Função de Perda	*Categorical Cross Entropy*
Percentual Dados Ativos	10%
β **da *RNC de Movimentação***	0,5
β **da *RNC de Controle***	1,1

Tabela 57 – Hiperparâmetros Utilizados no Treinamento do *IAP-MC-Agent* com **RGB**

Tamanho do Mini-Lote	32
Número de Épocas	10
Otimizador	*Adam*
Taxa de Aprendizagem	0,001
Função de Perda	*Categorical Cross Entropy*
Percentual Dados Ativos	10%
β **da *RNC de Movimentação***	0,8
β **da *RNC de Controle***	1,5

Assim, em um primeiro momento, é efetuada uma avaliação sobre o impacto que as arquiteturas geradas pelo método AG-RNC-M exercem no desempenho de agentes jogadores (Experimento 1). Em seguida, uma outra análise comparativa é realizada sobre as representações de estados descritas na seção 5.2 a partir do método IAP (Experimento 2). A métrica utilizada foi pontuação de jogo. Os experimentos foram executados em uma arquitetura composta por uma máquina com uma GPU NVIDIA GeForce 940MX e 8 GB de RAM.

8.4.1 Experimento 1: Avaliação do Método AG-RNC-M

O objetivo aqui é avaliar o impacto causado pelo AG-RNC-M na construção dos módulos de tomada de decisão de agentes jogadores. Nesse sentido, foi efetuada uma análise comparativa, em termos da pontuação de jogo, entre as variações do *ID-MC-Agent* que utilizam a arquitetura de RNC proposta em (HUSSEIN et al., 2018) (manualmente) e aquelas que utilizam as arquiteturas geradas pelo AG-RNC-M (automaticamente) com base nas duas representações aqui estudadas. A avaliação da pontuação de jogo foi realizada no contexto de um torneio composto de 50 partidas envolvendo tais agentes e a máquina da *FIFA*. Enfatiza-se que tal máquina usada contra todas as variações possuía o mesmo nível de jogo (ou melhor, mesma dificuldade) de maneira a não beneficiar nenhum agente em particular.

De modo a facilitar a análise do impacto causado em cada representação, esse experimento foi decomposto em duas fases que serão explanadas a seguir.

Fase 1 - Avaliação das Variações baseadas na Representação de Estado Escala de Cinza:

A Tabela 58 mostra, respectivamente, a média total e o desvio padrão das variações de *ID-MC-Agent* com **Escala de Cinza** no torneio efetuado em termos da pontuação de jogo. Os resultados indicam uma superioridade de desempenho da variação produzida pela definição automática das RNCs. Assim, um *Teste T* com um *nível de significância* ($\alpha = 0,05$) foi executado sobre tais resultados para validar a comparação de desempenho entre as referidas variações. Neste sentido, a seguinte hipótese nula H_0 foi criada: ambas as variações possuem o mesmo nível de desempenho.

Tabela 58 – Resultados de *ID-MC-Agent* com **Escala de Cinza** e suas variações em termos da pontuação de jogo

Agente	Média	Desvio Padrão
ID-MC-Agent com **Escala de Cinza** e **Manual**	1100,00	1161,456
ID-MC-Agent com **Escala de Cinza** e **Automática**	1298,00	1042,855

Como indicado na Tabela 59, o *valor-t* negativo (t(98) = -0,897) indica que a média da pontuação de jogo relativa ao *ID-MC-Agent* com **Escala de Cinza** e **Automática** é ligeiramente maior do que a média obtida pelo *ID-MC-Agent* com **Escala de Cinza** e **Manual** com *graus de liberdade* igual a 98. No entanto, como *valor-p* $= 0,372$ é maior do que o *nível de significância* ($\alpha = 0,05$), a hipótese nula não pode ser rejeitada. Portanto, esses resultados sugerem que a variação produzida por meio do AG-RNC-M tem o mesmo desempenho daquela implementada de acordo com (HUSSEIN et al., 2018).

Tabela 59 – *Teste T* aplicados aos resultados de *ID-MC-Agent* com **Escala de Cinza**

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
ID-MC-Agent **com Escala de Cinza e Manual** **x** *ID-MC-Agent* **com Escala de Cinza e Automática**	-,897	98	,372

Fase 2 - Avaliação das Variações baseadas na Representação de Estado RGB:

A Tabela 60 mostra, respectivamente, a média total (Média) e o desvio padrão (Desvio Padrão) das variações de *ID-MC-Agent* com **RGB** no torneio efetuado em termos da pontuação de jogo. Os resultados indicam uma semelhança muito grande de desempenho das variações provindas tanto da definição automática quanto da definição manual das RNCs. Assim, um *Teste T* com um *nível de significância* ($\alpha = 0,05$) foi executado sobre tais resultados para validar a comparação de desempenho

entre tais variações. Neste sentido, a seguinte hipótese nula H_0 foi criada: ambas as variações possuem o mesmo nível de desempenho.

Tabela 60 – Resultados de *ID-MC-Agent* com **RGB** e suas variações em termos da pontuação de jogo

Agente	Média	Desvio Padrão
ID-MC-Agent com **RGB** e **Manual**	1752,00	1030,007
ID-MC-Agent com **RGB** e **Automática**	1710,00	1009,395

Como indicado na Tabela 61, o *valor-t* positivo (t(98) = 0,206) indica que a média da pontuação de jogo relativa ao *ID-MC-Agent* com **RGB** e **Manual** é ligeiramente maior do que a média obtida pelo *ID-MC-Agent* com **RGB** e **Automática** com *graus de liberdade* igual a 98. No entanto, como *valor-p* $= 0,837$ é maior do que o *nível de significância* ($\alpha = 0,05$), a hipótese nula não pode ser rejeitada. Portanto, esses resultados sugerem que tais variações possuem o mesmo nível de desempenho.

Tabela 61 – *Teste T* aplicados aos resultados de *ID-MC-Agent* com **RGB**

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
***ID-MC-Agent* com RGB e Manual x *ID-MC-Agent* com RGB e Automática**	,206	98	,837

Os resultados obtidos permitiram validar o método AG-RNC-M como uma ferramenta auxiliadora na construção de RNCs que atuam como tomadoras de decisão em agentes jogadores em um cenário complexo de jogo (modo de confronto). Em ambas as fases avaliadas, as variações produzidas por meio do referido método obtiveram resultados muito competitivos em relação àquelas que utilizam RNCs definidas manualmente. Apesar das variações embasadas no AG-RNC-M não terem melhorado significativamente o desempenho no modo de confronto, é interessante observar que elas são muito mais leves em termos do número de parâmetros em relação a arquitetura de RNC extraída de (HUSSEIN et al., 2018). De fato, esta última arquitetura (definida manualmente) é composta por 3.181.400 parâmetros, enquanto a maior RNC encontrada pelo método automático aqui proposto possui apenas 155.372 parâmetros (aproximadamente 20 vezes menor). Isso influencia diretamente no espaço (memória) necessário para armazenar a rede neural e também pode auxiliar no tempo de processamento de cada *frame* em tempo real, pois espera-se que, quanto menor a rede, mais rápida a sua inferência, o que pode ser crucial na escolha de boas ações em determinadas situações de jogo.

Além disso, o método AG-RNC-M reduz consideravelmente o esforço humano no processo de criação de RNCs. De fato, a definição manual de arquiteturas apropriadas para um problema em específico exige um nível de expertise considerável

por parte dos projetores, uma vez que não é trivial definir os hiperparâmetros de uma arquitetura, o que também torna o seu processo de validação oneroso. Dessa forma, por ser totalmente automático, o método aqui proposto pode ser generalizado para outros contextos que envolvem tarefas de classificação de imagens, independentemente do conhecimento ou não sobre RNCs e até mesmo sobre AG por parte de pesquisadores. No entanto, a grande fragilidade dessa abordagem é relativa ao cálculo da aptidão dos indivíduos baseado na técnica *k-fold cross validation*, o que corresponde a um processo bastante oneroso em termos de tempo computacional, apesar de garantir uma menor variância da melhor arquitetura gerada. Isto é, ela é mais confiável e espera-se que seja mais robusta do que se a técnica evolutiva considerasse, por exemplo, o método *Holdout* (SAMMUT; WEBB, 2010) para avaliar os indivíduos (principalmente ao lidar com bases de dados menores, como o conjunto de *ações de controle* de 3000 demonstrações).

Assim, os autores estão estudando a seguinte modificação no método AG-RNC-M relacionada à função de aptidão: ao invés de computar a acurácia média a partir do uso do método *k-fold cross validation*, efetuar o treinamento da arquitetura de RNC com todos os exemplos do conjunto de dados e considerar, diretamente, o desempenho em termos da pontuação de jogo como um dos termos no cálculo da aptidão. Destaca-se que o número de parâmetros da arquitetura é mantido como o outro termo do AG. Contudo, é necessário enfatizar que, embora isso diminua o custo computacional, essa abordagem pode trazer mais chances de *overfitting*.

8.4.2 Experimento 2: Análise Comparativa entre as Representações de Estado

O objetivo aqui é avaliar a qualidade da tomada de decisão de *IAP-MC-Agent* e suas variações - implementadas a partir de suas respectivas versões de *ID-MC-Agent* que foram produzidas por meio do AG-RNC-M (*ID-MC-Agent* com **Escala de Cinza** e **Automática** e *ID-MC-Agent* com **RGB** e **Automática**) - em termos de representação de estado. Nesse sentido, cada variação de *IAP-MC-Agent* foi submetida a um torneio constituído de 50 partidas contra a máquina do *FIFA*.

A Tabela 62 mostra, respectivamente, a média total e o desvio padrão das variações de *IAP-MC-Agent* no torneio efetuado em termos da pontuação de jogo. Os resultados indicam uma grande superioridade de desempenho da variação que considera a representação **RGB** sobre aquela baseada na representação **Escala de Cinza**. A fim de validar tal comparação de desempenho, um *Teste T* com um *nível de significância* ($\alpha = 0,01$) foi executado. Neste sentido, a seguinte hipótese nula H_0 foi criada: as variações implementadas de acordo com as representações **Escala de Cinza** e **RGB** possuem o mesmo nível de desempenho.

Tabela 62 – Resultados de *IAP-MC-Agent* e suas variações em termos da pontuação de jogo

Agente	Média	Desvio Padrão
IAP-MC-Agent com **Escala de Cinza**	1622,00	746,226
IAP-MC-Agent com **RGB**	2172,00	795,405

Como indicado na Tabela 63, o *valor-t* positivo (t(98) = -3,566) indica que a média da pontuação de jogo relativa ao *IAP-MC-Agent* com **RGB** é consideravelmente maior do que a média obtida pelo *ID-MC-Agent* com **Escala de Cinza** com *graus de liberdade* igual a 98. Como *valor-p* $= 0,001$ é menor do que o *nível de significância* ($\alpha = 0,01$), a hipótese nula pode ser rejeitada. Portanto, esses resultados confirmam que a representação **RGB** possui uma maior capacidade de gerar informações relevantes para a tomada de decisão de agentes jogadores no modo de confronto com um intervalo de confiança de 99%.

Tabela 63 – *Teste T* aplicados aos resultados de *IAP-MC-Agent*

Experimento	*valor-t*	*graus de liberdade*	*valor-p*
IAP-MC-Agent **com Escala de Cinza** **x** *IAP-MC-Agent* **com RGB**	-3,566	98	,001

Os resultados aqui obtidos permitiram verificar que a representação de ambiente baseada em imagens cruas com informação de cor propicia uma melhor qualidade de percepção do estado atual de jogo e, consequentemente, um melhor desempenho a um agente jogador atuante no cenário modo de confronto do que aquela que não leva em consideração a informação de cor. Os autores acreditam que a representação sem cor é totalmente viável para a criação de um agente inteligente que atue de modo razoável no problema abordado (visto o desempenho obtido pelo *IAP-MC-Agent* com **Escala de Cinza**), mas que levar em conta a informação de cor é crucial para melhorar a qualidade do processo de aprendizado e de tomada de decisão de um agente (visto os melhores resultados alcançados pelo *IAP-MC-Agent* com **RGB**). Pelo fato das imagens coloridas serem representadas por três valores, enquanto as imagens em escala de cinza por apenas um, é possível que informações sobre elementos relevantes à tarefa de tomada de decisão sejam prejudicadas e até perdidas (por exemplo, entre cores que são transformadas em valores de escala de cinza bem próximos).

É possível observar que o *IAP-MC-Agent* baseado na representação **Escala de Cinza** aqui produzido obteve um desempenho praticamente idêntico àquele construído no capítulo 7. Dessa forma, apesar de ter sido utilizada uma maior quantidade de demonstrações e uma taxa de aprendizado ativo mais alta, os resultados não foram aprimorados. Com relação ao *IAP-MC-Agent* com **RGB**, o desempenho melhorou consideravelmente ao utilizar as informações de cor das imagens. Em termos práticos, ele é o melhor agente aqui

gerado e demonstra uma atuação bem próxima do nível humano em diversas execuções, porém não tem uma performance consistente por conta das propriedades complexas do ambiente (dinâmico, multiagente e desconhecido). Todos esses fatos corroboram com as limitações inerentes às representações aqui utilizadas, as quais não consideram a sequência ou a dinâmica do ambiente no processo de tomada de decisão.

Assim, os autores acreditam que os resultados podem ser melhorados a partir da utilização de técnicas que auxiliem a tornar os agentes capazes de "conhecerem" melhor o ambiente de modo a atenuar o impacto das propriedades complexas desse cenário, como as redes neurais recorrentes - as quais levam em conta na tomada de decisão estados de jogo vistos previamente, além do *frame* (estado) recente. Além disso, também é sugerido a remoção de informações irrelevantes no processo de tomada consideradas nas representações aqui estudadas. Por exemplo, na Figura 48, a parte interna do retângulo vermelho indica as informações que são realmente relevantes para o agente considerar em sua tomada de decisão. A parte externa do retângulo não precisa ser levada em consideração no processo de escolha de uma ação, todavia, ela foi utilizada nos agentes aqui construídos.

Figura 48 – Exemplo de informações irrelevantes na representação de estado baseada em imagem crua

Capítulo 9

Conclusão e Trabalhos Futuros

Este trabalho efetuou uma investigação de técnicas de AM - tanto da abordagem de ARP quanto da abordagem de AI - combinadas a técnicas de representação de ambiente baseadas em VC - a partir do uso de RNCs e também de TDOs para processar eficientemente as imagens brutas de jogo - no processo de construção de agentes que resolvem problemas, em tempo real, com propriedades de ambiente completamente observáveis, estocásticos, desconhecidos e que variam em termos das propriedades de agente único ou multiagente, e estático ou dinâmico. Para tanto, o *FIFA* foi utilizado como estudo de caso, por envolver cenários desafiadores compostos pelas propriedades citadas e também por ter sido relativamente pouco explorado no contexto de AM. Particularmente, os seguintes cenários do jogo *FIFA* foram abordados: cobrança de faltas sem goleiro; cobrança de faltas com goleiro; modo de confronto. Assim, foram estudados diferentes métodos de aprendizado, tais como o DQN (relativo à abordagem de ARP), a ID e a IAP (correspondentes à abordagem de AI), além de distintas formas de representação de ambiente baseadas em imagens brutas e em TDO. Além disso, o presente trabalho explorou técnicas evolutivas baseadas em AG com o propósito de definir arquiteturas apropriadas de RNC de maneira automática, o que resultou na produção do método AG-RNC-M.

Assim sendo, em um primeiro momento, uma interface de conexão foi construída e validada. Ela mostrou-se fundamental para o desenvolvimento do presente trabalho. Em seguida, o método DQN foi estudado nos cenários de faltas, onde obteve um ótimo desempenho naquele que não envolve a presença de um goleiro, enquanto não conseguiu alcançar um bom resultado no cenário com goleiro por conta da modelagem simples da função de recompensa. Dessa forma, o método IAP foi utilizado com sucesso para resolver a tarefa envolvendo este último cenário. Além disso, ele foi validado juntamente com a ID no modo de confronto (cenário mais complexo aqui abordado), onde obteve um ótimo comportamento quando combinado ao método AG-RNC-M e à representação por imagens brutas coloridas.

Com relação às hipóteses do presente trabalho, os métodos de aprendizagem foram avaliados combinados às representações de ambiente. Particularmente, as representações

provenientes de imagens brutas com informação de cor (RGB) obtiveram os melhores resultados em geral. Ademais, a presente pesquisa atestou o potencial que as TDOs possuem em aprimorar a qualidade da percepção de ambiente influenciando diretamente no desempenho de agentes jogadores em ambientes complexos. O grande problema das TDOs é referente ao tempo de processamento demandado para gerar uma representação de ambiente, devido ao uso de recursos computacionais de mais alto custo em relação às representações baseadas em imagens cruas (pelo fato de serem provenientes de RNCs pesadas). Dentre os métodos de aprendizagem considerados, destaca-se que aqueles embasados em AI foram fundamentais para lidar com a propriedade de desconhecido inerente ao ambiente de jogo *FIFA*. Com relação ao método AG-RNC-M, os resultados obtidos mostraram o enorme ganho na construção de RNCs automatizadas em termos do número de parâmetros, o que é interessante no contexto de agentes jogadores que atuam em ambientes complexos em tempo real, diminuindo o espaço necessário para armazenar a RNC gerada e acelerando o tempo de processamento. Além disso, o referido método tem potencial de ser utilizado em diversos problemas que envolvem classificação de imagens.

Apesar dos bons desempenhos obtidos pelos agentes jogadores implementados no presente trabalho, os autores identificaram diversas fragilidades relacionadas a cada abordagem ao fim de cada capítulo. De modo geral, são sugeridas as seguintes técnicas na intenção de produzir melhores resultados:

- ❏ Motivados pela dinâmica bem-sucedida seguida por outras equipes de pesquisa de jogos (tais quais a *DeepMind* e *OpenAI*), os autores pretendem aproveitar a experiência adquirida com o estudo realizado neste trabalho para investigar a combinação de métodos de AI com ARP, o que atenuará as limitações da supervisão nos agentes automáticos.

- ❏ Investigar uma nova forma de representação de ambiente que combine as informações dos *pixels* brutos produzidas pelas imagens de jogo com as informações espaciais dos elementos geradas pelas TDOs a fim de enriquecer ainda mais a percepção de estado de um agente inteligente.

- ❏ Investigar a utilização de redes neurais recorrentes no processo de tomada de decisão em agentes jogadores.

- ❏ Efetuar um estudo acerca de abordagens incrementais de AI (contexto de *on-line learning*) de modo a auxiliar no processamento eficiente das demonstrações produzidas por um supervisor humano.

Por fim, destaca-se que as técnicas aqui investigadas servirão de base para o trabalho de Doutorado a ser iniciado pela equipe desenvolvedora da presente pesquisa de Mestrado, o qual terá como objetivo geral a confecção de um agente autônomo inteligente capaz de extrair informações das imagens do jogo para treinar seu modelo de decisão com base no

perfil de seus adversários em tempo real. Esta abordagem será utilizada em jogos digitais projetados para crianças com distúrbios psicológicos em um contexto de medicina, nos quais os agentes inteligentes que interagem com os jogadores humanos (crianças) adaptam-se às especifidades cognitivas de cada criança, melhorando a sua experiência durante o entretenimento e estimulando o aprimoramento de suas limitações.

Referências

ABBET, I.-R.-I. P. **Mash-Simulator**. 2014. Disponível em: <https://github.com/idiap/mash-simulator>.

AGUIAR, M. A.; JULIA, R. M. da S. Sdm-go: An agent for go with an improved search process based on monte-carlo tree search and sparse distributed memory. **2013 IEEE 16th International Conference on Computational Science and Engineering**, p. 424–431, 2013. Disponível em: <https://doi.org/10.1109/CSE.2013.71>.

ANDERSEN, P.-A.; OLSEN, M. G.; GRANMO, O.-C. Flashrl: A reinforcement learning platform for flash games. **CoRR**, 2018.

ARBIB, M. A. **The Handbook of Brain Theory and Neural Networks**. MIT Press, 2003. Disponível em: <https://doi.org/10.7551/mitpress/3413.001.0001>.

ARULKUMARAN, K. et al. Deep reinforcement learning: A brief survey. **IEEE Signal Processing Magazine**, v. 34, n. 6, p. 26–38, 2017. Disponível em: <https://doi.org/10.1109/MSP.2017.2743240>.

AUMONT, J.; MARIE, M. **Dicionário teórico e crítico de cinema**. Papirus, 2006. Disponível em: <https://books.google.com.br/books?id=aWz42y0RZksC>.

BAIN, M.; SAMMUT, C. A framework for behavioural cloning. In: **Machine Intelligence 15**. [S.l.: s.n.], 1995.

BAKER, B. et al. Designing neural network architectures using reinforcement learning. In: **5th International Conference on Learning Representations, ICLR 2017**. [S.l.: s.n.], 2017.

BARANOWSKI, T. et al. Games for health for children-current status and needed research. **Games for Health Journal**, v. 5, n. 1, p. 1–12, 2016. Disponível em: <https://doi.org/10.1089/g4h.2015.0026>.

BERGER, E. et al. Towards a simulator for imitation learning with kinesthetic bootstrapping. 01 2008.

BHANDARI, D.; MURTHY, C. A. Genetic algorithm with elitist model and its convergence. **IJPRAI**, v. 10, n. 6, p. 731–747, 1996. Disponível em: <https://doi.org/10.1142/S0218001496000438>.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BROCKMAN, G. et al. Openai gym. **CoRR**, 2016.

Buche, C.; Even, C.; Soler, J. Autonomous virtual player in a video game imitating human players: The orion framework. In: **2018 International Conference on Cyberworlds (CW)**. [s.n.], 2018. p. 108–113. Disponível em: <https://doi.org/10.1109/CW.2018.00029>.

CAMPOS, L. mauro Lima de; GARCIA, J. C. P. Redes neurais apoiando a tomada de decisão na análise de crédito bancário e detecção do câncer de mama. **Revista Gestão Tecnologia**, v. 19, n. 1, p. 90–112, 2019. Disponível em: <https://doi.org/10.20397/2177-6652/2019.v19i1.1451>.

CHOWDHURY, G. G. Natural language processing. **Annual Review of Information Science and Technology**, v. 37, n. 1, p. 51–89, 2003. Disponível em: <https://doi.org/10.1002/aris.1440370103>.

COLEMAN, A. M. **A dictionary of psychology**. 3rd. ed. [S.l.]: Oxford University Press Oxford ; New York, 2009. 882 p.

DAVIS, L. **Handbook of Genetic Algorithms**. [S.l.]: Van Nostrand Reinhold, 1991.

DOMINGOS, R. M. M. **Neuroevolucão de um controlador neural e dinâmico para um robô móvel omnidirecional de quatro rodas**. Dissertação (Mestrado) — Universidade Federal de Goiás, 2018.

DOSOVITSKIY, A. et al. CARLA: An open urban driving simulator. In: **Proceedings of the 1st Annual Conference on Robot Learning**. [S.l.]: PMLR, 2017. (Proceedings of Machine Learning Research, v. 78), p. 1–16.

DUNN, O. J. Multiple comparisons using rank sums. **Technometrics**, Taylor Francis, v. 6, n. 3, p. 241–252, 1964. Disponível em: <https://doi.org/10.1080/00401706.1964.10490181>.

FARIA, E. **Teste de Significância Estatística em AM**. 2017. Disponível em: <http://www.facom.ufu.br/~elaine/disc/MFCD/TesteEstatistico.pdf>.

FARIA, M. P. P.; JULIA, R. M. S.; TOMAZ, L. B. P. Proposal of an automatic tool for evaluating the quality of decision-making on checkers player agents. In: **Anais do XV Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2018. p. 389–400. Disponível em: <https://doi.org/10.5753/eniac.2018.4433>.

_____. Deep active imitation learning in fifa free-kicks player platforms based on raw image and object detection state representations. In: **31st International Conference on Tools with Artificial Intelligence**. [s.n.], 2019. Disponível em: <https://doi.org/10.1109/ICTAI.2019.00242>.

_____. A deep reinforcement learn-based fifa agent with naive state representations and portable connection interfaces. In: **The Thirty-Second International Florida Artificial Intelligence Research Society Conference (FLAIRS-32)**. [S.l.: s.n.], 2019. p. 282–285.

_____. Evaluating the performance of the deep active imitation learning algorithm in the dynamic environment of fifa player agents. In: **18th IEEE International Conference on Machine Learning and Applications**. [s.n.], 2019. Disponível em: <https://doi.org/10.1109/ICMLA.2019.00043>.

GAMA, J. et al. **Extração de conhecimento de dados: data mining**. 2. ed. Lisboa: Sílabo, 2015.

GAMING, I. **Esports: An Insight into the Competitive Gaming Industry Part 1**. 2018. Disponível em: <https://medium.com/@IGGalaxy/esports-an-insight-into-the-next-billion-dollar-industry-164ef1434ab7>.

GASTWIRTH, J. L.; GEL, Y. R.; MIAO, W. The impact of levene's test of equality of variances on statistical theory and practice. **Statist. Sci.**, The Institute of Mathematical Statistics, v. 24, n. 3, p. 343–360, 2009. Disponível em: <https://doi.org/10.1214/09-STS301>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Book in preparation for MIT Press. Disponível em: <http://www.deeplearningbook.org>.

GORMAN, B. **Imitation learning through games: theory, implementation and evaluation**. Tese (Doutorado) — Dublin City University, 2009.

GRAYBILL, F. A.; IYER, H. K.; BURDICK, R. K. **Applied Statistics: A First Course in Inference**. [S.l.]: Prentice Hall, 1998.

GU, J. et al. Recent advances in convolutional neural networks. **Pattern Recognition**, v. 77, p. 354 – 377, 2018. Disponível em: <https://doi.org/10.1016/j.patcog.2017.10.013>.

HARMER, J. et al. Imitation learning with concurrent actions in 3d games. In: **2018 IEEE Conference on Computational Intelligence and Games (CIG)**. [s.n.], 2018. p. 1–8. Disponível em: <https://doi.org/10.1109/CIG.2018.8490398>.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. [S.l.]: Prentice Hall, 1999.

HE, K. et al. Deep residual learning for image recognition. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2016. Disponível em: <https://doi.org/10.1109/CVPR.2016.90>.

HEIBERGER, R. M.; NEUWIRTH, E. One-way anova. In: **R Through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics**. New York, NY: Springer New York, 2009. p. 165–191. Disponível em: <https://doi.org/10.1007/978-1-4419-0052-4_7>.

HEMANTH, D.; ESTRELA, V. **Deep Learning for Image Processing Applications**. [S.l.]: IOS Press, 2017. (Advances in Parallel Computing).

HESSEL, M. et al. Rainbow: Combining improvements in deep reinforcement learning. In: **AAAI**. [S.l.: s.n.], 2018.

HIJAZI, S. L.; KUMAR, R.; ROWEN, C. **Using Convolutional Neural Networks for Image Recognition**. 2015. Disponível em: <https://ip.cadence.com/uploads/901/cnn_wp-pdf>.

HOWARD, A. G. et al. **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. 2017.

HUANG, J. et al. Speed/accuracy trade-offs for modern convolutional object detectors. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 3296–3297, 2017. Disponível em: <https://doi.org/10.1109/CVPR.2017.351>.

HUBER, P. J.; RONCHETTI, E. M. **Robust statistics; 2nd ed.** Hoboken, NJ: Wiley, 2009. (Wiley Series in Probability and Statistics). Disponível em: <https://doi.org/10.1002/9780470434697>.

HUSSEIN, A. et al. Deep imitation learning for 3d navigation tasks. **Neural Computing and Applications**, v. 29, n. 7, p. 389–404, Apr 2018. Disponível em: <https://doi.org/10.1007/s00521-017-3241-z>.

_____. Imitation learning: A survey of learning methods. **ACM Comput. Surv.**, v. 50, p. 21:1–21:35, 2017. Disponível em: <https://doi.org/10.1145/3054912>.

IBMCORP. **IBM SPSS Statistics for Windows, Version 20.0**. 2011. Armonk, NY: IBM Corp.

IHRITIK. **MATLAB|RGB image representation**. 2020. Disponível em: <https://www.geeksforgeeks.org/matlab-rgb-image-representation/>.

JUNIOR, E. V.; JULIA, R. M. Btt-go: An agent for go that uses a transposition table to reduce the simulations and the supervision in the monte-carlo tree search. In: **FLAIRS Conference**. [S.l.: s.n.], 2014.

KEMPKA, M.; WYDMUCH, M.; RUNC, G. Vizdoom: A doom-based ai research platform for visual reinforcement learning. **2016 IEEE Conference on Computational Intelligence and Games (CIG)**, 2016. Disponível em: <https://doi.org/10.1109/CIG.2016.7860433>.

KERMANY, D. S. et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. **Cell**, v. 172, n. 5, p. 1122 – 1131.e9, 2018.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2015.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. [S.l.]: Curran Associates, Inc., 2012. p. 1097–1105.

KRUSKAL, W.; WALLIS, W. Use of ranks in one-criterion variance analysis. **Journal of the American Statistical Association**, p. 583–621, 1952. Disponível em: <https://doi.org/10.1080/01621459.1952.10483441>.

LAMPLE, G.; CHAPLOT, D. S. Playing fps games with deep reinforcement learning. In: **Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2017. p. 2140–2146.

LECUN, Y.; BENGIO, Y.; HINTON, G. E. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>.

LIMNIOS, N.; OPRIŞAN, G. **Semi-Markov Processes and Reliability**. Birkhäuser Boston, 2001. Disponível em: <https://doi.org/10.1007/978-1-4612-0161-8>.

LIN, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. **Machine Learning**, v. 8, n. 3, p. 293–321, 1992. Disponível em: <https://doi.org/10.1007/BF00992699>.

LITJENS, G. et al. A survey on deep learning in medical image analysis. **Medical Image Analysis**, v. 42, p. 60 – 88, 2017. Disponível em: <https://doi.org/10.1016/j.media.2017.07.005>.

LIU, H. et al. Hierarchical representations for efficient architecture search. 2018.

LIU, W. et al. Ssd: Single shot multibox detector. In: **Computer Vision – ECCV 2016**. Springer International Publishing, 2016. p. 21–37. Disponível em: <https://doi.org/10.1007/978-3-319-46448-0_2>.

_____. A survey of deep neural network architectures and their applications. **Neurocomputing**, v. 234, p. 11 – 26, 2017. Disponível em: <https://doi.org/10.1016/j.neucom.2016.12.038>.

LIU, Y. et al. Imitation from observation: Learning to imitate behaviors from raw video via context translation. **2018 IEEE International Conference on Robotics and Automation (ICRA)**, p. 1118–1125, 2018. Disponível em: <https://doi.org/10.1109/ICRA.2018.8462901>.

LULIO, L. C. **Técnicas de visão computacional aplicadas ao reconhecimento de cenas naturais e locomoção autônoma em robôs agrícolas móveis**. 2011. Dissertação (Mestrado em Manufatura) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

MANNOR, S.; PELEG, D.; RUBINSTEIN, R. The cross entropy method for classification. In: **Proceedings of the 22Nd International Conference on Machine Learning**. New York, NY, USA: ACM, 2005. (ICML '05), p. 561–568. Disponível em: <https://doi.org/10.1145/1102351.1102422>.

MARTINS, W. et al. Tutoriais inteligentes baseados em aprendizado por reforço: Concepção, implementação e avaliação empírica. **Simpósio Brasileiro de Informática na Educação - SBIE - Mackenzie**, 2007.

MCCULLOCH, W. S.; PITTS, W. **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Cambridge, MA, USA: MIT Press, 1988. 15–27 p.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)**. Berlin, Heidelberg: Springer-Verlag, 1996.

MILLER, B. L. et al. Genetic algorithms, tournament selection, and the effects of noise. **Complex Systems**, v. 9, p. 193–212, 1995.

MNIH, V. et al. Asynchronous methods for deep reinforcement learning. In: **ICML**. [S.l.: s.n.], 2016.

_____. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529–533, 2015. Disponível em: <https://doi.org/10.1038/nature14236>.

MOUSAVI, S. S.; SCHUKAT, M.; HOWLEY, E. Deep reinforcement learning: An overview. In: BI, Y.; KAPOOR, S.; BHATIA, R. (Ed.). **Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016**. Cham: Springer International Publishing, 2018. p. 426–440. Disponível em: <https://doi.org/10.1007/978-3-319-56991-8_32>.

_____. Deep reinforcement learning: An overview. **CoRR**, 2018. Disponível em: <https://arxiv.org/abs/1806.08894>.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. [S.l.]: Omnipress, 2010. p. 807–814.

NETO, H. C. **Ls-draughts: um sistema de aprendizagem de jogos de damas baseado em algoritmos genéticos, redes neurais e diferenças temporais**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2007.

NETO, H. C.; JULIA, R. M. S. Ace-rl-checkers: decision-making adaptability through integration of automatic case elicitation, reinforcement learning, and sequential pattern mining. **Knowledge and Information Systems**, v. 57, n. 3, p. 603–634, Dec 2018. Disponível em: <https://doi.org/10.1007/s10115-018-1175-0>.

OPENAI. **More on Dota 2**. 2017. Disponível em: <https://openai.com/blog/more-on-dota-2/>.

_____. **OpenAI Five**. 2018. Disponível em: <https://openai.com/blog/openai-five/>.

ORTEGA, J. et al. Imitating human playing styles in super mario bros. **Entertainment Computing**, v. 4, n. 2, p. 93 – 104, 2013. Disponível em: <https://doi.org/10.1016/j.entcom.2012.10.001>.

PENHA, D. de P. **Rede neural convolucional aplicada à identificação de equipamentos residenciais para sistemas de monitoramento não-intrusivo de carga**. Dissertação (Mestrado) — Universidade Federal do Pará, Instituto de Tecnologia, 2018.

PICCA, D.; JACCARD, D.; EBERLÉ, G. Natural language processing in serious games: A state of the art. **International Journal of Serious Games**, v. 2, n. 3, 2015. Disponível em: <https://doi.org/10.17083/ijsg.v2i3.87>.

PONTI, M. A.; COSTA, G. B. P. da. **Como funciona o Deep Learning**. 2018. Disponível em: <https://arxiv.org/abs/1806.07908>.

REAL, E. et al. Large-scale evolution of image classifiers. In: **Proceedings of the 34th International Conference on Machine Learning - Volume 70**. [S.l.: s.n.], 2017. p. 2902–2911.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: LIU, L.; ÖZSU, M. T. (Ed.). **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 532–538.

ROSS, S.; BAGNELL, D. Efficient reductions for imitation learning. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. [S.l.]: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 661–668.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <https://doi.org/10.1038/323533a0>.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial - 3ª Ed**. [S.l.]: Elsevier, 2013.

SALKIND, N. J. Tukey's honestly significant difference (hsd). **Encyclopedia of research design**, 2010. Thousand Oaks, CA: SAGE Publications.

SAMMUT, C. et al. Learning to fly. In: **Proceedings of the Ninth International Workshop on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992. (ML92), p. 385–393. Disponível em: <https://doi.org/10.1016/B978-1-55860-247-2.50055-3>.

SAMMUT, C.; WEBB, G. I. Holdout evaluation. In: **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 506–507. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_369>.

SANDLER, M. et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [s.n.], 2018. p. 4510–4520. Disponível em: <https://doi.org/10.1109/CVPR.2018.00474>.

SANTOS, G.; JULIA, R. M. Go-ahead: Improving prior knowledge heuristics by using information retrieved from play out simulations. **Florida Artificial Intelligence Research Society Conference**, 2016.

SCHAAL, S. **Is imitation learning the route to humanoid robots?** 1999. Disponível em: <https://doi.org/10.1016/S1364-6613(99)01327-3>.

SCHULMAN, J. et al. Proximal policy optimization algorithms. **CoRR**, 2017. Disponível em: <http://arxiv.org/abs/1707.06347>.

SEBASTIAN, S. **The Digital Games Industry and its Direct and Indirect Impact on the Economy**. 2017. (Bachelor's thesis) - Degree Programme in International Business and Logistics.

SEO, Y. Electronic sports: A new marketing landscape of the experience economy. **Journal of Marketing Management**, Routledge, v. 29, n. 13-14, p. 1542–1560, 2013. Disponível em: <https://doi.org/10.1080/0267257X.2013.822906>.

SHETH, A. et al. Cognitive services and intelligent chatbots: Current perspectives and special issue introduction. **IEEE Internet Computing**, IEEE, p. 6–12, 2019. Disponível em: <https://doi.org/10.1109/MIC.2018.2889231>.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para engenhearia e ciências aplicadas**. [S.l.]: Artliber, 2010.

SILVER, D.; BAGNELL, J. A. D.; STENTZ, A. T. High performance outdoor navigation from overhead data using imitation learning. **Proceedings of Robotics Science and Systems**, 2008. Disponível em: <https://doi.org/10.15607/RSS.2008.IV.034>.

SILVER, D. et al. Mastering the game of Go with deep neural networks and tree search. **Nature**, Nature Publishing Group, v. 529, n. 7587, p. 484–489, jan. 2016. Disponível em: <https://doi.org/10.1038/nature16961>.

_____. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. **CoRR**, 2017.

_____. Mastering the game of go without human knowledge. **Nature**, Macmillan Publishers Limited, part of Springer Nature. All rights reserved., v. 550, p. 354–, out. 2017. Disponível em: <https://doi.org/10.1038/nature24270>.

Srinivas, M.; Patnaik, L. M. Genetic algorithms: a survey. **Computer**, v. 27, n. 6, p. 17–26, 1994. Disponível em: <https://doi.org/10.1109/2.294849>.

SUN, Y. et al. **Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification**. 2018.

_____. Completely automated cnn architecture design based on blocks. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–13, 2019. Disponível em: <https://doi.org/10.1109/TNNLS.2019.2919608>.

SUTTON, R. S.; BARTO, A. G. **Introduction to Reinforcement Learning**. 1st. ed. Cambridge, MA, USA: MIT Press, 1998.

SYNCED. **Humans Call GG! OpenAI Five Bots Beat Top Pros OG in Dota 2**. 2019. Disponível em: <https://medium.com/syncedreview/humans-call-gg-openai-five-bots-beat-top-pros-og-in-dota-2-8508e59b8fd5>.

TOMAZ, L. B. P. **D-MA-Draughts: um sistema multiagente jogador de damas automático que atua em um ambiente de alto desempenho**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2013.

TOMAZ, L. B. P. **ADABA: a new Alpha-Beta distribution approach - application to the Checkers game domain**. Tese (Doutorado) — Universidade Federal de Uberlândia, 2019.

TOMAZ, L. B. P.; FARIA, M. P. P.; JULIA, R. M. S. Aphid-draughts: Comparing the synchronous and asynchronous parallelism approaches for the alpha-beta algorithm applied to checkers. In: **2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)**. [s.n.], 2017. p. 1243–1250. Disponível em: <https://doi.org/10.1109/ICTAI.2017.00188>.

TOMAZ, L. B. P.; JULIA, R. M. S.; DUARTE, V. A. A multiagent player system composed by expert agents in specific game stages operating in high performance environment. **Applied Intelligence**, v. 48, n. 1, p. 1–22, 2018. Disponível em: <https://doi.org/10.1007/s10489-017-0952-x>.

TORRADO, R. R.; BONTRAGER, P.; TOGELIUS, J. Deep reinforcement learning for general video game ai. **CoRR**, 2018. Disponível em: <https://arxiv.org/abs/1806.02448>.

TRIVEDI, C. **Building a Deep Neural Network to play FIFA 18**. 2018. Disponível em: <https://towardsdatascience.com/building-a-deep-neural-network-to-play-fifa-18-dce54d45e675>.

_____. **Using Deep Q-Learning in FIFA 18 to perfect the art of free-kicks**. 2018. Disponível em: <https://towardsdatascience.com/using-deep-q-learning-in-fifa-18-to-perfect-the-art-of-free-kicks-f2e4e979ee66>.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestre. **Proceedings of the XXIX Conference on Graphics, Patterns and Images**, p. 1 – 4, 2016.

VENKATESH, A. **Object Tracking in Games using Convolutional Neural Networks**. Dissertação (Mestrado) — California Polytechnic State University, San Luis Obispo, 2018.

VINYALS, O. et al. **AlphaStar: Mastering the Real-Time Strategy Game StarCraft II**. 2019. Disponível em: <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. Tese (Doutorado) — King's College, Oxford, 1989.

WATTANASOONTORN, V. et al. Serious games for health. **Entertainment Computing**, v. 4, n. 4, p. 231 – 247, 2013. Disponível em: <https://doi.org/10.1016/j.entcom.2013.09.002>.

WIJMAN, T. **The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market**. 2019. Disponível em: <https://newzoo.com/insights>.

WILCOXON, F. Individual comparisons by ranking methods. In: _____. **Breakthroughs in Statistics: Methodology and Distribution**. Springer New York, 1992. p. 196–202. Disponível em: <https://doi.org/10.1007/978-1-4612-4380-9_16>.

WYMANN, B. et al. **TORCS: The open racing car simulator**. 2015.

Xie, L.; Yuille, A. Genetic cnn. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. [s.n.], 2017. p. 1388–1397. Disponível em: <https://doi.org/10.1109/ICCV.2017.154>.

YANNAKAKIS, G. N.; TOGELIUS, J. **Artificial Intelligence and Games**. 1st. ed. Springer Publishing Company, Incorporated, 2018. Disponível em: <https://doi.org/10.1007/978-3-319-63519-4>.

YAO, Y. et al. **Adversarial Language Games for Advanced Natural Language Intelligence**. 2019. Disponível em: <https://arxiv.org/abs/1911.01622>.

ZHAO, Y. et al. **Winning Isn't Everything: Enhancing Game Development with Intelligent Agents**. 2019. Disponível em: <https://arxiv.org/abs/1903.10545>.

ZHAO, Z.-Q. et al. **Object Detection with Deep Learning: A Review**. 2018. Disponível em: <https://arxiv.org/abs/1807.05511>.

ZOPH, B.; LE, Q. V. Neural architecture search with reinforcement learning. In: **Proceedings of the 2017 International Conference on Learning Representations**. [S.l.: s.n.], 2017.