UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Jonatan Washington Pereira da Silva

Um Currículo de Aprendizado por Reforço para o Cenário "*Run to Score with Keeper*" do *Google Research Football Environment*

Recife

2019

Jonatan Washington Pereira da Silva

Um Currículo de Aprendizado por Reforço para o Cenário "*Run to Score with Keeper*" do *Google Research Football Environment*

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciências da Computação

Orientador: Pablo Azevedo Sampaio

Coorientador: Valmir Macario Filho

Recife

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586c Silva, Jonatan Washington Pereira
Um currículo de Aprendizado por Reforço para o cenário “Run to Score with Keeper” do Google Research Football environment / Jonatan Washington Pereira Silva. - 2019.
51 f. : il.

Orientador: Pablo Azevedo Sampaio.
Coorientador: Valmir Macario Filho.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2019.

1. Processo de decisão de Markov. 2. Aprendizado por reforço. 3. Aprendizado por currículo. 4. Algoritmo de otimização da política proximal. I. Sampaio, Pablo Azevedo, orient. II. Filho, Valmir Macario, coorient. III. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por **JONATAN WASHINGTON PEREIRA DA SILVA** às 15:30 do dia 10 de dezembro de 2019, no Auditório do Departamento de Computação - DC – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado " **Um Currículo de Aprendizado por Reforço para o Cenário "Run to Score with Keeper" do Google Research Football Environment**", orientado por Pablo Azevedo Sampaio e aprovado pela seguinte banca examinadora:

Pablo Azevedo Sampaio
DC/UFRPE

Valmir Macário Filho
DC/UFRPE

George Gomes Cabral
DC/UFRPE

Dedico este trabalho primeiramente à Deus, meu refúgio e fortaleza, e a todos que contribuíram de forma direta ou indireta para a conclusão desta etapa em minha vida, principalmente a minha família e professores, que sempre me forneceram apoio e orientação.

Agradecimentos

Agradeço primeiramente à Deus, pois sem Ele nada sou, aos meus pais que muito esforçaram-se para me oferecer uma educação de qualidade, a minha esposa e filha que me deram força para concluir esta jornada, meus orientadores Pablo Sampaio e Valmir Macário pelo apoio e direcionamento, todos os professores, desde pré-escola até o ensino superior, que contribuíram para o meu desenvolvimento e a todos familiares e amigos que auxiliaram na minha jornada.

“Confie no Senhor de todo o seu coração e não se apoie em seu próprio entendimento”
(Bíblia Sagrada)

Resumo

O aprendizado por reforço é um conjunto de técnicas que permitem a um agente interagir com um determinado ambiente. Os agentes observam o estado do ambiente e executam uma ação, a ação é avaliada por meio de uma recompensa obtida. O agente tem como objetivo maximizar esta recompensa. Diversas questões como: locomoção em três dimensões e jogos eletrônicos foram abordados pelo aprendizado por reforço (KURACH et al., 2019). O treinamento de agentes para um jogo de futebol normalmente possui recompensas esparsas, o que retarda o aprendizado (MATIISEN et al., 2019). Uma técnica que pode contornar este obstaculo é o aprendizado por currículo proposto em (BENGIO et al., 2009). O aprendizado por currículo é uma técnica que aborda sub-tarefas mais simples da tarefa principal e aumenta gradativamente o nível de dificuldade ao longo do tempo. Neste trabalho apresentamos dois currículos, identificados como: 5-15-30-50 e 3-10-20-67, para o cenário *Run to Score with Keeper* da *Football Academy*. Mostramos que os currículos, em média, obtiveram melhores resultados se comparados ao treinamento apenas no cenário principal, sem currículo. O currículo 3-10-20-67 obteve um melhor resultado mesmo considerando o desvio padrão.

Palavras-chave: Processo de decisão de Markov, MDP, Aprendizado por reforço, Aprendizado por currículo, Algoritmo de otimização da política proximal, PPO.

Abstract

Reinforcement learning is a group of techniques that allow an agent to interact with a particular environment. Agents observe the state of the environment and perform an action, the action is evaluated through a reward obtained. The agent objective is to maximize this reward. Various issues such as three-dimensional locomotion and electronic games have been addressed by reinforcement learning (KURACH et al., 2019). The Trainament of agents for a soccer game usually has sparse rewards, what slows learning (MATIISEN et al., 2019). One technique that can solve this obstacle is the curriculum learning proposed in (BENGIO et al., 2009). This technique use simplest tasks of the main task and the increase difficult level with the time. In This work we present two curriculum, identified as 5-15-30-50 e 3-10-20-67, for the scenario Run To Score With Keeper of Football Academy. We have shown that curriculums on average achieved better results compared to training only in the main scenario, without curriculum. Curriculum 3-10-20-67 achieved a better result even considering the pattern deviation.

Keywords: Markov decison process, MDP, Reinforcement Learning, Curriculum Learning, Proximal Policy Optimization Algorithm, PPO.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MDP	*Markov decison process*, processo de decisão de Markov
PPO	*Proximal Policy Optimization Algorithm*, algoritmo de Otimização da Politica Proximal
CNN	*Convolutional Neural Network* , Rede Neural Convolucional

Sumário

1 Introdução

1.1 Problema de pesquisa

O futebol simulado é uma tarefa complexa, dinâmica, colaborativa e competitiva (RABIEE; GHASEM-AGHAEE, 2004; FARAHNAKIAN; MOZAYANI, 2009). O aprendizado por reforço é muito usado em tarefas de futebol simulado, pois consegue obter um resultado significativo ao tratar com a complexidade inerente dessas tarefas (FARAHNAKIAN; MOZAYANI, 2009). O aprendizado por reforço engloba um conjunto de técnicas que permitem um agente escolher ações e interagir com o ambiente, a fim de maximizar as recompensas totais recebidas pelas execução das ações (SUTTON; BARTO, 2011). Geralmente no aprendizado por reforço, a tarefa é modelada como um MDP, processo de decisão de Markov. Um MDP fornece uma estrutura matemática capaz de modelar o processo de tomada de decisão em situações que os resultados são parcialmente aleatórios (WHITE, 2001). Uma função que mapeia cada estado em uma ação é denominado de política. Uma família de algoritmos com importantes resultados recentes é a *Policy Gradient* (SCHULMAN et al., 2017). Neles, são usados redes neurais para representar a politica (SUTTON et al., 2000). Um algorítimo desta família que obteve bons resultados é o PPO (SCHULMAN et al., 2017), Algoritmo de Otimização da Politica Proximal.

O treinamento de agentes para um jogo de futebol completo pode ser muito desafiador (KURACH et al., 2019). Para que um time consiga realizar um gol, pode requerer uma grande quantidade de ”ações”, tornando a recompensa recebida pelo agente esparsa, o que retarda o aprendizado (MATIISEN et al., 2019). Uma solução para este problema é o aprendizado por currículo (ZAREMBA; SUTSKEVER, 2014), onde o aprendizado começa com atividades mais simples, aprendendo aspectos mais fáceis da tarefa ou sub-tarefas menores. Quando o conhecimento evolui, gradualmente é aumentado o grau de dificuldade ou são utilizadas novas sub-tarefas (BENGIO et al., 2009). Neste trabalho nós apresentamos dois currículos para o cenário *Run To Score With Keeper* da *Football Academy* do ambiente *Google Research Football*.

1.2 Justificativa

O aprendizado por reforço vem tendo um rápido progresso (KURACH et al., 2019), com aplicação em diversas áreas, principalmente em jogos, obtendo resultados sobre-humanos. Abaixo temos uma listas de trabalhos que utilizam o aprendizado por reforço:

1. Carros autônomos: em (BANSAL; KRIZHEVSKY; OGALE, 2018), vemos a aplicação de aprendizado por reforço para treinamento de uma politica de direção autônoma, robusta o suficiente para conduzir um veículo real.

2. Gerenciamento de recursos em clusters de computadores: projetar algoritmos para alocar recursos limitados a diferentes tarefas é uma tarefa desafiadora e requer heurísticas geradas por humanos. Em (MAO et al., 2016) encontramos uma maneira de como usar o aprendizado por reforço para aprender automaticamente a alocar e programar recursos de computador para trabalhos em espera, com o objetivo de minimizar a desaceleração média do trabalho.

3. Controle de semáforo: em (AREL et al., 2010), os pesquisadores projetaram um controlador de semáforo para resolver o problema de congestionamento. Testados apenas em ambiente simulado, seus métodos mostraram resultados superiores aos métodos tradicionais e lançaram uma luz sobre os possíveis usos da aprendizado por reforço de múltiplos agentes no projeto de sistemas de tráfego.

4. Robótica: há um tremendo trabalho na aplicação de aprendizado por reforço em robótica. Em (KOBER; BAGNELL; PETERS, 2013) encontramos exemplos da aplicação deste método de aprendizado de máquina em robótica. Especialmente em (LEVINE et al., 2016), os autores usam o aprendizado por reforço para o treinamento conjunto dos sistemas de percepção e controle do robô. Para este fim, foi desenvolvido um método que pode ser usado para aprender políticas que mapeiam observações brutas de imagens diretamente dos torques nos motores do robô. As imagens RGB foram entrada de uma rede neural convolucional profunda e as saídas foram os torques do motor.

5. Configuração do Sistema Web: existem muitos parâmetros configuráveis em um sistema da Web, e o processo de ajuste dos parâmetros requer um amplo conhecimento e vários testes de rastreamento e erro. Em (BU; RAO; XU, 2009) mostra o uso de aprendizado por reforço na configuração e reconfiguração autônoma de parâmetros aplicada a sistemas web multicamada em ambientes dinâmicos baseados em máquinas virtuais.

6. Química: o aprendizado por reforço também pode ser aplicado na otimização de reações químicas. Em (ZHOU; LI; ZARE, 2017) apresentaram um modelo que superou um algoritmo de última geração na otimização de reações químicas também generalizou para mecanismos subjacentes diferentes.

7. Recomendações personalizadas: em (ZHENG et al., 2018) foi aplicado aprendizado por reforço em um sistema de recomendação de notícias.

8. Lances e publicidade: com o grande sucesso do comercio eletrônico e o avanço da tecnologia, a publicidade em tempo real vem ganhando espaço. Pesquisadores do grupo Alibaba publicaram o artigo (JIN et al., 2018), e nele afirmaram que sua solução distribuída de múltiplos agentes baseada em cluster (DCMAB) alcançou resultados promissores no gerenciamento de publicidade em tempo real, de tal forma que planejam realizar um teste online na plataforma Taobao (OU; DAVISON, 2009).

9. Jogos: o aprendizado por reforço é tão conhecido atualmente devido o seu excelente resultado em jogos, chegando, as vezes, alcançar um desempenho superior ao humano. O mais famoso é o AlphaGo (SILVER et al., 2016) e o AlphaGo Zero (SINGH; OKUN; JACKSON, 2017). O AlphaGo, foi treinado com inúmeros jogos humanos, já alcançou um ótimo desempenho. No entanto, posteriormente os pesquisadores usaram uma abordagem mais pura da aprendizagem por reforço, treinando a inteligencia artificial, sem utilizar dados de partidas humanas como base. Os pesquisadores deixaram o novo agente, o AlphaGo Zero, jogando com o AlphaGo até finalmente derrotar-lo por 100-0. Em (MNIH et al., 2013) é apresentado um modelo de aprendizado por reforço profundo para aprender políticas de controle para sete jogos do Atari 2600 do *Arcade Learning Environment* (BELLEMARE et al., 2013), o modelo utilizado é uma rede neural convolucional, treinada com uma variante do Q-learning, cuja entrada são os pixels provinientes de um pré-processamento dos quadros brutos, primeiro são convertidos para escala de cinza e amostragem reduzida de 210 × 160 pixels para uma imagem de 110 × 84 e a saída é uma função de valor futuro estimando. Os artigos (HAUSKNECHT; STONE, 2015) e (MNIH et al., 2015) possuem resultados melhores para o treinamento do jogo Atari 2600. Até mesmo Jogos profissionais como Dota 2 (ANDERSEN; GOODWIN; GRANMO, 2018) e Starcraft II (VINYALS et al., 2017) possuem aplicações de aprendizado por reforço.

No artigo (XIONG et al., 2008) é apresentado um novo algoritmo de aprendizado por reforço, que é uma junção dos algoritmos Q-learning, Nash-Q, CE-Q e WoLF-PHC. Esse algoritmo é utilizado para treinar a troca de passes entre os jogadores, onde a recompensa é baseada no destino final da bola, quando a bola chega ao companheiro de equipe a recompensa é máxima, e quando o adversário fica com a bola há uma recompensa negativa (punição).

Em (AKIYAMA et al., 2016) é utilizado o código aberto da equipe Helios (NERI et al., 2012) como base para o seu time. Foi modificado o comportamento do ataque em duas áreas específicas do campo mostradas na Figura 1, para a utilização de aprendizagem por reforço usando o algoritmo Q-learning. São mapeados 7 estados e 7 ações em duas matrizes de aprendizado, uma para cada área do ataque, onde as matrizes fo-

ram inicializadas com 0 e esses valores foram atualizados ao final de cada jogo treino. Foram realizados 10 jogos de treino e 10 jogos de teste com três equipes. Os resultados das somas dos placares contra as 3 equipes foram positivos, e uma delas foi a Wrigth Eagle time campeão mundial em 2011.

Figura 1 – Áreas de ataque, fonte: (NERI et al., 2012)

Nos artigos (FARAHNAKIAN; MOZAYANI, 2009) e (RABIEE; GHASEM-AGHAEE, 2004) foram usados o algoritmo Q-learning para treinar o chute ao gol, o código da equipe é Uva Trilearn (KOK et al., 2003) é usado como base para o novo time. A diferença entre os trabalhos são os parâmetros utilizados no aprendizado: em Fahimeh Farahnakian e Nasser Mozayan é utilizado a distância entre a bola e o goleiro e a probabilidade que é obtida da pesquisa da equipe da Uva, já em Zam Rabiee e Nasser Ghasem-Aghaee é usado o ângulo formado entre o corpo e pescoço do goleiro.

Em (FABRO; REIS; LAU, 2014) é utilizado o algoritmo de aprendizado por reforço Q-learning para selecionar a melhor ação que pode ser executada por um jogador. As ações são provenientes do *Setplay Framework*, que é um conjunto predefinido de ações coordenadas e colaborativas. Os resultados se mostraram satisfatórios contra equipes intermediárias, mas não alcançou um bom rendimento contra as equipes mais bem colocadas no campeonato de 2013.

O artigo (YOON; BEKKER; KROON, 2017) propõe um novo algoritmo de aprendizado por reforço, que é utilizado na tomada de decisão de robôs da RoboCup Small Size League (SSL) com a finalidade de melhorar o chute ao gol. O algoritmo proposto obteve um bom desempenho sob condições especificas.

Não foram encontrados na literatura trabalhos que apresentam um currículo para *Football Academy* do ambiente *Google Research Football*. Possivelmente pelo fato do ambiente ser muito recente (2019). A utilização do aprendizado por currículo

pode acelerar a obtenção de conhecimento pelo agente no ambiente *Google Research Football*, e assim conseguir resultados melhores e mais rápidos do que os apresentados em (KURACH et al., 2019).

1.3 Objetivo geral

Desenvolver um currículo composto por cenários da *Football Academy* e capaz de acelerar o desenvolvimento do conhecimento em um algoritmo de aprendizado por reforço aplicado a um jogo de futebol do ambiente *Google Research Football*.

1.4 Organização do trabalho

O próximo capitulo possui a fundamentação teórica deste trabalho, contendo uma sucinta explicação de conceitos importantes para compreensão do objetivo do trabalho. No capitulo 3 será apresentado o currículo proposto, exibindo o número de passos utilizados no treinamento dos cenários selecionados da *Football Academy*. No quarto capitulo são expostos o os resultados obtidos por todos os currículos experimentados e comparados com o resultado do treinamento sem currículo. O último capitulo dispõe das conclusões deste trabalho e potenciais trabalhos futuros.

2 Fundamentação teórica

Nesse capitulo veremos os seguintes subtópicos: 1. MDP; 2. Aprendizado por reforço; 3. Redes Neurais; 4. Algoritmos *Policy Gradient*; 5. Algoritmo PPO; 6. Aprendizado por currículo; 7. Ambientes benchmark de futebol; 8. *Football Academy*; 9. Comentários.

2.1 MDP

Um processo de decisão de Markov (MDP) baseia-se nas noções de **estado**, que representa a situação atual do agente em um ambiente, a **ação** que é um evento que afeta a dinâmica do processo e uma **recompensa** observada para cada transição entre estados a partir da execução de uma ação. O objetivo do agente é maximizar sua recompensa total descontada ao longo do tempo (GARCIA; RACHELSON, 2013; PUTERMAN, 1990; BERNSTEIN et al., 2002). A solução de um MDP implica em procurar uma política π*, que produz uma recompensa máxima. Na Figura 2 vemos um exemplo de um MDP composto por dois estados, S_1 e S_2. O S_1 possui duas ações possíveis $a_{1,2}$ e $a_{1,1}$, executando $a_{1,2}$ o agente tem 100% de chance de ir para o estado S_2 e receber 10 como recompensa, já executando $a_{1,1}$ possui 50% de chance de ir para S_2 e 50% de continuar em S_1, nos dois casos o agente recebe 5 de recompensa.

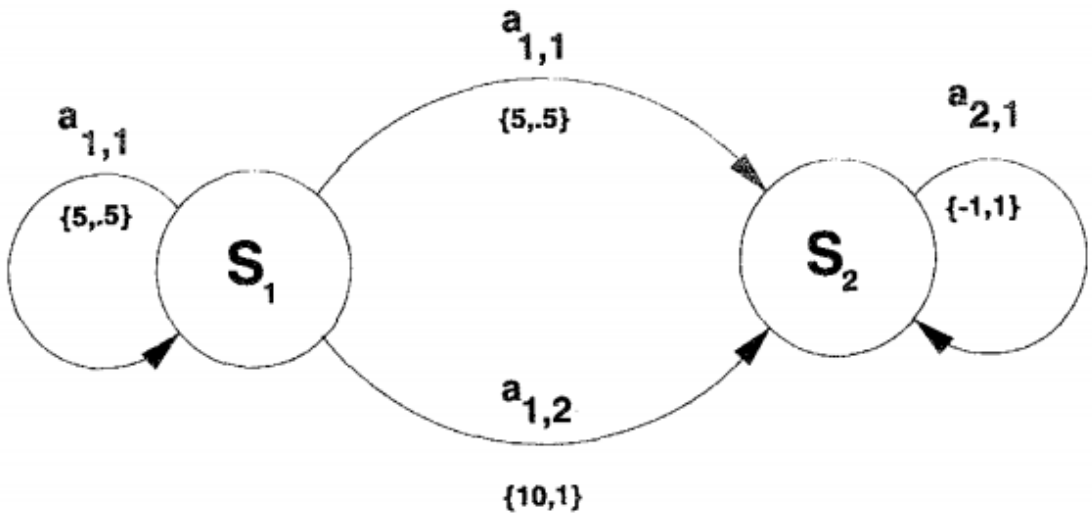

Figura 2 – Representação simbólica de um MDP
(PUTERMAN, 1990)

Sendo s o estado atual, a uma ação do estado s e r a recompensa obtida pela execução da política π. Temos que para uma determinada política, podemos definir duas funções:

1. Valor do estado: $V_\pi(s)$;
2. Valor da ação: $Q_\pi(s, a)$;

Podemos determinar o valor de um estado s_t como :
$V_\pi(s_t) = E(G_t)$, onde: $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n$, que é o retorno futuro descontado por γ a partir de s_t aplicando as ações da política π, γ é a constante de desconto das recompensas futuras, que pode variar entre maior que 0 e menor que 1. $E(X) = \sum p_i \times x_i$, onde X é uma variável aleatória discreta, e p_i é a probabilidade de retornar o valor x_i. Na equação de Bellman (GIVAN; PARR, 2001) temos recursivamente que: se s_t for um estado terminal: $V_\pi(s_t) = 0$, se não: $V_\pi(s_t) = E(r_t + \gamma V(s_{t+1}))$.

2.2 Aprendizado por reforço

Aprendizado por Reforço é aprender quais ações deve-se realizar, como mapear estados para ações, de maneira que venha maximizar o valor de uma recompensa numérica. Diferente de outras formas de aprendizado de máquina, o aprendiz, ou agente, não é informado sobre quais ações ele deve executar, mas deve aprender como obter uma maior recompensa, experimentalmente (SUTTON; BARTO, 2011). Um agente de aprendizado por reforço aprende interagindo dinamicamente com seu ambiente (SUTTON; BARTO, 2011; KAELBLING; LITTMAN; MOORE, 1996; SEN; WEISS, 1999), o Aprendizado por Reforço é um método geral (abrange vários problemas), e que geralmente possui a tarefa modelada como um MDP.

O ambiente e o agente são os elementos principais de um sistema de Aprendizado por Reforço, mas também podemos identificar outros sub-elementos importantes, que são: política, função de recompensa, função de valor e um modelo do ambiente (estado). Um estado é um modelo do ambiente que é percebido pelo agente. A função de recompensa define qual a recompensa de uma ação em um determinado estado dado um objetivo (SUTTON; BARTO, 2011; KITANO, 1998).

Outro conceito importante é o de *exploration* e *exploitation* (NERI et al., 2012; RABIEE; GHASEM-AGHAEE, 2004; SUTTON; BARTO, 2011): *exploration* é exploração no sentido de reconhecimento sondagem; já *exploitation* é exploração como o sentido de aproveitamento, obtenção de vantagem. Isto é, quando o agente realiza *exploration*, ele executa ações não realizadas antes para descobrir novos caminhos que podem ser melhores do que os já realizados; e na *exploitation* ele executa as ações com maior recompensa conhecida (SUTTON; BARTO, 2011; RABIEE; GHASEM-AGHAEE, 2004). Com isso observamos que quando a política já foi devidamente treinada, o ideal é que a taxa de *exploitation* seja maior que a de *exploration*.

2.3 Redes Neurais

Redes neurais são um conjunto de algoritmos, que baseiam-se na maneira que os neurônios funcionam (KRÖSE et al., 1993). Primeiro, uma coleção de ”neurônios”de software é criada e conectada, permitindo que eles enviem mensagens uns para os outros. Em seguida, solicita-se à rede que resolva um problema, ela tenta resolver repetidamente, sempre fortalecendo as conexões que levam ao sucesso e diminuindo as que levam ao fracasso. Projetados para reconhecer padrões, as redes neurais interpretam os dados através de um tipo de percepção da máquina, rotulando ou agrupando dados brutos. Os dados de entrada são numéricos, geralmente contidos em vetores, nos quais todos os dados do mundo real, sejam imagens, sons, textos e etc, podem ser convertidos.

2.3.1 Redes neurais convolucionais

Uma Rede Neural Convolucional é um algoritmo de aprendizado que pode receber uma imagem de entrada, e extrair características importantes dela, além de ser capaz de diferenciar estas características umas das outras. O pré-processamento necessário em uma CNN é muito menor e o resultado mais significativo em comparação com outros algoritmos de classificação (LECUN et al., 1998). Enquanto outros métodos, como *Perceptron*, os filtros são projetados manualmente, com treinamento suficiente, as CNN têm a capacidade de aprender esses filtros.

CNNs obtiveram resultados significativos em várias áreas da Inteligência Artificial, destacam-se: Visão Computacional e Processamento de Linguagem Natural (KIM, 2014). A arquitetura de uma CNN foi inspirada pela organização do córtex visual (RAWAT; WANG, 2017), onde neurônios individuais respondem a estímulos apenas em uma região restrita do campo visual. Uma coleção desses campos se sobrepõem para cobrir toda a área da visual. O termo convolucional denota a aplicação de uma operação matemática denominada convolução, que é um operador linear que será detalhado na próxima sub-seção.

2.3.1.1 Camada de Convolução

As camadas convolucionais servem como filtros de características e, assim, aprendem as representações de características da imagem de entrada. Os neurônios nas camadas convolucionais são organizados em mapas de características (RAWAT; WANG, 2017). Os filtros, mapas e entradas são matrizes, sendo o mapa resultado de um cálculo do filtro sobre a entrada. Inicialmente o filtro (*Kernel*) recebe valores aleatórios, no treinamento da CNN os seus valores são ajustados para valores otimizados, que consigam gerar mapas de características.

Na Figura 3, a imagem de entrada é representada pela matriz 3 x 3 e o *Kernel* representado pela matriz 2 x 2, geralmente o *Kernel* é uma matriz menor, pois a convolução permite reduzir a matriz que representa a entrada, assim reduzindo os parâmetros a cada camada. É realizada a primeira etapa da convolução, o filtro passa por uma parte da matriz de entrada, com a mesma dimensão. Onde cada elemento sobreposto é multiplicado pelo elemento na mesma linha e coluna do *Kernel*. Em seguida é feita a soma resultando no primeiro elemento da matriz de características que está sendo criada.

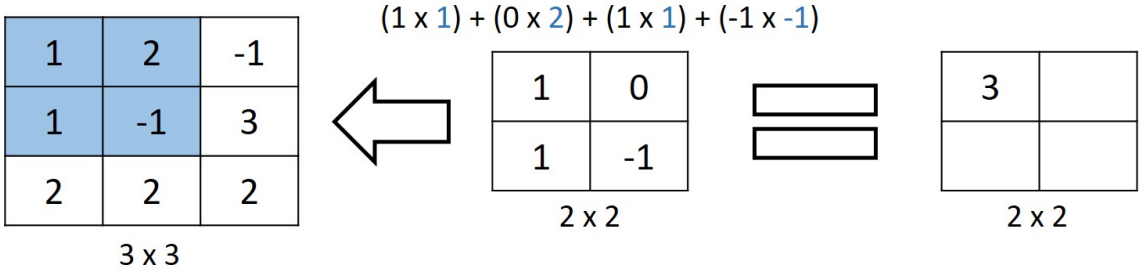

Figura 3 – Passo 1 do exemplo de convolução. Fonte: Autor

A próxima execução do filtro vai depender do tamanho do passo que vai ser dado da esquerda para direita, dependendo do tamanho das matrizes esse tamanho é variável. Nesse exemplo, o tamanho do passo, ou *Stride*, vai ser igual a 1, um dos efeitos de sua escolha é o tempo de processamento da convolução que pode aumentar ou diminuir. Nas Figuras 2, 3 e 4, são mostradas as etapas seguintes dessa convolução, elas também podem ocorrer em paralelo, movendo o *Kernel* pela imagem e realizando as operações formando a matriz resultante à direita. Na realização da convolução, é possível que seja perdida as margens da imagem original, o tratamento a ser dado para isso pode ser simplesmente ignorar, zerar ou replicar os seus valores entre outros.

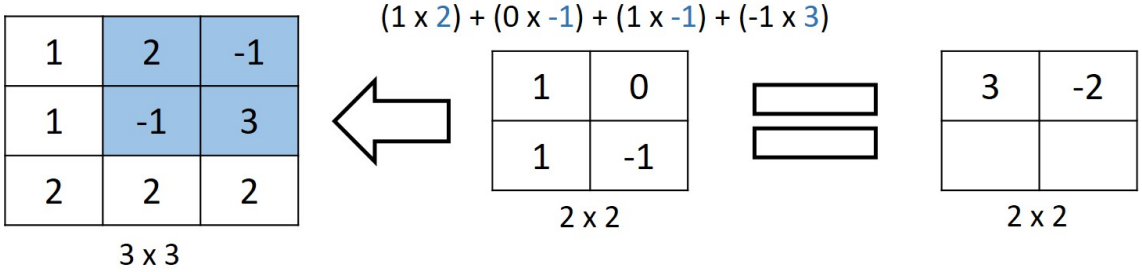

Figura 4 – Passo 2 do exemplo de convolução. Fonte: Autor

Neste exemplo conjecturado, é gerada uma matriz 2 x 2, essa matriz é chamada de mapa de característica, ou *Feature map*. Em uma convolução o número de mapas de características gerados é especificado, ou seja podem ser geradas diferentes mapas de características com base no seu respectivo *Kernel*. No exemplo mostrado a

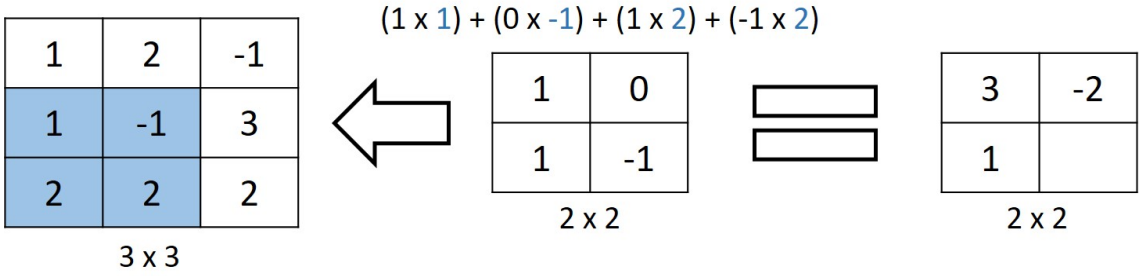

Figura 5 – Passo 3 do exemplo de convolução. Fonte: Autor

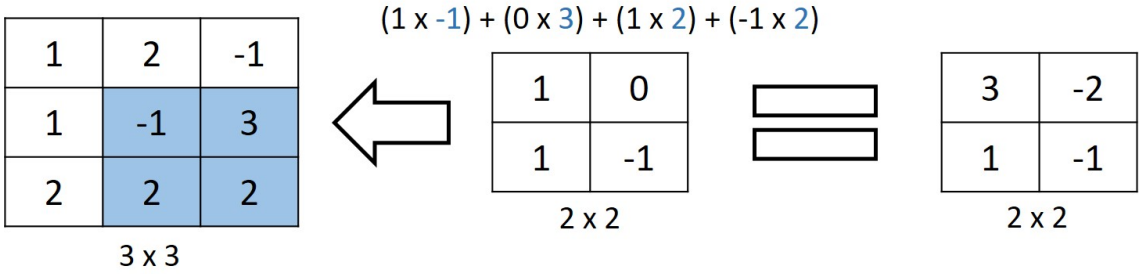

Figura 6 – Passo 4 do exemplo de convolução. Fonte: Autor

imagem só tem um canal, sendo uma imagem apenas com tons de cinza. Se fosse uma imagem colorida, o *Kernel* teria de ser multiplicado pelas três matrizes, que representariam cada cor do pixel formado no padrão RGB, Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*), gerando assim três canais para o mapa de característica.

2.3.1.2 Camada de *Pooling*

Semelhantemente à camada convolucional, a camada *Pooling* é responsável por reduzir o tamanho da entrada. Isso é para reduzir o custo computacional necessária para processar os dados através da redução de dimensionalidade. Além disso, é útil para extrair características dominantes, mantendo assim o processo de treinamento eficaz do modelo. Geralmente o *Pooling* é aplicado no mapa de característica.

Existem dois tipos de principais de *Pooling*: *Max Pooling* e *Average Pooling*. O *Max Pooling* retorna o valor máximo da parte da imagem coberta pela matriz do *Pooling*. Por outro lado, o *Average Pooling* retorna a média de todos os valores da parte da imagem coberta pela *Pooling*. O *Pooling* percorre a matriz de entrada da esquerda para direita, pulando n passos por execução. Na figura 7 temos um exemplo da aplicação de *Max Pooling* e *Average Pooling*, por exemplo na área verde temos 15 como valor máximo e 10 como valor médio.

O *Pooling* também funciona como um supressor de ruído e ajuda a evitar o sobre-ajuste, ou *Overfitting*. *Overfitting* é quando a rede neural se adapta demais aos dados de treinamento fazendo com que seu erro diminua nessa etapa, mas na etapa

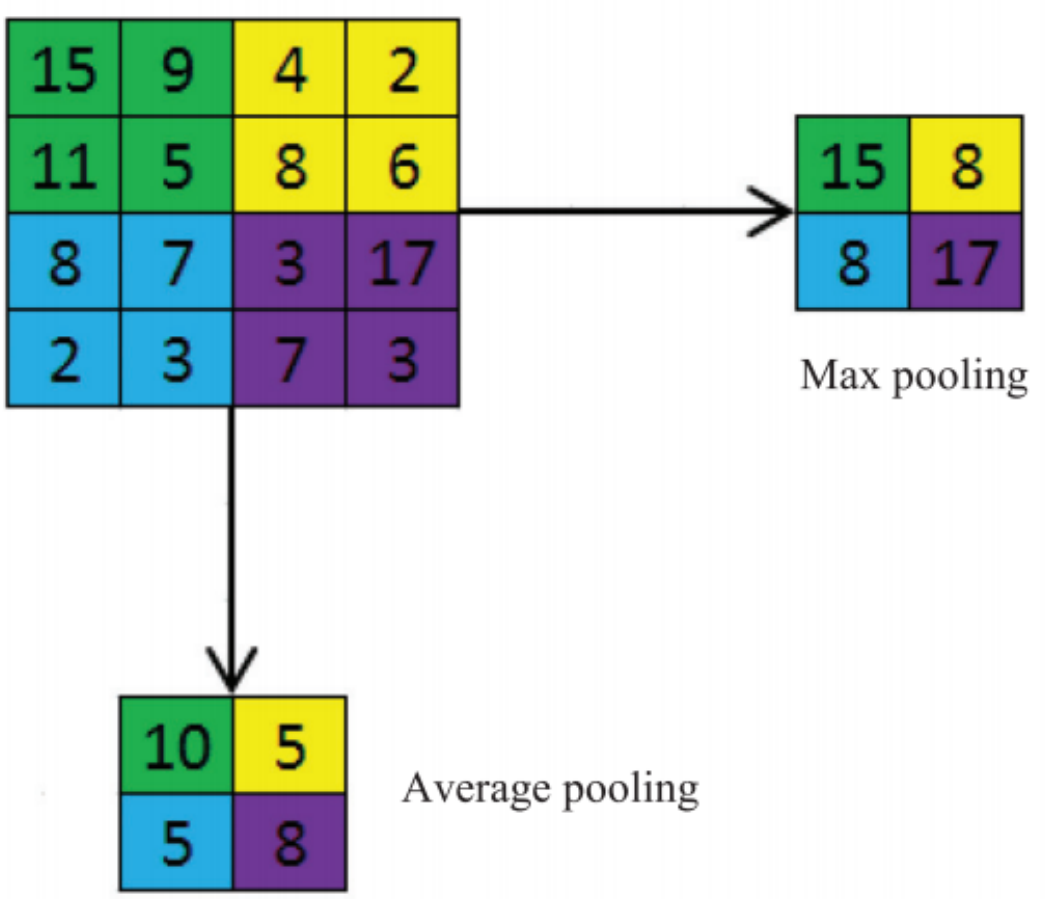

Figura 7 – Exemplo de *Max Pooling* e *Average Pooling*.
(RAWAT; WANG, 2017)

de teste o erro continua alto, ou seja, a rede passa a não generalizar dados diferentes, não vistos no processo de treinamento. Existe também o conceito de subajuste, ou *Underfitting*, acontece quando o erro na etapa de treinamento não reduz, isso significa que a rede não consegue aprender padrões.

2.3.1.3 Camada Totalmente Conectada

Essa camada basicamente pega um volume de entrada e gera um vetor com dimensão N, onde N é o número de classes do problema. Cada número neste vetor dimensional N representa a probabilidade de uma determinada classe. Por exemplo, na Figura 8 temos um N igual a 5, pois existem 5 classes possíveis, *Car*, *Bus*, *Train*, *Plane*, *Ship*, e se o vetor resultante para um programa de classificação de dígitos for [0.75, 0.1, 0.1, 0, 0.05], isso representa que a possibilidade da classe ser *Car* é de 75%, 10% de ser *Bus*, 10% de ser *Train*, 5% de ser *Ship* e nenhuma possibilidade de ser *Plane*.

A maneira como essa camada funciona é analisando a saída da camada anterior e determinando quais recursos mais se correlacionam com uma determinada

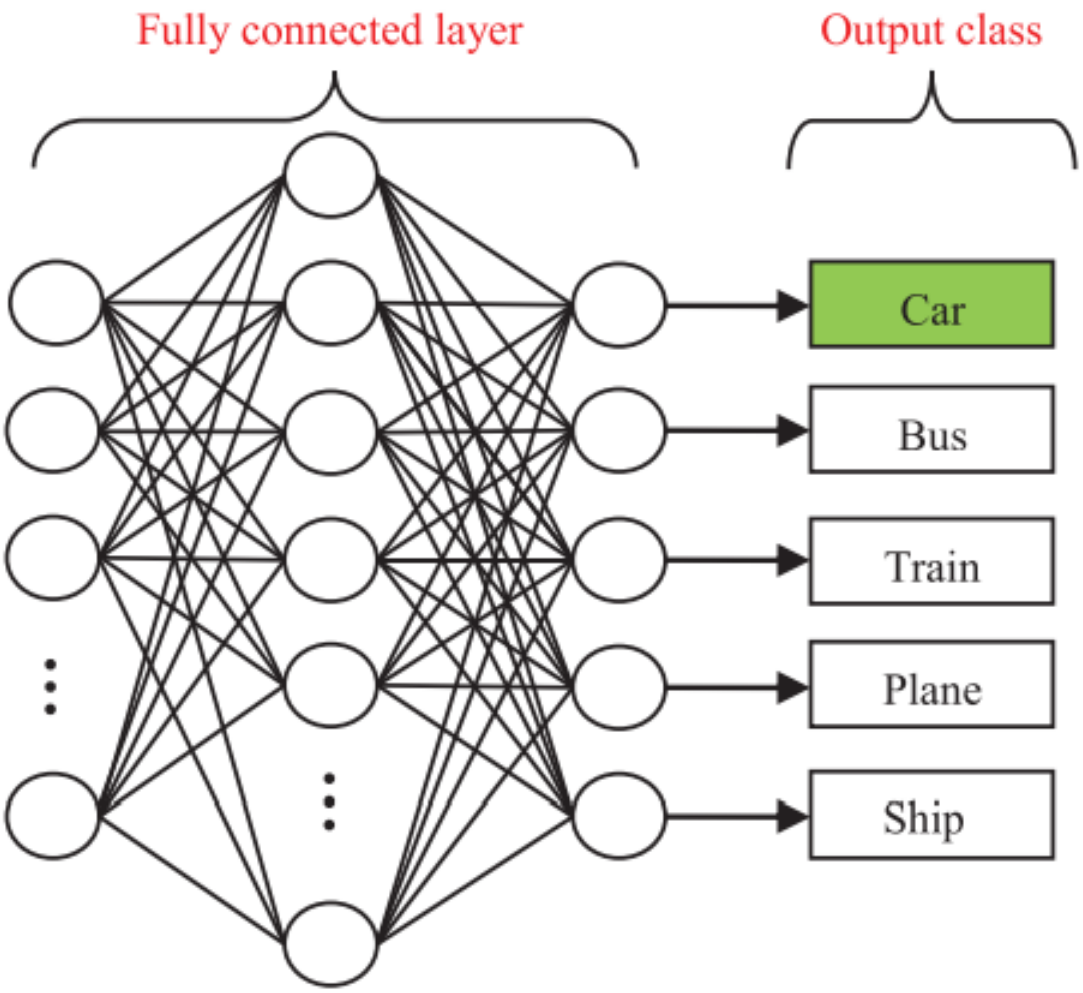

Figura 8 – Representação de uma Camada Totalmente Conectada (RAWAT; WANG, 2017)

classe. No exemplo da Figura 8, para classificar um *Car*, ele terá altos valores nos mapas de ativação que representam recursos de alto nível, como uma roda ou quatro rodas, formato da porta, etc. Basicamente, uma camada totalmente conectada analisa quais características de alto nível se correlacionam mais fortemente a uma classe específica.

2.3.2 Função de ativação

As funções de ativação são importantes para uma Rede Neural Artificial aprender e entender algo realmente complicado. Principalmente quando há mapeamentos funcionais complexos, lineares ou não, entre as variáveis de entrada e saída. Elas introduzem propriedades não lineares na rede e seu principal objetivo é converter um sinal de entrada de um nó em um sinal de saída. Esse sinal de saída agora é usado como entrada na próxima camada da rede. Existem vários tipos de funções de ativação,

como Sigmoide, Tangente hiperbólica e ativação linear retificada.

A função de ativação linear retificada (*Rectified Linear Unit* - ReLU), será apresentada, pois será a utilizada neste trabalho. Ela foi introduzida por (HAHNLOSER et al., 2000). Ela é geralmente a função mais utilizada entre as camadas intermediárias em redes neurais convolucionais (CHATFIELD et al., 2014). A função $ReLU(x) = max(0, x)$ recebe um valor x como entrada e transforma valores negativos em zero. Na Figura 9 temos o gráfico da função ReLU.

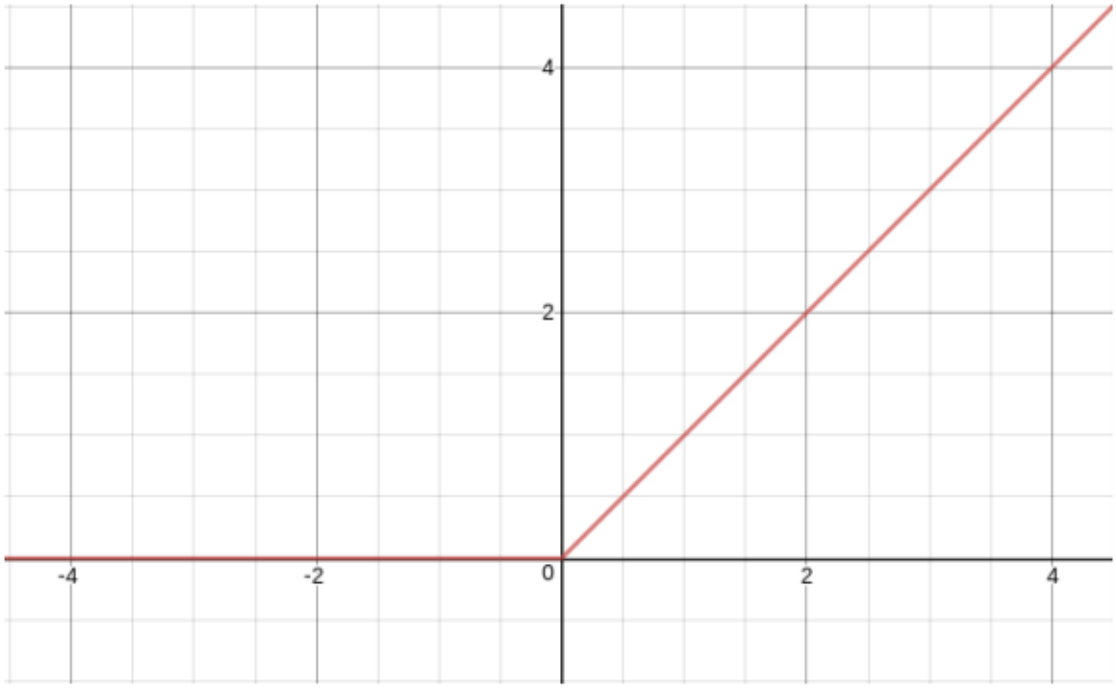

Figura 9 – Gráfico da Função de Ativação ReLU
(AGARAP, 2018)

2.3.3 Treinamento

Quando começamos o treinamento da rede neural, inicializamos os pesos aleatoriamente. No processo de treinamento, começamos com uma rede neural com desempenho ruim e terminamos com uma rede de alta precisão. A melhoria da rede acontece com ajustes nos pesos da rede. O problema do treinamento é equivalente ao problema de minimizar a Função de Perda, o conceito de Função de Perda será apresentado no parágrafo a seguir.

A função de perda é uma função que nos diz o quão boa é a nossa rede neural para uma determinada tarefa. Para calcular a função de perda, executamos a rede para cada exemplo de treinamento, que já está classificado, do conjunto de dados, e verificamos o quão precisa é a classificação dada pela Rede Neural. Se o resultado da Função Perda for alto, a rede ainda não aprendeu, queremos minimizar a Função de

Perda. Existem muitos algoritmos que otimizam funções. Um conjunto desses algoritmos são baseados em gradiente.

2.3.3.1 Taxa de aprendizado

A taxa de aprendizado é um hiperparâmetro que controla a velocidade de ajuste dos pesos. Quanto menor o valor, mais lento percorremos a função, isso significar que levaremos muito tempo para convergir, especialmente se ficarmos presos em uma região plana. Quanto maior o valor maior o salto ao longo da função o que pode levar a uma não convergência. Geralmente os valores estão ente 0 e 1.

2.4 Algoritmos *Policy-Gradient*

Os métodos de políticas gradientes são fundamentais para as recentes descobertas no uso de redes neurais profundas em diversas questões como locomoção em três dimensões e jogos eletrônicos (SCHULMAN et al., 2017). A obtenção de bons resultados em métodos de gradiente de políticas pode ser um desafio, pois este método é sensível à escolha do tamanho do episódio. A utilização de episódios muito pequenos gera um progresso extremamente lento, já muito grande o sinal é sobrecarregado pelo ruído, além da possibilidade de haver grandes quedas no desempenho. Essa técnica costuma ter uma eficiência de amostragem baixa, demorando muito para convergir (SCHULMAN et al., 2017).

A política é representada por um modelo derivável, geralmente uma Rede Neural, onde é usado um algoritmo de gradiente para atualizar os parâmetros da rede e assim encontrar uma política ótima. Vamos usar θ para representar os parâmetros da política (os pesos da Rede Neural) e θ_t para retratar θ no iteração t. α representa a taxa de aprendizado. Podemos denominar a probabilidade de tomar uma ação a no estado s de $\pi_\theta(a|s)$, com o objetivo de descobrir a regra de atualização de θ à θ_{t+1}, de uma maneira que eventualmente atinja a política ideal.

Se a^* a ação da politica ótima no estado s, queremos que $\pi_\theta(a^*|s)$ seja o mais próximo possível de 1. Para isso, podemos simplesmente realizar subida de gradiente em $\pi_\theta(a^*|s)$; portanto, a cada iteração, atualizamos θ da seguinte maneira:

$$\theta_{t+1} = \theta_t + \alpha \nabla \pi_{\theta_t}(a^*|s)$$

Podemos ver o gradiente $\nabla \pi_{\theta_t}(a^*|s)$ como sendo a direção na qual mover θ_t para aumentar o valor de $\pi_\theta(a^*|s)$ o mais rápido. Observe que, de fato, estamos usando subida gradiente, pois queremos aumentar um valor, não diminuí-lo, como é habitual ocorrer na função de perda no aprendizado por reforço.

Na prática, não sabemos qual ação é melhor, desta maneira podemos destacar

ações sub-ótimas e desta maneira a política nunca convergirá. Uma solução seria destacar as ações de maneira proporcional ao valor estimado da ação pela própria política π_θ: G_t. Contanto que o valor estimado da ação estejam fundamentadas na realidade, mais valorização haverá na ação ideal a^*. Dessa forma, é garantido que, eventualmente, nossa política irá convergir. Com isso temos uma nova subida de gradiente:

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \pi_{\theta_t}(a|s)$$

Embora possamos impulsionar fortemente as ações que têm um melhor valor, também impulsionaremos as ações que tiverem valores iniciais altos para a política π_θ, o que pode acontecer principalmente devido a inicialização aleatória dos pesos na Rede Neural. Essas ações podem acabar sendo extremamente valorizadas, apesar de serem ruins. Para solucionar este problema dividimos a atualização pela probabilidade da ação. Desta forma temos a seguinte atualização:

$$\theta_{t+1} = \theta_t + \alpha \frac{G_t \nabla \pi_{\theta_t}(a|s)}{\pi_\theta(a|s)}$$

Devido à regra da cadeia e ao fato de a derivada do $\log(x)$ ser $\frac{1}{x}$, como é amplamente conhecido. Então, nós temos a seguinte atualização:

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \log \pi_\theta(a|s)$$

A seguir daremos uma definição da função de perda, que pode ser utilizada por métodos de descida de gradiente e que é calculada após a execução de um episodio completo com T passos.

$$loss(\theta) = -\frac{1}{T} \sum_{i=1}^{T} G_t \times \log \pi(a_t|s_t, \theta)$$

Dessa maneira, os algoritmos de *Policy-Gradient* executam uma determinada quantidade de passos no ambiente seguindo a política determinística π. Ao final da execução, as informações dos estados, ações e recompensas, são utilizados para o calculo da função de perda e a política é atualizada por algum algoritmo de gradiente.

2.4.1 *Policy-Gradient com Baselines*

Suponhamos que uma ação nos dá uma recompensa cumulativa de pelo menos 1000, temos dois estados s_1 e s_2, digamos que para a ação a_1 o valor de G_t é 1001 e 1002 para a ação a_2. Como podemos ver, o valor da função de valor descontada para a_1 e a_2 é irrelevante, fazendo com que o resultado da atualização não garanta se a ação atual é melhor. Um solução para este problema é a função Vantagem A_t que subtrai do valor da ação o valor do estado, $A_t = G_t - V(s)$. Essa função nos diz quanto melhor ou pior a ação de um estado é comparada à ação de acordo com a política. Obtendo a seguinte Função de Perda:

$$loss(\theta) = -\frac{1}{T} \sum_{i=1}^{T} A_t \times \log \pi(a_t|s_t, \theta)$$

2.4.2 O método *Actor Critic*

A abordagem Ator Crítico, é um algoritmo *Policy-Gradient* com *baseline* (*Advantage*), que ao invés de executar um episodio completo, executa apenas K passos por vez, antes de fazer uma atualização dos pesos usando o gradiente descendente.

2.5 Algoritmo PPO

O algoritmo Otimização de Política Proximal e um algoritmo que usa *Policy-Gradient* que foi introduzido pela equipe OpenAI em 2017 (SCHULMAN et al., 2017) e rapidamente se tornou um dos métodos de RL mais populares. O PPO encontra um equilíbrio entre facilidade de implementação, complexidade de amostra e facilidade de ajuste, tentando calcular uma atualização a cada etapa que minimiza a função de valor e garante que o desvio da política anterior seja relativamente pequeno.

A principal contribuição do PPO é garantir que uma nova atualização da política não a mude muito em relação à política anterior. Isso leva a uma menor variação no treinamento, garantindo um treinamento mais suave além de garantir que o agente não siga um caminho irrecuperável de ações sem sentido.

Na implementação do algoritmo PPO, são mantidas duas políticas. A primeira é a política atual $\pi_\theta(a_t|s_t)$ que queremos aperfeiçoar. A segunda política $\pi_{\theta_k}(a_t|s_t)$ é a utilizada no ultimo treinamento que possui como principal objetivo avaliar o nível de mudança entres as duas políticas. $r_t(\theta)$ é uma proporção entre as probabilidades das ações na nova política e na antiga, assim temos que $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$. Essa proporção mede a diferença entre duas políticas. Assim podemos construir a função objetivo abaixo:

$$L^{CLIP}(\theta) = E_t[\min r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

A função de $CLIP$ é responsável por ”cortar”a probabilidade quando ela varia muito, para mais ou menos. ϵ é um hiperparâmetro utilizado pela função $CLIP$, ele geralmente assume os valores 0,1 ou 0,2. Se a taxa de proporção entre a nova política e a política antiga estiver fora do intervalo $(1-\epsilon)$ e $(1+\epsilon)$, a função de objetivo será cortada. A Figura 10 mostra o gráfico da função L^{CLIP} em termos de r.

2.6 Aprendizado por currículo

Os humanos precisam de mais ou menos duas décadas para serem treinados como adultos funcionais da nossa sociedade. Esse treinamento é altamente organizado, baseado em um sistema educacional com um currículo que introduz conceitos diferentes em momentos diferentes, explorando conceitos aprendidos anteriormente

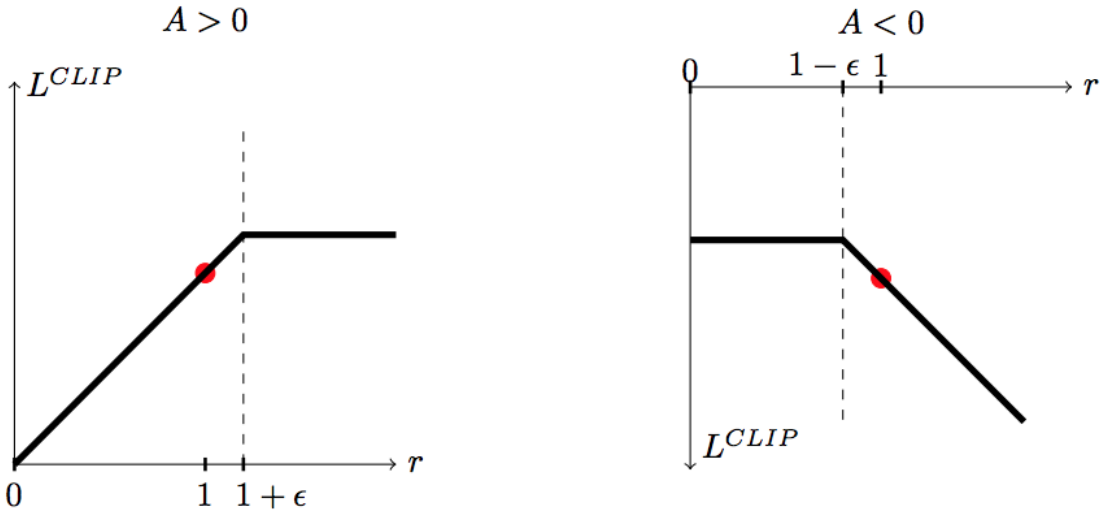

Figura 10 – Gráfico da função L^{CLIP} em termos de r.
(SCHULMAN et al., 2017)

para facilitar o aprendizado de novas abstrações (BENGIO et al., 2009). Mas este mesmo processo pode ser utilizado para os computadores? Sim, nos artigos (ELMAN, 1993) e (KRUEGER; DAYAN, 2009) encontramos a ideia da utilização de uma metodologia similar ao currículo para aprendizado de máquina .

Quando o ambiente oferece apenas recompensas muito esparsas, o aprendizado é muito lento, fazendo com que o algoritmo demore muito tempo para convergir em uma solução satisfatória (WU; TIAN, 2016). Para solucionar este problema podemos usar a aprendizagem curricular, onde a ideia básica é começar por problemas menores, aprendendo aspectos mais fáceis da tarefa ou utilizar sub-tarefas mais fáceis, posteriormente aumentar gradualmente o nível de dificuldade da tarefa ou novas sub-tarefas mais difíceis (BENGIO et al., 2009).

2.7 Ambientes de benchmark de futebol

2.7.1 Robocup 2/3D

A RoboCup foi criada em 1996 por um grupo de pesquisadores japoneses, americanos e europeus de Inteligencia Artificial (IA), e tem o como finalidade promover a pesquisa em IA e robótica, fornecendo uma tarefa em comum, o Futebol, para avaliação de inúmeras teorias, algoritmos e arquiteturas de agentes (KITANO, 1998). A RoboCup conta com três ligas de futebol robótico: Simulação, robôs de tamanho Pequeno e robôs de tamanho médio. A liga de simulação simplifica e abstrai o nível físico, para que os pesquisadores possam se concentrar no desenvolvimento de agentes inteligentes de nível estratégico que realizam comportamentos cooperativos. Essa liga é dividida em duas sub-ligas: 3D e 2D, o campo é composto por jogos entre duas equi-

pes de 11 jogadores em um campo com as mesmas proporções da Copa do mundo da FIFA (NARDI et al., 2014), na Figura 11 é mostrado o campo da sub-liga 2D e na Figura 12 é mostrado o campo da sub-liga 3D.

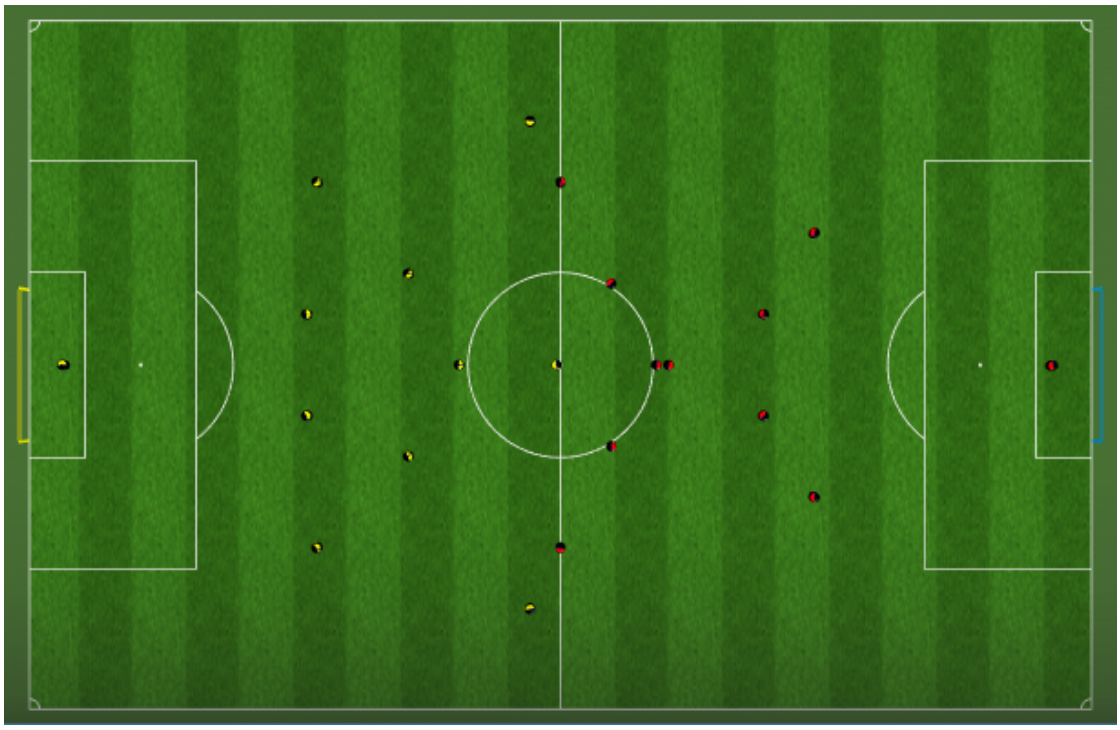

Figura 11 – Campo RoboCup 2D. Fonte: Autor

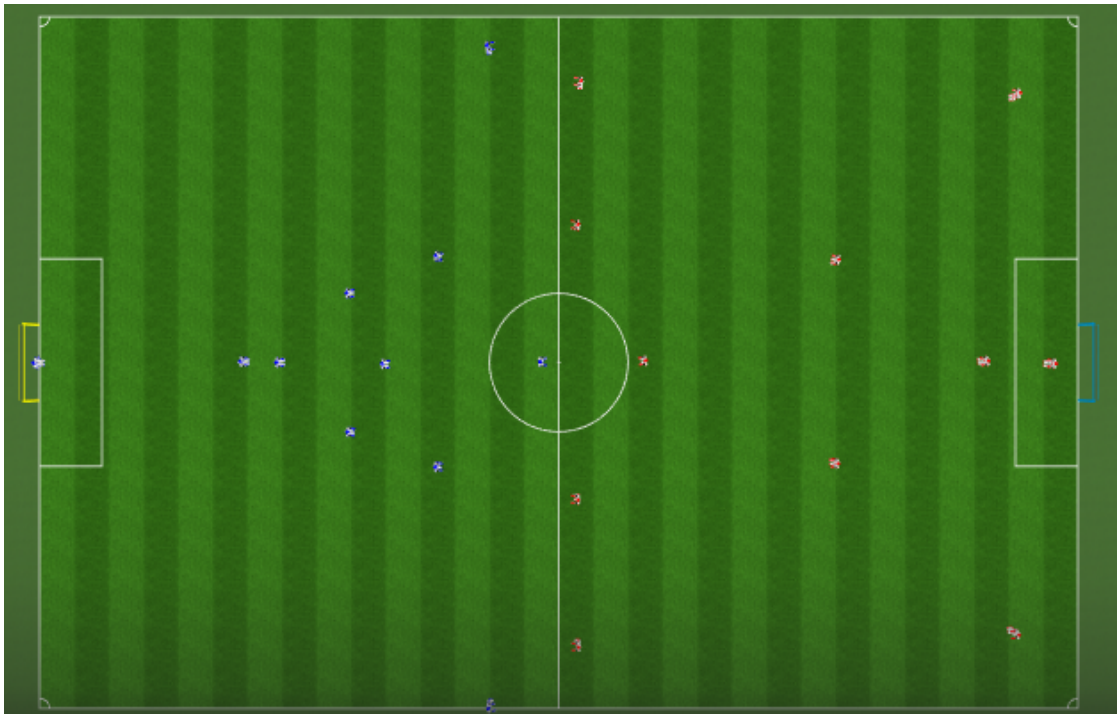

Figura 12 – Campo RoboCup 3D. Fonte: Autor

O funcionamento da simulação acontece por meio do *Soccer Server*, este é um sistema que possui a arquitetura cliente-servidor e permite agentes autônomos joga-

rem uma partida de futebol (CUÉLLAR, 2012). Os principais elementos que interagem com o *Soccer Server* são: os agentes e o monitor, essa comunicação é feita por meio de soquetes UDP/IP como é mostrado na Figura 13. Cada agente possui um processo separado e se comunicam com o servidor através de uma porta, o monitor possui uma porta exclusiva que fornece uma visão global do jogo. O jogador possui apenas a percepção do seu campo de visão, que é uma área triangular a sua frente, e quanto maior a distancia do jogador e percepção perde qualidade. Uma equipe pode ter no máximo 12 clientes (agentes), sendo 11 jogadores e 1 técnico (XIONG et al., 2008). Os jogadores enviam mensagens para o servidor com ações que desejam realizar (chutar, virar, correr, etc.), e o *Soccer Server* lida com as requisições atualizando o ambiente de acordo com as regras do jogo e as leis da física, além de fornecer aos clientes informações do jogo. O simulador é um sistema em tempo real que trabalha com intervalos de tempo discretos e possui uma gerencia de incerteza, que atribui a uma ação um resultado nem sempre perfeito (KITANO et al., 1997a; KITANO et al., 1997b). Cada intervalo tem determinada duração e para que uma ação seja executada em um intervalo é preciso que chegue em tempo hábil ao *Soccer Server*. Uma descrição mais detalhada do servidor está disponível no manual do simulador (ITSUKI, 1995).

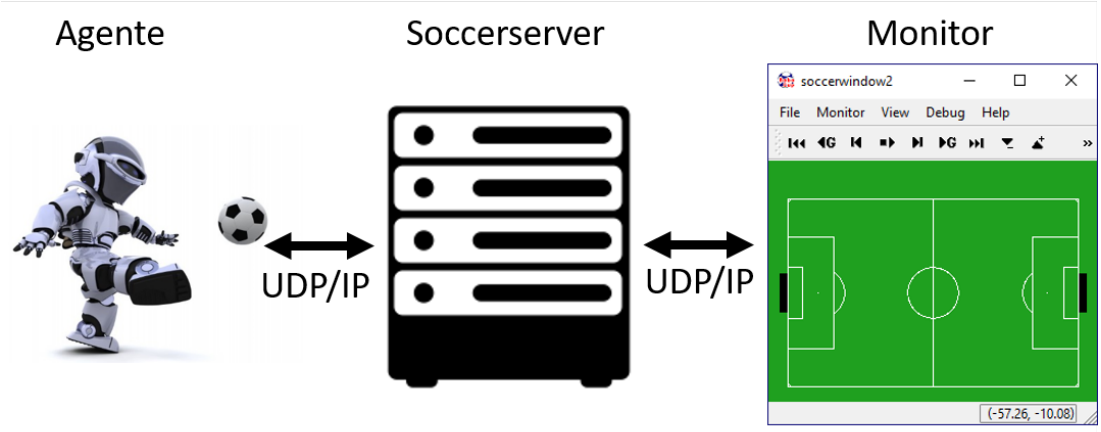

Figura 13 – Arquitetura da comunicação do *Soccer Server*. Fonte: Autor

2.7.2 Ambiente Google Research Football

O *Google Research Football* é um ambiente simulado de Futebol 3D baseado nas leis da física e com aspectos semelhantes a jogos eletrônicos, como FIFA. Foi criado com o objetivo de treinar e avaliar algoritmos de aprendizado por reforço. Possui código aberto, um mecanismo de jogo otimizado e um conjunto de cenários de aprendizado por reforço progressivamente mais difíceis, a chamada *Football Academy*. Possui diversas contribuições, dentre elas destacam-se: propor um benchmark de futebol avaliar algoritmos de aprendizado por reforço e fornecer uma API simples para perso-

nalizar e definir novos cenários de aprendizado por reforço do futebol (KURACH et al., 2019).

O jogo completo possui 11 jogadores em cada lado, e tem a duração medida em termos de número de quadros, totalizado 3 mil quadros por jogo (10 quadros por segundo em 5 minutos). Os jogadores possuem características, como velocidade e precisão, diferentes. O time adversário é controlado por um algoritmo baseado em regras, e o nível de dificuldade Θ pode variar entre 0 e 1, acelerando ou diminuindo o tempo de reação e a tomada de decisão do jogador. São pré-definidas três dificuldades, *Easy* ($\theta = 0,05$), *Medium* ($\theta = 0,6$), *Hard* ($\theta = 0,95$). A Figura 14 mostra a interface gráfica da ferramenta.

O ambiente possui as seguintes ações disponíveis: movimentação em 8 direções, passe curto, passe longo, arremesso, passe alto, correr, interceptar bola, parar movimentação e parar corrida. O ambiente possui duas funções de recompensa, *Scoring* e *Checkpoint*. A primeira função corresponde à recompensa natural, onde recebe 1 de recompensa quando o time marca um gol e recebe -1 quando sofre um gol. Na função *Checkpoint* há uma divisão do campo adversário em 10 regiões, de acordo com a distância euclidiana para gol adversário, pela primeira vez que o time possui a bola em uma região ganha uma recompensa de 0.1, no máximo pode receber 1. Qualquer recompensa *Checkpoint* não adquirida, também é adicionado ao marcar um gol, a fim de não penalizar os agentes que não passam por todos os pontos de verificação antes do gol, ou seja, atirando de fora de uma região do ponto de verificação, a recompensa por gol feito ou recebido também é utilizada.

Figura 14 – Campo do Google Research Football
(KURACH et al., 2019)

2.8 Academia de Futebol

Como o treinamento de agentes para um jogo de Futebol completo pode ser desafiador, o ambiente *Google Research Football* fornece a *Football Academy*, em português ”academia de futebol”. Ela é composta por um conjunto diversificado de cenários com diferentes dificuldades. Permitindo que pesquisadores avaliem novas ideias de pesquisa, permite também testar conceitos de alto nível e fornece uma base para avaliar ideias de pesquisa de aprendizado por currículo. Usando uma API simples, os pesquisadores podem definir ainda mais seus próprios cenários e treinar agentes para resolvê-los (KURACH et al., 2019). Abaixo podemos ver uma lista com todos os 11 cenários academia de futebol. Em (KURACH et al., 2019) encontramos resultados de experimentos iniciais reportados pela Google. São utilizados algoritmos de aprendizado por reforço, a exemplo do PPO, para treinamento de um jogo completo e em todos os cenários da *Football Academy*.

- *empty_goal_close*;
- *empty_goal*;
- *run_to_score*;
- *run_to_score_with_keeper*;
- *pass_and_shoot_with_keeper*;
- *run_pass_and_shoot_with_keeper*;
- *3_vs_1_with_keeper*;
- *corner*;
- *counterattack_easy*;
- *counterattack_hard*;
- *single_goal_versus_lazy*;

2.9 Comentários

Neste trabalho pretendemos resolver o problema das recompensas esparsas, que demoram para serem recebidas pelo agente, em ambientes de futebol. Este problema retarda o aprendizado, fazendo com que a aplicação de aprendizado por reforço em ambientes de futebol seja desafiadora. Contudo técnicas de aprendizado obtiveram

bons resultados em alguns desses ambientes. Diante desses bons resultados aplicaremos o aprendizado por currículo para acelerar o treinamento de agentes em um ambiente novo, com poucos resultados, o *Google Research Football Environment* (2019).

3 Um currículo para o *Google Research Football*

No artigo do Google (KURACH et al., 2019) foram reportados resultados inicias, que servem como referência para trabalhos futuros. Para cada função recompensa proposta no artigo, *Scoring* e *Checkpoint*, foram feitos treinamentos para um jogo completo, com 20 e 500 milhões de passos nas três dificuldades: *Easy*, *Medium*, *Hard*. Da mesma forma foram realizados treinamentos para todos os cenários da *Football Academy*, utilizando 1, 5 e 50 milhões de passos. Na Figura 15 temos a diferença média de gols para os cenários da *Football Academy* utilizando o algoritmo PPO com a recompensa *Checkpoint*. Nosso objetivo com a utilização do aprendizado por currículo é acelerar o treinamento, se comparado com os resultados iniciais reportados pelo Google. Assim obtendo melhores resultados com o mesmo número de passos.

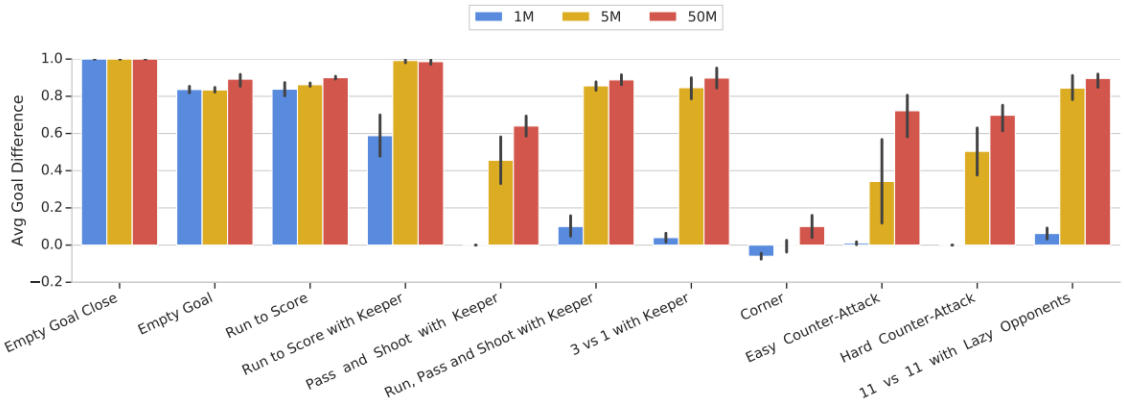

Figura 15 – Diferença média de gol na *Football Academy* com o algoritmo PPO usando a recompensa *Checkpoint*.
(KURACH et al., 2019)

A princípio, iríamos construir um currículo para o jogo de futebol completo, utilizando cenários selecionados da *Football Academy*. Entretanto o alto poder computacional necessário para a realização dos treinamentos, inviabilizou este primeiro propósito. Focamos em um cenário intermediário *Run to Score with Keeper*, mas com a mesma ideia de: usar currículo para obter um resultado melhor do que o do Google neste cenário.

3.1 Cenários Utilizados

Cada um dos dois currículos propostos é composto por um conjunto total de 1 milhão de passos, distribuídos em 4 cenários da *Football Academy*, com o objetivo

de maximizar a média da diferença de gols do ultimo cenário. Os cenários utilizados foram: *Empty Goal Close*, *Empty Goal*, *Run to Score* e *Run to Score with Keeper*, que são mostrados nas Figuras 16, 17, 18 e 19 consecutivamente. Abaixo temos uma lista com uma breve descrição dos cenário utilizados:

1. *empty_goal_close*: o jogador começa dentro da grande área com a bola e precisa marcar contra um gol vazio.
2. *empty_goal*: o jogador começa no meio do campo com a bola e precisa marcar contra um gol vazio.
3. *run_to_score*: o jogador começa no meio do campo com a bola e precisa marcar contra um gol vazio, mas possui cinco jogadores adversários o marcando.
4. *run_to_score_with_keeper*: o jogador começa no meio do campo com a bola e precisa marcar contra um gol com goleiro e mais cinco jogadores adversários o marcando.

Figura 16 – Cenário *Empty Goal Close*. Fonte: Autor

3.2 Arquitetura e parâmetros

O algoritmo PPO foi configurado com 8 *threads*, foram utilizados os parâmetros da Tabela 1 e a arquitetura da Rede Neural é apresentada na Figura 20. A arquitetura é composta por camadas convolucionais $3x3$ com *stride* 1 e camadas *Max Pooling* $3x3$ com *stride* 2. Também possui blocos residuais, um conceito relativamente recente

Figura 17 – Cenário *Empty Goal*. Fonte: Autor

Figura 18 – Cenário *Run to Score*. Fonte: Autor

(2015), que essencialmente, permitem o fluxo de informação da camada inicial para a última. Os blocos residuais são compostos camadas convolucionais 3x3 com *stride* 1, com funções de ativação ReLU. No final, possui uma rede totalmente conectada. As saídas da rede são as características do método *Actor Critic* e dos *Policy-Gradient* com *baseline*: a política $\pi(a_t)$ e o valor do estado V_t. Os parâmetros e arquitetura da rede são os mesmo propostos em (KURACH et al., 2019).

Figura 19 – Cenário *Run to Score with Keeper*. Fonte: Autor

Parametro	**Valor**
Repetições de ação	1
Intervalo de corte	0.08
Fator de desconto (γ)	0.993
Coeficiente de entropia	0.003
GAE (λ)	0.95
Corte do gradiente normal	0.64
Taxa de aprendizado	0.000343
Número de atores	8
Otimizador	Adam
Período de treinamento por atualização	2
Mini-batches de treinamento por atualização	8
N-step	512
Coeficiente da função de valor	0.5

Tabela 1 – Tabela de parâmetros usados nos experimentos. Fonte: Autor

3.3 Currículos propostos

O primeiro currículo proposto foi denominado de **5-15-30-50**, que possui 50 mil passos do cenário *Empty Goal Close*, 150 mil do cenário *Empty Goal*, 300 mil do cenário *Run to Score* e 500 mil passos no cenário *Run to Score with Keeper*. Este currículo foi fundamentado apenas na definição de cada cenário, foi observado no *log* gerado, que os primeiros três cenários possuíam mais passos que o necessário para a obtenção de um bom resultado, ou seja, recompensa 2 nos últimos 100 episódios. Visando uma melhoria no resultado, foram analisados os resultados do primeiro currículo, e proposto um segundo currículo, denominado de **3-10-20-67** é composto por: 30 mil passos

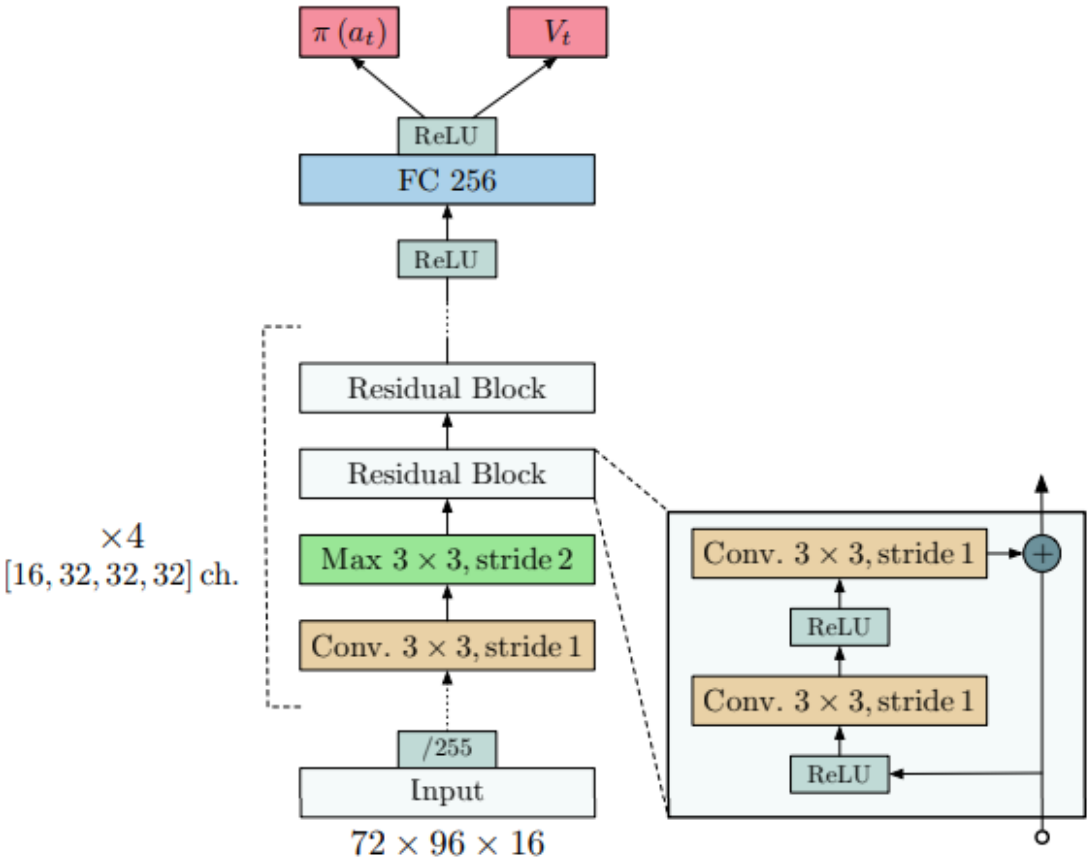

Figura 20 – Arquitetura da Rede Neural utilizada nos esperimentos (KURACH et al., 2019)

do cenário *Empty Goal Close*, 100 mil do cenário *Empty Goal*, 200 mil do cenário *Run to Score* e 670 mil passos no cenário *Run to Score with Keeper*. Na Tabela 2 temos a quantidade de passos de utilizados em cada cenário.

Curriculo	**5-15-30-50**	**3-10-20-67**
1 *Empty Goal Close*	50 mil	30 mil
2 *Empty Goal*	150 mil	100 mil
3 *Run to Score*	300 mil	200 mil
4 *Run to Score with Keeper*	500 mil	670 mil

Tabela 2 – Tabela com a quantidade de passos treinados para cada cenário. Fonte: Autor

3.4 Implementação

Para execução dos experimentos foi utilizado a algoritmo PPO de referência da OpenAI. Tivemos que limitar o número de *threads* para 8 por uma restrição na capacidade do ambiente de treinamento, descrito na seção 4.1 deste trabalho. O arquivo de *log* padrão gerado pelo ambiente possuía apenas a média das recompensas de um

conjunto de episódios. Com a finalidade de obtermos a recompensa individual de cada episodio foi necessário reescrever o código de geração do *log*. Para isso, gastamos um tempo significativo, devido a alta complexidade do algoritmo e do código, para a aplicação de engenharia reversa e realizar as modificações necessárias.

4 Resultados e discussão

4.1 Ambiente utilizado para teste

O ambiente utilizado para realização dos experimentos foi um Notebook Dell Inspiron 14 3000 com Memória de 8 GB DDR3, processador Intel Core i5-4210U 1.7GHz e com o sistema operacional Ubuntu 18.04.1.

4.2 Organização dos experimentos

Juntamente aos currículos propostos, também foi realizado o treinamento do ultimo cenário,*run_to_score_with_keeper* , com 1 milhão de passos sem currículo, tendo como finalidade a reprodução dos resultados obtidos pelo Google. Conseguimos a reprodução dos resultados dentro da margem de erro, o que garante que estamos utilizando o mesmos parâmetros e algoritmo. Além de uma comparação fidedigna dos resultados com os currículos apresentados e os experimentos realizados em (KURACH et al., 2019). Cada treinamento, tanto com o currículo quanto o sem currículo, continham 1 milhão de passos e foi realizado 5 vezes ao final calculamos a média e o desvio padrão dos resultados obtidos. Cada experimento durou em média 18 horas para executar na ambiente descrito na seção 4.1, totalizando 270 horas de treinamento, já que foram realizados 15 experimentos.

4.3 Comparação dos resultados

A Figura 21 contem o a média da diferença de gols obtidas pelos 5 treinamentos para os dois currículos apresentados neste trabalho e o treinamento sem currículo no cenário *run_to_score_with_keeper*. Comparando o resultado do primeiro currículo proposto (5-15-30-50), com o treinamento sem currículo vemos que em média o currículo possui um melhor resultado, entretanto se levarmos em consideração o desvio padrão o currículo não necessariamente possui uma maior média da diferença de gols. Observando o resultado do segundo currículo (3-10-20-67), vemos que mesmo levando em conta a desvio padrão, o currículo apresenta uma maior média da diferença de gols que o treino sem o currículo do ultimo cenário. Na Figura 22 é mostrada a diferença de gols no cenário *run to score with keeper* dos 5 experimentos realizados para o treinamento sem currículo. Na Figura 23 são exibidas as diferenças de gols no cenário *run to score with keeper* do currículo 5-15-30-50 para os 5 treinamentos realizados. A Figura 24 contém a diferença de gols no cenário *run to score with keeper* para os

5 experimentos realizados pelo currículo 3-10-20-67. Não conseguimos avançar mais nos testes por limitação de poder computacional.

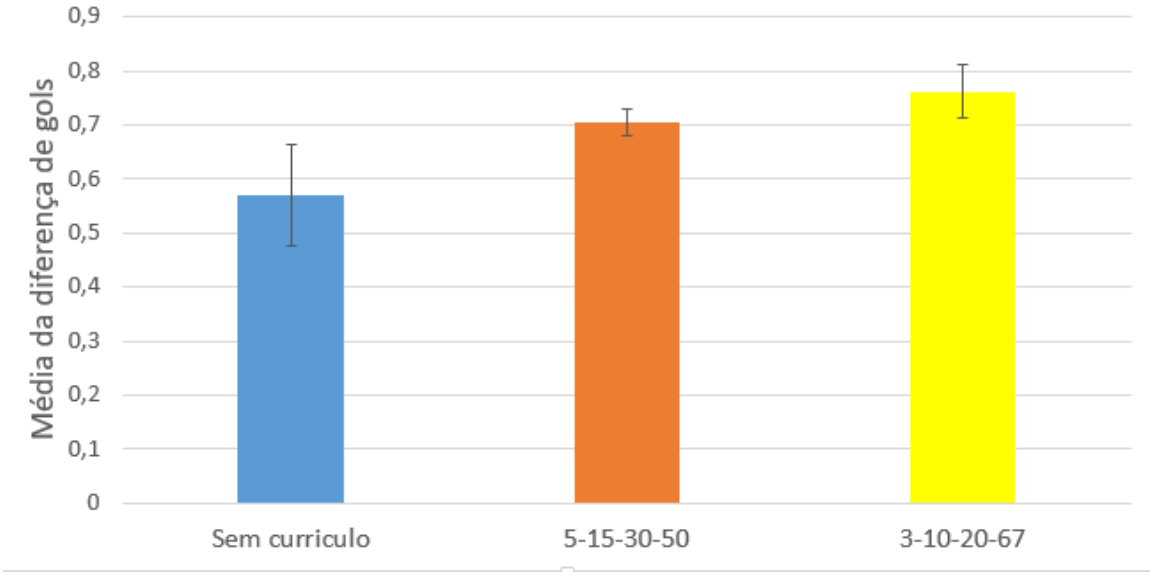

Figura 21 – Comparação da média da diferença de gols no cenário *run to score with keeper*. Fonte: autor

Se levarmos em consideração os resultados obtidos da utilização de um currículo nos cenários *empty goal* e *run to score*, obtivemos um resultado excelente. Na Figura 25 temos os resultados apresentados em (KURACH et al., 2019) para o treinamento sem currículo dos cenários *empty goal* e *run to score*, denominados na Tabela 2 de **2** e **3** respectivamente, para o treinamento utilizando 1, 5 e 50 milhões de passos. Também temos os resultados para os mesmos cenários, mas utilizando o currículo 3-10-20-67, com 130 mil passos utilizados no treinamento do cenário 2 e 330 mil para o cenário 3. Nos resultados do Google, sem currículo, mesmo sendo utilizados 50 milhões de passos para o treinamento não foi alcançado o recompensa máxima (1,0). Já com a utilização do aprendizado por currículo, obtivemos a recompensa máxima com menos de 200 mil passos para o cenário 2 e 400 mil para o 3. Esses resultados apontam para a eficácia da utilização de currículo no aprendizado por reforço aplicado ao ambiente *Google Research Football*.

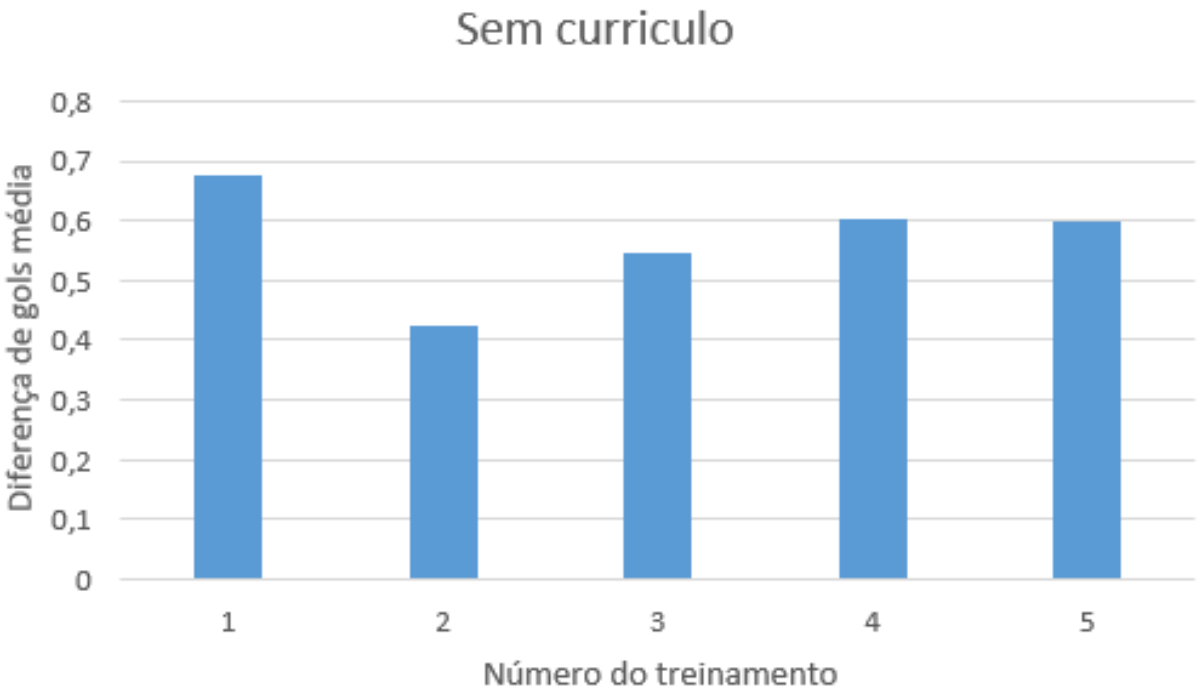

Figura 22 – Diferença de gols no cenário *run to score with keeper* para os experimentos do treinamento sem currículo. Fonte: autor

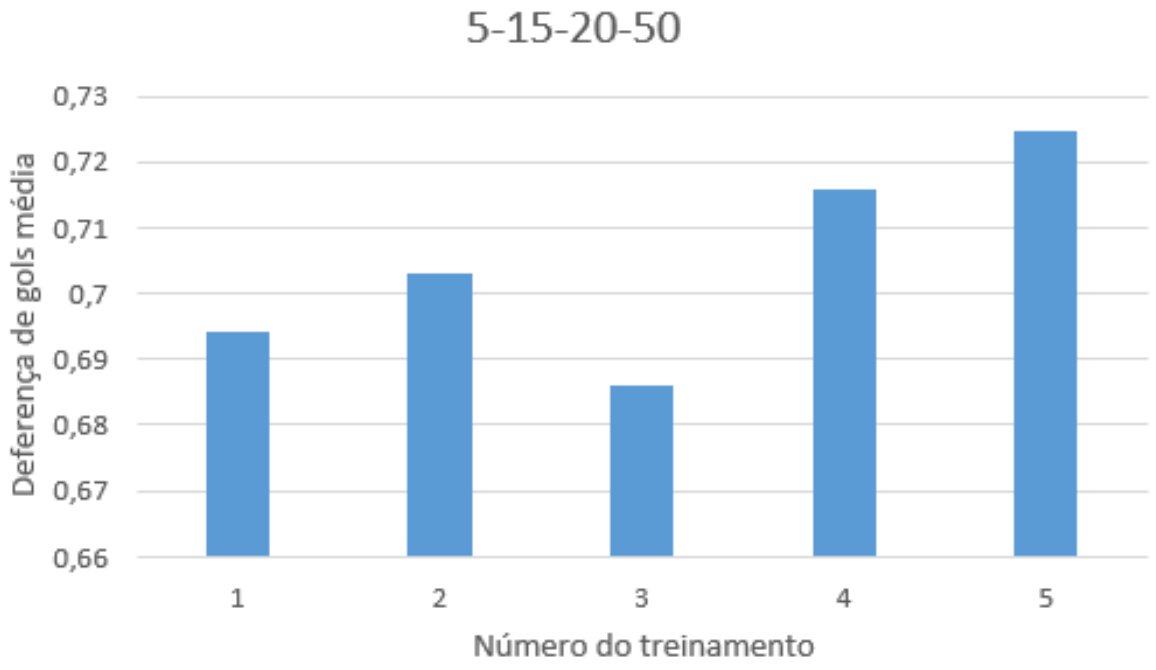

Figura 23 – Diferença de gols no cenário *run to score with keeper* para os experimentos do currículo 5-15-30-50. Fonte: autor

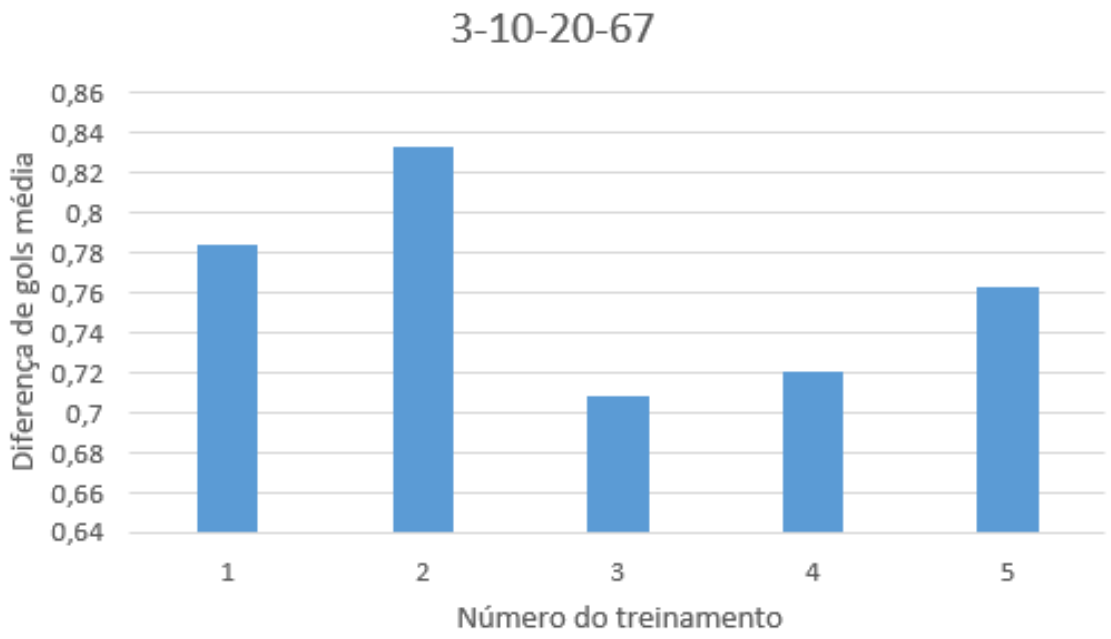

Figura 24 – Diferença de gols no cenário *run to score with keeper* para os experimentos do currículo 3-10-20-67. Fonte: autor

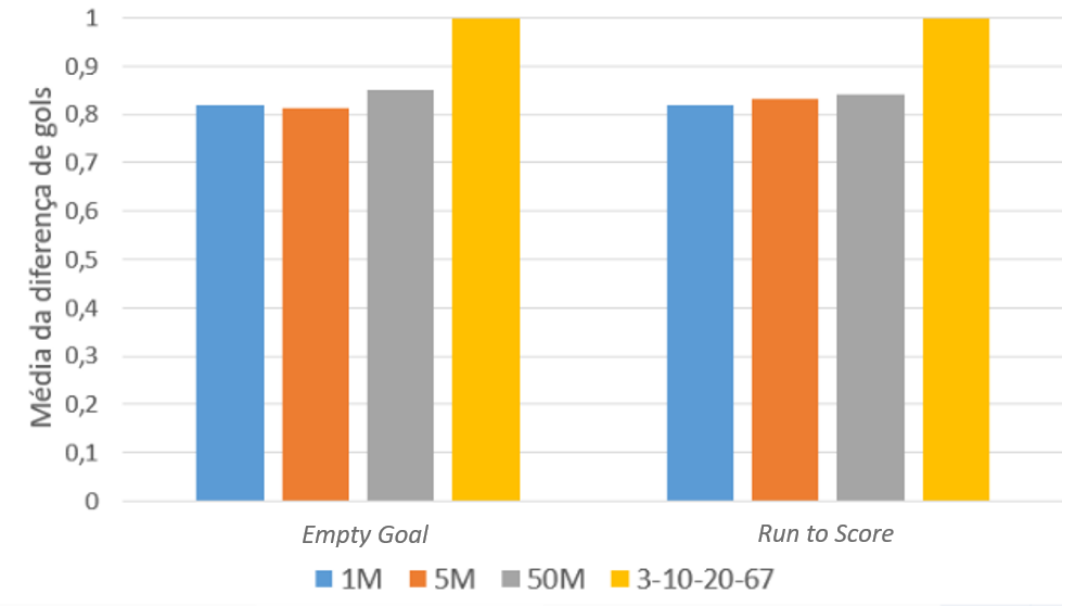

Figura 25 – Diferença de gols nos cenários *empty goal* e *run to score*, denominados de 2 e 3 respectivamente, utilizando o currículo 3-10-20-67. Em comparação com os resultados obtidos no artigo (KURACH et al., 2019) para os cenários *empty goal* e *run to score* treinados sem currículo. Fonte: autor

5 Conclusões e trabalhos futuros

5.1 Conclusão

Este trabalho abordou um problema do método de aprendizado por reforço *Policy Gradient*, que na maioria das vezes necessitam de uma grande quantidade de passos para adquirir um bom ”conhecimento”. Para solucionar esta dificuldade do método *Policy Gradient*, foi utilizado o aprendizado por currículo, que consiste em começar por partes menores do problema e assim conseguir obter um melhor resultado mais rapidamente. Utilizamos o aprendizado por currículo para acelerar o treinamento de um agente no cenário *Run to Score with Keeper* da *Football Academy* com 1 milhão de passos. Foram propostos dois currículos, denominados de 5-15-30-50 e 3-10-20-67, compostos pelos seguintes cenários: *Empty Goal Close*, *Empty Goal*, *Run to Score* e *Run to Score With Keeper*.

Mostramos que a utilização do currículo leva a um melhor resultado do que o treinamento feito apenas no último cenário (*Run to Score with Keeper*), sem currículo. O primeiro currículo obteve um resultado 23,58% melhor que a implementação sem a utilização de currículo, e o segundo currículo um resultado 33,57% melhor. Se considerarmos o cenário *Empty Goal* ou *Run to Score* como cenário principal do treinamento, obtemos o resultado melhor (máximo) e com menos de 1% dos passos utilizados em (KURACH et al., 2019). Esses resultados indicam a grande eficácia da utilização de currículo no aprendizado por reforço aplicado ao ambiente *Google Research Football*.

Destacamos que este trabalho exigiu uma grande quantidade de tempo de treinamento. Cada treinamento durou em média 18 horas, como foram realizados 15 execuções, totalizamos 270 horas de treinamento. Para obtenção de resultados em um jogo completo, são necessários 20 milhões de passos o que resultaria em aproximadamente 5.400 horas de treinamento ou 225 dias. Por isso, fica como trabalho futuro a utilização de um currículo para o jogo completo.

5.2 Trabalhos futuros

- Utilizar o aprendizado por currículo para treinamento de uma partida completa de futebol.
- Aplicação de um aprendizado por currículo automático, gerando um aprendizado mais eficiente. Em (MATIISEN et al., 2019) encontramos um currículo automático denominado *Teacher-Student Curriculum Learning*.

- Modificar a função de recompensa. O aperfeiçoamento desta função, também é uma importante abordagem que pode acelerar o aprendizado.

Referências

AGARAP, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. Citado na página 24.

AKIYAMA, H. et al. Helios2013 team description paper. In: *RoboCup 2016 Symposium and Competitions: Team Description Papers*. [S.l.: s.n.], 2016. Citado na página 14.

ANDERSEN, P.-A.; GOODWIN, M.; GRANMO, O.-C. Deep rts: a game environment for deep reinforcement learning in real-time strategy games. In: IEEE. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. [S.l.], 2018. p. 1–8. Citado na página 14.

AREL, I. et al. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, IET, v. 4, n. 2, p. 128–135, 2010. Citado na página 13.

BANSAL, M.; KRIZHEVSKY, A.; OGALE, A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. Citado na página 13.

BELLEMARE, M. G. et al. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, v. 47, p. 253–279, 2013. Citado na página 14.

BENGIO, Y. et al. Curriculum learning. In: ACM. *Proceedings of the 26th annual international conference on machine learning*. [S.l.], 2009. p. 41–48. Citado 3 vezes nas páginas 5, 12 e 28.

BERNSTEIN, D. S. et al. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, INFORMS, v. 27, n. 4, p. 819–840, 2002. Citado na página 17.

BU, X.; RAO, J.; XU, C.-Z. A reinforcement learning approach to online web systems auto-configuration. In: IEEE. *2009 29th IEEE International Conference on Distributed Computing Systems*. [S.l.], 2009. p. 2–11. Citado na página 13.

CHATFIELD, K. et al. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. Citado na página 24.

CUÉLLAR, A. G. Use of graphs with bayesian networks and q-learning for pass-into-space decision making in robocup soccer simulation 2d. 2012. Citado na página 30.

ELMAN, J. L. Learning and development in neural networks: The importance of starting small. *Cognition*, Elsevier, v. 48, n. 1, p. 71–99, 1993. Citado na página 28.

FABRO, J. A.; REIS, L. P.; LAU, N. Using reinforcement learning techniques to select the best action in setplays with multiple possibilities in robocup soccer simulation teams. In: IEEE. *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*. [S.l.], 2014. p. 85–90. Citado na página 15.

FARAHNAKIAN, F.; MOZAYANI, N. Reinforcement learning for soccer multi-agents system. In: IEEE. *2009 International Conference on Computational Intelligence and Security*. [S.l.], 2009. v. 2, p. 50–52. Citado 2 vezes nas páginas 12 e 15.

GARCIA, F.; RACHELSON, E. Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, Wiley Online Library, p. 1–38, 2013. Citado na página 17.

GIVAN, B.; PARR, R. An introduction to markov decision processes. *Purdue University*, 2001. Citado na página 18.

HAHNLOSER, R. H. et al. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, Nature Publishing Group, v. 405, n. 6789, p. 947, 2000. Citado na página 24.

HAUSKNECHT, M.; STONE, P. Deep recurrent q-learning for partially observable mdps. In: *2015 AAAI Fall Symposium Series*. [S.l.: s.n.], 2015. Citado na página 14.

ITSUKI, N. Soccer server: a simulator for robocup. In: CITESEER. *JSAI AI-Symposium 95: Special Session on RoboCup*. [S.l.], 1995. Citado na página 30.

JIN, J. et al. Real-time bidding with multi-agent reinforcement learning in display advertising. In: ACM. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. [S.l.], 2018. p. 2193–2201. Citado na página 14.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996. Citado na página 18.

KIM, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. Citado na página 19.

KITANO, H. *RoboCup-97: robot soccer world cup I*. [S.l.]: Springer Science & Business Media, 1998. v. 1395. Citado 2 vezes nas páginas 18 e 28.

KITANO, H. et al. Robocup: A challenge problem for ai. *AI magazine*, v. 18, n. 1, p. 73–73, 1997. Citado na página 30.

KITANO, H. et al. The robocup synthetic agent challenge 97. In: SPRINGER. *Robot Soccer World Cup*. [S.l.], 1997. p. 62–73. Citado na página 30.

KOBER, J.; BAGNELL, J. A.; PETERS, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 32, n. 11, p. 1238–1274, 2013. Citado na página 13.

KOK, J. R. et al. Uva trilearn 2003 team description. *Proceedings CD RoboCup*, v. 2003, 2003. Citado na página 15.

KRÖSE, B. et al. An introduction to neural networks. Citeseer, 1993. Citado na página 19.

KRUEGER, K. A.; DAYAN, P. Flexible shaping: How learning in small steps helps. *Cognition*, Elsevier, v. 110, n. 3, p. 380–394, 2009. Citado na página 28.

KURACH, K. et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019. Citado 14 vezes nas páginas 5, 6, 7, 12, 16, 31, 32, 34, 36, 38, 40, 41, 43 e 44.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Taipei, Taiwan, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 19.

LEVINE, S. et al. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, JMLR. org, v. 17, n. 1, p. 1334–1373, 2016. Citado na página 13.

MAO, H. et al. Resource management with deep reinforcement learning. In: ACM. *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. [S.l.], 2016. p. 50–56. Citado na página 13.

MATIISEN, T. et al. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, IEEE, 2019. Citado 4 vezes nas páginas 5, 6, 12 e 44.

MNIH, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. Citado na página 14.

MNIH, V. et al. Human-level control through deep reinforcement learning. *Nature*, Nature Publishing Group, v. 518, n. 7540, p. 529, 2015. Citado na página 14.

NARDI, D. et al. Robocup soccer leagues. *AI Magazine*, v. 35, n. 3, p. 77–85, 2014. Citado na página 29.

NERI, J. R. F. et al. A proposal of qlearning to control the attack of a 2d robot soccer simulation team. In: IEEE. *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*. [S.l.], 2012. p. 174–178. Citado 4 vezes nas páginas 7, 14, 15 e 18.

OU, C. X.; DAVISON, R. M. Technical opinion-why ebay lost to taobao in china: The global advantage. *Communications of the ACM*, ACM Special Interest Group, v. 52, n. 1, p. 145–148, 2009. Citado na página 14.

PUTERMAN, M. L. Markov decision processes. *Handbooks in operations research and management science*, Elsevier, v. 2, p. 331–434, 1990. Citado na página 17.

RABIEE, A.; GHASEM-AGHAEE, N. A scoring policy for simulated soccer agents using reinforcement learning. In: CITESEER. *2nd International Conference on Autonomous Robots and Agents, New Zealand*. [S.l.], 2004. Citado 3 vezes nas páginas 12, 15 e 18.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, MIT Press, v. 29, n. 9, p. 2352–2449, 2017. Citado 3 vezes nas páginas 19, 22 e 23.

SCHULMAN, J. et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. Citado 4 vezes nas páginas 12, 25, 27 e 28.

SEN, S.; WEISS, G. Learning in multiagent systems. *Multiagent systems: A modern approach to distributed artificial intelligence*, MIT Press Cambridge, MA, p. 259–298, 1999. Citado na página 18.

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, 2016. Citado na página 14.

SINGH, S.; OKUN, A.; JACKSON, A. Artificial intelligence: Learning to play go from scratch. *Nature*, Nature Publishing Group, v. 550, n. 7676, p. 336, 2017. Citado na página 14.

SUTTON, R. S.; BARTO, A. G. Reinforcement learning: An introduction. Cambridge, MA: MIT Press, 2011. Citado 2 vezes nas páginas 12 e 18.

SUTTON, R. S. et al. Policy gradient methods for reinforcement learning with function approximation. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2000. p. 1057–1063. Citado na página 12.

VINYALS, O. et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. Citado na página 14.

WHITE, C. *Markov decision processes*. [S.l.]: Springer, 2001. Citado na página 12.

WU, Y.; TIAN, Y. Training agent for first-person shooter game with actor-critic curriculum learning. 2016. Citado na página 28.

XIONG, L. et al. A new passing strategy based on q-learning algorithm in robocup. In: IEEE. *2008 International Conference on Computer Science and Software Engineering*. [S.l.], 2008. v. 1, p. 524–527. Citado 2 vezes nas páginas 14 e 30.

YOON, M.; BEKKER, J.; KROON, S. New reinforcement learning algorithm for robot soccer. *orion*, ORSSA, v. 33, n. 1, p. 1–20, 2017. Citado na página 15.

ZAREMBA, W.; SUTSKEVER, I. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014. Citado na página 12.

ZHENG, G. et al. Drn: A deep reinforcement learning framework for news recommendation. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 2018 World Wide Web Conference*. [S.l.], 2018. p. 167–176. Citado na página 13.

ZHOU, Z.; LI, X.; ZARE, R. N. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, ACS Publications, v. 3, n. 12, p. 1337–1344, 2017. Citado na página 13.