UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

# Aprendizado por reforço profundo para robótica social usando sinais sociais e emoções faciais

**José Pedro Ribeiro Belo**

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

**José Pedro Ribeiro Belo**

# Aprendizado por reforço profundo para robótica social usando sinais sociais e emoções faciais

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Roseli Aparecida Francelin Romero

**USP – São Carlos**
**Fevereiro de 2023**

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

BB452a

Belo, José Pedro Ribeiro
Aprendizado por reforço profundo para robótica social usando sinais sociais e emoções faciais / José Pedro Ribeiro Belo; orientadora Roseli Aparecida Francelin Romero. -- São Carlos, 2023.
151 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. aprendizado por reforço profundo. 2. interação humano-robô. 3. robótica social. 4. emoções. 5. sinais sociais. I. Romero, Roseli Aparecida Francelin, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

**José Pedro Ribeiro Belo**

# Deep reinforcement learning for social robotics using social signals and facial emotions

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Roseli Aparecida Francelin Romero

**USP – São Carlos**
**February 2023**

*Dedico este trabalho aos meus pais,*
*que sempre me deram apoio, força e incentivo*
*para a realização dos meus sonhos.*

# AGRADECIMENTOS

Primeiramente agradeço aos meus pais, Aparecida e José Carlos, que me deram base e força para a conclusão desta etapa em minha vida. Agradeço também à toda minha família, por todo apoio, incentivo e carinho.

Em memória aos que se foram durante esta jornada, minha madrinha Edneia, meus avôs José Estevam, Therezinha, Pedro e Albertina.

Agradeço à minha noiva, Ana Paula, por todo apoio, carinho, compreensão, paciência e companheirismo. Sem seu suporte, eu jamais teria conseguido.

Agradeço, especialmente, à Prof.ª Dr.ª Roseli Romero por ter me orientado, pelo incentivo e por confiar no meu trabalho ao longo do desenvolvimento desta Tese de Doutorado.

Agradeço também à Helio Azevedo, que sempre se dispôs a ajudar e contribuir tanto nesta tese, quanto em vários outros assuntos. À Josué Ramos, que também teve grande papel no desenvolvimento desta tese.

Agradeço também aos colegas do grupo de Computação Bioinspirada (Biocom) e do Laboratório de Aprendizado de Robôs (LAR), pela disposição em ajudar, discutir ideias, pela amizade e companheirismo, em especial à Iury Andrade, Felipe Padula, Gean Trindade, Guilherme Nardari, Saulo Mastelini, Moisés Santos, Angélica Ribeiro e Kenzo Sakiyama.

Aos professores do ICMC, que sempre estiveram dispostos a ensinar, contribuindo para a minha formação e de certa forma para o projeto. Agradeço a todos os colegas dentro e fora do ICMC, com os quais tive a oportunidade de conviver e trocar experiências.

Por fim, agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo nº 2018/25782-5, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo financiamento total e parcial desta tese. Muito obrigado!

*"O universo não foi feito à medida do ser humano,*
*mas tampouco lhe é adverso:*
*é-lhe indiferente"*
*(Carl Sagan)*

# RESUMO

BELO, J. P. R. **Aprendizado por reforço profundo para robótica social usando sinais sociais e emoções faciais**. 2023. 151 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

A robótica social representa um ramo da interação humano-robô dedicado ao desenvolvimento de sistemas para controlar os robôs para operar em ambientes não estruturados com a presença de seres humanos. Robôs sociais devem interagir com seres humanos entendendo sinais sociais e respondendo adequadamente a eles. A maioria dos robôs sociais ainda são pré-programados, não tendo grande capacidade de aprender e responder com ações adequadas durante uma interação com humanos. Métodos mais elaborados usam movimentos corporais, direção do olhar e linguagem corporal. Nesta tese os sinais socialmente aceitáveis comumente utilizados durante uma interação são considerados para o treinamento de um robô social. Um sistema inteligente foi desenvolvido para tornar um robô capaz de decidir, de forma autônoma, quais comportamentos emitir em função do estado emocional humano. Para isso, a primeira contribuição deste trabalho é uma arquitetura denominada Social Robotics Deep Q-Network (SocialDQN) é proposta para ensinar robôs sociais a se comportarem e interagirem adequadamente com humanos com base em sinais sociais, especialmente em estados emocionais humanos. Ela oferece um arcabouço para a utilização de sinais sociais visando controlar as ações do robô e seu aprendizado é realizado por meio de *Deep Reinforcement Learning*(DRL). Uma segunda contribuição é o simulador SimDRLSR, que é o primeiro simulador a prover uma ferramenta para modelar humanos e seus comportamentos por meio de sinais sociais. O desenvolvimento e validação da rede SocialDQN foram realizados com o apoio desse simulador. Os resultados obtidos em diversos testes realizados em ambiente real demonstraram que o sistema aprendeu satisfatoriamente a maximizar as recompensas e, consequentemente, o robô se comportou de forma socialmente aceitável.

**Palavras-chave:** aprendizado por reforço profundo, interação humano robô, robótica social, emoções, sinais sociais, simulador.

# ABSTRACT

BELO, J. P. R. **Deep reinforcement learning for social robotics using social signals and facial emotions**. 2023. 151 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Social robotics represents a branch of human-robot interaction dedicated to developing systems to control robots to operate in unstructured environments in the presence of humans. Social robots must interact with humans by understanding social signals and responding appropriately. Most social robots are still pre-programmed, unable to learn and respond with appropriate actions during an interaction with humans. More elaborate methods use body movements, gaze direction, and body language. In this thesis, the socially acceptable signals commonly used during an interaction are considered for the training of a social robot. An intelligent system was developed to make a robot capable of autonomously deciding which behaviors to emit depending on the human emotional state. For this, the first contribution of this work is an architecture called Social Robotics Deep Q-Network (SocialDQN) is proposed to teach social robots to behave and interact appropriately with humans based on social signals, especially in human emotional states. It offers a framework for the use of social signals to control the robot's actions and its learning is carried out through Deep Reinforcement Learning (DRL). A second contribution is the SimDRLSR simulator, which is the first simulator to provide a tool to model humans and their behavior through social signals. The development and validation of the SocialDQN network were carried out with the support of this simulator. The results obtained in several tests carried out in a real environment showed that the system learned satisfactorily to maximize rewards and, consequently, the robot behaved in a socially acceptable way.

**Keywords:** Deep reinforcement learing, human-robot interaction, social robotics, emotions, social signs.

# LISTA DE ILUSTRAÇÕES

# LISTA DE QUADROS

# LISTA DE ALGORITMOS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AL | *Active Learning* |
| ConvNets | *Convolutional Neural Networks* |
| DL | *Deep Learning* |
| DNN | *Deep Neural Networks* |
| DQN | *Deep Q-Network* |
| DRL | *Deep Reinforcement Learning* |
| DRL-IM | DRL intrinsecamente motivada |
| FER | *Facial Expression Recognition* |
| FOV | *Field of View* |
| GPL | *General Public License* |
| IA | Inteligência Artificial |
| IHR | Interação Humano-Robô |
| IRL | *Interactive Reinforcement Learning* |
| LSTM | *Long Short-Term Memory* |
| MDP | *Markov Decision Processes* |
| MDQN | *Multimodal Deep-Q-Network* |
| ML-Agents | *Unity Machine Learning Agents Toolkit* |
| MLP | *Multilayer Perceptron* |
| MTCNN | *Multi-task Cascaded Convolutional Networks* |
| ReLU | *rectified linear unit* |
| ResNet | *Residual Network* |
| RHS | *Robot House Simulator* |
| RL | *Reinforcement Learning* |
| RNAs | Redes Neurais Artificiais |
| RNN | *Recurrent Neural Networks* |
| SimDRLSR | S*imulator for Deep Reinforcement Learning and Social Robotics* |
| SocialDQN | *Deep Q-Network for Social Robotics* |

# SUMÁRIO

CAPÍTULO

# 1

# INTRODUÇÃO

Inicialmente aplicada predominantemente em ambientes industriais, a robótica está cada vez mais presente em ambientes residenciais e domésticos. Os robôs deixaram de ser meras ferramentas para se tornarem assistentes nas tarefas cotidianas em parceria direta com os seres humanos. Para permitir esta evolução, a comunidade de robótica deve enfrentar vários desafios, em particular, na área de Interação Humano-Robô (IHR) e, especialmente, na subárea de Robótica Social (FONG; NOURBAKHSH; DAUTENHAHN, 2003).

IHR é um campo de estudo dedicado a entender, projetar e avaliar sistemas robóticos para interagir e ser usado por humanos (GOODRICH; SCHULTZ, 2007). A robótica social direciona seu interesse para as interações entre humanos e robôs, estabelecendo uma comunicação semelhante à utilizada entre humanos. As aplicações da robótica social vão desde tarefas que envolvem busca e salvamento até educação, robótica assistiva e diversas outras aplicações que requerem alguma habilidade social.

A partir do momento que robôs e humanos se comunicam, é desejável que esta interação ocorra no mesmo nível cognitivo (SPARC Robotics, 2016). A cognição é o processo sistêmico que permite que um determinado agente entenda como as coisas podem ser, não apenas agora, mas em algum momento no futuro, usando esse entendimento para influenciar suas ações. Prever o futuro exige que o robô se lembre do passado, então o aprendizado é fundamental para todos os sistemas cognitivos (SPARC Robotics, 2016).

A capacidade de interpretar sinais emocionais é essencial para a interação social (KESSELS *et al.*, 2014). As emoções podem auxiliar na interpretação dos estados internos de um indivíduo e, consequentemente, auxiliar na previsão de ações futuras (BREAZEAL; DAUTENHAHN; KANDA, 2016). As emoções são categorizadas como puramente cognitivas, a representação mental de uma experiência emocional inclui componentes motores e viscerais, bem como componentes cognitivos (DANTZER, 1993). Assim, os robôs devem reconhecer e interpretar sinais emocionais e mapear internamente essas informações para interagir com

humanos.

Também é desejável que robôs sociais possam interagir com humanos entendendo sinais sociais e respondendo apropriadamente para promover uma interação "natural" entre humanos e robôs. Essa habilidade depende tanto dos níveis de habilidade de interação quanto dos níveis de habilidade cognitiva (SPARC Robotics, 2016). A maioria dos robôs sociais ainda é pré-programada, sem a capacidade de aprender e atualizar informações (QURESHI *et al.*, 2016). Métodos mais elaborados usam movimentos corporais, direção do olhar e linguagem corporal. No entanto, esses métodos geralmente negligenciam os sinais de interação necessários, como estado emocional e sinais sociais mais complexos. Esses sinais podem fornecer informações vitais para avaliar o sucesso da interação do usuário.

O uso de Aprendizado por Reforço (do inglês, *Reinforcement Learning* (RL)) representa uma forma de robôs sociais aprenderem a se comportar e interagir adequadamente com humanos, seguindo um paradigma de autoaprendizagem, buscando maximizar algum aspecto da interação. Por meio de técnicas de Inteligência Artificial (IA), como Redes Neurais Profundas (do inglês, *Deep Neural Networks* (DNN)) ou *Deep Learning* (DL), os robôs podem capturar dados do ambiente com muito mais detalhes, abstraindo padrões complexos de informações visuais, incluindo informações de cognição social avançadas extraídas automaticamente. A combinação dessas duas técnicas é conhecida como Aprendizado por Reforço Profundo (do inglês, *Deep Reinforcement Learning* (DRL)).

Em RL, os agentes dependem da execução do maior número possível de ações em diferentes configurações e condições do ambiente, a fim de explorar e aprender relações entre estados e ações. Dada esta particularidade, a utilização de sistemas computacionais para simular ambientes e agente robóticos pode ser uma alternativa plausível. Além disto, a área da robótica, em geral, aproveita da vantagem dos simuladores em prover um ambiente controlado para desenvolvimento, treinamento, teste e validação de sistemas sem o uso desnecessário de robôs reais, pois isto pode reduzir a vida útil do equipamento. Por este e outros motivos, a utilização de simuladores robóticos é uma prática adotada comumente na área de RL.

Desta forma, esta tese de doutorado está associada ao desenvolvimento uma arquitetura baseada em DRL e *Deep Q-Network* (DQN) para que robôs sociais possam identificar comportamentos interativos humanos e agir adequadamente considerando sinais sociais, como foco de atenção e emoções, e de imagens capturadas do ambiente. O robô deve aprender a identificar quando o humano está disposto a interagir e qual ação selecionar para cada caso a partir desses sinais. Em relação às emoções, foram adotadas as seis emoções básicas de Ekman (EKMAN; FRIESEN, 1971), expressas através do rosto humano e são independentes da cultura. Essas emoções básicas são: *felicidade*, *medo*, *nojo*, *surpresa*, *raiva* e *tristeza*. A arquitetura desenvolvida é denominada *Deep Q-Network for Social Robotics* (SocialDQN)[1].

[1] Em português, Rede Q-Profunda para Robótica Social

Além da arquitetura SocialDQN, nesta tese também é proposto um simulador denominado S*imulator for Deep Reinforcement Learning and Social Robotics* (SimDRLSR)[2]. O SimDRLSR oferece um ambiente simulado para teste e desenvolvimento de métodos que seguem o paradigma de aprendizado não supervisionado na robótica social. O objetivo é fornecer um ambiente para o desenvolvimento de técnicas de aprendizado automático de comportamentos interativos humanos e, a partir daí, reagir naturalmente e adequada a esses comportamentos visando a aplicação em robôs sociais. Em particular, este simulador serve como alicerce para desenvolvimento, teste e validação da SocialDQN.

O ambiente de simulação é uma implementação derivada do *Robot House Simulator* (RHS) (BELO; AZEVEDO; ROMERO, 2017; BELO; AZEVEDO; ROMERO, 2018), o qual foi desenvolvido a partir do motor de desenvolvimento de jogos Unity 3D (Unity Technologies, 2022). O RHS é uma proposta implementada no Laboratório de Robôs (LAR) do ICMC-USP, voltado para validação de arquiteturas cognitivas para robótica social.

A seguir, os objetivos, as justificativas, contribuições e limitações da arquitetura proposta serão apresentados.

## 1.1 Objetivo

O objetivo desta tese é o desenvolvimento de uma sistema computacional capaz de prover uma arquitetura de treinamento, teste e validação para que robôs sociais se portarem de forma socialmente aceita a comportamentos interativos de seres humanos. Para isto, são considerados sinais sociais, em especial a emoção humana.

Esse objetivo é alcançado através da definição e desenvolvimento de dois módulos computacionais, uma arquitetura DQN para robótica social, a SocialDQN, e do simulador SimDRLSR. O primeiro visa abstrair informações a partir de imagens e sinais sociais, buscando aprender padrões comportamentais de humanos para guiar as ações do robô. O segundo define um ambiente simulado, capaz de emular humanos agindo no ambiente, bem como um robô social hábil para interagir com estes humanos. A utilização de um simulador traz diversos benefícios ao treinamento de sistemas inteligentes, como agilizar, paralelizar e padronizar o processo de treinamento, além de permitir definir um *baseline* para validação do sistema com diferentes parâmetros.

Nesta tese, são investigadas as técnicas de RL e DL para aprender comportamentos interativos de seres humanos e capacitar o robô a se comportar adequadamente e natural a estes estímulos. Sinais sociais provenientes de características emocionais extraídas da face de humanos são consideradas. A arquitetura desenvolvida se difere dos demais sistemas de RL para robótica social, explorando sinais sociais que envolvem a emoção facial e foco de atenção. Ademais, a

[2] Em português, Simulador para Aprendizado por Reforço Profundo e Robótica Social

proposta do simulador SimDRLSR é inédita na área de robótica social e aprendizado por reforço envolvendo interação e compreensão de comportamentos humanos.

A arquitetura denominada *Multimodal Deep-Q-Network* (MDQN), proposta em Qureshi *et al.* (2016), é utilizada para guiar o desenvolvimento inicial e validar o simulador SimDRLSR. Esta integração visa utilizar a MDQN como *benchmark* para validar comportamentos humano e robótico na simulação, bem como verificar a capacidade inicial do simulador em prover um ambiente para desenvolvimento do módulo SocialDQN. Após esta validação, a evolução da SocialDQN é dada com auxílio do simulador, inicialmente buscando alcançar os resultados obtidos na MDQN, e após isso, refinar e implementar módulos para aperfeiçoar e otimizar o aprendizado do agente.

Esta proposta também tem como objetivo realizar uma transferência de conhecimento da SocialDQN, treinada a partir do simulador SimDRLSR, para o robô Pepper, da *Softbank Robotics*. Nesta atividade, são também verificadas a capacidade do arquitetura proposta para atuação em robôs sociais reais, bem como levantar eventuais limitações da abordagem proposta.

O objetivo principal pode ser detalhado através dos seguintes itens a serem desenvolvidos:

- criar um módulo de aprendizado comportamental adaptativo por meio de técnicas de aprendizado por reforço e aprendizagem profunda, DRL;
- desenvolvimento do simulador SimDRLSR, a partir do motor de jogos Unity 3D, visando atender os requisitos para treinar, testar e validar a MDQN (inicialmente) e a SocialDQN (posteriormente);
- integrar a arquitetura MDQN (QURESHI *et al.*, 2016) ao modelo proposto, o que envolve adaptar e atualizar *frameworks* e bibliotecas para isto;
- modelar agente robótico e humanos no simulador;
- implementar ações e comportamentos para o robô e humanos simulados;
- desenvolver módulos de captura de imagens em escala de cinza e profundidade no simulador;
- modelar comportamentos interativos nos agentes simulados;
- definir funções de recompensa e penalidade;
- integração e comunicação entre módulos;
- testar e validar o simulador SimDRLSR na arquitetura MDQN de Qureshi *et al.* (2016);
- implementar a arquitetura da rede de aprendizado da SocialDQN;
- mapeamento dos estados do ambiente através de imagens e características emocionais;

- desenvolver um detector de eventos de modo a auxiliar no mapeamento do estado do ambiente e modelagem da recompensa;
- testar e validar a SocialDQN no simulador SimDRLSR;
- desenvolver e utilizar sub-módulos no robô Pepper para dar suporte na interação com o ambiente e usuários; e
- testar e validar o aprendizado da SocialDQN a partir do robô Pepper em um ambiente real.

## 1.2 Justificativa

Dada a complexidade em classificar comportamentos humanos, o treinamento de modelos utilizando algoritmos de aprendizado supervisionado se torna uma tarefa quase impraticável. Seres humanos conseguem detectar a intenção de interação de outras pessoas utilizando sinais sociais, tais como foco de atenção, expressões faciais, direção do movimento, entre vários outros. Dada a dificuldade de extrair e mapear tais comportamentos, os algoritmos de aprendizado por reforço podem oferecer uma solução interessante para prover ao robô a capacidade de decidir (ação) baseada no comportamento humano. Desta forma, o robô aprende com seus acertos e erros recebendo ou perdendo recompensas.

Contudo, algoritmos tradicionais de RL por si sós não conseguem extrair informações complexas de comportamentos humanos, necessitando de técnicas mais robustas para tal tarefa. Nos últimos anos, os algoritmos de DL vem apresentando resultados muito eficientes para extrair informações em dados topológicos, como imagens, sem a necessidade de pré-processar estes dados. Dada a junção das duas técnicas, RL e DL, é possível utilizar a visão para o que robô abstraia comportamentos complexos para o problema abordado.

Outro ponto abordado nesta tese é utilização de emoções humanas para auxiliar no aprendizado do agente. Segundo Kessels *et al.* (2014), a capacidade de interpretar sinais emocionais é essencial para a interação social. Já Breazeal, Dautenhahn e Kanda (2016) cita que as emoções ajudam na previsão de ações durante uma interação humana. A SocialDQN tem em vista tirar proveito destas questões a partir de técnicas e modelos de reconhecimento de emoções faciais disponíveis na literatura.

A utilização de robôs físicos, interagindo diretamente um ambiente com humanos reais, pode demandar um custo inviável, pois o treinamento expõe o agente robótico a desgastes físicos e as pessoas envolvidas no treinamento do robô podem perder o interesse na interação. Isto é agravado pelo fato de sistemas de RL iniciarem suas interações de forma aleatória no início do aprendizado, o que pode gerar pouca aceitação social. Como mencionado anteriormente, é possível resolver estas questões por meio de simuladores para robótica. Entretanto, a área de robótica social ainda sofre com a ausência de simuladores que moldem comportamentos interati-

vos humanos. Dados os problemas e lacunas apresentadas, a formulação e desenvolvimento de um simulador para robótica social e RL, através do SimDRLSR, se justifica.

## 1.3 Contribuições e limitações

Nesta tese, é proposto o desenvolvimento de dois módulos, visando contribuir na área de robôs sociais com paradigma de autoaprendizagem a partir de informações de alta dimensionalidade. Apesar de ambos os módulos se complementarem, cada um pode beneficiar diferentes campos de pesquisa e atuação.

Usualmente, a área de RL tira proveito de simuladores, jogos eletrônicos e ambientes emulados. Contudo, poucos são voltados para a área de robótica social com foco em emular comportamentos humanos. Desta forma o simulador contribui fornecendo um ambiente de simulação de humanos e comportamentos interativos. A proposta e elaboração deste ambiente simulado consiste na primeira grande contribuição desta tese.

Já a arquitetura SocialDQN, contribui diretamente para que robôs aprendam comportamentos complexos de seres humanos. Isto é possível graças ao advento das DNN, que visam abstrair padrões de informações de alta-dimensionalidade. Adicionalmente, a SocialDQN tira vantagens através de sinais sociais pertinentes à interação, como mencionado anteriormente.

Esta tese segue um ciclo de desenvolvimento que envolve a elaboração das arquiteturas, modelagem, desenvolvimento e a validação. A SocialDQN é validada a partir do treinamento e testes no simulador SimDRLSR. Por sua vez, o simulador é inicialmente validado a partir da MDQN. Todos os códigos, configurações e *scripts* associados ao ciclo de desenvolvimento foram liberados conforme a licença GNU *General Public License* (GPL) v3.0, no repositório GitHub:

- SocialDQN: github.com/JPedroRBelo/SocialDQN
- SimDRLSR: github.com/JPedroRBelo/simDRLSR
- MDQN em Python:github.com/JPedroRBelo/pyMDQN
- Ferramenta de Validação: github.com/JPedroRBelo/validation_tool_socialdqn

A liberação dos códigos-fonte abertamente à comunidade contribui para a evolução do projeto por outros pesquisadores, permite amparar o desenvolvimento e validação de técnicas e métodos na área de robótica social e IA, além de possibilitar a reprodução dos resultados alcançados nesta tese.

Os resultados obtidos representam uma evolução de modelos promissores na área de DRL e robótica social, por parte da SocialDQN. Por outro lado, o simulador SimDRLSR representa os primeiros passos para elaboração de um ambiente para emular comportamentos humanos, em especial comportamento de interação, percepção e emissão de sinais sociais. Nesse contexto, é

possível levantar algumas limitações nesta abordagem inicial, especialmente no que tange ao mapeamento da atuação humana.

No SimDRLSR, os comportamentos humanos são simplificados e representam um esforço inicial em oferecer suporte para algorítimo de IA para interação com humanos. Esta simplificação se baseia na observação de que, segundo Breazeal (2002), as intenções humanas são dissimuladas e seus comportamentos são moldados por normas sociais, dificultando o mapeamento destes comportamentos em modelos computacionais.

Outra limitação é dada através da simplificação dos sinais sociais utilizados. Nesta tese, a emoção humana e foco de atenção são utilizadas para moldar estes sinais. Apesar da emoção humana auxiliar na previsão de ações futuras de seres humanos (BREAZEAL; DAUTENHAHN; KANDA, 2016), esta característica pode variar conforme o ambiente, foco de atenção, atividade, etc. do mesmo. Entretanto, a utilização de sinais sociais tem um intuito te contribuir no aprendizado do agente em paralelo à utilização de imagens do ambiente, ou seja, o agente deve utilizar este sinal social como uma "dica" para interagir com o humano. Além disso, a SocialDQN permite a configuração e utilização de vários outros sinais sociais, dependendo exclusivamente de detectores externos destes sinais. Vale ressaltar que estes detectores não são foco direto de estudo nesta tese, utilizado somente algoritmos disponíveis na literatura para reconhecimento de emoções no robô real.

## 1.4 Organização do Trabalho

Esta tese está organizada da seguinte forma. No Capítulo 2, alguns conceitos técnicos e teóricos utilizados são descritos, tais como RL, DL, DRL e emoções. Já no Capítulo 3 é realizada uma revisão na literatura sobre pesquisas envolvendo robótica social, autoaprendizado, sinais sociais e emoções. Na sequência, no Capítulo 4, são apresentados os materiais, métodos e ferramentas adotados nesta tese, como a MDQN, simulador RHS, robô Pepper, reconhecimento de emoções, etc.

Ainda no Capítulo 4, também é apresentada a arquitetura proposta, envolvendo o simulador SimDRLSR e o modelo SocialDQN. A circunstanciação do simulador SimDRLSR é realizada no Capítulo 5. Em seguida, no Capítulo 6, é apresentada a arquitetura SocialDQN, bem como as emoções consideradas, mapeamento do ambiente, recompensas utilizadas, ações, entre outros aspectos. É apresentado também o desenvolvimento do módulo de interação com o robô real. No Capítulo 7, é descrita uma série de experimentos relativos à SocialDQN, treinada no simulador SimDRLSR. Inicialmente, a MDQN é utilizada tanto para validar o simulador quanto a SocialDQN. Após isto, a SocialDQN é testada sob diferentes aspectos através do SimDRLSR. Dentre estes, é verificada a aceitação social da rede SocialDQN em simulação em relação à MDQN. Em seguida, são apresentados experimentos com a rede SocialDQN interagindo em um ambiente com várias pessoas, onde também foram realizados testes de validação social. O capí-

tulo é finalizado com várias discussões sobre os resultados obtidos. Finalmente, no Capítulo 8, são apresentadas as conclusões, artigos originados desta tese e trabalhos futuros.

CAPÍTULO

2

# REFERENCIAL TEÓRICO

Neste Capítulo, são apresentados os fundamentos teóricos relacionados ao tema de pesquisa desta tese de doutorado. Os assuntos abordados neste capítulo são, aprendizado por reforço, redes Neurais profundas, aprendizado por reforço profundo, e, por fim, emoções humanas, descrevendo como estas podem ser mapeadas e reconhecidas. Este último tópico visa auxiliar na detecção de eventos e ajuste do aprendizado do sistema proposto.

## 2.1 Aprendizado por reforço

Os algoritmos de RL, ou aprendizado por reforço, procuram maximizar um sinal de recompensa ao executar ações dadas as situações (estados). Neste contexto, um agente deve descobrir quais ações geram melhores recompensas sem informações prévias, isto através de tentativa e erro. Em casos mais elaborados, as ações podem afetar as recompensas imediatas e subsequentes para estados futuros (SUTTON; BARTO, 2018).

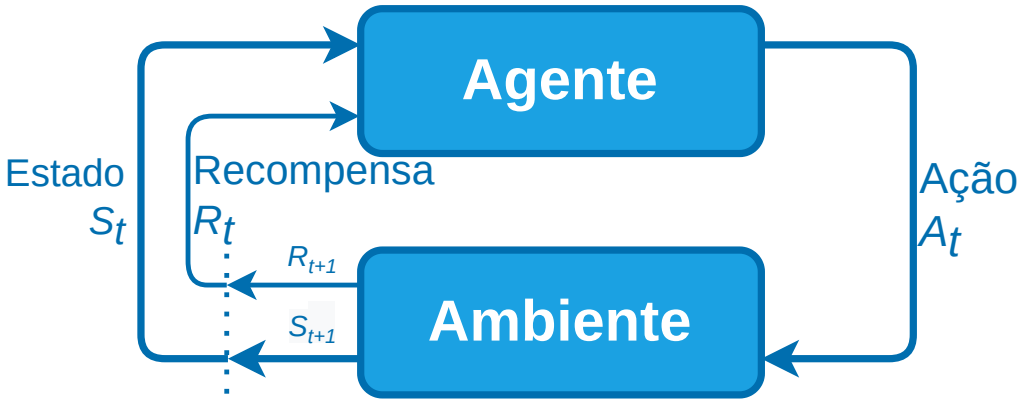
Estes algoritmos visam guiar o aprendizado de um agente na ausência de exemplos rotulados, ou seja, aprender de forma não-supervisionada (NORVIG; RUSSELL, 2014). Em oposição, o aprendizado supervisionado tem como característica a disposição de conjuntos de treinamento com exemplos rotulados fornecidos por um supervisor externo. Neste caso, cada exemplo descreve uma situação, seguido de uma classificação (rótulo), a qual indica a ação que o algoritmo deve executar para dada situação. O objetivo é fazer com que o sistema consiga generalizar suas predições para que ele atue corretamente em situações não exploradas no conjunto de treinamento. Em problemas interativos, às vezes é impraticável coletar exemplos do comportamento desejado que representam todas as situações nas quais os agentes precisam agir. Desta forma, em um ambiente desconhecido, um agente deve conseguir aprender sozinho com sua própria experiência (SUTTON; BARTO, 2018).

Segundo Sutton e Barto (2018), a compreensão de RL depende muito do conceito de

estado, o qual funciona como entrada para a função de política e valor. A definição formal de estado é dada pelo Processo de Decisão de Markov (do inglês, *Markov Decision Processes* (MDP)), onde um estado pode ser definido como qualquer informação disponível ao agente sobre seu ambiente.

Em um modelo MDP, um agente interage com o ambiente continuamente, o primeiro seleciona ações e o segundo responde a essas ações apresentando novas situações ao agente. O ambiente também gera recompensas, valores que o agente tem em vista maximizar ao longo do tempo por meio da escolha de ações. Formalmente, a cada etapa $t$ ($t = 0, 1, 2, 3, \ldots$) o agente recebe uma representação do estado do ambiente $S_t \in \mathscr{S}$, e executa uma ação $A_t \in \mathscr{A}(s)$. Ao executar uma ação, o agente recebe uma recompensa $R_{t+1} \in \mathscr{R} \subset \mathbb{R}$, além do próximo estado $S_{t+1}$. A Figura 1 representa o modelo de uma MDP em uma interação agente e ambiente.

Figura 1 – Interação de um agente com o ambiente em um MDP




Fonte: Adaptada de Sutton e Barto (2018).

O MDP e o agente interagem e acumulam experiência $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3 \ldots$ ao longo do tempo. Estas experiências permitem o agente decidir qual ação $a_t$ do conjunto de ações $\mathscr{A}(s)$ a ser executada no estado $s_t$, utilizando uma política $\pi(a_t|s_t)$ para selecionar a ação. A política é utilizada para mapear a probabilidade de selecionar uma ação dado um estado, visando obter uma estimativa do quão bom é o desempenho desta execução. O agente então recebe uma recompensa $r_t$ e avança para o próximo estado $s_{t+1}$, dada a função de recompensa $\mathscr{R}(s,a)$ e a probabilidade de transição de estado $\mathscr{P}(s_{t+1}|s_t, a_t)$. Esse processo continua até que o agente atinja um estado terminal em um problema episódico. Ao final do ciclo, é retornada a recompensa acumulada com desconto com o fator $\gamma \in (0, 1]$. A função de recompensa pode ser expressa da seguinte forma:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{2.1}$$

O agente visa maximizar a expectativa de retorno a longo prazo de cada estado. Desta forma, a *função de valor* é uma previsão da recompensa esperada, acumulativa, descontada e futura, medindo a qualidade de cada estado ou par de ação e estado (LI, 2018). A função valor é definida como o retorno esperado para a política $\pi$ no estado $s$, definida por:

$$v_\pi(s) \doteq \mathbb{E}\left[R_t | s_t = s\right] \tag{2.2}$$

onde $R_t$ representa a equação apresentada na Equação 2.1. A função valor pode ser definida através do retorno esperado selecionando a ação $a$ no estado $s$ dada a política $\pi$, dada pela seguinte equação:

$$q_\pi(s,a) = \mathbb{E}\left[R_t | s_t = s, a_t = a\right]. \tag{2.3}$$

A função valor $v_\pi(s)$ pode ser decomposta na equação de *Bellman*:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \tag{2.4}$$

A equação de Bellman expressa uma relação entre o valor de um estado e os valores de seus estados sucessores, ela calcula a média de todas as possibilidades, ponderando cada uma por sua probabilidade de ocorrência. Nesta equação, o valor do estado inicial deve ser igual ao valor descontado do próximo estado esperado, acrescido da recompensa esperada ao longo do caminho.

Alguns autores, ao invés de procurar maximizar a recompensa $R$, visam minimizar o custo $C$ (ou punição) (BERTSEKAS *et al.*, 1995). Contudo, nesta tese é adotada a abordagem de Sutton e Barto (2018) onde busca-se maximizar $R$. Assim definido, pode-se assumir que $v_*(s)$ é o valor máximo do estado alcançável por qualquer política para o estado $s$

$$v_* = \max_\pi v_\pi(s) = \max_a q_{\pi *}(s,a) \tag{2.5}$$

que pode ser decomposta em uma equação de Bellman:

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]. \tag{2.6}$$

A decomposição também pode ser feita com a função ação-valor

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')], \tag{2.7}$$

e a função ação-valor ótima é dada por

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')]. \tag{2.8}$$

A política ótima é dada como $\pi^*$, sendo uma função que seleciona sempre uma ação que maximize recompensas futuras em qualquer estado. Esta política ótima é um problema que pode ser resolvido de forma *on-policy* ou *off-policy*. O primeiro utiliza e ajusta a política $\pi$ em relação aos valores de ação, ajustando a função ação-valor $q_\pi(s,a)$. Algoritmos deste tipo, por exemplo, o SARSA (RUMMERY; NIRANJAN, 1994), avaliam a política utilizando amostras da mesma política, refinando a política gananciosamente em relação aos valores da ação. O SARSA atualiza a política $Q(S,A)$ segundo a equação:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)]. \tag{2.9}$$

Na equação, o SARSA é representado por estado atual, ação, recompensa, próximo estado e próxima ação, além de utilizar a política atual tanto para encontrar a ação a ser tomada quanto como regra de atualização. No Algoritmo 1, adaptado de Sutton e Barto (2018), é apresentada a operação do SARSA para o cálculo da ação-valor $Q$.

**Algoritmo 1** – Algoritmo SARSA

Inicializar $Q(s,a)$ arbitrariamente, para todo $S \in \mathscr{S}^+$, $a \in \mathscr{A}(s)$, exceto quando $Q(final,.) = 0$

**para** cada episódio **faça**
  Inicializar $S$
  $A \leftarrow$ selecione ação dado $S$ usando política derivada de $Q$ ($\varepsilon$-greedy, por exemplo)
  **para** cada passo do episódio **faça**
    Execute ação $A$, observe a recompensa $R$ com base em $S'$
    $A' \leftarrow$ selecione ação dado $S'$ usando política derivada de $Q$ ($\varepsilon$-greedy, por exemplo)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
    $S \leftarrow S'$
    $A \leftarrow A'$
  **fim para**
**fim para**

No segundo tipo, *off-policy*, o agente aprende a política ideal (função valor), podendo utilizar uma política qualquer. O *Q-learning* (WATKINS; DAYAN, 1992), por exemplo, obtém uma política diferente da política que gera as amostras, ou seja, o Q-learning procura valores de ação para a política ideal diretamente, não necessariamente se ajustando à política que gera os dados (LI, 2018). A função $Q(S,A)$ para o tempo $t$ no algoritmo Q-learning é atualizada da seguinte forma:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \max_{a} Q(S_{t+1},a) - Q(S_t,A_t)]. \tag{2.10}$$

Note que na Equação 2.10 o Q-learning refina a política ação-valor $Q$ através da operação *max*, aproximando de uma função ótima $q*$. A cada etapa $t$ o agente observa o estado $S_t$ e executa uma ação. Ao passar para o estado $S_{t+1}$ o algoritmo atribui ao agente uma recompensa $r_t$. A equação é dotada de uma taxa de aprendizado $\alpha$ e um fator de desconto $\gamma$. O Algoritmo 2 apresenta passo a passo o aprendizado da política ideal.

**Algoritmo 2** – Algoritmo Q-learning

Parâmetros: tamanho do passo $\alpha \in (0,1]$, $\varepsilon > 0$
Inicializar $Q(s,a)$ arbitrariamente, para todo $S \in \mathscr{S}^+$, $a \in \mathscr{A}(s)$, exceto quando $Q(final,.) = 0$

**para** cada episódio **faça**
  Inicializar $S$
  **para** cada passo do episódio, sendo $S$ não terminal **faça**
    $A' \leftarrow$ selecione ação dado $S'$ usando política derivada de $Q$ ($\varepsilon$-greedy, por exemplo)
    Execute ação $A$, observe a recompensa $R$ com base em $S'$
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  **fim para**
**fim para**

Para possibilitar a convergência a uma política ótima, o Q-learning ainda conta com um parâmetro $\varepsilon$, que configura um valor de exploração. Isto assegura que o algoritmo não fique preso a uma dada política $\pi$, explorando aleatoriamente ações que eventualmente podem maximizar, ou não, recompensas futuras. Na abordagem $\varepsilon - greed$, com $\varepsilon \in (0,1)$, um agente seleciona uma ação gananciosamente para o estado $s$, com probabilidade $1-\varepsilon$ e seleciona uma ação aleatória com probabilidade $\varepsilon$.

A partir do momento que ambientes ficam mais complexos, os algoritmos clássicos, como Sarsa e Q-learning, não são capazes mapear seus estados/ações de forma hábil e viável. Segundo Sutton e Barto (2018), algoritmos de otimização de política por gradiente conseguem suprir informações necessárias considerando conjunto de ações e estados, modeladas, geralmente, por redes neurais ou outras técnicas de aprendizado de máquina. Este tipo de algoritmo aprende uma política parametrizada que pode selecionar ações sem consultar uma função de valor.

Os métodos baseados em políticas geralmente têm melhores propriedades de convergência do que baseados em valor, sendo eficazes em espaços de ação contínuos ou de alta dimensão e podem aprender políticas estocásticas. Contudo, estes métodos eventualmente convergem para um ótimo local, sendo ineficientes para encontrar e avaliar alta variância (LI, 2018).

O aprendizado por reforço baseado em políticas é um problema de otimização, cujo objetivo é encontrar $\theta$ que maximiza a medida de desempenho $J(\theta)$ para a política $\pi_\theta$. O objetivo do algoritmo é procurar por um máximo local em $J(\theta)$ pelo gradiente ascendente em respeito aos parâmetros $\theta$:

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)} \tag{2.11}$$

onde $\alpha$ é o tamanho do passo e $\widehat{\nabla J(\theta_t)} \in \mathbb{R}^d$ é uma estimativa estocástica cuja expectativa se aproxima do gradiente da medida de desempenho em relação ao parâmetro $\theta$. O teorema do gradiente de política (SUTTON; BARTO, 2018) fornece uma expressão analítica para o gradiente de desempenho em relação ao parâmetro de política:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla(a|s,\theta) \tag{2.12}$$

onde os gradientes são vetores de coluna de derivadas parciais em relação aos componentes do parâmetro $\theta$ e $\pi$ corresponde à política correspondente ao vetor de parâmetro $\theta$.

A família de algoritmos *REINFORCE* (WILLIAMS, 1992) são métodos que otimizam a política $\pi(a|s;\theta)$ diretamente, ao invés da função valor, atualizando os parâmetros $\theta$ pelo gradiente ascendente utilizando um retorno $R_t$ como amostra imparcial de $Q(s_t|a_t)$. Em uma variação do REINFORCE, é utilizada uma *baseline* $b_t(s_t)$ que é subtraída do retorno, geralmente, para reduzir a variância da estimativa do gradiente, com o intuito de fornecer a direção do gradiente.

O teorema do gradiente de política, apresentado na Equação 2.12, segundo Sutton e Barto (2018) pode ser adaptado para incluir os valores de ação com um *baseline* $b(s)$, esta podendo ser qualquer função. Esta generalização é dada por:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla(a|s,\theta) \tag{2.13}$$

que pode ser utilizada para derivar uma regra de atualização para o algoritmo REINFORCE com *baseline*:

$$\theta_{t+1} \doteq \theta_t + \alpha(G_t - b(S_t)) \frac{\nabla_\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}. \tag{2.14}$$

O algoritmo completo para o REINFORCE usando uma função de valor-estado dada uma *baseline* para o aprendizado é ilustrado no Algoritmo 3 (SUTTON; BARTO, 2018).

O REINFORCE com *baseline* utiliza a função valor-estado apenas como base para o estado cuja estimativa está sendo atualizada. Esta abordagem é imparcial e converge assintoticamente para um mínimo local, além de aprender lentamente e ser inviáveis para implementações *on-line* e problemas contínuos. Métodos *Actor-Critic*, contudo, utilizam a função valor-estado como crítico, que segundo Sutton e Barto (2018), somente desta forma é que um viés e uma dependência assintótica de qualidade da aproximação podem ser introduzidos eficientemente, reduzindo a variação e acelerando o aprendizado.

**Algoritmo 3** – Algoritmo REINFORCE com *baseline*

Entrada: política $\pi(a|s,\theta), \hat{v}(s,\mathbf{w})$
Parâmetros: tamanho do passo $\alpha > 0$, $\beta > 0$
Saída: política $\pi(a|s,\theta)$
Inicializar parâmetro $\theta$ de política e os pesos $\mathbf{w}$ de valores-estado
**para** cada episódio **faça**
  Gere um episódio $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, seguindo $\pi(\cdot|\cdot,\theta)$
  **para** cada passo do episódio $t = 0, 1, \ldots, T-1$ **faça**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \delta \nabla \log \pi(A_t|S_t,\theta)$
  **fim para**
**fim para**

A atualização dos parâmetros $\theta$ nos algoritmos *actor-critic* é dada da seguinte forma:

$$\theta_{t+1} \doteq \theta_t + \alpha \delta_t \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}. \tag{2.15}$$

Em relação à Equação 2.14, a atualização de parâmetros do *actor-critic* substitui o retorno completo do REINFORCE pelo retorno de uma etapa. No Algoritmo 4 são apresentadas as etapas executadas por métodos *actor-critic*.

**Algoritmo 4** – Algoritmo *Actor-Critic*

Entrada: política $\pi(a|s,\theta), \hat{v}(s,\mathbf{w})$
Parâmetros: tamanho do passo $\alpha^{\theta} > 0$, $\alpha^{\mathbf{w}} > 0$
Inicializar parâmetro $\theta$ de política e os pesos $\mathbf{w}$ de valores-estado
**para** cada episódio **faça**
  Inicializa $S$
  $I \leftarrow 1$
  **para** cada passo do episódio, sendo $S$ não terminal **faça**
    $A \sim \pi(\cdot|S,\theta)$
    Execute ação $A$, observe a recompensa $R$ com base em $S'$
    $\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^{\theta} I \delta \nabla \log \pi(A|S,\theta)$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
  **fim para**
**fim para**

*Métodos de Redes Neurais Artificiais (RNAs) e, recentemente, DL estão sendo utilizados para auxiliar algoritmos de otimização de política, funções valor, funções de recompensa, funções de transições de estado, etc. Os métodos de aprendizado profundo tomam informações sensoriais brutas como entrada e processam-nas para aprender vários níveis de representação

automaticamente, onde cada nível de representação corresponde a um nível um pouco mais alto de abstração (LI, 2018).

## 2.2 Redes Neurais Artificiais Profundas

As RNAs são técnicas inspiradas nas estruturas neurais do cérebro biológico e conseguem aprender a partir de experiência prévia (HAYKIN, 2010). São sistemas paralelos distribuídos com unidades de processamento simples capaz de calcular determinadas funções matemáticas geralmente não lineares (BRAGA; CARVALHO; LUDERMIR, 2007).

Segundo Tubb (1993), a habilidade de aprender com a experiência, melhorar seu desempenho e se adaptar a novos ambientes, a capacidade de lidar com ruído, tolerar falhas e de fazer processamento paralelo, são as principais vantagens das RNAs. Elas são utilizadas principalmente pela sua estrutura paralela, fornece um desempenho superior comparado a outros métodos, e à sua habilidade de aprender e generalizar o conhecimento obtido (HAYKIN, 2010).

No ano de 1943, McCulloch e Pitts (1943) publicaram o primeiro modelo matemático de um neurônio artificial, o qual é bastante simples, formado por conjunto de entradas, uma unidade de processamento e uma ou mais saídas que correspondem aos detritos, corpo do neurônio e axônio, respectivamente. Alguns anos depois, foi desenvolvida a técnica de Aprendizado Hebbiano, onde o peso de uma conexão sináptica deve ser ajustado se houver sincronismo entre os níveis dos dados de entrada e saída (HAYKIN, 2010).

No final da década de 1950, foi desenvolvido uma rede com vários neurônios, denominada *perceptron*, sendo adicionado uma função de limiar, assim estabelecendo a primeira geração de RNAs que possuem como característica saídas booleanas (ROSENBLATT, 1958). Além disto, a rede podia aprender, ajustando sistematicamente seus pesos sinápticos (LUGER; STUBBLEFIELD, 1990).

A segunda geração das RNAs, surgiu com as contribuições de Widrow (1962) com um algoritmo de aprendizado "regra delta" em um neurônio com função de ativação diferençável. Outra contribuição foi o desenvolvimento do algoritmo *backpropagation*, na década de 1980 por Rumelhart *et al.* (1988).

As redes *Multilayer Perceptron* (MLP) surgiram com a necessidade de resolver problemas não linearmente separáveis, onde as redes *perceptron* conseguem resolver somente problemas lineares, pois esta rede é constituída de apenas de entrada e saída, e a MLP, além destas, é provida por uma ou mais camadas ocultas (HAYKIN, 2010). Os resultados com estas redes permitiram o surgimento e o desenvolvimento das redes neurais profundas e consequentemente da área de DL.

As redes profundas surgiram com objetivo de prover modelos capazes de classificar abundância de dados, extraindo recursos hierárquicos e usando uma combinação de várias camadas com transformações não lineares. Cada camada transforma a representação da camada anterior

em uma representação mais alta e abstrata (LECUN; BENGIO; HINTON, 2015). Estas redes vêm ganhando notoriedade por resolver diversos problemas de visão computacional (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; LECUN; BENGIO; HINTON, 2015; SZEGEDY *et al.*, 2015), reconhecimento de fala (HINTON *et al.*, 2012; GRAVES; MOHAMED; HINTON, 2013) e inteligência artificial, em geral. O interessante destas redes é que elas conseguem trabalhar com dados brutos (sem pré-processamento) e realizar transformações aprendidas a partir dos dados, suprimindo ruídos e aprendendo características discriminativas.

Várias arquiteturas de redes profundas foram propostas para a resolução de problemas reais. Dentre elas podemos citar as Redes Neurais Convolucionais (do inglês *Convolutional Neural Networks* (ConvNets)) (LECUN; BENGIO; HINTON, 2015), projetadas para trabalhar com dados matriciais, como imagens, espectrograma de áudio e vídeo, etc., as Redes Residuais (*Residual Network* (ResNet)) (HE *et al.*, 2016), que adicionam conexões atalho para aprender funções residuais, as Redes Neurais Recorrentes (*Recurrent Neural Networks* (RNN)) (GRAVES, 2012), com ligações cíclicas que permitem utilizar informações passadas na computação de saídas futuras e redes de memória de longo prazo (*Long Short-Term Memory* (LSTM)) (HOCHREITER; SCHMIDHUBER, 1997), responsáveis por reconhecer dependências temporais de longa duração. As duas primeiras serão melhor abordadas nas próximas seções por estarem ligadas de alguma forma com a nossa proposta.

A convolução é um operador que calcula o produto vetorial entre duas funções, a matriz de pesos e a região de sobreposição. A Equação 2.16 apresenta esta operação:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.16}$$

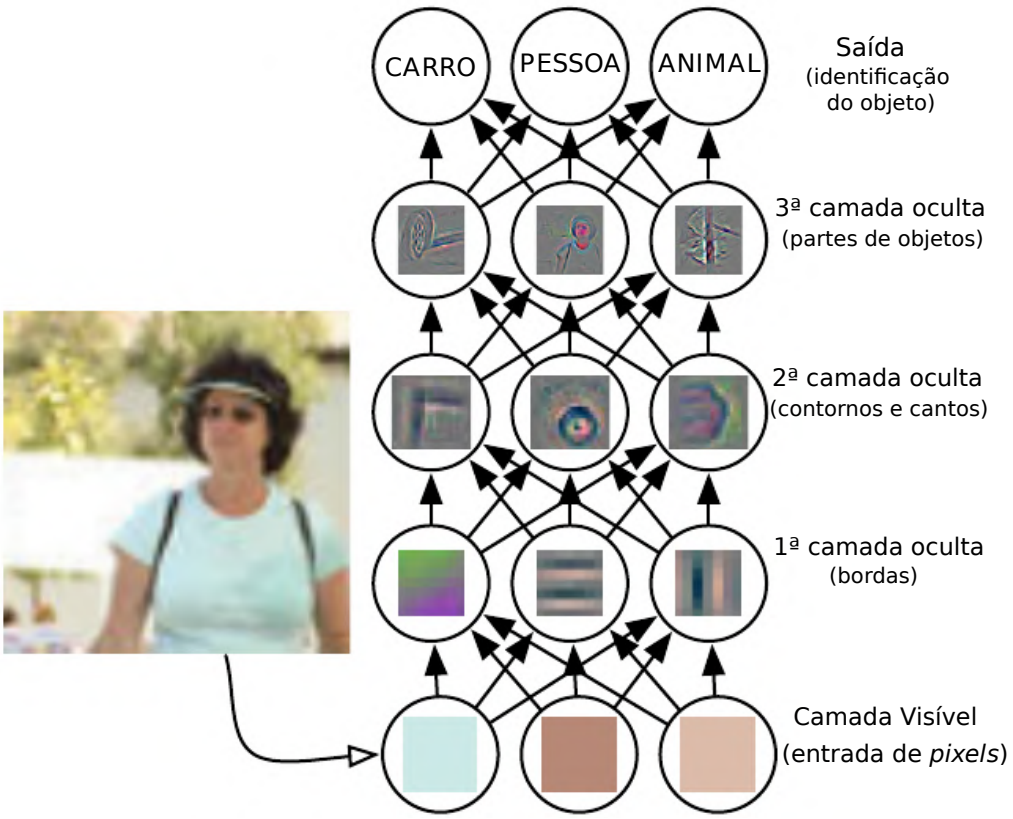
onde $S$ e $I$ são matrizes (imagens, por exemplo) de entrada e saída, e $K$ é um *kernel* (ou filtro) bi-dimensional.

As ConvNets são caracterizadas por diversas camadas convolucionais e de *pooling*, sendo estas responsáveis por realizar a extração de padrões e características dos dados, seguida por várias camadas totalmente conectadas. As primeiras camadas são responsáveis por extrair padrões simples, como traços e linhas, no caso da utilização de imagens, bem como o aprendizado de contraste e cores entre píxeis. A cada camada, a rede consegue aprender padrões mais complexos, como objetos, formas geométricas, pessoas, rostos, etc.

O processo de *pooling* é responsável por gerar uma subamostragem da camada anterior, reduzindo a dimensão dos dados para propagar apenas informações relevantes. As matrizes geradas entre as camadas são conhecidas como mapas de características, representados como neurônios dispostos em uma grade tridimensional.

Além das peculiaridades apresentadas, as ConvNets possuem neurônios que se conectam parcialmente com uma determinada região da camada anterior, isto é denominado como campos receptivos locais. Isto permite a extração de características específicas para cada local,

Figura 2 – Exemplo de representações aprendidas em um modelo de redes neurais profundas

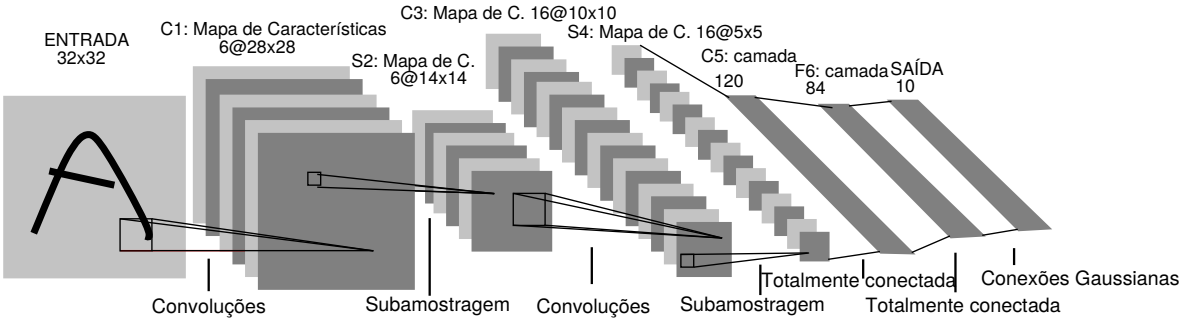



Fonte: Adaptada de Goodfellow, Bengio e Courville (2016).

viabilizando a especialização em padrões mais simples (como apresentado anteriormente), e a partir da combinação destas camadas é possível aprender representações complexas. Na Figura 2, esta característica é ilustrada, com as primeiras camadas abstraindo padrões mais simples, como bordas, contornos e cantos, e a última camada, padrões mais complexos, como partes de objetos. Todos os neurônios pertencentes a um mesmo mapa de características compartilha os mesmos pesos, sendo estes mapas gerados pelas saídas dos neurônios das camadas anteriores (GOODFELLOW; BENGIO; COURVILLE, 2016). Na Figura 3 é apresentada um exemplo de arquitetura da ConvNet para reconhecimento de dígitos manuscritos.

O treinamento de uma rede convolucional é realizado exatamente como uma rede MLP padrão, onde é utilizada a descida de gradiente estocástico e os gradientes são calculados conforme a regra da cadeia. Em relação à saída da rede, são aplicadas funções de ativação com objetivo de tornar a classificação mais suave, possibilitando que pequenas alterações nos pesos contribuam suavemente na saída final (BLANCO, 2019). As funções de ativações mais utilizadas são a sigmoide, *softmax* e a *rectified linear unit* (ReLU). Contudo Goodfellow, Bengio e Courville (2016) citam que a função ReLU é a mais recomendada para as redes neurais modernas. Esta

Figura 3 – Arquitetura da *LeNet-5*, uma *ConvNet* para reconhecimento de dígitos. Cada plano é um mapa de características.




Fonte: Adaptada de LeCun *et al.* (1998).

função é definida a partir da seguinte equação:

$$f(x) = \max(0, x). \tag{2.17}$$

Com a evolução das redes neurais e com o advento das ConvNets, várias arquiteturas surgiram como AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), *Inception* (SZEGEDY *et al.*, 2015), *VGGNet* (SIMONYAN; ZISSERMAN, 2014) e a *ResNet* (HE *et al.*, 2016). Além disto, as ConvNets permitiram o avanço de outros modelos de aprendizado, tais como os de aprendizado por reforço, originando os métodos de DRL. Este tema será melhor abordado na próxima seção.

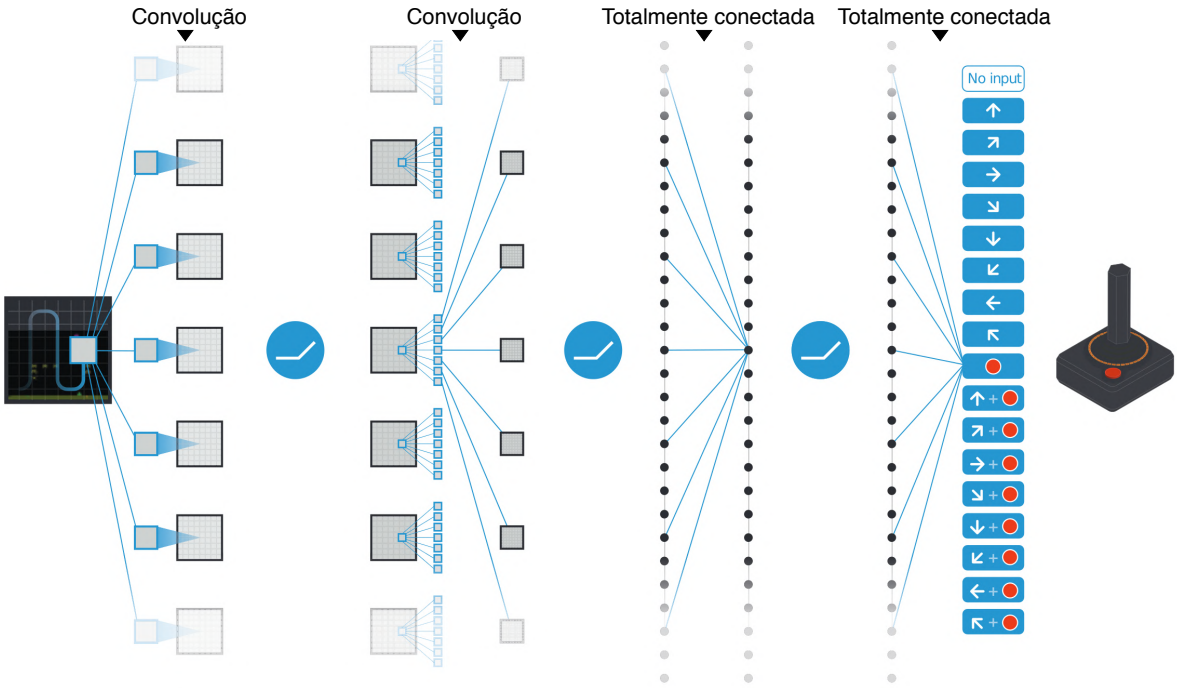
## 2.3 Aprendizado por Reforço Profundo

Em RL, os agentes são submetidos a ambientes complexos, onde seus sensores devem captar informações de alta dimensão e utilizá-las para generalizar a experiência para novas interações, principalmente em situações que abordam a complexidade do mundo real. Graças ao advento e sucesso da DL em derivar representações em dados de altas dimensões, os algoritmos de RL finalmente puderam ser aplicados em ambientes cadas vez mais complexos. A utilização de técnicas destas duas áreas abriram caminho para o desenvolvimento de uma nova área denominada Aprendizado por Reforço Profundo, DRL, principalmente pelo desenvolvimento da DQN (MNIH *et al.*, 2015).

A DQN, proposta em Mnih *et al.* (2015), foi testada no domínio de 49 jogos do *videogame* Atari 2600. A rede recebia apenas os píxeis e a pontuação do jogo como entradas, conseguindo superar o desempenho de todos os algoritmos propostos anteriormente e alcançar um nível comparável ao jogador humano profissional. Os autores ressaltam que esse trabalho preenche a lacuna entre entradas e ações sensoriais de alta dimensão, resultando no primeiro agente artificial

capaz de aprender a se destacar em uma variedade diversificada de tarefas desafiadoras.

Figura 4 – Arquitetura DQN para jogos de Atari 2600.




Fonte: Adaptada de Mnih *et al.* (2015).

Na Figura 4 é apresentada a arquitetura DQN originalmente proposta, com três camadas convolucionais e duas totalmente conectadas com uma saída para cada ação, sendo estas ações comandos do *joystick* para o *videogame*. Cada camada escondida é seguida por uma função de ativação ReLU.

Segundo Li (2018), existem alguns trabalhos anteriores às DQN, como de Tesauro (1994) e Riedmiller (2005), que propunham integrar redes neurais à RL. Contudo, o modelo RL era instável ou até mesmo divergente quando a função ação-valor era aproximada com uma função não linear, como RNAs. No modelo proposto, foram realizadas mudanças em três conceitos que ajudaram a estabilizar o sinal de treinamento e aprimorar o desempenho da rede. Abaixo é apresentada a função (*loss*) aprimorada por Mnih *et al.* (2015), utilizada para atualizar a Q-learning na interação $i$:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\ U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a',\theta_i^-) - Q(s,a,\theta_i)\right)^2\right] \tag{2.18}$$

onde $\gamma$ é o fator de desconto determinando pelo horizonte do agente, $\theta_i$ são parâmetros da Q-network na interação $i$ e $\theta_i^-$ são os parâmetros da rede usados para calcular os valores da rede alvo (*target network* em inglês) na interação $i$. Os parâmetros da rede alvo $\theta_i^-$ são atualizados apenas com os parâmetros da Q-network a cada etapa $C$ sendo mantidos fixos entre atualizações

individuais. Estas características salientam a primeira contribuição, fornecendo um sinal de treinamento mais estável e impedindo que as atualizações da rede deem grandes saltos.

A segunda mudança realizada foi limitar as recompensas para deixá-las mais coerentes, elas são "cortadas" entre $-1$ e $+1$ , evitando aumentar os valores estado-ação drasticamente. Já a última mudança envolve a utilização de uma memória de repetição $\mathscr{M}$, a qual armazena as experiências na forma $e_t =< s_t, a_t, r_t, s_{t+1} >$, denominadas como *mini-batch* (mini-lotes, em português), a cada tempo $t$. Isto permite o agente extrair experiências aleatórias da memória de repetição ao invés de usar experiências atuais, interrompendo as correlações entre experiências sucessivas e melhorando a eficiência dos dados (FRANÇOIS-LAVET *et al.*, 2018; GLATT, 2019).

Apesar de ter seu sucesso baseado em jogos eletrônicos, os algoritmos de DRL também se mostraram bastante eficientes nas áreas da robótica, processamento de linguagem natural, visão computacional, em geral, finanças, gerenciamento de negócios, saúde, educação, etc. (LI, 2018).

## 2.4 Emoções Humanas

A capacidade de interpretar sinais emocionais é essencial para a interação social (KESSELS *et al.*, 2014). Desta forma, para participar de uma interação com humanos, os robôs devem conseguir reconhecer e interpretar sinais de emoção e mapear internamente estas informações. Geralmente, as emoções podem ajudar na interpretação dos estados internos de um indivíduo e, consequentemente, ajudar na previsão de ações futuras (BREAZEAL; DAUTENHAHN; KANDA, 2016).

A emoção é geralmente definida como um estado mental (OATLEY, 2000) de natureza reativa, normalmente breve e intensa, causada por uma experiência de medo, surpresa, alegria, etc. ou um sentimento de afeto, como dor, desejo, esperança, etc. (CABANAC, 2002; EKMAN, 1999). Embora as emoções sejam categorizadas como puramente cognitivas, a representação mental de uma experiência emocional inclui componentes motores e viscerais, bem como componentes cognitivos (DANTZER, 1993).

Diferente de emoção, o humor é mais estável e constante, tendendo a ser mais abrangente e não tão vinculado a circunstâncias específicas, geralmente enviesando estratégias cognitivas de longo prazo (PERGHER *et al.*, 2006 apud ELLIS; MOORE, 1999). Já o afeto é definido como "o componente emocional de uma ideia", ou seja, a é uma representação emocional que acompanha uma ideia ou representação mental (PERGHER *et al.*, 2006).

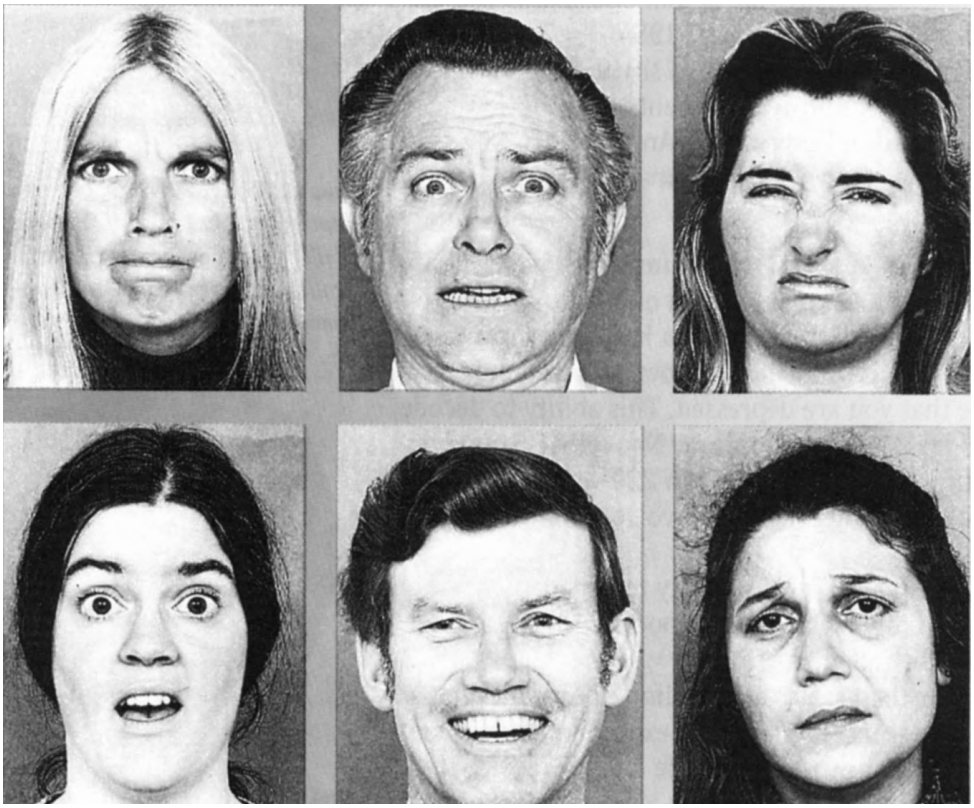
Do ponto de vista evolutivo, as emoções são respostas automáticas baseadas em estímulos visuais, pois tendem a produzir comportamentos geralmente adaptativos, sem exigir pensamento deliberado (GREENE, 2013). Estas perspectivas evolutivas foram iniciadas com o livro "*The*

*Expression of the Emotions in Man and Animals*" de Charles Darwin (DARWIN; PRODGER, 1998), no século XIX. Darwin foi pioneiro em vários métodos para estudar expressões não verbais, a partir dos quais concluiu que algumas expressões tinham universalidade transcultural.

A questão transcultural foi estudada em Ekman e Friesen (1971), onde os autores demonstraram a universalidade das emoções básicas, dado que elas estão presentes em diversos contextos culturais. Estas emoções básicas são: *happiness* (alegria), *fear* (medo), *disgust* (nojo), *surprise* (surpresa), *anger* (raiva) e *sadness* (tristeza). Plutchik (2001) se baseiam no trabalho de Ekman para criar uma perspectiva biologicamente orientada através da "Roda das Emoções" (*Wheel of Emotions*), sugerindo oito emoções primárias agrupadas em bases positivas e negativas.

A seguir, serão apresentadas as formas mais aceitas na literatura de classificar e agrupar emoções, incluindo os trabalhos de Ekman e Friesen (1971) e Plutchik (2001).

Figura 5 – Seis emoções básicas de Ekman. No sentido horário, do canto superior esquerdo: raiva, medo, nojo, tristeza, felicidade, surpresa.



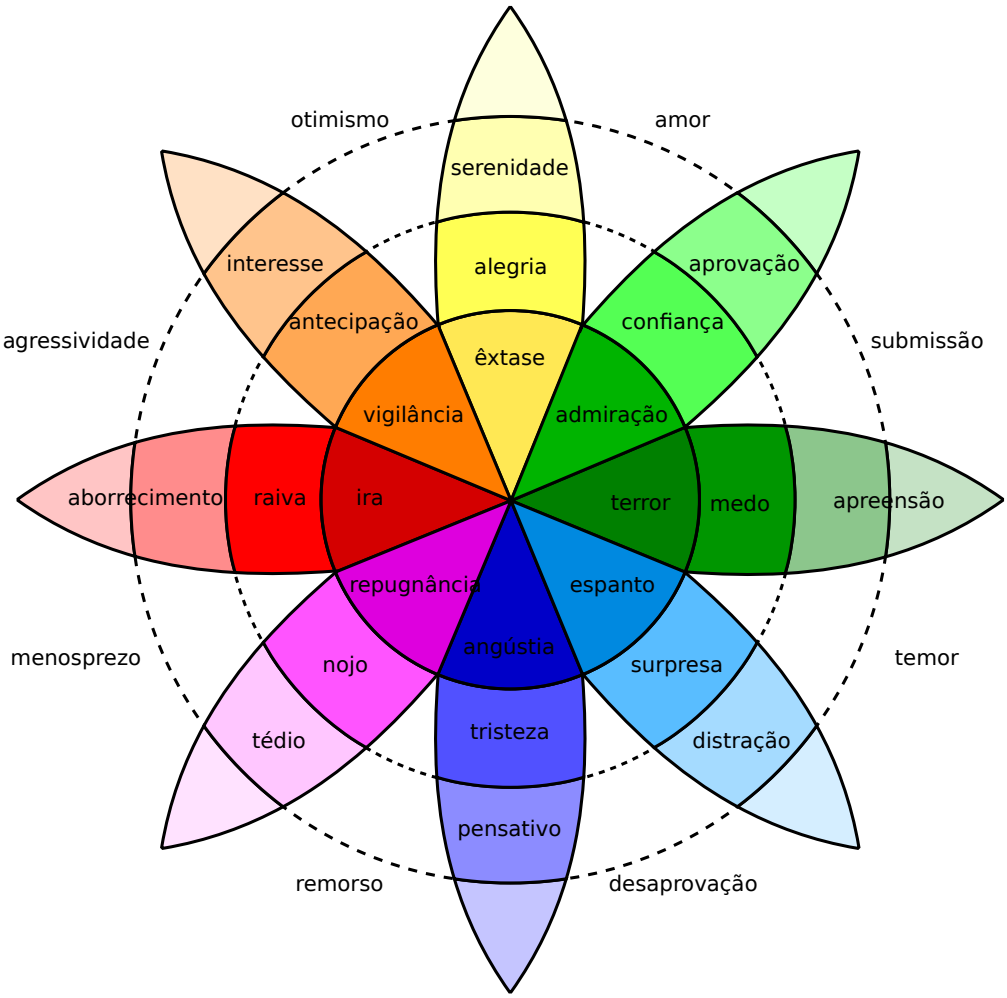
Fonte: Ekman (1999).

### 2.4.1 Classificação

Como apresentado anteriormente, os robôs devem conseguir reconhecer e interpretar e mapear internamente emoções humanas. Desta forma, é importante ter um modelo que possa representar estas emoções.

O conjunto de emoções de Ekman e Friesen (1971) representam as expressões universais, inatas e independentes de aspectos culturais. Cada uma destas é demonstrada na Figura 5. Contudo, em Ekman e Davidson (1994) são definidas mais onze emoções, *amusement* (diversão), *contempt* (desprezo), *contentment* (contentamento, satisfação), *embarrassment* (embaraço, constrangimento), *excitement* (excitação), *guilt* (culpa), *pride in achievement* (orgulho na conquista de algo), *relief* (alívio), *satisfaction* (satisfação), *sensory pleasure* (prazer sensorial) e *shame* (vergonha). Porém, os autores afirmam que nem todas as emoções deste segundo grupo podem ser codificadas por expressões faciais, dificultando a detecção delas por sistemas computacionais.

Figura 6 – Roda das emoções




Fonte: Adaptada de Plutchik (2001).

A roda das emoções proposta por Plutchik (2001), agrupa as emoções entre positivas e negativas: alegria contra tristeza; raiva contra medo; confiança contra nojo; e surpresa contra antecipação. Estas emoções básicas podem ser combinadas para formar emoções complexas, as quais formam o espectro da experiência emocional humana. Por exemplo, medo e surpresa

podem se misturar para formar temor. A "Roda das Emoções" é apresentada na Figura 6, note a combinação de emoções e a separação entre emoções positivas e negativas.

Figura 7 – Modelo contínuo bidimensional para mapeamento de emoções


Fonte: Adaptada de Russell, Lewicka e Niit (1989).

Diferente dos modelos apresentados, os modelos contínuos não limitam as emoções a um conjunto finito. O modelo de Russell, Lewicka e Niit (1989) mapeia emoções por meio de variáveis contínuas em um espaço bidimensional, onde uma dimensão representa o quão agradável é a emoção (valência) e a outra a propensão do indivíduo em agir dada a emoção experimentada (atividade). Na Figura 7 é apresentada esta representação bidimensional, onde algumas emoções discretas são mapeadas.

Nesta proposta, é utilizada a classificação discreta de emoções, dado que muitas destas são difíceis de serem detectadas utilizando métodos atuais, principalmente quando são utilizados técnicas pouco invasivas, como reconhecimento facial.

## 2.5 Considerações finais

Neste capítulo foram apresentados os fundamentos teóricos de alguns métodos de aprendizado supervisionado e por reforço, utilizados no desenvolvimento desta tese. Começando com o método de aprendizado por reforço, passando por aprendizado profundo e a junção destes dois métodos, denominada aprendizado por reforço profundo ou DRL. Finalmente, foram

apresentados aspectos das emoções humanas, utilizadas nesta proposta como sinal de reforço para o sistema de DRL.

No próximo capítulo, será apresentado um levantamento de trabalhos que propõem interação entre robôs e humanos, com foco em aprendizado por reforço e emoções humanas.

CAPÍTULO

# 3

# TRABALHOS RELACIONADOS

Esta tese está relacionada ao tópico de pesquisa: sistemas de IHR que devem aprender a se comportar socialmente seguindo o paradigma de autoaprendizagem, observando informações de alta dimensionalidade do ambiente e/ou sinais sociais. Neste Capítulo, são apresentados vários trabalhos relacionados a este tópico, considerando sinais sociais, tais como, detecção de face, emoções, direção do olhar, detecção de fala, etc.

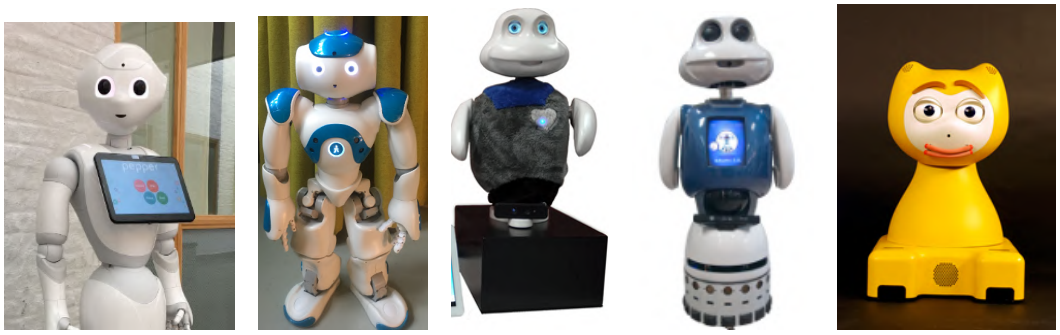
## 3.1 Robótica Social, Auto-Aprendizado, Sinais Sociais e Emocionais

Robôs autônomos devem atuar com base em suas próprias decisões para cumprir dados objetivos (MATARIĆ; ARKIN, 2007). Desta forma, o robô deve saber qual ação deve executar em cada situação, e para isto, ele deve aprender essa relação entre situações e ações na falta deste conhecimento (CASTRO-GONZÁLEZ *et al.*, 2014).

Em Broekens (2007) são apresentadas evidências quantitativas do fato de que um humano pode aumentar o desempenho do aprendizado em tempo real, por meio de expressões faciais como reforço do sistema, em um ambiente de aprendizado não trivial. Neste trabalho as emoções positivas são utilizadas para dar recompensas e as negativas para punir um sistema RL. Este trabalho é validado utilizando o ambiente 2D *grid-world*, onde o agente deve coletar "alimentos", evitando obstáculos e paredes. Neste cenário, um humano observa as ações do robô, reagindo por meio de emoções (expressões faciais). Os autores concluem que esta abordagem apresenta fortes evidências do potencial benefício da comunicação afetiva com humanos no ciclo de aprendizado por reforço. Além disso, eles ressaltam que os resultados alcançados contribuem na evidência de que os robôs podem ser treinados e seus comportamentos otimizados usando dicas sociais naturais através do uso de emoções.

A pesquisa de Akalin e Loutfi (2021) traz contribuições importantes ao compilar diversas

Figura 8 – Alguns dos robôs mais utilizados para interação social com humanos.

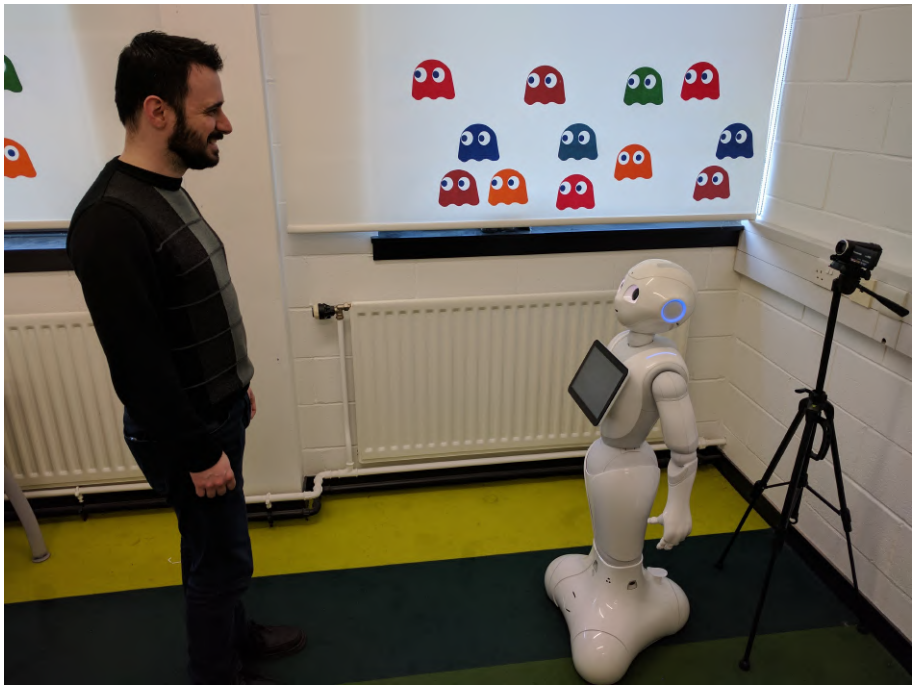


(a) Pepper (b) NAO (c) Mini (d) Maggie (e) iCat

Fonte: Adaptada de Akalin e Loutfi (2021).

abordagens e trabalhos voltados para robótica social. Segundos os autores, os trabalhos desta área enfrentam desafios em relação à obtenção de recompensas imediatas em ambientes reais, dificultando o aprendizado de agentes por reforço. O trabalho foca em aplicações em robôs reais, apresentando alguns dos agentes mais utilizados na literatura para atuação na interação com humanos. Na Figura 8 são ilustrados alguns destes agentes, tais como os robôs Pepper, NAO, Mini, Maggie e iCat.

Figura 9 – Robô Pepper interagindo com humano em um experimento controlado. São utilizadas 12 variáveis para mapear o estado do ambiente e um conjunto de ações para interação verbal e não verbal.



Fonte: Papaioannou *et al.* (2017).

A utilização de RL para robôs que usam diálogos e execução de tarefas é explorada em Papaioannou *et al.* (2017). Os autores utilizam o robô Pepper, Figura 9, para interação

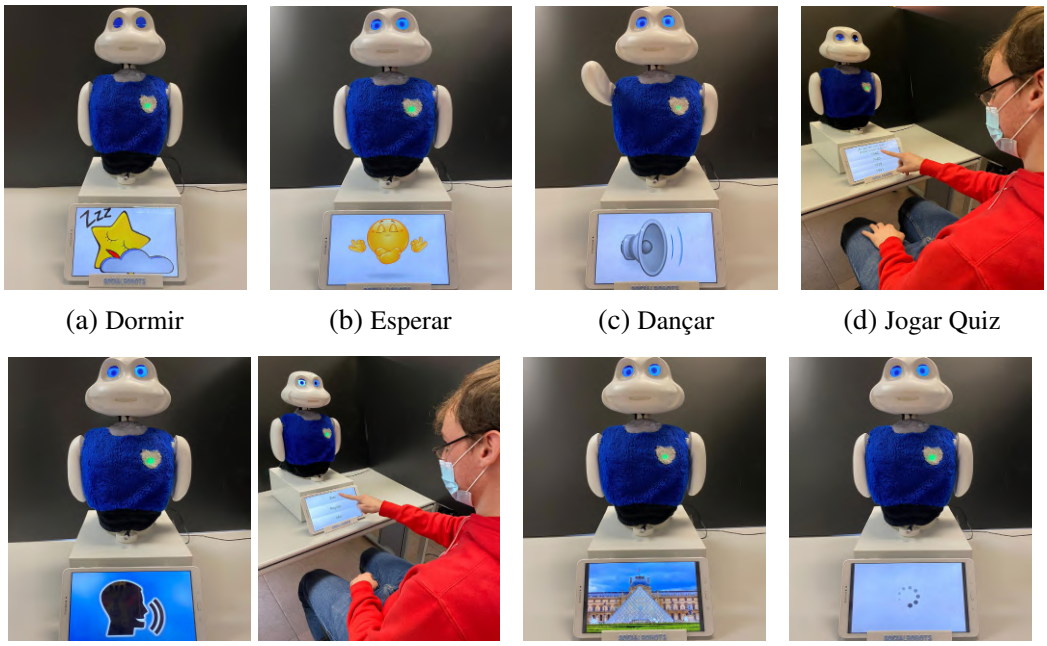
com humanos, o qual aprende a alternar entre a execução de diálogos e tarefas através do algoritmo *Q-learning*, usando diálogos e dados sensoriais como informações para o estado do sistema. Este estado é composto por 12 variáveis, que armazenam a distância entre robô e humano, engajamento, tarefa executada, dentre outras. É utilizado um conjunto de ações, como *executar tarefa*, *cumprimentar*, *despedir*, *diálogo*, *mostrar direções*, *esperar*, etc. Estas ações são focadas para atender pessoas que chegam em um shopping e querem informações sobre lojas, cupons de desconto, etc. Cada ação pode ser subdividia em tarefas, dependentes do contexto do diálogo com o humano, como, por exemplo, se o humano pedir um cupom de desconto, o robô apresenta um código em sua tela. Segundo os autores, por meio de experimentos e questionários com pessoas reais, o sistema proposto se mostrou mais agradável em relação à interação com usuários, sendo que estes tiveram sensação de que o sistema atendeu às expectativas em relação a uma abordagem linear, sem o aprendizado e adaptação. Contudo, o sistema não considera comportamentos corporais e sinais sociais para a interação com o humano.

Outra questão na área de IHR é a coleta de dados para treinamento de agentes. No trabalho apresentado por Andriella, Torras e Alenyà (2019), é relatado que tal problema se agrava quando o foco da interação é com adultos mais velhos com demência, pois tal interação pode levar horas e sobrecarregar o usuário. Com isto, os autores propõem um *framework* para mapear comportamentos e personas humanos, visando simular ações e interações humanas, com foco em atividades de treinamento cognitivo para idosos com demência. As personas são caracterizadas por mapearem quatro dimensões relativas à capacidade do indivíduo: memória, reatividade, atenção e audição. Além disso, é utilizado um mecanismo que gerencia informações simples relacionadas a tarefas executadas pelos humanos. Os autores validam a simulação através de um agente *Q-learning*, que deve aprender o nível de assistência que ele deve dar ao humano na execução de dada tarefa terapêutica. Os níveis vão desde o robô encorajar a pessoa, a dar algumas dicas sobre a execução da tarefa, até a sugestão de como realizá-la corretamente ou dizer explicitamente a resposta para um problema. Um assistente define no simulador as personas e uma tarefa. As personas são estáticas durante toda a simulação, porém as ações relacionadas às tarefas são selecionadas utilizando distribuições de probabilidade. Estas distribuições consideram a variação da complexidade da tarefa ao longo do tempo e afeta a estimativa da probabilidade do usuário de adivinhar a ação correta em um determinado estado. Com estas informações, o agente deve aprender se deve ajudar ou não o humano em determinada atividade. No trabalho, a maior contribuição do simulador é prover um treinamento exaustivo para que o agente tenha uma noção inicial de como selecionar os níveis de assistência, sendo ainda necessário fazer um ajuste fino para interação com humanos reais.

Como apresentado nos trabalhos anteriores, métodos baseados em RL trazem diversas vantagens para a área da robótica social, especialmente em ambientes reais onde há uma variedade de eventos e fenômenos dinâmicos, às vezes difíceis de prever ou mapear computacionalmente. Contudo, com o aumento do espaço de estados e número de ações, o mapeamento de estados-ações pode crescer exponencialmente, o que pode tornar este problema intratável, diminuindo

drasticamente o processo de aprendizado. Esta questão foi levantada em Maroto-Gómez *et al.* (2021), que propõe a utilização de métodos clássicos de RL, como Q-learning, junto a um modelo probabilístico, cujo objetivo é mapear transições do ambiente e predições de recompensa.

Figura 10 – Robô social Mini executando comportamentos para regular estados internos



(a) Dormir (b) Esperar (c) Dançar (d) Jogar Quiz

(e) Falar Sozinho (f) Falar com pessoa (g) Procurar notícias (h) Tocar música

Fonte: Adaptada de Maroto-Gómez *et al.* (2021).

O modelo em questão armazena as interações do agente com o ambiente a fim de permitir o planejamento (simulação) dos efeitos causados por cada uma das ações. Desta forma, o agente é treinado tanto com interações no mundo real quanto com informações geradas pelo modelo probabilístico. A recompensa simulada é obtida através de uma distribuição normal baseada nas recompensas obtidas em interações passadas. Maroto-Gómez *et al.* (2021) mencionam que a abordagem adotada permite o agente explorar estados pouco observados, dando uma estabilidade maior ao treinamento. Segundo eles, isto permite uma melhora substancial na velocidade e estabilidade da convergência do aprendizado. Para validação da arquitetura proposta, é utilizado um robô social nomeado Mini, o qual possui tronco, cabeça e pequenos braços, além de uma tela para interação com humanos. Este robô é geralmente usado para ajudar na terapia de idosos com problemas cognitivos leves. Os estados são mapeados usando informações internas ao robô e estímulos externos, como presença de um humano e se há música sendo executada em dado momento. O estado interno do robô é responsável por tentar modelar alguns aspectos biológicos, como cansaço, tédio, necessidade social e "vontade" de aprender, os quais são calculados de acordo com estímulos (estados) externos. As ações que compõe o comportamento do robô envolvem dormir, esperar, dançar, colocar um jogo para o usuário, conversar sozinho, conversar com humano, procurar notícias na internet e ligar o tocador de músicas, conforme apresentado na Figura 10. Através de experimentos, Maroto-Gómez *et al.* (2021) mostram que a abordagem

adotada se mostra promissora para auxiliar no treinamento de agentes baseados em algoritmos clássicos de RL.

Em cenários onde humanos e robôs colaboram, é importante que estes agentes percebam e entendam o comportamento humano a fim de atribuir contexto às suas interações (BREAZEAL, 2003). Segundo Churamani *et al.* (2022), neste tipo de interação é necessário que os robôs sociais compartilhem e se adaptem à dinâmica de comportamentos afetivos, porém as abordagens atuais se concentram na percepção instantânea e atuação através de ações estáticas. Desta forma, os autores propõem uma nova estrutura para geração de comportamentos orientados a afetos sociais. No trabalho, é utilizado um modelo híbrido para avaliar expressões faciais e fala dos usuários, um modelo afetivo, para incorporar traços comportamentais como paciência e atuação emocional, a fim de modelar a avaliação afetiva do robô, e um modelo de RL para que o robô aprenda através da interação. Os autores validaram o trabalho através de um robô que atua com voluntários a partir de um jogo chamado "Jogo do Ultimato", que consiste em uma banca negociar recursos com uma pessoa, esta podendo recusar ou aceitar a oferta, conforme ilustrado na Figura 11. O robô analisa a resposta e o humor humano e, considerando estas informações, ajusta seu estado "afetivo", aplicando níveis diferentes de generosidade e comportamento altruísta ao negociar com as pessoas. Segundo Churamani *et al.* (2022), isso é benéfico para o robô interagir dinamicamente com os usuários, em vez de seguir políticas de comportamento estáticas e predeterminadas.

Figura 11 – Robô aprende a negociar com humano baseado em expressões faciais e voz


Fonte: Adaptada de Churamani *et al.* (2022).

A fim de priorizar ações que mantenham o bem estar emocional humano, em Hao *et al.* (2019) foi abordada a questão da "regulação emocional" na área de robôs de serviços. O objetivo do trabalho é atuar no ambiente reduzindo os estados de emoção negativa e mantendo as situações de emoção positiva. Para isto, os autores propõem uma técnica denominada *Multi-Objective Weighted Reinforcement Learning* [1]. É utilizado um processo de hierarquia analítica *fuzzy* para calcular o peso de cada função alvo de recompensa sob diferentes condições, além de analisar

[1] Aprendizado por Reforço Ponderado Multiobjetivo

fatores que afetam a transferência de emoções das pessoas, incluindo o estado emocional atual do usuário, sua personalidade e estímulos externos. Além da recompensa, estas informações são utilizadas para mapear os estados do ambiente, representadas por tuplas simples pré-definidas. Através de um conjunto de ações, como "tocar uma música despojada", "dar um presente", etc., o agente tenta regularizar as emoções das pessoas. Experimentos simulados foram conduzidos com voluntários, com objetivo de avaliar a satisfação de usuários como o sistema. No geral, os autores avaliaram que a proposta teve um nível médio de satisfação, com alguns resultados negativos associados a pessoas propensas a terem "personalidades negativas".

A questão da regulação emocional também é tratada em Li *et al.* (2019), trabalho que tem como proposta um sistema robótico para ensinar palavras a crianças. O robô apresenta uma palavra em um *tablet* e o aluno deve registrar o significado desta. Esta resposta é utilizada como entrada de um sistema baseado em RL, bem como a valência da emoção do aluno. Estas informações permitem que o sistema selecione comportamentos adaptativos, que incluem a mudança da expressão facial do robô, da resposta a ser dada ao aluno, próxima palavra a ensinar e dificuldade destas perguntas. O sistema recorre a alguns conceitos de teoria cognitiva, como memória, motivação e emoção. No caso desta última, os autores calculam o quão "positiva" é a emoção do aluno, diferente de outros trabalhos que detectam e classificam as emoções em categorias. Contudo, os autores apontam que o sistema não se porta bem com alunos com personalidade negativa, assim como no trabalho descrito anteriormente.

Além de emoções, trabalhos na área de IHR demonstram interesse em verificar o nível de engajamento durante interações, a fim de adaptar o comportamento do agente. Em Rudovic *et al.* (2019), por exemplo, é proposto um sistema de automatização na detecção do engajamento de crianças interagindo com um robô. Segundo os autores, este é um problema desafiador, dada a diversidade de formas como as expressões de engajamento se dão, sendo particulares de cada pessoa, por meio de gestos faciais e corporais, bem como devido às mudanças de iluminação, oclusão parcial, entre vários outros fatores. Para tratar este problema, os autores utilizam DRL e *Active Learning* (AL) para aprender e estimar o engajamento através de dados de vídeo. Os autores utilizam redes convolucionais e LSTM para abstração de conhecimento sobre o engajamento. Um humano externo insere a recompensa do sistema para cada predição, além de auxiliar na rotulação do vídeo com o tipo do engajamento para o treinamento caso o sistema não consiga efetuar isto automatizadamente. A saída da rede, quando treinada, consegue determinar o grau de engajamento em três tipos: baixo, médio e alto.

Em Lathuilière *et al.* (2019) o objetivo do trabalho é em direcionar o foco de atenção do robô para grupos de pessoas a partir de experiências audiovisuais, independentemente do número de pessoas e de suas posições. Os autores utilizam redes recorrentes (LSTM) combinadas com DQN para a seleção de política ideal para a tarefa. Para treinamento, são utilizadas simulações com cenários realistas envolvendo participantes conversando e também em silêncio. As ações do sistemas consistem em rotacionar a cabeça para direita, esquerda, cima e para baixo. As

recompensas são baseadas no número de pessoas e nas fontes de áudio presentes no campo de visão do robô. Nos experimentos, foi utilizado um robô NAO, onde o sistema se mostrou bastante robusto, com resultados que indicam que o comportamento de direção do olhar do robô é socialmente aceitável.

As expressões faciais foram pesquisadas em Lin *et al.* (2020), trabalho que as utiliza como recompensa para um robô baseado em RL para aprender corretamente alguns comportamentos. Diferente dos outros trabalhos, os autores utilizam uma vertente chamada de Aprendizado por Reforço Interativo (em inglês, *Interactive Reinforcement Learning* (IRL)) cujo objetivo é permitir que um instrutor guie o aprendizado do agente através de instruções diretas de como o agente deve agir. Os autores desse trabalho utilizam emoções agrupadas em "Positivas" e "Negativas", além de agregarem gestos para guiar o aprendizado do agente. A validação do método proposto é realizada por meio de ambientes dos jogos eletrônicos *Tetris* e *LoopMaze*, utilizando como recompensa para as ações do robô os gestos e emoções emitidas explicitamente por humanos.

Figura 12 – Espaço de ações do agente para observar comportamentos humanos

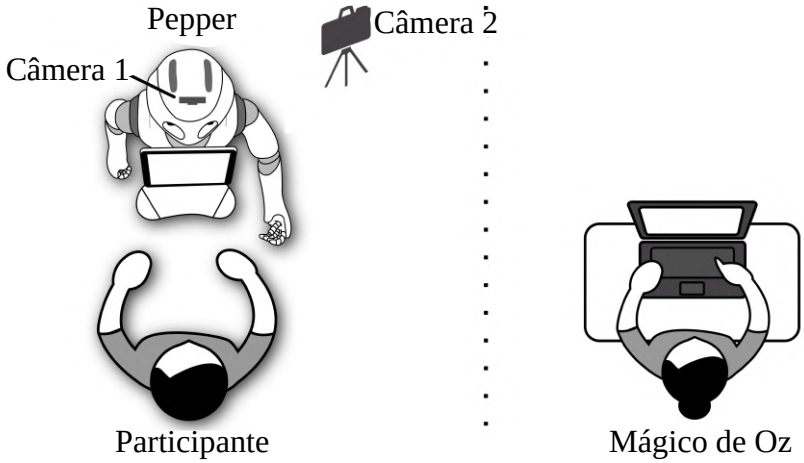

(a) Imagens capturadas pelo robô no ambiente virtual (b) Espaço de ação

Fonte: Adaptada de Kumrai *et al.* (2020).

A utilização de DQN em robótica social não é só explorada para determinar comportamentos interativos para robôs, mas também para auxiliar na percepção do mesmo em relação a reconhecimento de atividades humanas. Em Kumrai *et al.* (2020) a abordagem DQN é utilizada para posicionar um robô simulado, com objetivo de observar e detectar atividades de pessoas, buscando maximizar a precisão do reconhecimento e minimizar o consumo de energia ao se movimentar. É utilizado um ambiente virtual para o treinamento, com um humano realizando tarefas manuais (Figura 12a). São extraídas informações do "esqueleto" humano a partir de imagens, que servem como entradas da rede DQN. A entrada ainda é composta por duas imagens, cada uma com um recorte destacando as mãos humanas, visando o reconhecimento de objetos. Como algumas atividades podem ser atreladas a manipulação de objetos, a detecção destes elementos são triviais para a rede. A saída desta é uma ação que o robô deve executar. No caso do trabalho em questão, a ação corresponde a uma posição onde o robô deve se mover, seguindo uma pessoa em uma determinada distância, e deve rotacionar de forma que sua câmera permita

que a pessoa seja focalizada no centro da imagem (Figura 12b). Como a posição entre o humano e o robô é predefinida, as possíveis posições do agente seguem um formato de circunferência em torno do humano. Por fim, os estudos de casos mostraram que a utilização de DQN para posicionar o robô trouxe melhora significativa para a detecção das atividades humanas.

Figura 13 – Configuração do experimento, onde um usuário joga com um participante e um controlador gerencia o robô através de "Mágico de Oz"




Fonte: Bagheri *et al.* (2021).

A pesquisa proposta em Bagheri *et al.* (2021) teve objetivo de desenvolver de uma estrutura de empatia, que determina os estados emocionais dos usuários a partir de expressões faciais. Através de RL, a arquitetura tem em vista capacitar para que robôs selecionem a expressão empática mais apropriada para diferentes estados afetivos. Os autores utilizam quatro emoções básicas e três possíveis tipos de personalidade para mapeamento dos estados do ambiente. A recompensa é dada ao agente no caso do humano se sinta mais "positivo" após a ação do robô. Estas ações mapeiam os seguintes comportamentos empáticos: "Mímica", "Motivacional", "Distração" e "Alívio". Para cada um destes, o robô possui um conjunto de frases com objetivos de imitar emoções do usuário, motivar, distrair, trazer algum alívio ou desestressar o humano. Como a dimensionalidade de estados e ações são computacionalmente tratáveis, de modo que uma tabela simples consegue mapear a função de ação-valor, os autores optam por utilizar uma abordagem clássica de RL. Voluntários foram submetidos a experimentos, interagindo diretamente com um robô Pepper, o qual detecta as emoções através da face humana. Na Figura 13 é ilustrada a configuração do experimento realizado em Bagheri *et al.* (2021), com um robô Pepper jogando jogos com um, enquanto um humano controla alguns comportamentos do robô via "Mágico de Oz"[2].

Modelar as complexas normas sociais estabelecidas entre os humanos define um desafio de pesquisa significativo para o IHR. O uso de *hand-crafted policy* representa uma estratégia

[2] Na técnica do Mágico de Oz, um pesquisador simula as respostas do sistema nos bastidores, enquanto um participante interage com um sistema sem ter conhecimento desta intermediação, acreditando que a interação é genuinamente gerada pelo sistema (BELLA; HANINGTON, 2012).

praticamente inviável por limitar o escopo da interação (QURESHI *et al.*, 2018). Uma abordagem promissora para modelar normas sociais consiste em estratégias para *aprender com demonstrações*.

Neste sentido, em Qureshi *et al.* (2016) é proposta a arquitetura MDQN, mencionada no Capítulo 1, que permite que um robô aprenda habilidades de interação comportamentais por meio de RL. A estratégia utilizada envolve o uso de DQN para aprender o protocolo humano com a ação cumprimentar como recompensa externa. Assim como alguns trabalhos apresentados anteriormente, o robô Pepper é utilizado para interação com pessoas. Para isto, são utilizadas as ações *esperar*, *olhar*, *acenar* e *cumprimentar*. Em Qureshi *et al.* (2018) esta arquitetura foi expandida usando a expressão facial *sorriso* e *contato visual* como formas de recompensar o agente. Os autores modelaram uma rede de previsão de ação condicional (Pnet) e uma rede de estado de valor de ação (Qnet). Este conjunto possibilitou a proposta de um *framework* de aprendizado por reforço profundo intrinsecamente motivado para aprender habilidades de interação social no mundo real de recompensas escassas. Na Figura 14 é ilustrada a interação entre o robô Pepper e algumas pessoas nos experimentos realizados por Qureshi *et al.* (2016).

Figura 14 – Robô aprendendo a interagir socialmente com pessoas

Fonte: Qureshi *et al.* (2016).

Na mesma linha da arquitetura MDQN, Clark-Turner e Begum (2018) propuseram um sistema também baseado em DQN, capaz de aprender por demonstrações. Para isso, um robô tele-operado NAO e um conjunto de participantes foram considerados nos experimentos. Os participantes tiveram o olhar, voz e gestos analisados pelo robô através de *Análise de Comportamento Aplicado* para intervenção de saudação social. Ao capturar as sequências de imagem e áudio, a rede analisou os sinais e as respostas dos participantes selecionando um conjunto de três ações, *Prompt*, *Reward* e *End*. A ação *Prompt* corrige a resposta quando o participante exibe um comportamento que não é socialmente aceitável em resposta a uma saudação social. *Reward* é responsável por dar uma recompensa positiva em resposta à ação

positiva do participante. Por fim, a ação *End* é executada após uma resposta correta ou após o participante ter respondido incorretamente várias vezes. O modelo criado apresentou uma precisão de 68,1% durante a simulação e obteve resultados semelhantes em um ambiente real.

No trabalho proposto em Gao *et al.* (2019) a área de DRL é explorada para permitir robôs a se aproximarem de grupo de pessoas respeitando normas sociais. Este trabalho propôs um modelo chamado *Staged Social Behavior Learning*, para consolidar um modelo anterior de estratégias de aproximação usando simulação e aplicando o conhecimento gerado em experimentos com participantes humanos. O modelo utilizou um esquema DL associado a uma função de recompensa que segue a teoria proxêmica (HALL *et al.*, 1968).

A partir dos trabalhos relacionados ao uso de RL em IHR, pode-se concluir que eles têm em comum a fusão de diferentes tipos de RNAs para processar as informações multimodais do ambiente, como foco de atenção, fala, gestos, emoção, etc. Grande parte dos trabalhos abordam diversas estratégias de RL para mimetizar e aprender o comportamento humano.

O presente trabalho contribui para esta área de pesquisa propondo a arquitetura SocialDQN e o simulador SimDRLSR, visando acelerar o aprendizado e treinamento de agentes, considerando também as emoções humanas como informações de entrada para o sistema DRL.

## 3.2 Discussão sobre os trabalhos da literatura

Os trabalhos apresentados estão relacionados diretamente com robótica social, buscando de alguma forma utilizar aspectos sociais humanos para refinar os comportamentos dos agente robóticos. No trabalho de Hao *et al.* (2019), os autores têm como abordagem a regulação emocional, cujo objetivo é personalizar as ações do robô a fim de melhorar o humor dos usuários. Apesar do sistema proposto utilizar um modelo *Q-learning* simples, com informações de estados e recompensas mapeadas de forma simples, o trabalho traz uma conclusão bastante interessante sobre emoções e personalidades negativas, que diz respeito a menor tendência destas pessoas em ficarem satisfeitas com o serviço prestado. Esta insatisfação pode ocorrer por vários motivos, desde condições externas e pessoais, as quais estão fora do controle do robô, até como o agente aborda e interage com o humano. Neste último caso, é necessário que o robô consiga identificar pessoas com emoções negativas e aprender quando ele deve interagir com elas. A hipótese desta abordagem é que em determinados momentos a não interação com o humano é mais vantajosa do que uma interação mal sucedida ou indesejada.

O que pode ser observado em vários trabalhos é a utilização de simuladores para treinar os agentes a se comportarem de acordo com algum aspecto humano, tais como em Broekens (2007), Andriella, Torras e Alenyà (2019), Hao *et al.* (2019), Lathuilière *et al.* (2019), Lin *et al.* (2020), Kumrai *et al.* (2020) e Gao *et al.* (2019). Em Lathuilière *et al.* (2019) é citado que uso da simulação dispensar o uso de sensores, pois o robô aprende a associar padrões e comportamentos usando apenas informações visuais. Por exemplo, no trabalho citado, o robô aprende a identificar

pessoas conversando dispensando o uso de um microfone (estéreo ou não), usando este sensor apenas durante a simulação para modelar a recompensa. Além disto, os autores citam que a adoção de simulação tem como vantagem o treinamento do agente DQN "sem a necessidade de alocar várias horas de tediosas interações".

Em Akalin e Loutfi (2021), os autores observam que existem desvantagens do aprendizado através da atuação direta com um humano. Uma delas é que este tipo de tarefa pode ser tedioso e impraticável para usuários, pois é necessário diversos ciclos de interação demasiadamente repetitivas para que o agente aprenda, o que pode levar à fadiga e desinteresse por parte das pessoas. Além disso, uma quantidade consideravelmente grande de tempo pode causar um desgaste no hardware do robô. Desta forma, os autores mencionam que uma alternativa viável para esta questão é a utilização de um ambiente simulado para treinamento destes algoritmos. As vantagens desta abordagem vão deste a repetição incessante de cenários até a possibilidade de acelerar as simulações, permitindo o agente executar proporcionalmente mais experiências de aprendizado.

Apesar das vantagens da simulação, Akalin e Loutfi (2021) ainda ressaltam que simular e modelar comportamentos, normas sociais e incertezas do mundo real são muito difíceis. Desta forma, um dos objetivos associados a esta tese, é o desenvolvimento de um simulador para robótica social e aprendizado por reforço cujo objetivo é reduzir a lacuna existente na área de IHR que diz respeito a simulação de comportamentos sociais e interativos humanos.

Outro aspecto relevante encontrado ao analisar os trabalhos, é como alguns deles capturam e mapeiam informações sobre o ambiente, influenciando diretamente na abordagem de aprendizado utilizada. Como, por exemplo, os trabalhos de Broekens (2007), Papaioannou *et al.* (2017), Andriella, Torras e Alenyà (2019), Maroto-Gómez *et al.* (2021), Churamani *et al.* (2022), Hao *et al.* (2019), Li *et al.* (2019), Lin *et al.* (2020) e Bagheri *et al.* (2021), recorrem a algoritmos clássicos de RL, como *Q-Learning*, os quais são computacionalmente menos complexos e exigem menos que o agente explore o ambiente exaustivamente em relação a métodos baseados em DNN. Isto se dá graças a utilização de outras técnicas para o processamento e discretização prévia de informações ambientais. Contudo, isto só pode ser feito ao ter conhecimento de quais informações devem ser abstraídas do ambiente. A partir do momento que é necessário interagir com humanos, diversos aspectos devem ser considerados, dado que é necessário analisar, por exemplo, comportamentos corporais, direção do movimento, velocidade, foco de atenção, expressões faciais e tarefa atual do humano. Além disto, alguns comportamentos e intensões são dissimuladas e muito complexas de serem categorizadas, bem como, elas podem ser mais importantes ou não para as ações exercitadas pelo agente.

Boa parte dos trabalhos em Robótica Social atuam em tarefas que supõem que humano e robô já tenham estabelecido algum grau de engajamento previamente. Por exemplo, em Rudovic *et al.* (2019), o robô interage com crianças dispostas previamente a frente do agente, da mesma forma que em Clark-Turner e Begum (2018) e Papaioannou *et al.* (2017) alguns voluntários são

posicionados individualmente em frente aos robôs para que os autores validem suas hipóteses. Outra parte dos trabalhos atua na fase de pré-interação, visando analisar o ambiente, indivíduos ou grupos de pessoas para que agentes sociais atuem de forma socialmente aceita. Por exemplo, em Gao *et al.* (2019) os autores exercem esforços para que um robô aprenda a se aproximar de um grupo de pessoas, já em Qureshi *et al.* (2016) e Qureshi *et al.* (2018) o foco é direcionado para que o agente busque a atenção do humano e interaja conforme o comportamento humano extraído por imagens. Da mesma forma que esses trabalhos, esta tese também tem como objetivo contribuir na fase de pré-interação, onde o agente deve analisar comportamentos humanos que indiquem quando e como o robô deve agir para estabelecer uma interação.

Analisando o conjunto de trabalhos apresentados, é possível verificar a adoção tanto de paradigmas de RL clássicos quanto de abordagens mais avançadas e robustas. De forma geral, é notável que a escolha do paradigma de aprendizado visa lidar com a complexidade de informações capturadas ambiente, real ou não. Trabalhos que utilizam algoritmos de RL clássicos, como *Q-Learning*, por exemplo, geralmente utilizam um escopo limitado de informações do ambiente, o que pode negligenciar aspectos importantes para o aprendizado. Por outro lado, isso simplifica o treinamento do agente, pois a exploração do ambiente para aprender relações entre estados-ações se torna menos exaustiva. Como mencionado anteriormente, comportamentos humanos são complexos e inerentes a normas sociais. Desta forma, a utilização de técnicas mais complexas, como DQN, permite generalizar e abstrair padrões pertinentes da interação a partir de informações de alta dimensionalidade. Contudo, esta complexidade implica em um treinamento mais exaustivo e extenso, necessitando a configuração e validação de inúmeros parâmetros.

Desta forma, nesta tese são combinadas informações de alta dimensão (imagens) e dados pré-processados referentes a interação social (emoções humana), visando beneficiar o aprendizado do agente de forma que ele se comporte de forma socialmente aceita. Para isto, são consideradas as contribuições de vários trabalhos apresentados, em especial de Qureshi *et al.* (2016) e Qureshi *et al.* (2018), que se aproximam muito com o objetivo deste trabalho.

## 3.3 Considerações finais

Neste capítulo, foram apresentados alguns trabalhos relacionados com paradigma de aprendizado baseado em experiências com autorregulação para interação com humanos. Adicionalmente, foram considerados produções científicas que lidam com informações pertinentes ao comportamento humano, interação social e emoção humana. Através do estudo, análise e discussão desses trabalhos, foi possível verificar lacunas na área, bem como formas, técnicas e meios para atingir os objetivos desta tese.

A seguir, serão apresentados os Materiais e Métodos associados para o desenvolvimento da arquitetura proposta.

CAPÍTULO

# 4

# MATERIAIS E MÉTODOS

A pesquisa desenvolvida nesta tese tem como base o emprego e elaboração de diversas ferramentas e plataformas. Estas tecnologias visam oferecer apoio para desenvolvimento, treinamento e validação dos sistemas propostos. A seguir, serão abordados: a rede MDQN, utilizada como referência inicial para a SocialDQN; o simulador RHS, base direta para desenvolvimento do SimDRLSR; o robô Pepper, que atua como um agente social na fase de validação com o ambiente real; e modelos pré-treinados para reconhecimento de emoções visando auxiliar o robô Pepper na fase de interação com pessoas reais.

## 4.1 Multimodal Deep Q-Network

O modelo MDQN visa contribuir para que robôs sociais possam interpretar corretamente os comportamentos humanos para agir adequadamente com eles. Neste modelo, duas redes DQN são utilizadas para a extração automática de características para estimar a função ação-valor, uma para sequência de 8 imagens em tons de cinza e outra para imagens de profundidade. Esta rede foi proposta em Qureshi *et al.* (2016) e aperfeiçoada em Qureshi *et al.* (2018), conforme apresentado no Capítulo 3. Apesar de serem utilizadas contribuições de ambos os trabalhos, apenas Qureshi *et al.* (2016) dispõe de códigos e documentação (apesar de limitada) da rede MDQN abertamente para que a comunidade científica replique os resultados, evolua e utilize a arquitetura proposta. Desta forma, a implementação do método proposto nesta tese considera os códigos disponíveis em Qureshi *et al.* (2016) através do repositório dos autores[1].

Basicamente, a MDQN tem como objetivo realizar extração automática de características através das ConvNets para aproximar a função de valor de ação do método Q-learning (MNIH *et al.*, 2015). Como apresentado anteriormente, vários trabalhos vem mostrando a capacidade do método DQN em aprender por meio de entradas visuais de alta dimensão, atuando em várias

[1] <https://github.com/ahq1993/Multimodal-Deep-Q-Network-for-Social-Human-Robot-Interaction>

áreas da robótica, incluindo na área de IHR. A entrada para estas redes são imagens provenientes de uma das câmeras 2D e imagens de profundidade do sensor do robô. A rede MDQN utiliza estas entradas para selecionar um comportamento em um conjunto de 4 ações. Estas são: *wait*, *look towards human* (ou *look*), *handwave* (ou *wave*) e *handshake*. Nesta tese, serão adotadas as traduções "esperar", "olhar", "acenar" e "cumprimentar", respectivamente, para cada uma dessas ações.

Na primeira ação, "esperar", o robô altera sua orientação da cabeça aleatoriamente. Na segunda, "olhar", o robô direciona seu foco de visão para um humano. A ação de "acenar" envolve tanto o gesto com a mão quanto a saudação verbal "*Hello*" (Olá). A última ação, "cumprimentar", o robô projeta o braço com o intuito de cumprimentar a pessoa. Caso correspondido, o robô pronuncia uma frase de saudação para o humano.

O módulo principal da MDQN é representado pela função de aproximação ação-valor baseado em redes neurais profundas, tendo como entrada um estado $s$ e saída é um vetor *K-dimensional* onde o *k-enésimo* elemento corresponde à *k-enésima* ação. Os estados são tuplas de vetores de alta dimensão de imagens observadas do ambiente, sendo esta tupla composta por um canal de imagens em escala de cinza e um canal com imagens de profundidade. Desta forma, a rede é treinada para que cada valor de saída seja ajustado dado o retorno esperado.

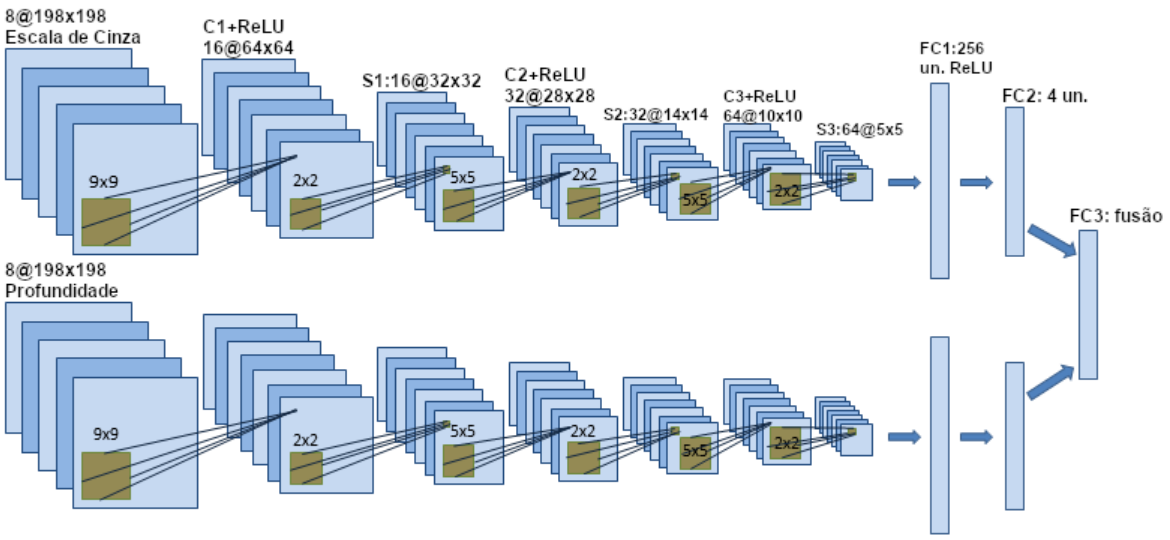
Tanto em Qureshi *et al.* (2016) e Qureshi *et al.* (2018) são utilizados o robô Pepper para treinar e validar os modelos propostos. Desta forma, a MDQN é desenvolvida com base nas capacidades e limitações do robô físico. Entretanto, este robô dispõe apenas de um sensor de toque atrás de cada uma das mãos, incapaz de detectar o toque humano durante um aperto de mão. Portanto, nos trabalhos em questão, foi desenvolvido um sensor externo em formato de luva, que se encaixa na mão direita do robô. Esta detecção é muito importante, pois permite modelar a recompensa do agente DQN.

Em RL, o aprendizado é realizado com base em recompensas geradas após a conclusão bem sucedida ou não de uma ação. Ao tocar a mão do robô, o sistema modela a recompensa da MDQN com um valor positivo durante a ação "cumprimentar". Caso contrário, se um humano não corresponder o gesto do agente, uma recompensa negativa (ou punição) é gerada.

O treinamento da MDQN é realizado dado os conceitos apresentados no Seção 2.1 em relação à *Q-Learning*. Além disto, os conceitos de DQN são agregados no modelo, como repetição da experiência (*experience replay*), para treinar o agente. A arquitetura, de fato, é representada por duas redes Q (*Q-Networks*) e suas estruturas implementam duas redes convolucionais, cada uma para um fluxo de dados (imagens em escala de cinza e profundidade), com uma saída dupla que é fundida gananciosamente, selecionando a ação com maior *q-value*. Estas redes são apresentadas na Figura 15.

A *Q-network* da MDQN possui dois fluxos de imagens, uma em escala de cinza e outra de profundidade, contendo alguns conjuntos de estruturas (conforme Figura 15), onde cada

Figura 15 – *MDQN*: implementação da Q-Network




Fonte: Adaptada de Qureshi *et al.* (2018).

conjunto é composto por uma camada convolucional, função de ativação não linear ReLU e agregação (*max pooling*) $2 \times 2$ de passo (*stride*) $2 \times 2$, que se repetem até a camada totalmente conectada FC1. As camadas convolucionais C1, C2, C3 e totalmente conectada FC1 possuem, respectivamente, filtros de $16 \times 9 \times 9$ com passo igual a 3, filtros $32 \times 5 \times 5$ com passo igual a 1, filtros $32 \times 5 \times 5$ com passo igual a 1 e $256 \times 5 \times 5$ com passo igual a 1. Para cada fluxo de imagem, esta configuração é aplicada, ou seja, o sistema é composto por duas redes independentes.

A configuração de rede em questão possui uma camada totalmente conectada de FC1 de 256 unidades em cada um dos fluxos, a qual é aplicada a função ReLU, reduzida para 4 unidades. Com isto, os dois fluxos de dados são fundidos, formando FC3, onde cada uma das 4 unidades representam uma ação que o agente deve executar.

Os autores separam o processo de treinamento em duas fases, coleta de dados e aprendizado. O objetivo desta divisão é evitar atrasos durante a interação com humanos enquanto a rede é treinada (QURESHI *et al.*, 2016). Na coleta de dados, o sistema interage com o ambiente usando *Q-network* $Q(s,a;\theta)$, executando uma ação usando a estratégia gananciosa ($\varepsilon - greed$) dadas as observações da cena atual (imagens de profundidade e em escala de cinza). O detector de eventos procura a ocorrência de um evento $e$ e a transição $(s(t),a(t),e(t+1),s(t+1))$ é armazenada na memória de repetição $M$. Esta memória mantém as N experiências mais recentes, as quais são utilizadas na fase de treinamento para atualizar os parâmetros da rede.

Na fase de aprendizado, a memória de repetição $M$ é utilizada para treinar a *Q-network*, a qual é otimizada através do método *RMSprop*. A recompensa para a rede é computada usando

os valores obtidos durante aquela interação, também recuperada da memória de repetição.

Em Qureshi *et al.* (2016) e Qureshi *et al.* (2018) o robô Pepper é utilizado com o sistema proposto para interagir com pessoas durante 14 dias, cada dia sendo considerado um episódio. A fase da coleta dos dados durava cerca de 4 horas e era seguida pela fase de treinamento. Foram utilizados 28.000 passos de interação, onde o parâmetro de exploração $\varepsilon$ decai linearmente de 1 até 0,1. As interações são armazenadas na memória de repetição $M$ para o treinamento do modelo, conforme mencionado anteriormente. O Algoritmo 5 apresenta a estrutura do *framework* MDQN.

**Algoritmo 5** – MDQN proposta em Qureshi *et al.* (2016)

Inicializar a memória de repetição $M$ com tamanho $N$
Inicializar *Q-network* $Q(s,a;\theta)$ a ser aprendida com parâmetros $\theta$
Inicializar *Q-network* $Q(s,a;\theta^-)$ alvo com parâmetros $\theta^- = \theta$
**para** cada episódio **faça**
  Inicializar o estado inicial como $s_1$ ▷ Fase de coleta de dados
  **para** cada passo do episódio **faça**
    Com $\varepsilon$-greedy, selecione $a_t = \begin{cases} \text{ação aleatória} & \text{com probabilidade } \varepsilon \\ \max_a Q(s(t),a;\theta) & \text{caso contrário} \end{cases}$
    $s(t+1), r_t \leftarrow ExecuteAction(a(t))$ ▷ Executar Ação
    Armazena a transição $(s_t, a_t, r_t, s_{t+1})$ em $M$
  **fim para**
  Aleatorizar a memória $M$ ▷ Fase de aprendizado
  **para** cada passo do episódio **faça**
    Selecione aleatoriamente um *minibuffer* $B$ de $M$
    **enquanto** $B$ **faça**
      Selecione uma amostra $m$ das transições $(s_k, a_k, e_{k+1}, s_{k+1})$ de $B$ sem substituição
      $y_j = \begin{cases} r_k & \text{se k+1 for terminal} \\ r_k + \gamma \max_{a'} \hat{Q}(s_{k+1}, a'; \theta^-), & \text{caso contrário} \end{cases}$
      Executar a descida no gradiente na função de perda $(y_k - Q(s_k, a_k; \theta))^2$ em respeito aos parâmetros $\theta$ da *Q-network*
    **fim enquanto**
  **fim para**
  Após cada C episódios sincronizar $\theta^-$ com $\theta$.
**fim para**

Segundo os autores, algumas ações tomadas pelo robô durante a validação do sistema mostram que o modelo proposto aprendeu a selecionar corretamente estas ações conforme a situação. Por exemplo, quando não havia ninguém ao redor do agente ou quando as pessoas estavam ocupadas, o robô aprendeu a executar a ação "esperar". Quando o robô está levemente engajado com as pessoas, ele aprendeu a "olhar" para as pessoas. Já a ação "acenar" é executada quando as pessoas estão longe e não estão olhando para o robô. A quarta ação é tomada quando a pessoa esta totalmente engajadas com o agente. Desta forma, os resultados indicam que o sistema consegue reconhecer e responder adequadamente aos complexos comportamentos interativos

dos seres humanos, tais como linguagem corporal, trajetória de caminhada, tarefa em andamento, etc.

Em Qureshi *et al.* (2018) é proposto um *framework* de DRL intrinsecamente motivada (DRL-IM), onde um agente obtém recompensas baseadas em motivação intrínseca por meio do modelo preditivo condicional à ação. Os autores apontam que o *framework* consegue prover a robôs habilidades sociais a partir das experiências de interação humano-robô em ambientes reais não controlados. Os resultados dessa pesquisa indicam que o robô aprendeu tanto habilidades sociais semelhantes às humanas, quanto também aprendeu a tomar decisões mais naturais e humanamente aceitáveis.

De forma geral, a evolução da MDQN apresentada em Qureshi *et al.* (2018) contribui no módulo de cálculo da recompensa do agente. Entretanto, os autores não divulgaram o código fonte para a comunidade, tão pouco base de dados e documentação técnica, inviabilizando a replicação e reprodução dos resultados apresentados pelos autores. Desta forma, nesta tese, são utilizados os códigos-fonte mais recentes, os quais não trazem a contribuição do módulo de recompensa intrinsecamente motivada. Isto não representa um problema, dado que nesta tese um dos focos de atuação é na abstração dos estados do ambiente em forma de sinais sociais e não no módulo de recompensas.

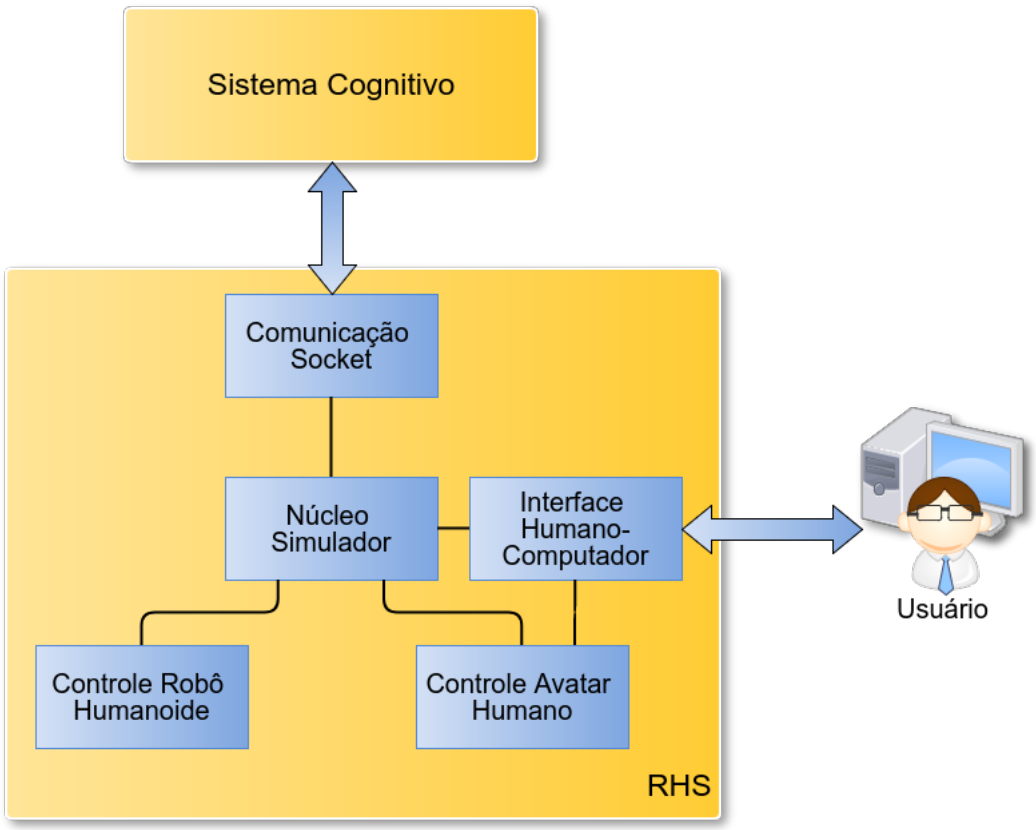
## 4.2 Robot House Simulator

Simuladores são ferramentas úteis para validar implementações antes de integrar um determinado sistema em um robô real. Além disso, é interessante evitar o uso desnecessário de robôs, pois sua vida útil diminui lentamente, prejudicando a técnica ou modelo testado devido ao desgaste do robô (ECHEVERRIA *et al.*, 2012; KUO *et al.*, 2013).

Na dissertação de mestrado em Belo (2018) é proposto o simulador RHS, destinado a validar sistemas de tomada de decisão voltados à robótica social em ambientes domésticos. As vantagens do simulador RHS são o uso de comandos robóticos e informações sensoriais de alto nível, visando simplificar o padrão de comunicação e o sistema de controle, sendo as informações sensoriais modeladas através da ontologia *OntPercept* (AZEVEDO; ROMERO; BELO, 2017). O simulador RHS foi desenvolvido através do motor de desenvolvimento de jogos Unity 3D (Unity Technologies, 2022).

O robô humanoide simulado é responsável por capturar informações do ambiente, seguindo a ontologia *OntPercept*, e as envia para o sistema de controle a ser validado, como um sistema cognitivo, por exemplo. Por sua vez, o sistema interpreta estes dados e constrói comandos para o agente simulado. O avatar humano é controlado por um usuário através de teclado, mouse e interface gráfica. Neste último recurso, o usuário pode inclusive digitar frases para que o avatar humano simule comandos verbais, permitindo a comunicação com o agente robótico e, consequentemente, com a arquitetura inteligente. Na Figura 16 é apresentada uma

Figura 16 – Visão Geral do simulador RHS




Fonte: Belo (2018).

visão geral da comunicação entre módulos do simulador RHS, sistema cognitivo e usuário.

Como mencionado, o sistema inteligente envia comandos de alto nível ao robô. O simulador RHS apresenta um conjunto de possíveis comandos e parâmetros para a execução de tarefas no ambiente. No Quadro 1 este comandos são agrupados, seguidos por uma breve descrição. O robô é dotado com comandos de interação, movimentação pelo ambiente, foco de visão (ou direção do olhar), comunicação e comandos operacionais. Cada comando deve estar associado a uma posição ou objeto de interesse. Por exemplo, o comando *"take cookie"* (pegar bolacha) faz com que o agente pegue um pacote de bolachas com uma das mãos. Para isto, o sistema cognitivo deve abstrair e processar as informações do ambiente para decidir se é possível manipular tal objeto. Caso o alvo seja inalcançável, o próprio simulador detecta que tal comando não é possível e gera uma resposta negativa para a arquitetura inteligente.

Na Figura 17a e na Figura 17b são mostradas a visão das câmeras de usuário associadas a cada agente. Na primeira, é possível visualizar as informações capturadas pelo agente, bem como a ação sendo executada por ele. Já na segunda, a é possível visualizar a interface gráfica de comandos do usuário para controlar o avatar humano. Na Figura 17c é possível visualizar os dois cômodos mapeados, sala de estar e cozinha. Já Figura 17d, o avatar humano executa uma ação de pegar o pacote de bolachas do robô após o usuário inserir tal comando na interface.

Quadro 1 – Comandos disponíveis no simulador RHS

| Categoria | Comando | Descrição |
|---|---|---|
| Interação | Pegar | Pega objeto com determinada mão. |
| | Deixar | Deixa objeto de determinada mão em uma dada localização. |
| | Ativar | Altera o estado de um objeto para aberto/ligado. |
| | Desativar | Altera o estado de um objeto para fechado/desligado. |
| | Experimentar | Prova o "sabor" de um objeto. |
| | Cheirar | Sente o "odor" de um objeto. |
| Movimentação | Movimentar | Movimenta o robô para um objeto ou posição. |
| | Rotacionar | Rotaciona o robô em seu próprio eixo dado o ângulo (em graus). |
| | Girar | Rotaciona o robô para que ele encare determinado objeto ou posição. |
| Foco de Visão | Olhar para | Rotaciona a cabeça de forma que o robô foque em um objeto ou posição. |
| | Procurar | Realiza uma varredura com a cabeça, considerando a visão, a procura de um objeto. |
| | Reiniciar olhar | Reenceta o foco da visão. |
| Comunicação | Falar | Simula a fala no formato de *string* para o usuário. |
| Operacional | Cancelar | Cancela todos os comandos da fila, inclusive o em execução, caso exista. |

Esta tese de doutorado tem como um de seus objetivos o desenvolvimento de um ambiente de desenvolvimentismo e validação de sistemas para robótica social e aprendizado por reforço profundo, o SimDRLSR. Para isto, o desenvolvimento deste sistema tira proveito das tecnologias, vantagens e elementos do simulador RHS. No entanto, no SimDRLSR o objetivo é que o avatar humano seja controlado por *scripts*, visando automatizar seus comportamentos, dispensando que usuários interajam exaustivamente com o sistema. Para isto, os comandos apresentados na Quadro 1 são utilizados no avatar humano. Já para o robô, este é dotado com as ações herdadas da MDQN, indisponíveis no RHS. Estas adaptações e implementações de novos módulos representam uma parte ínfima de todo o trabalho realizado para o desenvolvimento do simulador proposto. Sendo assim, no próximo capítulo serão apresentados os esforços realizados para alcançar os objetivos envolvendo o simulador SimDRLSR.

## 4.3 Robô Softbank Pepper

Segundo Christensen (2016), os robôs interativos devem ser projetados com a capacidade de entender o comportamento social humano, bem como exibir comportamentos sociais que seguem as normas aceitas da interação humana. Além disto, capacidades como o diálogo complexo, de interpretar e gerar sinais não-verbais e conseguir entender e expressar emoção são fundamentais para atingir o nível de sociabilidade exigido. Nesse contexto, o robô Pepper (Figura 18) se destaca dos demais sistemas robóticos por melhor atender os requisitos apresentados.

Figura 17 – Robot House Simulator

(a) Agente robótico (b) Avatar humano

(c) Sala e Cozinha (d) Avatar humano recebendo um objeto

Fonte: Belo (2018).

Uma unidade do robô Pepper foi adquirido pelo laboratório LAR através de projeto FAPESP número 2017/01687-0.

O robô Pepper (Figura 18) tem aparência semi-humanoide e possui capacidade de interagir emocionalmente com o usuário através de sua voz e movimentos gestuais expressivamente e natural. Esse robô usa informações imediatas para adequar suas ações ao estado do usuário.

Os quatro microfones localizados na cabeça do robô permitem detectar a origem de sons, além de identificar as emoções na voz de usuários. A partir disto, o robô realiza o rastreamento do rosto, caso uma pessoa for encontrada. Através do rosto, o robô consegue reconhecer o usuário e suas emoções. Ademais, o robô é capaz também de identificar objetos e detectar movimentos.

O robô Pepper é dotado de sensores táteis em suas mãos, peito e cabeça. Possui duas câmeras 2D e um sensor infravermelho para capturar profundidade. A navegação é auxiliada por dois transmissores e receptores de ultrassom, seis sensores a laser e três detectores de obstáculos, localizados em sua perna. Estes sensores permitem o robô Pepper detectar e desviar de objetos e obstáculos até um alcance de 3 metros.

Figura 18 – Robô semi-humanoide Pepper


Fonte: Elaborada pelo autor.

A interação com o ambiente é realizada de diversas formas. O robô consegue falar atualmente em inglês, mandarim e japonês, movimentar seus braços com o intuito de se comunicar, locomover-se no ambiente através de suas três rodas omnidirecionais e apresentar informações em seu *tablet*, localizado na altura do peito. É possível, também, personalizar a voz do Pepper para expressar emoções através da alteração de seu tom.

Neste projeto, a ideia é que o robô Pepper seja dotado com vários comportamentos (ações) de interação com pessoas, tais como acenar, cumprimentar, rastrear pessoas e esperar que alguém se aproxime ou entre em seu campo de visão, e através do modelo proposto, que o robô consiga aprender de forma autônoma quando executar cada uma das ações baseado nas informações visuais e sociais do ambiente. Estas informações serão dadas através da captura de imagens e reconhecimento de emoções humanas.

Como apresentado no Capítulo 1, o objetivo desta tese é a utilização de simulação para auxiliar no desenvolvimento, treinamento e validação da SocialDQN. A utilização do robô Pepper é dada na fase de transferência de conhecimento e validação qualitativa com pessoas e ambiente reais. Esta é uma tarefa importante, pois ela demonstra a capacidade de o simulador SimDRLSR e do modelo SocialDQN para atuar em ambientes reais sem a necessidade de treinamento em ambientes reais, somado os benefícios que o modelo proposto traz para interações sociais a partir da utilização de emoções humanas.

Apesar de parecer uma ferramenta promissora para pesquisas em robótica social, o robô Pepper apresenta diversas limitações de hardware e, principalmente, de software. O sistema operacional deste robô é baseado em um *Linux Gentoo*, porém os usuários não possuem permissão

para efetuar operações de super usuário ou de administradores, impossibilitando a atualização e instalação de diversos pacotes e bibliotecas de programação e recursos do sistema. Este problema se torna ainda mais evidente com a descontinuação de diversas bibliotecas e frameworks existentes no sistema do robô, principalmente por este sistema utilizar Python 2.7 como uma únicas linguagens de programação, que foi descontinuada e não recebe atualizações desde 2020.

Desta forma, a utilização de um servidor externo se faz necessário para trabalhar eficientemente e prática com o robô Pepper. Desta forma, nesta tese são utilizados dois nós de processamento, robô Pepper e servidor externo, ambos se comunicando por *sockets* TCP/IP, ficando o robô responsável por atuar e capturar informações do ambiente. Por sua vez, o servidor é responsável por processar, analisar e classificar as informações recebidas. Esta configuração permite a utilização de bibliotecas e frameworks atualizados, bem como a não limitação de hardware do robô.

Outra limitação do robô Pepper é em relação à detecção de face e reconhecimento de emoções. Este robô possui um módulo de detecção de "humor", capaz de detectar as emoções "felicidade", "raiva", "tristeza" e "neutra", bem como a valência da emoção e o foco de atenção humano. A limitação se encontra nas condições para esta operação, pois é necessário que o humano esteja de frente ao robô, com aproximadamente 1 metro de distância entre eles. Esta característica impede, por exemplo, do robô analisar emoções de pessoas transitando em um ambiente, ou até mesmo pessoas mais distantes. Desta forma, a utilização de um servidor externo, como mencionado anteriormente, se faz necessário visando utilizar modelos pré-treinados para reconhecimento de emoções através da face humana.

## 4.4 Reconhecimento de emoções através da face

Uma das principais características desta tese é a utilização de sinais sociais, em especial a emoção extraída através da face humana. Como apresentado anteriormente, as emoções podem indicar possíveis intenções ou estados internos humanos, informações que podem ser utilizadas a favor de robôs sociais.

Existem métodos mais flexíveis para detecção de emoções e que não dependem do robô Pepper para isto. O módulo *Facial Expression Recognition* (FER) disponibilizado em <https://pypi.org/project/fer/> provê uma biblioteca em Python, de fácil utilização e classificação de emoções. Ela agrupa um modelo de rede convolucional com pesos pré-treinados através da base de dados FER2013 (GOODFELLOW *et al.*, 2013), uma das bases estado-da-arte mais conceituadas da área (LI; DENG, 2020).

O módulo FER consegue categorizar emoções em expressões faciais em uma das sete classes: "raiva", "nojo", "medo", "feliz", "triste", "surpresa", e "neutra". O modelo é treinado com dados que consistem em imagens de rostos em tons de cinza de tamanho $48 \times 48$ *pixels*. Os rosto nestas imagens são centralizados para ocuparem aproximadamente a mesma quantidade de

espaço em cada imagem. Desta forma, o módulo mencionado realiza a extração do rosto das imagens para padronizá-las conforme o especificado no modelo.

É possível configurar o módulo FER de duas formas para detectar e extrair faces de imagens. A primeira através do classificador *Haar Cascade*, do framework OpenCV (VIOLA; JONES, 2004; HUAMáN, 2022), e a outra através da *Multi-task Cascaded Convolutional Networks* (MTCNN) (ZHANG *et al.*, 2016).

O *Haar Cascade* tem como base o algoritmo *AdaBoost* cujo objetivo é filtrar características relevantes a fim de determinar a existência de dado padrão em uma imagem. No caso aqui retratado, este padrão diz respeito a elementos da face humana. O processo de treinamento desse tipo de algoritmo é considerável bastante lento, mas, em contrapartida, são selecionados um conjunto pequeno de características para detecção, permitindo uma classificação quase instantânea na fase de detecção (ROCHA *et al.*, 2018).

Por sua vez, a MTCNN tem como base a correlação dos métodos de detecção facial e alinhamento facial. Ela recorre a três redes convolucionais em cascata, a *P-Net*, *R-Net* e *O-Net*, e possui um bom desempenho em tempo real. Contudo, em relação ao  a MTCNN é um pouco mais lenta, mas possui uma acurácia significativamente melhor. Mais detalhes da MTCNN podem ser encontrados em Zhang *et al.* (2016).

Nesta tese, o reconhecimento de emoções é utilizado na fase de validação com o robô real. Na fase de treinamento e na validação em simulação, a emoção é detectada pelo próprio simulador, que consegue acessar esta informação trivialmente, dado que este tipo de ambiente é totalmente controlado.

Inicialmente, o objetivo era utilizar a detecção de emoções disponível no robô Pepper, contudo, como mencionado, a detecção de emoções do robô é limitada a somente pessoas que estejam próximas o bastante do agente, além de não funcionar tão bem na detecção de rostos parcialmente oclusos. Desta forma, o módulo FER se torna uma escolha interessante para este projeto. Aliado a isso, é utilizado o método MTCNN para detecção de faces, dada a prioridade em classificar corretamente as emoções.

A seguir, será apresentada a arquitetura proposta do sistema de treinamento e validação para robótica social e aprendizado por reforço profundo, bem como será ilustrado o setor de atuação do módulo FER.

## 4.5 Arquitetura proposta

Durante a apresentação deste capítulo, foram abordadas diferentes tecnologias, materiais e métodos consolidados na literatura. O objetivo é agregar estes artifícios para alcançar os objetivos propostos. A utilização de técnicas e materiais já consolidados na literatura trazem robustez e segurança para o desenvolvimento deste projeto.

Na Figura 19 é apresentada uma visão geral da arquitetura proposta, composta pela SocialDQN, simulador SimDRLSR e o robô físico Pepper. A MDQN é utilizada como alicerce e guia de desenvolvimento da SocialDQN, que herda as ações, módulo de imagens para mapeamento parcial do estado, parâmetros da rede, dentre outras características.

Os estados da SocialDQN ainda contam com a representação vetorial da emoção detectada na face de um humano presente no ambiente. Esta informação é enviada à rede neural, que seleciona uma ação a ser executada. O módulo de comunicação da SocialDQN envia esta informação para o agente (simulado ou físico), que atua no ambiente e gera uma recompensa. Este dado é enviado à SocialDQN, que repete o ciclo de interação.

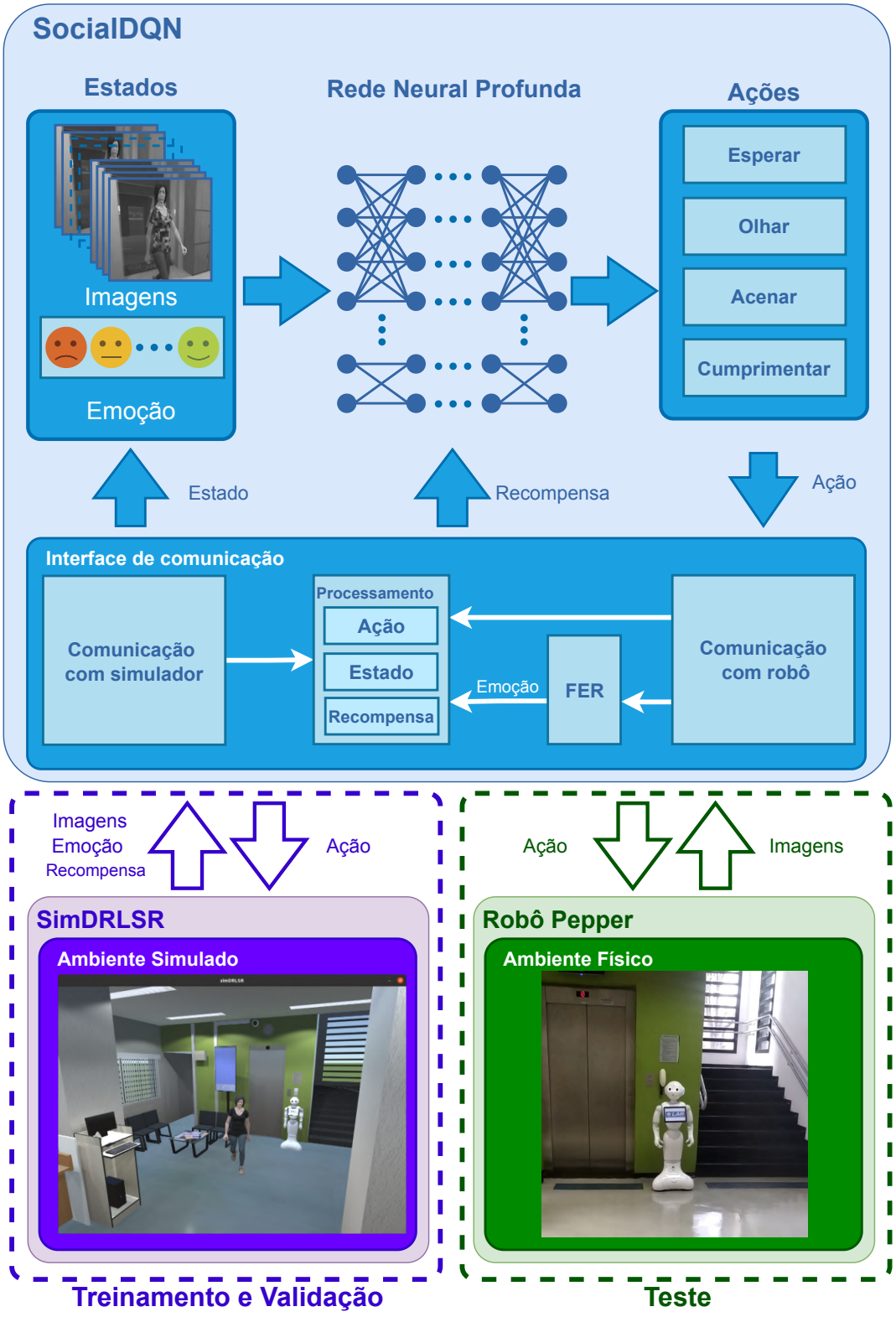
Na arquitetura proposta há dois módulos que representam o ambiente, um para treinamento e validação do agente SocialDQN e um para testes. O simulador SimDRLSR é responsável por gerar e gerenciar os elementos do ambiente virtualmente no primeiro módulo. Já o segundo, é representado pelo robô Pepper, que fica responsável por interagir no ambiente real.

Note que, somente no módulo de testes não há retorno da recompensa e nem da emoção humana. O primeiro se justifica porque nos testes não há aprendizado do agente, então a recompensa não é necessária. Já o segundo, a emoção é realizada externamente, ficando a SocialDQN responsável por detectar a emoção através do módulo FER. No simulador, a emoção é simplesmente recuperada através das variáveis da ferramenta, dado que o ambiente é totalmente controlado.

Dependendo do ambiente de atuação, a SocialDQN possui diferentes módulos de comunicação, visando configurar de forma específica cada um dos ambientes. Além disso, o módulo de comunicação pré-processa informações para envio e recebimento de ações, estados e recompensas do sistemas externos.

Nos próximos capítulos, serão apresentados detalhadamente cada um dos módulos do sistema proposto, tais como o simulador SimDRLSR, a arquitetura SocialDQN, bem como a integração entre estes módulos através de experimentos. O desenvolvimento de recursos para integração com o robô Pepper, tais como reconhecimento de emoções, será tratado com a arquitetura SocialDQN, e será validada nos experimentos.

Figura 19 – Arquitetura proposta para treinamento, teste e validação de sistemas de aprendizado por reforço profundo para robótica social




Fonte: Elaborada pelo autor.

CAPÍTULO

# 5

# SIMULADOR PARA APRENDIZADO POR REFORÇO PROFUNDO VOLTADO PARA ROBÓTICA SOCIAL

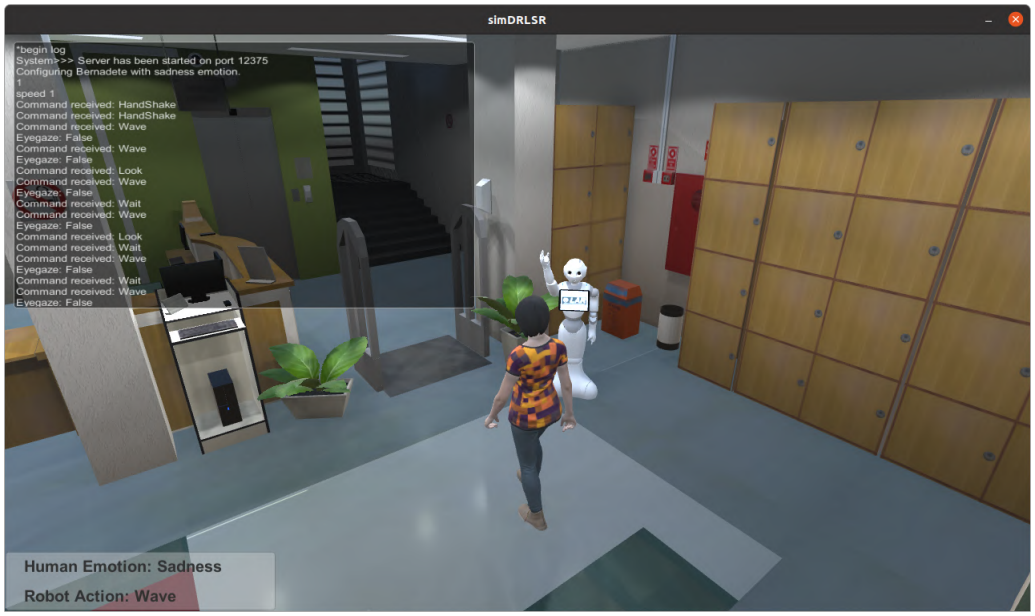
O SimDRLSR é um simulador cujo objetivo é fornecer uma ferramenta de treinamento e validação de sistemas para interação humano-robô com um paradigma de autoaprendizagem. Esta ferramenta fornece um ambiente para robôs sociais aprenderem e identificarem, através de imagens e sinais sociais processados, comportamentos humanos interativos e agirem de acordo com eles.

A concepção do simulador representa uma das primeiras etapas do plano de desenvolvimento proposto inicialmente. Isto é, mesmo antes da evolução da SocialDQN. Desta forma, o simulador proposto foi moldado, a princípio, conforme as especificações da MDQN proposta em Qureshi *et al.* (2016).

Na Seção 4.2 foi dissertado sobre o simulador RHS, o qual é voltado para validação de sistemas inteligentes e arquiteturas cognitivas para robótica social, sendo fruto direto dos trabalhos desenvolvidos no LAR do ICMC-USP. O RHS é utilizado como base de desenvolvimento do SimDRLSR dada a compatibilidade do foco de atuação de ambos os projetos.

Desta forma, o SimDRLSR herda dependências com algumas tecnologias. A principal delas é a plataforma utilizada para desenvolvimento, no caso, o motor de jogos Unity 3D (Unity Technologies, 2022). Esta ferramenta auxilia no desenvolvimento, programação, modelagem de objetos, construção de ambientes, física de corpos, renderização, dentro outros aspectos. A Unity pode ser adquirida gratuitamente e possui uma vasta documentação, bem como um repositório onde usuários podem compartilhar ou vender pacotes (*assets*) para desenvolvimento de objetos na plataforma. Na Figura 20, é ilustrada uma captura do simulador, onde é possível visualizar o agente interagindo com o ambiente e um humano transitando no mesmo.

Figura 20 – Interface gráfica do simulador SimDRLSR




Fonte: Elaborada pelo autor.

Apesar de ser uma ferramenta poderosa, a Unity não é especializada na modelagem e configuração de avatares humanos. Para isto, é possível utilizar a ferramenta Adobe Fuse[1] e a plataforma online Mixamo[2].

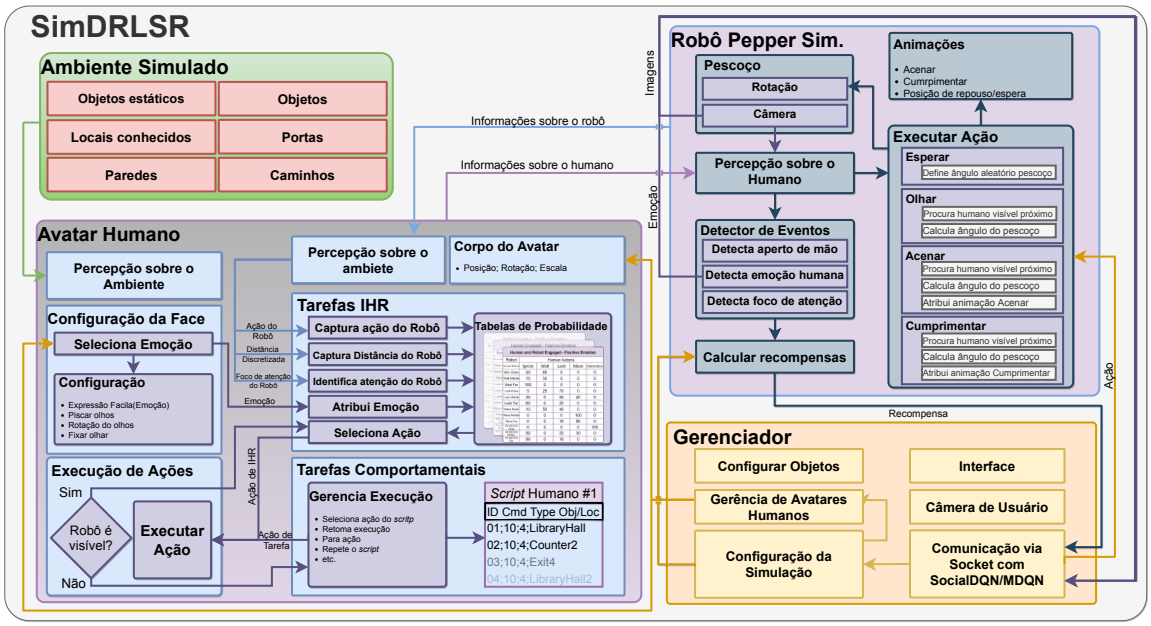
No simulador RHS, um usuário controla o avatar humano através de teclado e mouse, enquanto o agente robótico é controlado por um módulo externo a ser validado na plataforma. Um pouco diferente disto, os humanos no SimDRLSR são controlados automaticamente e configurados por *scripts* pré-definidos. Para isto, os avatares foram dotados com os comportamentos do agente robótico do RHS. Ou seja, no SimDRLSR, os humanos possuem as habilidades do robô RHS. Contudo, ao invés de ter uma arquitetura cognitiva controlando os humanos, existem *scripts* quem coordenam as ações humanas. Estas questões serão detalhadas em breve.

Na Figura 21 os componentes principais do simulador são apresentados, bem como as mensagens de comunicação de operação do sistema. Note que os componentes mais complexos são o *Robô Pepper Sim.* (robô simulado) e *Avatar Humano*. Esta complexidade é dada pela necessidade de modelar aspectos inerentes da interação entre humano e robô.

Além do robô e do avatar humano, o simulador possui outros dois módulos principais, o *Ambiente Simulado*, o qual representa a simulação de um ambiente físico, e o *Gerenciador*, responsável por comunicar com módulos externos, administrar, coordenar e sincronizar os agentes, bem como demais componentes da simulação. Além disso, o *Gerenciador* é responsável por fazer a interface com o usuário. Apesar do simulador não exigir um controle ativo de uma pessoa, é interessante que haja alguns elementos informativos, de controle de câmera e de

[1] https://www.adobe.com/br/products/fuse.htmly
[2] https://www.mixamo.com

Figura 21 – Visão Geral dos módulos do simulador SimDRLSR




Fonte: Elaborada pelo autor.

gerência da simulação.

A seguir, cada um dos módulos ilustrados na Figura 21 e suas particularidades serão apresentados com mais detalhes.

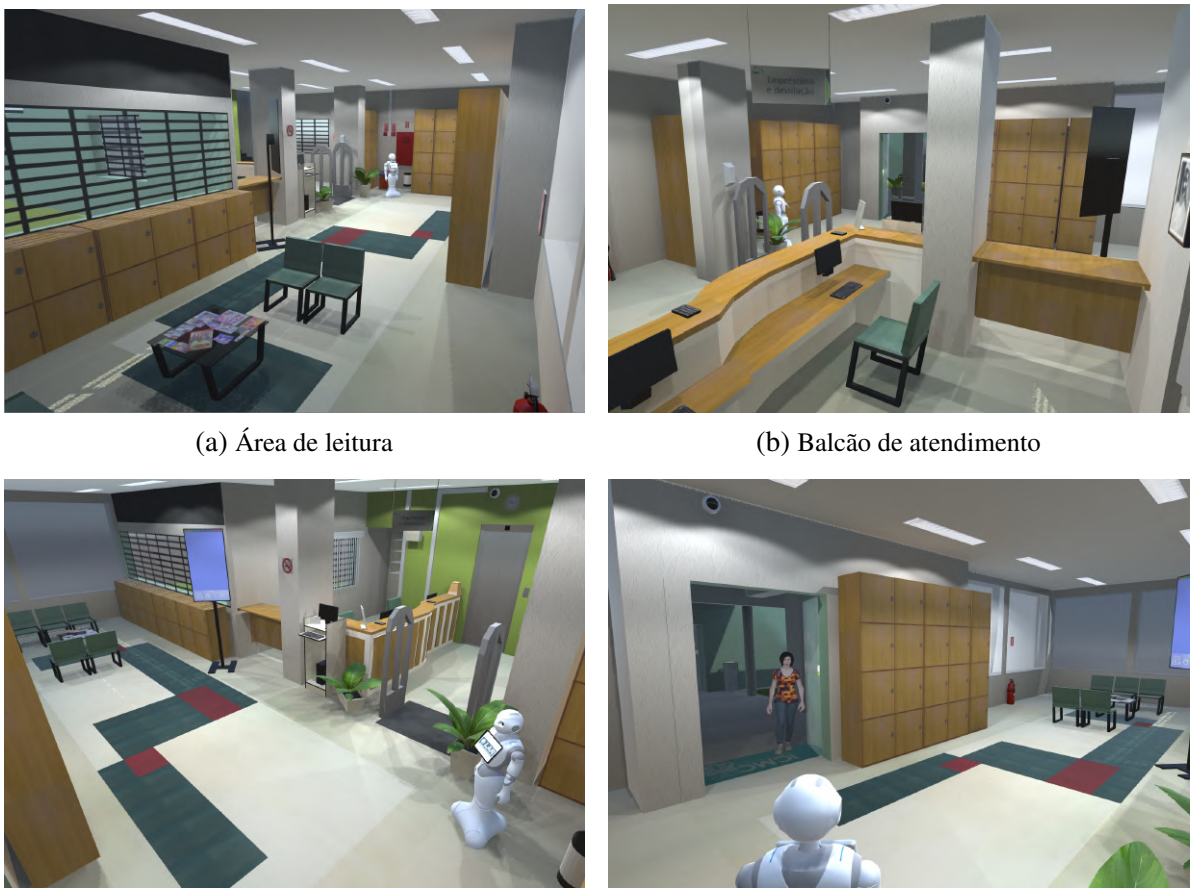
## 5.1 Ambiente Simulado

O ambiente é uma representação da biblioteca Achille Bassi presente no instituto ICMC-USP. Ele foi desenvolvido usando a própria Unity, com alguns *assets* gratuitos e livres disponíveis no repositório oficial da ferramenta.

Na Figura 22 o ambiente modelado é ilustrado em diversos ângulos. O cenário é composto por vários objetos e móveis passíveis de interação, possibilitando o avatar humano realizar tarefas comuns a pessoas reais, como abrir portas, sentar em cadeiras, pegar objetos, etc. A utilização de *Navmeshs* na Unity permite mapear possíveis caminhos para o humano transitar pelo ambiente inteligentemente, evitando obstáculos e transitando para outros cômodos através de portas e corredores. Esta funcionalidade facilita no desenvolvimento de ações para os avatares, pois todo planejamento de rota é feito em tempo real pela ferramenta, sendo necessário somente informar um ponto 3D para o avatar se movimentarem.

Além dos objetos visíveis, o ambiente foi configurado com vários pontos de interesse que auxiliam na navegação do avatar humano. Eles possuem identificadores únicos e servem como ponto de referência para inicializar o avatar em locais aleatórios e também na definição de novos comandos através de *scripts*.

Figura 22 – Primeiro cenário modelado no simulador SimDRLSR baseado na biblioteca Achille Bassi do ICMC/USP



(a) Área de leitura

(b) Balcão de atendimento

(c) Saguão de entrada

(d) Porta de entrada

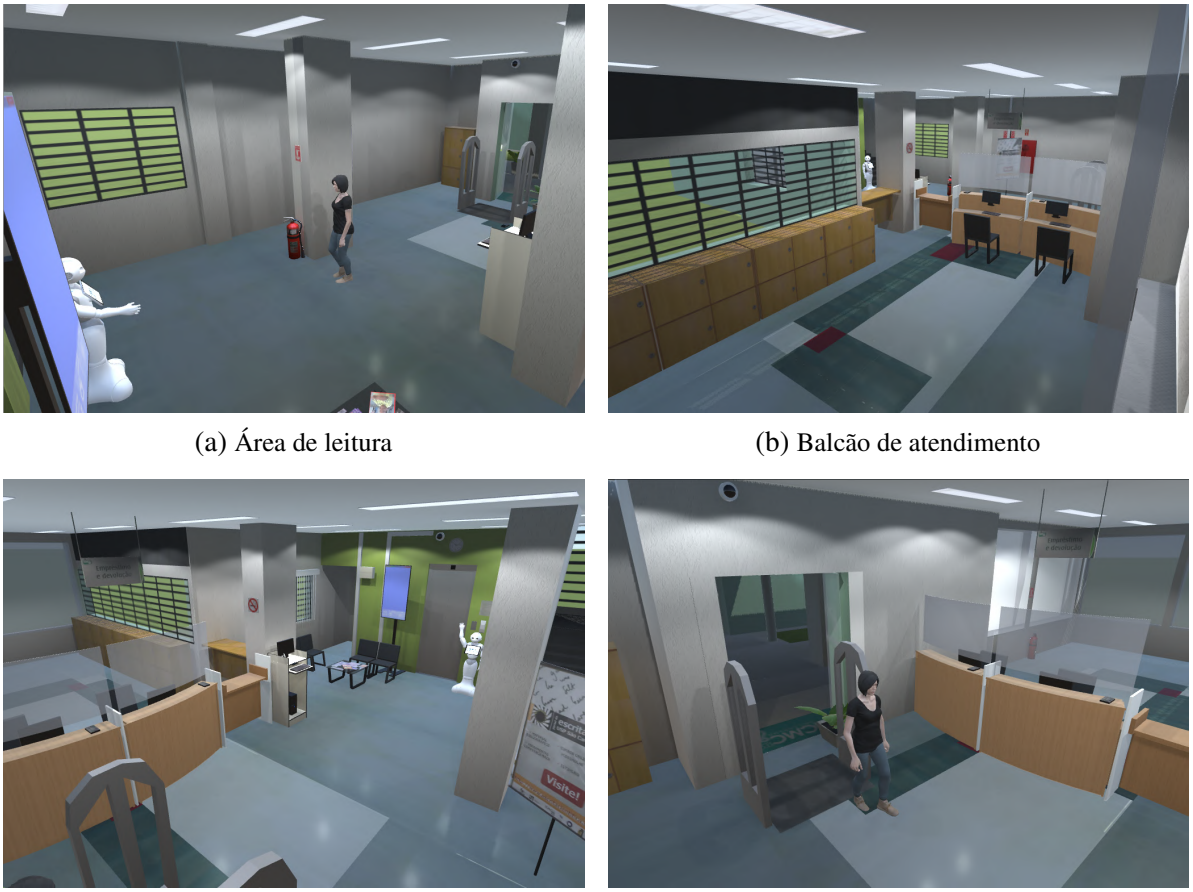
Fonte: Elaborada pelo autor.

De fato, durante o desenvolvimento, testes, validação e evolução de todo o projeto, o ambiente real sofreu algumas alterações. Desta forma, um segundo cenário foi mapeado, visando refletir a nova disposição dos móveis no ambiente. Na Figura 23 são compilados algumas capturas do cenário atualizado. Sucintamente, os dois ambientes e diferem pela disposição do balcão de atendimento e da área para leitura dos usuários da biblioteca. A atualização deste ambiente se fez necessário para a atuação do agente robótico na fase de validação com pessoas reais, visando beneficiar a transferência de conhecimento obtido no ambiente simulado para o robô real. A seguir, o robô simulado é apresentado.

## 5.2 Robô Pepper Simulado

O robô modelado no simulador é baseado no robô real *Softbank Pepper*, sendo respeitadas as limitações físicas de juntas e rotações. O robô foi desenvolvido para conseguir executar ações, capturar imagens em escala de cinza, profundidade, detectar eventos (sinais sociais), inicialmente suportando as necessidades da MDQN e da SocialDQN.

Figura 23 – Segundo cenário modelado no simulador SimDRLSR, baseado na disposição atual do ambiente físico da biblioteca Achille Bassi do ICMC/USP



(a) Área de leitura

(b) Balcão de atendimento

(c) Saguão de entrada

(d) Porta de entrada

Fonte: Elaborada pelo autor.

A partir de projetos abertos disponíveis na literatura, foram obtidos arquivos de configuração (*meshs*) para modelagem das partes do robô simulado. Cada membro e junta do robô Pepper teve que ser configurada individualmente, onde cada um destes foram agrupados para definir uma hierarquia. Por exemplo, o ombro esquerdo agrupa o braço, que por sua vez agrega o antebraço, este que compõe a mão esquerda com o punho, quer por fim, agrupa cada um dos dedos. A fisionomia do robô pode ser visualizada na Figura 24, o qual está executando a ação "acenar".

### 5.2.1 Ações do robô

A partir das especificações da MDQN, o robô foi dotado com as quatro ações apresentadas anteriormente, "esperar", "olhar", "acenar" e "cumprimentar". No simulador, estas ações têm uma duração de aproximadamente 3 segundos, com exceção da ação "cumprimentar", que depende diretamente da ação humana para se concretizar. Isto por essa ação ser dividia em partes. A primeira é responsável por fazer o agente projetar o braço e a mão para frente. Nesta etapa, o

Figura 24 – Robô Pepper simulado

Fonte: Elaborada pelo autor.

agente aguarda por cerca de 6 segundos que um humano toque em sua mão. Caso isso ocorra, o robô dá procedimento à ação, apertando a mão do humano. Caso contrário, o agente retorna o braço e a mão para suas posições iniciais.

Figura 25 – Robô Pepper e avatar humano se cumprimentando no simulador SimDRLSR

Fonte: Elaborada pelo autor.

As ações foram modeladas através de diferentes abordagens. Em “esperar” e “olhar”,

a rotação do pescoço é realizada através de *scripts*, capazes de executar a rotação do pescoço do agente robótico para fazer com que este direcione o olhar para dado ponto no ambiente. Para isto, são executados diversos cálculos de trigonometria e configurações para restringir o movimento do robô conforme as limitações do robô real. Na ação "esperar", é calculada uma posição aleatória para o agente focar sua visão. Já em "olhar", é procurado o humano visível mais próximo e a posição do rosto deste alvo é selecionada pelo sistema. Neste último caso, o valor é atualizado durante a execução da ação conforme o alvo se movimenta no ambiente.

No robô Pepper físico, a ação "olhar" usa informações sensoriais de som, câmera, tátil e movimento para a detecção da presença de humanos no ambiente. No simulador, este processo é realizado simplesmente buscando um humano dentro do campo de visão do robô e determinando a distância entre os agentes.

Já as ações "cumprimentar" e "acenar", são auxiliados por arquivos de animações, obtidos através da plataforma Mixamo (ADOBE, 2022). Estes elementos são responsáveis por configurar cada junta de avatares independentemente, modificando a posição e rotação em três eixos de atuação durante uma linha de tempo. Além disso, ao longo da execução dessas ações, os mecanismos utilizados na ação "olhar" são ativados em paralelo, visando deixar explícita a intenção de interação com dado humano através do foco de atenção do robô. Na arquitetura apresentada previamente (Figura 21), é possível visualizar um sumário das ações e suas sub-tarefas. Na Figura 25, é ilustrada a ação "cumprimentar" entre humano e agente robótico. Note que o robô olha para o rosto do humano durante a ação.

Como apontado, as ações representam um conjunto de processos de percepção, cálculo e operação, abstraindo operações menores a favor de execuções mais tangíveis ao nível social e operacional. O intuito disto é, principalmente, viabilizar o gerenciamento destas ações por parte de sistemas de aprendizado por reforço profundo, que tem sua capacidade de treinamento diretamente influenciada pelo número de ações a serem aprendizadas. Fazer com que estes sistemas executem cada sub-operação de uma ação (procurar humano, girar pescoço, projetar o braço para frente, etc.) pode trazer prejuízo ao modelo e dificultar a definição de recompensas ao sistema.

A partir da definição das ações, o sistema externo fica responsável por realizar a chamada destas a cada ciclo de interação. Em um sistema de RL, as ações do agente no ambiente podem gerar recompensas ou punições para o mesmo. No simulador SimDRLSR, a recompensa é modela através de estímulos capturados do ambiente pelo detector de eventos. Este sub-módulo é apresentado a seguir.

### 5.2.2 Detector de Eventos

O detector de eventos é um sub-módulo responsável por auxiliar na abstração de informações na perspectiva do agente robótico, em especial, na detecção de informações relativas à

interação com humano. Na versão atual do simulador, estes eventos são detectados analisando as seguintes ações e informações humanas: aperto de mão com o robô, foco de atenção e emoção extraída da face.

O evento "aperto de mão" ocorre quando a ação "cumprimentar" é correspondida por um avatar humano. Já o "foco de atenção" dá-se quando um avatar olha (ou permanece olhando) para o robô, mas este evento é somente detectado após o robô executar um aceno (ação "acenar"). Em ambos os casos, uma recompensa positiva é gerada ao detectar um dos eventos. Os valores destas recompensas são definidos através do módulo "Gerenciador", o qual recebe estes valores do sistema RL externo.

O terceiro evento é relacionado a sinais sociais e emoções. A emoção humana é uma informação atrelada a cada um dos humanos na simulação, desta forma, basta o robô buscar esta informação dado que se trata de uma simulação. Isto evita a necessidade de utilizar complexos algoritmos para detecção destes sinais.

Algumas restrições foram definidas para deixar a detecção das emoções mais próxima dos sistemas que trabalham em ambientes reais. Estas restrições envolvem verificar a emoção do humano mais próximo, que esteja visível no ângulo de visão do robô. Caso estas condições sejam satisfeitas, uma informação com a emoção é enviada ao sistema de aprendizado, caso contrário, a informação "sem face" é enviada ao módulo externo. O mapeamento das emoções e demais características do humano simulado serão apresentados em breve.

Como ilustrado na Figura 21, a detecção de eventos se inicia através da *Percepção sobre o Humano*, detecta e extrai informações pertinentes e as envia para o nó *Calcular recompensas*, com exceção da emoção humana, enviada ao *Gerenciador*, que se comunica e envia tal informação, somado a imagens capturadas do ambiente, para o módulo externo.

### 5.2.3 Captura de estados ambiente

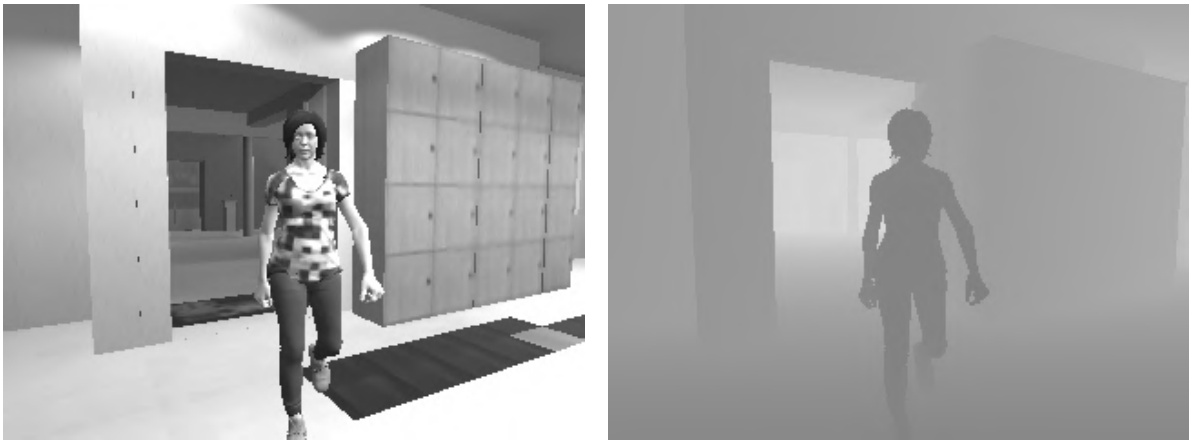
Em aprendizado por reforço, o estado é uma representação do mundo ou do ambiente onde determinada tarefa está sendo executada (SUTTON; BARTO, 2018). Na SocialDQN e na MDQN, são utilizadas imagens para essa representação.

As imagens são capturadas usando o pacote *Unity-ComputerVisionSim*[3] que oferece várias opções de captura, como segmentação, profundidade, cor, etc. Entre essas opções, são utilizadas imagens de profundidade e cores. No entanto, esta estrutura não capta imagens com um canal de cores em tons de cinza, um padrão usado pela MDQN. Desta forma, foram realizadas pequenas modificações no *framework* para capturar as imagens no padrão adotado.

São capturadas sequências de 8 imagens em escala de cinza e 8 imagens de profundidade. Cada imagem tem um espaçamento temporal de 125 milissegundos. Nas Figura 26a e 26b, são apresentados exemplos de imagens em escala de cinza e de profundidade, respectivamente. A

[3] <https://www.github.com/immersive-limit/Unity-ComputerVisionSim>

Figura 26 – Diferentes tipos de imagens capturadas pelo agente



(a) Escala de cinza (b) Profundidade

Fonte: Elaborada pelo autor.

MDQN utiliza de ambas as imagens, já a SocialDQN faz uso somente da sequência de 8 imagens em escala de cinza. Conforme as justificativas que serão apresentadas nesta tese, foi verificado que utilizar o fluxo de imagens de profundidade não é benéfico para a SocialDQN, por isso esse fluxo não é utilizado no modelo.

O robô oferece captura de imagens em escala de cinza e também de profundidade, com objetivo de mapear o estado do ambiente. Em validações iniciais, ambas as imagens eram salvas em disco, seguindo as limitações da MDQN. Contudo, essa abordagem depende que ambos os sistemas tenham acesso ao mesmo disco, limitando a utilização de nós de processamento independentes. Desta forma, as versões atuais do simulador permitem também o envio de mensagens via *socket* TCP/IP, assim como as mensagens de recompensas, ações e comandos de configuração.

A captura das imagens é realizada por uma câmera localizada na altura da testa do robô simulado. O campo de visão (em inglês, *Field of View* (FOV)) desta câmera é baseado na câmera do robô real, tendo aproximadamente 57º de FOV na horizontal e 44º de FOV na vertical. Atualmente, são capturadas 8 sequências de imagens em escala de cinza e profundidade, com intervalo de captura de aproximadamente 125ms. Elas são geradas com tamanho 640 $\times$ 480 *pixels*, podendo ser redimensionadas para 320 $\times$ 240 *pixels*. É importante ressaltar que as ações que modificam o foco de atenção do robô interferem na direção de captura destas imagens, influenciando diretamente na percepção do agente durante o treinamento.

Além das imagens, o agente captura, através do detector de eventos, a emoção do avatar humano. Como citado na Subseção 5.2.2, esta informação é resgatada diretamente, sem a necessidade de executar custos procedimentos de detecção e reconhecimento de faces e emoções. Isto só é possível graças ao ambiente controlado que a simulação consegue oferecer. Informações adicionais sobre as emoções humanas serão apresentadas a seguir, na seção sobre o avatar

humano.

## 5.3 Avatar Humano

O Avatar Humano possui o papel de emular comportamentos e interações pertinentes à interação social. Para isto, foram utilizadas ferramentas externas para auxiliar no desenvolvimento do modelo e das animações do avatar humano.

Figura 27 – Avatares humanos disponíveis no simulador SimDRLSR

Fonte: Elaborada pelo autor.

Através da ferramenta Adobe Fuse foi possível configurar diversos aspectos corporais, faciais, cor da pele, cor dos olhos, cabelo, e até mesmo de roupas e acessórios do humano. Na Figura 27 são ilustrados os diferentes avatares humanos modelados nessa ferramenta e importados para o SimDRLSR. As animações fazem parte da plataforma online Mixamo, que auxilia na criação e configuração das juntas do avatar, além de contar com milhares de animações que definem movimentos corporais, como movimentos de locomoção, sentar, acenar, cumprimentar, dançar, etc.

Na Figura 21 o *Avatar Humano* possui diversos módulos para interação com ambiente e robô. Os principais envolvem *Tarefas IHR*, responsável por selecionar ações de interação com o robô, *Tarefas Comportamentais*, para atuação no ambiente, *Execução de Ações*, para execução das ações selecionadas, e *Configuração da Face*, para configurar emoções, expressões faciais e direção do olhar (foco de atenção). Cada aspecto do avatar humano simulado será apresentado a seguir.

### 5.3.1 Expressões faciais e emoções

As emoções no simulador SimDRLSR são modeladas através da face do avatar humano. Uma das vantagens na utilização das ferramentas Adobe Fuse e Mixamo é o acesso e configuração da musculatura facial do avatar humano. A partir disso, é possível criar inúmeros aspectos e expressões faciais, incluindo emoções. Estas configurações são chamadas de *blend shapes*.

Figura 28 – Emoções expressadas através da face: neutra, medo, raiva (em cima, da esquerda para direita), surpresa, felicidade e tristeza (em baixo, da esquerda para direita).

Fonte: Elaborada pelo autor.

Nas *blend shapes* do simulador SimDRLSR é possível configurar detalhes do rosto para moldar emoções. A partir do trabalho de Ekman e Friesen (1971), e em especial da compilação de estudos em Paul Ekman Group (2022), foi possível configurar as *blend shapes* para criar a maioria das emoções básicas. Além disso, foram configuradas a expressão de piscar os olhos, o sorriso e a face neutra do avatar humano.

Contudo, algumas características ainda não estão disponíveis nas *blend shapes*, como algumas configurações de posicionamento do nariz e pálpebras, dificultando a configuração de emoções específicas. Com isto, a emoção “nojo” ainda não está implementada no simulador. Na Figura 28 são apresentadas as emoções atualmente mapeadas no SimDRLSR. Estas emoções são “neutra”, “medo”, “raiva”, “surpresa”, “felicidade” (feliz, alegre) e “tristeza”.

O simulador desenvolvido ainda permite o agrupamento de emoções, visando simplificar a configuração e execução de experimentos. É possível, por exemplo, agrupar emoções e expres-

sões faciais, tais como “felicidade” e expressão “sorrir”, em uma classe denominada “Positiva”, enquanto emoções “medo”, “raiva” e “tristeza” em uma classe denominada “Negativa”. Visando simplificar ainda mais, é possível configurar o simulador para detectar se a face do usuário é visível ou não. Neste caso, uma face visível seria caracterizada pela presença de uma emoção ou expressão facial. Ou seja, a classe “Rosto Visível” iria agregar todas as emoções, enquanto a classe “Não Visível” agregaria a informação “sem face”. Neste caso, o agente poderia inferir quando há um humano no cenário, evitando executar ações determinadas ações quando este esteja de costas ou até mesmo ausente do ambiente.

Outras configurações faciais incluem “piscar”, “movimentos aleatório dos olhos” e “direção do olhar do humano”. A direção do olhar modifica a orientação do pescoço e face quando o humano dirige sua atenção a um objeto, local, ou para o robô. Estas características têm como intuito deixar o comportamento facial do humano mais natural, dado que há esforços na área de robótica social em entender o comportamento corporal humano, sendo assim, é de grande importância a evolução nos simuladores robótico neste sentido. A seguir, o comportamento humano modelado no sistema é descrito detalhadamente.

### 5.3.2 Comportamento humano

O comportamento humano no simulador é dado através de *scripts* pré-definidos e de tabelas de probabilidade. Ações humanas *scripts* são arquivos de texto com comandos padronizados. São instruções dadas aos avatares para simular tarefas no ambiente simulado, como entrar no saguão da biblioteca, ir até à bancada, sair da biblioteca, ir até à lixeira, etc. Essas tarefas ajudam a emular pessoas se movimentando pelo ambiente. Os comandos inseridos nestas instruções seguem a estrutura apresentada anteriormente na Seção 4.2, apresentados do Quadro 1. Estes *scripts* pertencem ao primeiro grupo de tarefas, denominado como “Tarefas Comportamentais”.

Um possível *script* para configurar um avatar para se movimentar de um ponto a outro seguiria a seguinte estrutura: [*ID; COMMAND; PARAMETER_TYPE; LOCATION*], onde *COMMAND* receberia o valor *Move* (mover), *PARAMETER_TYPE* pode assumir os valores de *string*, ID ou posição 3D, e *LOCATION* seria a localização. Alguns locais conhecidos são predefinidos com um nome e ID fixos na simulação. Caso seja necessário que o avatar vá até à entrada da biblioteca, por exemplo, é definido o comando *[avatar01;Move; WithString; “LibraryHall”]*. Como mencionado, esse conjunto de ações é denominado “Tarefas Comportamentais”, pois delimitam uma rotina de tarefas estáticas para o avatar humano, independentemente da interação com o robô.

Diferente do grupo de comandos mencionado, o segundo grupo é voltado para interação direta com o robô, denominado como “Tarefas IHR”, cujo objetivo é definir ações para o avatar humano interagir com o robô. Estas ações são selecionadas com base em diversas variáveis como foco de atenção e ação do robô, distância da interação, engajamento, e emoção do humano. Além disso, cada ação é dada por probabilidades, moldadas através de tabelas.

As ações que compõe as “Tarefas IHR” servem como resposta a uma dada ação do robô. Dessa forma, algumas destas ações são semelhantes com as do agente robótico. Este conjunto de tarefas engloba as ações “esperar”, “olhar”, “mover”, “cumprimentar” e “ignorar”.

Na primeira ação, o humano espera o robô finalizar determinado comportamento. A segunda, “olhar”, é semelhante à anterior, porém é executada quando o humano ainda não está engajado na interação. Nesta ação, o avatar humano direciona o olhar para o robô, simulando um comportamento de curiosidade ou expectativa. Já em “mover”, o humano se movimenta em direção ao robô. Além disso, ele direciona o olhar para o agente. Na próxima, “Cumprimentar”, o avatar executa um aperto de mão com o robô, mas somente se o robô tomar a iniciativa de fazê-lo primeiramente e se ambos estejam suficientemente próximos. Por fim, a última ação permite o humano ignorar totalmente a ação do robô, executando comandos conforme a “Tarefas de Comportamento”. Esta ação serve tanto para evitar interagir com o robô (durante uma *tarefa comportamental*) quanto com a finalidade de finalizar uma interação com o robô.

Outra questão abordada na interação é o *engajamento*. Segundo Rudovic *et al.* (2019 apud SIDNER *et al.*, 2005), “engajamento é um processo em que várias partes estabelecem, mantêm e terminam de forma agradável uma conexão estabelecida durante uma interação conjunta”. Já em Rich *et al.* (2010) define que o engajamento ocorre, por exemplo, quando há contato visual entre agentes (pessoas e/ou robôs). No simulador, a variável que define o engajamento é responsável por mapear o tipo de envolvimento na interação, onde *apenas o robô*, *apenas o humano* ou *ambos os agentes* podem estar engajados. *Apenas o robô engajado* ocorre quando este tenta chamar atenção do humano, que pode estar executando uma *tarefa comportamental*. O segundo caso ocorre quando o humano está interagindo com o robô, mas este decide olhar para outra pessoa ou perde o foco de atenção com o humano (durante a execução de “esperar”, por exemplo, onde o robô olha para um local aleatório). Já o último caso, ocorre quando ambos estão com o foco de atenção e ações voltadas um para o outro.

Cada nível de engajamento é associado com uma tabela de probabilidade que define a ação utilizada pelo avatar humano na interação com o robô. Como exemplo, a Tabela 1 representa a probabilidade de executar uma dada ação dado que ambos agentes estejam engajados na interação. É possível observar que dois outros elementos contribuem na seleção da probabilidade: a ação executada pelo robô e a distância entre os agentes. Como exemplo, supondo que ambos os personagens estejam engajados na interação, o robô está executando a ação “cumprimentar” e a distância da interação é definida como *perto*, a probabilidade do humano também selecionar “cumprimentar” é de 100%.

Ademais, para cada tipo de emoção há um conjunto de tabelas de probabilidade para cada tipo de engajamento. Por exemplo, se o simulador for configurado com 7 classes de emoções, serão necessárias 21 tabelas de probabilidade, combinando cada emoção com cada um dos 3 tipos de engajamento. Desta forma, a emoção e o engajamento do avatar ditam seu comportamento nas *tarefas de IHR*. De outro ponto de vista, é possível dizer que as emoções no simulador definem

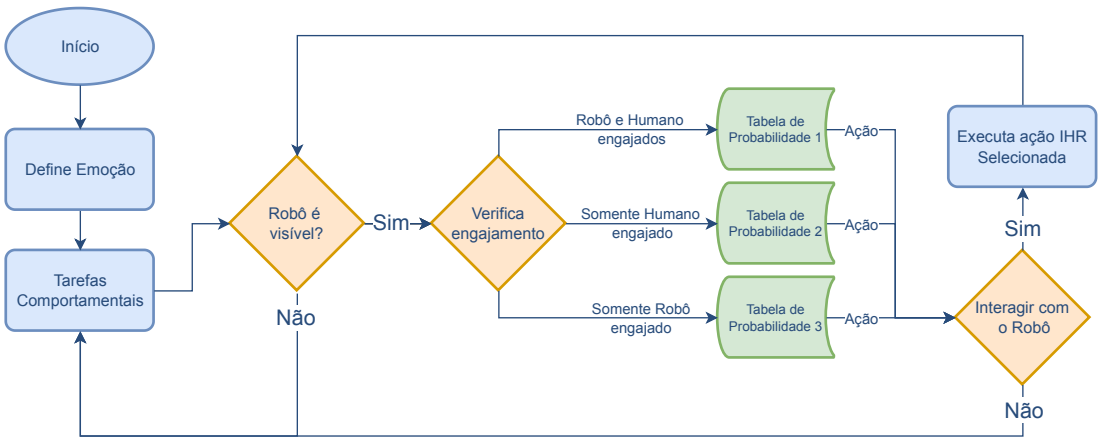
Tabela 1 – Tabela de probabilidade do avatar humano executar uma dada ação dada a ação do robô. Os valores abaixo dizem respeito a interação onde humano e robô estão engajados na interação.

| Ações do Robô | Distância | Ações Humanas (%) | | | | |
|---|---|---|---|---|---|---|
| | | Ignorar | Esperar | Olhar | Mover | Cumpr. |
| Esperar | Perto | 20 | 80 | 0 | 0 | 0 |
| | Meio | 70 | 30 | 0 | 0 | 0 |
| | Longe | 100 | 0 | 0 | 0 | 0 |
| Olhar | Perto | 5 | 25 | 70 | 0 | 0 |
| | Meio | 30 | 0 | 50 | 20 | 0 |
| | Longe | 80 | 0 | 20 | 0 | 0 |
| Acenar | Perto | 10 | 50 | 40 | 0 | 0 |
| | Meio | 0 | 0 | 0 | 100 | 0 |
| | Longe | 0 | 0 | 10 | 90 | 0 |
| Cumpr. | Perto | 0 | 0 | 0 | 0 | 100 |
| | Meio | 20 | 0 | 50 | 30 | 0 |
| | Longe | 80 | 0 | 20 | 0 | 0 |

a tendência do humano em interagir com o robô ou ignorá-lo. Por exemplo, se o humano está longe, não está engajado na interação e está triste, a probabilidade de olhar ou se mover para o robô é menor que se ele estivesse feliz. As emoções são atribuídas ao humano no início da simulação com tabelas de probabilidade pertinentes a esta emoção.

Estas tabelas de probabilidade representam um modelo simplificado de representar o comportamento humano, que é complexo, dependente de várias outras variáveis humanas. Apesar disso, os comportamentos providos pelo simulador são suficientes para auxiliar no desenvolvimento e validação da SocialDQN nesse trabalho, dada a capacidade das redes neurais profundas em abstrair e generalizar padrões complexos.

Figura 29 – Fluxo de decisão para selecionar a ação do avatar humano.




Fonte: Elaborada pelo autor.

Como mencionado, o humano possui dois conjuntos de ações, um para interação com o ambiente e outro para interação com o robô. O diagrama presente na Figura 29 representa o fluxo simples do comportamento do humano simulado. O sistema inicia configurando uma emoção ao humano. Logo após, o avatar executa a lista de ações predefinidas na "Tarefas Comportamentais". Caso o robô esteja visível, rapidamente o sistema verifica o tipo de engajamento da interação e consulta a tabela de probabilidade correspondente, que varia conforme a emoção do avatar. Logo após, é verificada a ação e distância do robô seguindo a tabela associada. Através desta verificação, é selecionada uma ação utilizando as probabilidades demarcadas na tabela. Caso o humano decida interagir com o robô, ele interrompe a ação atual e executa imediatamente a ação selecionada através da tabela de probabilidade. O humano então verifica novamente todas as variáveis pertinentes à interação com o robô e interage com este até decidir ignorá-lo. Quando isso acontece, as *tarefas comportamentais* são alocadas novamente e o humano continua a se movimentar e interagir com o ambiente.

Note que o engajamento pode mudar a cada interação. Geralmente, a comunicação se inicia com o robô engajado. Posteriormente, o humano adere à interação, levando a um engajamento mútuo. O robô pode ainda dar preferência por interagir com outra pessoa (caso haja mais de um avatar na simulação) ou mudar o foco de visão. Com isto, somente o humano está engajado, levando a seleção de uma tabela de probabilidade que incita o humano ignorar o robô.

É importante ressaltar que a definição e utilização de tabelas de probabilidade são um esforço inicial para modelar o comportamento humano sob diferentes circunstâncias. Ainda é necessário aprofundamento nas áreas que se comprometem a entender comportamentos e normas sociais, bem como o efeito das emoções nas ações e intenções humanas.

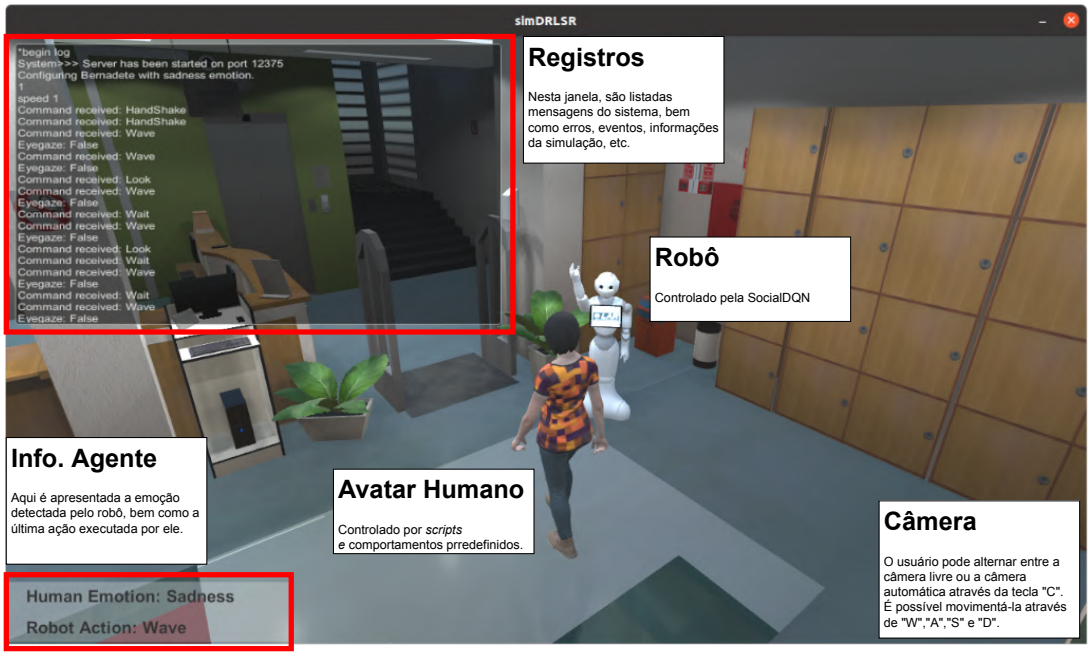
## 5.4 Módulos de gerenciamento e interação com usuário

O último módulo diz respeito ao gerenciamento do simulador. Este componente é responsável pela configuração de objetos, gerar interface do usuário, definir parâmetros de simulação, gerenciar avatares humanos, controlar a câmera do usuário e, não menos importante, realizar a comunicação de duas vias com o sistema RL através de *socket* TCP/IP.

Através do submódulo de comunicação, o simulador consegue receber mensagens de configuração do sistema, como velocidade da simulação, tamanho da tela, qualidade da renderização, valores para recompensas, tipo de sinal social a ser recebido (emoção, agrupamento de emoções, etc.), se o robô e o humano ficarão em posições aleatórias, entre várias outras.

Esse submódulo é responsável ainda por receber ações a serem executadas pelo agente no ambiente, além de coordenar o envio de recompensas, sinais sociais e imagens para o sistema inteligente. Neste processo, as mensagens são convertidas para bytes, visando padronizar a comunicação entre diferentes linguagens de programação.

Figura 30 – Interface de usuário e alguns elementos do simulador




Fonte: Elaborada pelo autor.

Outro papel do módulo de gerenciamento é a comunicação entre usuário e simulador. Na Figura 30, a interface do SimDRLSR é apresentada, onde são ressaltados os principais componentes visíveis através da simulação. Na tela de registros, o módulo de gerência consegue apresentar mensagens de erros, avisos e informações pertinentes à simulação. Na parte inferior são informadas a última ação executada pelo agente e a emoção que este detectou. Neste caso, esta informação é a mesma enviada para o sistema RL externo, e diz respeito a emoção do humano mais próximo com rosto visível. Na interface também é possível visualizar o agente robótico e o avatar humano, os quais não podem ser controlados pelo usuário.

## 5.5 Outras características

Na Unity, é possível executar o projeto pela própria ferramenta ou compilá-lo e gerar um executável. Nesta tese, é utilizada a versão compilada do SimDRLSR, pois isto permite economizar recursos do sistema, bem como configurar a qualidade e resolução da tela, e executar diversas instancias do simulador a fim de agilizar o treinamento do sistema de aprendizado.

Por fim, o simulador SimDRLSR está disponível abertamente para a comunidade através do repositório GitHub, no link <github.com/JPedroRBelo/simDRLSR>, sob licença GNU GPL v3.0. No repositório, ainda são disponibilizadas algumas versões compiladas durante o desenvolvimento do projeto.

## 5.6 Considerações finais

Neste capítulo, foi apresentado o simulador SimDRLSR, voltado para validação de sistemas de autoaprendizado e robótica social. Além de contribuir para estas áreas, o simulador tem como objetivo direto a implementação de um ambiente para treinamento, testes e validação da arquitetura SocialDQN.

Assim, como o simulador SimDRLSR, a arquitetura SocialDQN representa um dos esforços para alcançar os objetivos propostos nesta tese. Esta arquitetura será apresentada a seguir.

CAPÍTULO

6

# ARQUITETURA DE APRENDIZADO POR REFORÇO PROFUNDO E ROBÓTICA SOCIAL

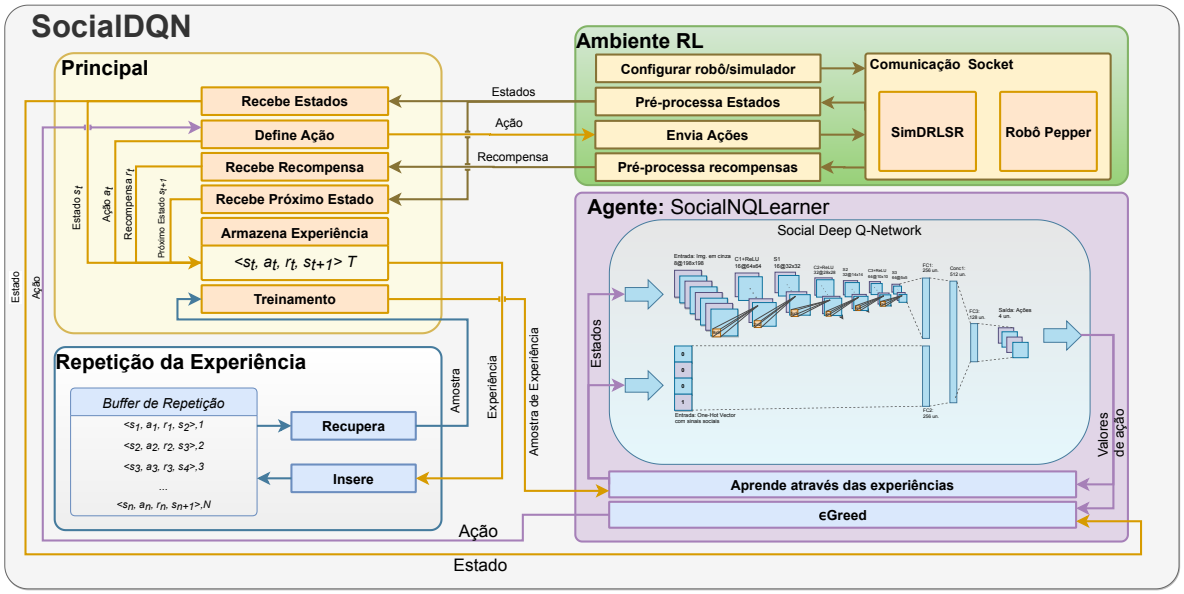
Neste capítulo, será apresentada a arquitetura SocialDQN, um dos pilares deste trabalho. Esta é uma arquitetura que agrega o uso de sinais sociais provenientes da interações entre humano e robô a conceitos bem estabelecidas na literatura, tais como *Q-learning*, DNN e a fusão destas duas técnicas, os modelos DQNs.

A utilização de imagens do ambiente para treinamento da rede representa uma forma de abstrair informações complexas da interação humana que podem ser inerentes a aspectos culturais e ambientais, gerando comportamentos que são difíceis de serem catalogados e classificados. Contudo, algumas informações da interação humana podem ser facilmente abstraídas por métodos computacionais, tais como, emoções e expressões faciais. Estas informações servem como dicas ao robô de como ele deve interagir com humanos. Desta forma, a SocialDQN agrega tanto imagens do ambiente quando informações emocionais para compor os estados em cada ciclo de interação. A cada um destes, o agente armazena experiências, com intuito de treinar um modelo de referência para robôs interagirem neste ambiente.

A partir da revisão na literatura, apresentada no Capítulo 3 foi possível levantar pesquisas, trabalhos e arquiteturas voltados para área de robótica social. Dentre as arquiteturas, a MDQN é o modelo que mais se aproxima dos objetivos definidos nesta tese. Portanto, o intuito é utilizar conceitos desta arquitetura para modelar elementos da SocialDQN, poupando esforços na validação de certos aspectos de interesse parcial desta tese. Isto permite a evolução e aprimoramento de mecanismos pouco explorados na literatura, como, por exemplo, a utilização de sinais sociais e emocionais para guiar o aprendizado de robôs sociais por reforço.

Diferente do modelo base, a SocialDQN define uma abordagem para treinamento do agente por milhares de episódios, abstraindo emoções pré-processadas para compor parte dos estados do ambiente. Os módulos que compõe internamento a SocialDQN são apresentados na Figura 31. Os componentes contidos na arquitetura serão descritos a seguir, tais como *Ambiente*

Figura 31 – A estrutura SocialDQN segue o padrão de um sistema tradicional de aprendizado por reforço. O agente captura os estados, seleciona uma ação por meio de uma política ($\varepsilon$-greedy), recebe uma recompensa, extrai informações de estado novamente e armazena essa interação em um *buffer* de repetição. Periodicamente, o sistema treina o modelo. O ciclo se repete até que o algoritmo atinja um estado terminal e possa reiniciar esse ciclo por $n$ iterações.




Fonte: Elaborada pelo autor.

*RL*, *Agente*, *Repetição da experiência* e *Principal*. Além disso, é possível observar os estados, recompensas, ações e demais aspectos pertinentes ao treinamento e comunicação com o robô. Em especial, também serão apresentados os estados capturados do ambiente gerados a partir do uso de emoções e imagens em escala de cinza.

A seguir, a estrutura principal do sistema é apresentada.

## 6.1 Estrutura da arquitetura

Esta arquitetura segue a estrutura clássica de um sistema de DRL onde agente, ambiente, rede neural, etc. são criados e inicializados a partir de parâmetros e hiper-parâmetros pré-definidos para a execução, treinamento e validação do modelo.

Na sequência, o sistema inicia capturando o estado do ambiente no tempo $t$. Esta informação é enviada ao agente *SocialNQLearner*, que seleciona uma ação a partir do algoritmo $\varepsilon$-greed. É importante ressaltar que o termo *agente* em RL difere do adotado no contexto do simulador SimDRLSR. Na SocialDQN, o agente é um módulo que recebe informações do estado e gera uma ação, utilizando uma rede neural profunda para a tomada de decisão e aprendizado.

A partir da seleção da ação pelo agente *SocialNQLearner*, esta informação é direcionada

ao módulo *Ambiente RL*, responsável pela comunicação com o robô real ou simulado. Uma recompensa do ambiente é recebida como resposta à ação do robô. O sistema mais uma vez captura o estado do ambiente, porém no tempo *t+1*. Na Figura 31 setas de conexão entre os módulos representam essa movimentação contínua de informação.

Baseado na rede MDQN, a arquitetura SocialDQN mapeia quatro ações para a interação do robô com o ambiente, *esperar*, *olhar*, *acenar*, e *cumprimentar*. A descrição destas ações pode ser consultadas na Seção 4.1. É esperado que o agente execute cada uma destas ações no contexto correto, tentando chamar atenção do humano, até que ele consiga obter a maior recompensa possível a partir de uma execução bem sucedida da ação *cumprimentar*.

O agente armazena no *Buffer de Repetição* as experiências geradas a partir de sua experiência no ambiente. Cada experiência, associada a execução no tempo $t$, é representada pela tupla $< s_t, a_t, r_t, s_{t+1} >$, sendo o termo $s_t$ a combinação da sequência de 8 imagens em escala de cinza e sinais sociais obtidas a partir da emoção, $a_t$ a ação tomada para esse estado, $r_t$ a recompensa dada pelo ambiente, e $s_{t+1}$ o estado no tempo $t+1$. A *Repetição de Experiência* armazena essa transição, sobrepondo as mais antigas caso a *Buffer de Repetição* atinja seu tamanho máximo.

O ciclo se repete, com o agente coletando informações do ambiente e executando ações, enquanto aprende as ações que maximizam o valor da recompensa. Diferente da rede MDQN, onde havia a etapa de coleta de dados e de treinamento, na rede SocialDQN o treino é realizado a cada $n$ interações com o ambiente. Isso garante um aprendizado mais suave do agente, pois o modelo é treinado constantemente, o que permite avaliar e atualizar os pesos da rede DQN de forma que ela não fique presa a mínimos locais.

A rede SocialDQN é treinada por diversos episódios nos quais o agente interage no ambiente até atingir um estado terminal. Isto é alcançado quando o robô consegue executar a ação *cumprimentar* com sucesso, ou seja, um humano corresponde a esta ação. Outro estado terminal é alcançado quando uma quantidade pré-definida de interações é atingido, podendo resultar em uma recompensa negativa.

O módulo *Principal* comunica diretamente com *Ambiente RL*, este último podendo ter várias cópias, cada uma conectada com uma instância do robô (real ou simulado). O módulo *Ambiente RL* tem como objetivo configurar o agente robótico, processar estados e recompensas, e padronizar a comunicação com o sistema externo.

No Algoritmo 6 é ilustrado o pseudo algoritmo da SocialDQN, baseado no trabalho pioneiro de Mnih *et al.* (2015) que propôs as DQNs. O algoritmo segue a mesma lógica de execução apresentada acima, mas é fundamental destacar alguns pontos referentes ao treinamento dos agentes. Para fins de padronização com o algoritmo apresentado em Mnih *et al.* (2015), é utilizado $\phi$ como uma função para processar os estados de entrada.

De várias exemplos de interação armazenados no *Buffer de Experiências*, a SocialDQN

**Algoritmo 6** – Pseudo algoritmo da SocialDQN

**Entrada:** 8 imagens em escala de cinza($s_{im}$) e um vetor *one-hot* com sinais sociais ($s_{em}$)
**Saída:** função ação-valor $Q$
Inicializa *Buffer* de Repetição $D$
Inicializa função ação-valor $Q$ com pesos aleatórios $\theta$
Inicializa função ação-valor alvo $\hat{Q}$ com pesos $\theta^- = \theta$
**para** episódio = 1, $M$ **faça**
  Inicializa $S_1 = \{s_{im_1}, s_{em_1}\}$ e sequência preprocessada $\phi = \phi(S_1)$
  **para** t = 1, $T$ **faça**
    Com $\varepsilon$-greedy, selecione $a_t = \begin{cases} \text{ação aleatória} & \text{com probabilidade } \varepsilon \\ \arg\max_a Q(\phi(S_t), a; \phi) & \text{caso contrário} \end{cases}$
    Execute ação $a_t$ na simulação e receba recompensa $r_t$, e estados $\{s_{im}, s_{em}\}$
    Defina $S_{t+1} = S_t, a_t, \{s_{im}, s_{em}\}$ e preprocesse $\phi_{t+1} = \phi(S_{t+1})$
    Armazena transição $(\phi_t, a_t, r_t, \phi_{t+1})$ em $D$
  **fim para**
  Selecione randomicamente uma amostra pequenas de transições $(\phi_j, a_j, r_j, \phi_{t+j})$ de $D$
  Set $y_j = \begin{cases} r_j & \text{para estado terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-), & \text{para estado não-terminal } \phi_{j+1} \end{cases}$
  Executa um passo de descida no gradiente em $(y_i - Q(\phi_j, a_j; \theta))^2$ ▷ Equação 2.18
  A cada $C$ passos reinicie $\hat{Q} = Q$, ou seja, defina $\theta^- = \theta$
**fim para**

seleciona aleatoriamente um conjunto de algumas dezenas de experiências para treinamento. Nesta etapa, a rede considera a função de perda (*loss*) $Q$ referente à rede alvo $\hat{Q}$. A rede $Q$ visa calcular a qualidade dos valores para a execução de uma ação $a_j$ dado o estado $\phi$. A rede alvo processa os estados futuros ($\phi\, j+1$), considerando a ação $a'$ que maximiza o retorno da rede. A SocialDQN treina seu modelo executando uma etapa de gradiente descendente de acordo com Equação 2.18.

O treinamento visa fazer com que a rede aprenda a estimar valores de ação dos estados atuais em relação aos estados futuros. O objetivo é que o agente aprenda a estimar sua recompensa ao realizar uma ação em um estado específico. No entanto, o agente deve realizar o maior número de interações possível, tentando aprender a relação entre estado e ação. Como são utilizadas imagens para mapear os estados, essa relação se torna computacionalmente inviável, justificando o uso de redes neurais para generalizar e abstrair padrões dessas relações.

## 6.2 Estados do ambiente

O objetivo do agente RL é, através das ações, alcançar estados que maximizem a recompensa cumulativa. Para isto, é desejado que o agente alcance um *estado terminal*, que pode indicar sucesso na tarefa desempenhada. Na SocialDQN, há duas situações que representam um estado terminal, a primeira acontece no momento que o humano corresponde o gesto *cumprimentar* do robô e o segundo quando este executa $n$ interações no ambiente sem ter sucesso nessa mesma

ação.

Independentemente de ser terminal ou não, um estado pode ser considerado uma representação total ou parcial do ambiente em dado momento. A forma como esta representação é realizada geralmente define um modelo para abstrair o conhecimento do agente. Um mapeamento total de um ambiente pode ser custoso e exigir uma quantidade absurda de sensores e sistemas computacionais para armazenar e processar tais informações. A utilização de imagens extraídas do ambiente podem trazer informações valiosas aos sistemas de aprendizado conforme revelado em vários trabalhos apresentados no Capítulo 3.

Nesta tese são utilizadas sequências de imagens em escala de cinza como um dos canais de entrada da rede DQN, responsável por mapear os valores de estado-ação. O outro canal é representado pela informação da emoção humana capturada no tempo $t$. A fim de formalizar cada uma destas representações de estados, será adotado aqui o termo $S_{im}$ para o estado representado pelas imagens em escala de cinza e $S_{em}$ para o mapeamento da informação emocional. O conjunto total destas duas informações em um dado tempo é representado por $s_t$, sendo $S$ o conjunto total de todos os estados.

A seguir, serão apresentadas as duas informações que compõe os estados de entrada da SocialDQN, a sequência de oito imagens em escala de cinza e, posteriormente, a informação emocional mapeada através de vetores *one-hot*.

### 6.2.1 Sequência de imagens

A SocialDQN herda alguns parâmetros de captura da arquitetura MDQN, como, por exemplo, a utilização de uma sequência de 8 imagens em escala de cinza de tamanho $198 \times 198$ *pixels*, convertidas a partir de imagens coloridas de tamanho $320 \times 240$ *pixels*. As imagens são capturadas com uma taxa 10 quadros por segundo através da câmera 2D presente na cabeça robô Pepper. Tanto em simulação quanto no robô real, estas configurações são mantidas.

Na rede MDQN é também utilizado um segundo canal de imagens para mapear a profundidade do ambiente. Essa abordagem tem um custo computacional alto, tanto para armazenar as interações do robô na memória de *Repetição de Experiências* quanto para processar e treinar estas interações. Com isso, o tempo de latência no envio das imagens para a rede DQN é o dobro, gerando um atraso na interação em tempo real do robô com humanos. Na arquitetura SocialDQN são utilizados somente o fluxo de imagens em escala de cinza para reduzir a latência na comunicação, um importante fator na área de robótica social. A estratégia de não utilização do canal de profundidade será discutido com mais detalhes no Capítulo 7, quando questões práticas da implementação serão discutidas.

Na Figura 32 são apresentadas 8 imagens sequenciais que representam um estado $S_{im}$ em dado tempo $t$. Através das imagens, é possível visualizar um humano se movendo pelo ambiente. Através desta configuração de imagens, o robô consegue inibir ou excitar determinadas ações

Figura 32 – Exemplo de imagens em escala de cinza que compõe o estado $S_{im}$. A sequência de imagens auxilia para que o robô consiga aprender conforme a direção de movimento humano. O fluxo de imagens segue da esquerda para a direita, de cima para baixo.

Fonte: Elaborada pelo autor.

com base na direção onde o humano se move. Por exemplo, supondo que alguém se aproxima do robô, a rede pode inferir que o humano esteja apto a interagir. Ou até mesmo se ele se afasta, o agente pode inferir que a pessoa está de partida e que o esforço de tentar interagir já é inútil.

Além das imagens, o diferencial da SocialDQN é a capacidade de utilizar sinais sociais pré-processados para a entrada da rede. Nesta tese, a validação desta propriedade é dada pela abstração de informações sobre a emoção extraída através da face humana. Este tópico será tratado a seguir.

### 6.2.2 Vetor One-Hot com informação de emoção extraída da face

O segundo componente de um estado $S$ é a representação processada da emoção detectada em um humano presente no ambiente. Este dado diz respeito à expressão facial do humano mais próximo ao robô, considerando que exista pelo menos um ou mais humanos na cena. Para este mapeamento, utilizamos as seis emoções básicas de Ekman.

As emoções consideradas são “alegria”, “surpresa”, “medo”, “raiva”, “nojo” e “tristeza”. Adicionalmente, é utilizada a classe “neutra” para indicar a falta de uma emoção ou expressão facial. Como apresentado anteriormente, em Russell, Lewicka e Niit (1989) são definidos diversos tipos de emoções, porém, muitas delas podem ser agregadas em uma só classe, utilizando as emoções de Ekman e Friesen (1971) como referência. Baseado nisto e também nas contribuições de Tozadore *et al.* (2018), são agregadas diversas expressões faciais e emoções em classes de

Quadro 2 – Classes de Emoções de Ekman e expressões faciais que as caracterizam

| Emoções de Ekman | Emoções, sentimentos e expressões faciais |
|---|---|
| Alegria | feliz, prazer, sorrindo, etc. |
| Surpresa | surpresa, espantado, chocado, etc. |
| Medo | medo, amedrontado, etc. |
| Raiva | raiva, fúria, nervoso, etc. |
| Nojo | nojo, aversão, etc. |
| Tristeza | triste, desapontamento, etc. |
| Neutra | Sem emoção ou expressão facial. |

emoções de Ekman. Este agrupamento é apresentado através da Quadro 2.

No quadro mencionado, a emoção de "Alegria", por exemplo, agrega expressão facial de "sorriso", feições de "prazer" e expressões de felicidade. Apesar desta última expressão poder ser considerada sinônimo da emoção a qual faz parte, este agregamento é importante para a possibilidade utilizar classificadores que adotam diferentes notações e classes de emoções. Um outro exemplo, a classe "Surpresa", envolve a emoção que a descreve, as expressões de espanto e choque (chocado), dentre outras. Para as demais emoções de Ekman, o mesmo é feito. Ademais, é utilizada a classe "Neutra" para designar a falta de emoção ou expressão facial humana.

A emoção humana, expressada através da face, é detectada pelo robô físico ou simulado e enviada para a rede SocialDQN através do módulo *Ambiente RL* (Figura 31). Esta abordagem permite que o processamento e detecção sejam realizados no próprio sistema de captura da imagem, evitando gargalos e atrasos na transmissão de dados massivos para o módulo de aprendizado. Como as imagens do estado $S_{im}$ tem tamanho reduzido ($320 \times 240$ *pixels*) o reconhecimento da emoção através destas imagens é inviável, pois as informações faciais representadas na imagem geralmente não são suficientes para esta detecção. Essa abordagem permite recuperar informações precisas do simulador, sem a possibilidade da validação do sistema se comprometer com certo grau de imprecisão dos modelos de reconhecimento de emoções disponíveis na literatura. Entretanto, parte dos objetivos desta tese visa a validação no robô real. Para este caso, é necessário realizar o reconhecimento facial externamente, o que será abordado eventualmente durante a apresentação deste documento.

Assim como no simulador, é utilizada uma classificação adicional denominada "sem face" para modelar um estado no qual não é possível identificar o rosto de um humano e, consequentemente, detectar a emoção. O uso desta informação é de grande importância para aprendizado da rede DQN, pois um estado "sem face" pode indicar uma improvável resposta humana a qualquer estímulo ou ação do robô no ambiente. Do contrário, um rosto visível pode indicar que o humano está com o rosto direcionado para o robô (parcial ou diretamente), aumentando as chances do robô estar no campo de visão do humano. Estas informações também dizem respeito à presença de uma pessoa suscetível à interação com o robô no ambiente. Com isso, é assumido que o estado $S_{em}$ agrega sinais sociais de emoção e de rosto humano visível.

Com as classes “neutra”, “sem face” e as seis emoções utilizadas, o estado $S_{em}$ pode representar atualmente oito possíveis valores. Diferente do estado $S_{im}$, que agrega oito imagens em sequência, a SocialDQN utiliza uma única informação para emoção durante aquele estado. Isto porque pode haver uma emoção diferente atrelada em uma ou mais dessas imagens, indicando vários humanos com estados emocionais diferentes ou até mesmo mudança desta condição durante a captura. Nestes casos, a rede SocialDQN considera a emoção que estiver mais presente nas oito imagens para dado estado.

Naturalmente, com a evolução da SocialDQN, serão adicionados mecanismos mais eficientes para se adequar a estas questões. Porém, como mencionado, nesta tese é assumido que a responsabilidade de capturar e processar estes sinais sociais é do robô ou sistema de reconhecimento externo e que a SocialDQN irá receber valores de sinais sociais que sejam consistentes e confiáveis.

Na rede SocialDQN, os sinais sociais são mapeados através de *one-hot vector encoding*, o que consiste mapear as classes em vetores com combinações que não se repetem, preenchidos com valores ‘0’ e um único valor ‘1’. Este mapeamento também pode ser representando através de uma matriz identidade, onde a presença do número ‘1’ em uma posição da linha da matriz indica a classe a que pertence à emoção analisada. Por exemplo, a classe “sem face” é representada por [1,0,0,0,0,0,0,0], “neutra” por [0,1,0,0,0,0,0,0], e assim sucessivamente, para cada uma classes disponíveis. O tamanho do vetor é dado pela quantidade de classes responsáveis por representar os sinais sociais.

Apesar da utilização de oito classes para o estado $S_{em}$, a rede utilizada na SocialDQN permite agregar ou reduzir a quantidade de sinais sociais no sistema. Por exemplo, é possível agrupar as emoções em “positiva”, “negativa”, “neutra” e “sem face”, possibilitando que sistemas menos robustos consigam utilizar a SocialDQN através da abstração de menos informações sobre os sinais sociais e emoções humanas. Por outro lado, é possível aumentar este vetor para que sistemas mais complexos possam adicionar outras informações, como sinais sociais provenientes da fala humana, gestos, quantidade de pessoas no ambiente, etc.

Além da emoção e das imagens, a rede SocialDQN considera as recompensas durante a fase de aprendizado. A seguir são apresentadas as recompensas utilizadas na arquitetura proposta.

## 6.3 Recompensas

As recompensas são valores que o ambiente gera dada a ação executada pelo agente, com intuito de indicar ao sistema o quão bom é dado comportamento em determinado estado. A recompensa pode ser modelada de várias formas. Nesta tese, duas das ações utilizadas possuem valores de recompensa atreladas a elas.

Assim como simulador SimDRLSR, a ação “cumprimentar” gera uma recompensa

positiva quando um humano corresponde ao gesto, caso contrário, uma recompensa negativa é retornada. Na ação "acenar", o objetivo é chamar atenção de algum humano. Desta forma, ao executar esta ação é verificado o foco de atenção do humano e, caso seja o robô, é gerado um valor de recompensa. O caso negativo pode ocorrer caso a pessoa esteja de costas, ocupado, não esteja presente no ambiente ou até mesmo não tenha tido interesse em olhar para o agente. Atualmente, as demais ações não possuem recompensas atreladas a elas, atribuídos valores neutros quando o agente as executa.

As recompensas são fixas durante o treinamento, mas podem ser configuradas antes do agente iniciar sua interação com o ambiente. Elas podem assumir valores negativos, positivos ou até mesmo neutros. Na implementação atual, é utilizado o valor '0' para recompensa positiva da ação "acenar", isto é necessário para evitar que o robô prefira maximizar a recompensa total acenando várias vezes para o humano em sequência, comportamento pouco aceito socialmente.

Tabela 2 – Valores de recompensas adotados na SocialDQN

| **Tipo de Recompensa** | **Valor** |
|---|---|
| Sucesso "Cumprimentar" | 1 |
| Falha "Cumprimentar" | -0.2 |
| Sucesso "Acenar" | 0 |
| Falha "Acenar" | -0.1 |
| Neutra (outras ações) | 0 |
| Falha no episódio | -1 |

Na Tabela 2 as atuais recompensas suportadas pela SocialDQN são listadas com seus respectivos valores. Note que, além das recompensas citadas, são utilizados outros dois valores para outras duas situações. O primeiro mapeia a recompensa na execução das demais ações, *esperar* e *olhar*. O segundo define uma punição para caso o agente termine o episódio após $n$ interações (estado terminal) sem conseguir executar a ação *cumprimentar* com sucesso. A adoção desta punição é forçar o agente a buscar interagir com humano, ou seja, fazer com que ele tenha uma "motivação" para chamar a atenção das pessoas.

Da mesma forma que as emoções, os eventos que geram as recompensa (toque na mão do robô, direção do olhar da pessoa, etc.) são capturados pelo robô ou sistema de processamento externo, não sendo responsabilidade da SocialDQN processar estes eventos. Vale mencionar que em Qureshi *et al.* (2016), que envole a primeira versão da MDQN, é somente trabalhado com a recompensa para modelar *falha* e *sucesso* da ação "cumprimentar". Já em Qureshi *et al.* (2018), uma rede modela essa recompensa, porém, os autores não disponibilizaram acesso aberto ao código e nem a especificidades da implementação desta rede, impossibilitando a consulta e desenvolvimento de tal recurso. Desta forma, as recompensas adicionais que definimos neste trabalho, representam uma contribuição adicional em relação às pesquisas anteriores na área de aprendizado por reforço e robótica social.

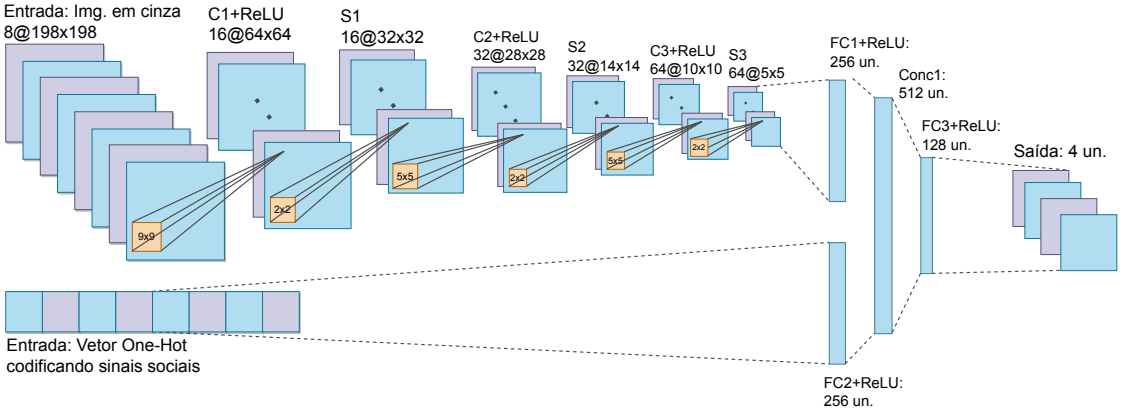
Ao longo deste capítulo, foram apresentados os estados, as recompensas e as ações

utilizadas pela SocialDQN. A seguir, será apresentado o módulo *Agente*, responsável por lidar com todas essas questões, visando gerar comportamentos que maximizem as recompensas.

## 6.4 Agente e Social Deep Q-Network

Em RL, a instância *Agente* é responsável por receber informações do ambiente em forma de estados e selecionar uma ação a partir disto. Em geral, a ação é responsável por fazer com que o agente transite entre estados, sendo a recompensa responsável por auxiliar este a escolher a ação que o leva a um estado desejável. Na rede SocialDQN, o agente é denominado *SocialNQLearner* (Figura 31).

Figura 33 – Social Deep Q-Network




Fonte: Elaborada pelo autor.

Na *SocialNQLearner*, a rede DQN é modelada. Esta rede possui duas entradas de dados, uma para cada um dos tipos de estados utilizados, $S_{im}$ e $S_{em}$. A DQN é uma função de aproximação ação-valor baseado em DL, tendo como entrada um estado e saída um vetor *K-dimensional*, onde o *k-enésimo* elemento corresponde à *k-enésima* ação. Desta forma, a DQN é treinada para que cada valor de saída seja ajustado dado o retorno esperado.

Na Figura 33 é apresentada a rede DQN proposta neste trabalho. A primeira entrada da rede precede uma série de camadas de convolução e filtros responsáveis por aprender os padrões visuais simples e complexos provenientes dos estados $S_{im}$. Os hiper-parâmetros utilizados para estas camadas são baseados na MDQN. Esta entrada recebe 8 imagens de tamanho $198 \times 198$, passam para a primeira convolucional, onde são aplicados 16 filtros de tamanho $9 \times 9$. Logo após, uma função *Rectifier Linear Unit Function* (C1+ReLU) é utilizada para gerar 16 mapas de tamanho $64 \times 64$. Na primeira camada de subamostragem (S1), é efetuada uma operação de *max-pooling* $2 \times 2$ a partir da saída da camada anterior. O processo de *max-pooling* consiste em discretizar uma amostragem, com objetivo de reduzir a dimensionalidade dos dados, permitindo que sejam realizadas suposições sobre uma sub-região dos dados. As camadas C2+ReLU,

S2, C3+ReLU e S3 seguem a mesma lógica das camadas anteriores, mas agora com filtros de tamanho 32 e 64, com *max-pooling* $5 \times 5$, sendo a saída conectada a uma camada linear totalmente conectada com 256 neurônios ReLU (FC1+ReLU).

Já a segunda camada, recebe um vetor *one-hot*, responsável por mapear a informação do sinal social detectado no ambiente. Esta entrada é direcionada a uma camada totalmente conetada (FC2+ReLU) com 256 neurônios ReLU. Esta camada e FC1+ReLU são concatenadas em uma de tamanho 512, responsável por agregar os dois fluxos de informações provenientes do estado do ambiente. Uma terceira e última camada (FC3+ReLU) é totalmente conectada a saída com 4 neurônios, cada uma representando uma possível ação da SocialDQN.

É utilizado o algoritmo $\varepsilon$-*greedy* para a política de seleção de ações. Este algoritmo tem como objetivo regular a execução de ações aleatórias versus através o aprendizado da DQN. Esta taxa pode varia entre 1 e 0, decaindo lentamente com o treinamento. Desta forma, no começo do treino, o agente tende a executar ações aleatórias, com objetivo de diversificar as experiências em relação à ação-estado. No final do treinamento, o objetivo é utilizar um valor $\varepsilon$-greedy menor, para que o agente aplique o conhecimento obtido, enquanto experimenta uma pequena quantidade de ações aleatórias a fim de explorar o ambiente em busca de melhores recompensas.

A fase de aprendizagem visa generalizar os valores de ação do estado, utilizando a recompensa do ambiente para ajustar o peso da rede. A rede local classifica os estados $s$ no tempo $t$ ($s_t$), já a rede alvo analisa os estados $s$ no tempo $t+1$ ($_{t+1}$) . Como apresentado em Seção 2.3, a rede local é treinada para realizar uma determinada ação $a'$ que maximiza os valores para o próximo estado na rede alvo. A ideia por trás disso é ajustar os pesos da rede conforme a equação Equação 2.18 apresentada anteriormente para uma rede alvo.

Os valores da rede alvo são utilizados como meta de valores a serem alcançados durante o treinamento. Esta abordagem visa fazer com que a rede local persiga os valores da rede alvo por alguns passos, tentando minimizar a perda entre as duas redes. Após algumas iterações, a rede alvo recebe os parâmetros da rede principal, que se mantém imutáveis novamente, dando continuidade ao processo até que o agente termine seu treinamento. Essa estratégia de utilizar uma rede alvo é usada no DQN para orientar e estabilizar o treinamento de agentes (KIM *et al.*, 2019).

Os valores de $Q$ para os estados $s_t$ são atualizados, segundo o retorno esperado, recompensas obtidas e um fator de desconto. Depois disso, o sistema calcula o valor da função de perda *loss* com base na diferença entre os valores da rede alvo e os da rede local. A perda é otimizada (Equação 2.18), e, após as interações de treinamento $n$, o sistema atualiza a rede alvo para servir de referência para treinamentos futuros.

## 6.5 Comunicação com o Ambiente

Os processos de treinamento, validação e teste consistem em aplicar a SocialDQN em um ambiente para captura de informações, atuação e coleta de recompensas. Isto é possível através do simulador SimDRLSR ou do robô físico Pepper. Contudo, nesta tese, a utilização do robô real se dá apenas na fase de testes. O treinamento e a validação foram realizados exclusivamente no simulador SimDRLSR. No Capítulo 7 serão apresentados experimentos envolvendo a SocialDQN em ambos os ambientes, mas antes disto, é interessante apresentar uma visão geral da comunicação com os ambientes simulado e físico e seus respectivos agentes robóticos.

Na MDQN, as imagens eram salvas em disco ao invés de serem enviadas via *socket*. A justificativa para isto é dada pelo fato do treinamento da MDQN era realizado ao final de um dia de interação do robô com o ambiente real, necessitando que tais informações fossem gravadas em disco ou algum tipo de armazenamento a longo prazo. Inicialmente, tal abordagem foi mantida nesta tese, contudo, por questões de velocidade de escrita e leitura em disco, bem como sincronização de acesso ao arquivo, tal solução precisou ser abandonada. A partir disto, todas as comunicações com o ambiente aconteceram via *socket* TCP/IP. Na SocialDQN, o treinamento é realizado logo após algumas interações, o que permite armazenar as experiências em memória volátil, evitando os problemas citados.

Independente do ambiente, a comunicação é realizada em uma rede local. No caso do simulador, esta comunicação pode ser realizada no mesmo nó de processamento da arquitetura SocialDQN. Já no ambiente físico, o robô Pepper possui um sistema com limitações de hardware e software que não permitem a execução da arquitetura em questão em seu processador local, sendo assim, ambos precisam se comunicar através de nós de processamento diferentes. A utilização do protocolo de *socket* mencionado permite que ambas as configurações sejam possíveis.

Para cada ambiente, há um módulo de comunicação responsável por tratar suas particularidades, mas há também pontos em comum entre sistemas. Ambos os módulos são responsáveis por configurar os valores de recompensa a serem recebidos, pré-processar os estados, enviar as ações geradas pela rede principal, pré-processar recompensas, etc. No geral, este módulo gerencia a comunicação através da padronização de mensagens recebidas via *socket*.

Em relação aos estados, em ambos os sistemas as imagens são recebidas pela arquitetura em tons de cinza. Estas imagens são processadas para o formato padrão da rede DQN, sendo estas convertidas para um tensor de tamanho $8 \times 198 \times 198$. Em relação à informação da emoção, esta também sofre um tratamento especial. A arquitetura recebe uma informação de emoção para cada uma das imagens, ou seja, são recebidas oito informações sobre as emoções detectadas no ambiente. Como podem haver discrepância entre estas informações, é considerada a emoção mais presente no conjunto recebido. A partir disto, a emoção é convertida para um vetor *one-hot*. O tensor com as imagens e o vetor com a emoção são inseridos em uma tupla e retornados para a

rede DQN.

Apesar da semelhança entre ambientes, cada módulo possui suas particularidades. A seguir, estas caraterísticas únicas de cada módulo são apresentadas.

### 6.5.1 Comunicação com simulador SimDRLSR

No ambiente simulado, a SocialDQN inicia a comunicação enviando parâmetros de configuração para o simulador. São enviados a velocidade de execução da simulação, o tipo de emoção, a posição do robô, a aparência do humano, o tipo de imagem a ser recebida (escala de cinza e/ou profundidade), bem como os valores de recompensas. O tipo de emoção se refere a configuração do simulador em configurar a emoção do avatar humano aleatoriamente ou se será pré-estabelecido um tipo fixo. A aparência do agente diz respeito à seleção de um avatar específico no conjunto de avatares disponíveis em simulação ou até mesmo a seleção aleatória.

O retorno de imagens de profundidade por parte do simulador SimDRLSR é opcional, podendo ser configurada também pela SocialDQN. Como esta arquitetura não faz uso desta informação, é interessante evitar gargalos na comunicação com informações desnecessárias. Contudo, a MDQN utiliza essa imagens, justificando a existência de tal flexibilidade. Isto permite tanto contribuir com a MDQN quanto para a fase de validação da SocialDQN, onde ambas as arquiteturas serão comparadas. Assim como as imagens em escala de cinza, a sequência de profundidades são convertidas em tensores, retornados para a rede DQN.

O módulo de comunicação possibilita a reinicialização do simulador, fazendo com que o ambiente virtual volte ao seu estado inicial, o mesmo acontece com o avatar e os objetos presentes na cena. Desta forma, todas as variáveis são reiniciadas e prontas para uma nova execução independente da anterior.

Por fim, é possível gerenciar o simulador ao nível de processo. Através de bibliotecas do sistema, é factível que a SocialDQN crie instâncias a partir de executáveis compilados do simulador SimDRLSR. Isto facilita o processo de gerenciamento de vários simuladores executando em paralelo, dado que é possível encerrar ou reiniciar a execução destas instâncias assim que a arquitetura SocialDQN considerar pertinente. Em casos de falha de comunicação ou até mesmo erros do sistema, a SocialDQN consegue identificar tais problemas, e, desta forma, finalizar e iniciar uma nova instância do simulador, evitando o consumo de recursos do sistema por processos irrecuperáveis.

### 6.5.2 Comunicação com robô real

O submódulo de comunicação com o robô Pepper é responsável por tratar especificidades deste sistema. Além desse módulo, é interessante também apresentar algumas informações sobre o sistema executado localmente no robô Pepper.

O *script* de execução do robô Pepper foi desenvolvido usando como base os arquivos disponibilizados no repositório oficial da MDQN [1]. Estes arquivos tem como objetivo definir os comandos que compõe as quatro ações da MDQN, bem como capturar imagens em escala de cinza e de profundidade do ambiente. Além disto, é compartilhado um código em C++ para a detecção do *aperto de mão* durante a execução da ação "Cumprimentar". Em Qureshi *et al.* (2016) o robô Pepper foi dotado de uma espécie de luva com sensores, capazes de detectar o toque na mão do agente através do referido código. Contudo, nesta tese, este tipo de evento é identificado através do sensor disponível no dorso da mão do Pepper.

Da mesma forma que realizado em simulação, o módulo de comunicação da SocialDQN envia ao robô físico algumas informações para configuração específicas do sistema. É possível configurar, por exemplo, a resolução das imagens capturadas. Contudo, resoluções maiores causam uma latência alta na captura e envio dessas informações. Por padrão, o robô Pepper foi configurado com a resolução de câmera 640 *pixels* $\times$ 480 *pixels*. Esta configuração foi adotada visando prover uma imagem com qualidade satisfatória para a detecção de faces e reconhecimento de emoções.

Inicialmente, pretendia-se utilizar os mecanismos disponibilizados pelo Pepper para detecção de emoções através da face do usuário. Contudo, por meio de testes empíricos, verificou-se que este módulo é pouco robusto, reconhecendo somente emoções de pessoas engajadas na interação e que estevam cerca de 1 metro do robô. O engajamento dependia de que a pessoa ficasse alguns segundos na frente do agente para que ele efetuasse a detecção. No contexto da SocialDQN, estas condições para reconhecimento da emoção se tornam inviáveis, dada toda a dinâmica de interação e objetivo da arquitetura. Desta forma, como já tratado neste documento, foi utilizado um modelo treinado para reconhecimento de emoções.

O módulo de reconhecimento de emoções através da face é executado a partir do módulo de comunicação da SocialDQN. Assim que esta arquitetura recebe a sequência de oito imagens, cada uma destas passa pelo processo de reconhecimento do modelo FER (apresentado na Seção 4.4). Este módulo retorna um vetor com a "confiança" para cada uma das seis emoções de Ekman mais a emoção neutra. Desta forma, é realizada uma média dos oito vetores gerados com os valores de confiança. A partir disto, a emoção com maior valor é recuperada. Contudo, se essa confiança for muito baixa (menos de 50%, por exemplo), a SocialDQN considera não haver um rosto da imagem, configurando a emoção como "Sem Face". Da mesma forma explicado anteriormente, esta informação é convertida em um vetor *one-hot* e retornada para a rede com os tensores de imagem.

É importante ressaltar que o robô físico foi utilizado nesta tese somente na fase de experimentos, dispensando a implementação de alguns recursos necessários para treinamento do sistema, como, por exemplo, a detecção do *foco de atenção* humano e a captura de imagens de profundidade, utilizada somente pela MDQN. A recompensa gerada pelo toque no dorso do

[1] <https://github.com/ahq1993/Multimodal-Deep-Q-Network-for-Social-Human-Robot-Interaction>

robô é utilizado somente para que o robô aperte a mão do humano. Desta forma, a recompensa retornada não é utilizada.

## 6.6 Configuração de treinamento

O processo desenvolvimento e evolução da SocialDQN envolve também o treinamento de sua rede, bem como a escolha de paradigmas de treino e validação. A partir de diversos treinamentos, testes e validações no simulador SimDRLSR, foi possível definir uma série de parâmetros de treinamento. No Capítulo 7 serão apresentados experimentos envolvendo as configurações de treinamento que serão apresentadas, contudo, é possível adiantar alguns resultados e conclusões para justificar algumas das decisões tomadas.

Após o desenvolvimento da primeira versão do simulador SimDRLSR, foi realizada uma validação deste sistema através da rede MDQN. Nesta etapa, foram mantidos os parâmetros utilizados pelos em Qureshi *et al.* (2016), com a rede sendo treinada em 14 episódios, com aproximadamente 2000 interações em cada episódio. Contudo, como será apresentado no próximo capítulo, o treinamento é instável, não sendo possível atingir um estado de convergência na recompensa cumulativa durante o aprendizado.

Baseado nesse resultado, foi decidido aumentar consideravelmente o número de episódios de treinamento, passando de 14 episódios para 15.000 e reduzindo de 2.000 interações para no máximo 25 em cada um dos episódios. Em ambos os casos, as redes são treinadas após cada episódio, mas com a adoção desta nova configuração, a rede é treinada por milhares de vezes. Como o treinamento é realizado após cada episódio, a interação com humanos não é prejudicada pela necessidade do agente treinar seu modelo. Esta abordagem foi adotada tanto nos testes posteriores com a MDQN quanto na concepção da SocialDQN.

Para seleção de ações, é adotada a política $\varepsilon$-greedy, que começa selecionando ações aleatórias com probabilidade igual a 1 (100%), decaindo com o passar dos episódios até atingir o valor 0,05 (5%). Este valor é alcançado por volta do episódio n.º 3.000.

Nesta tese, as emoções foram agrupadas nas classes “Positiva”, “Negativa” e “Neutra”. A primeira classe agrega as emoções “alegria” e “surpresa”. Já a segunda, agrupa as emoções negativas, seguindo os critérios apresentados anteriormente de acordo com Tozadore *et al.* (2018). Já a terceira classe, representa a falta de emoção no rosto do humano. Além disso, ainda é mantida a classe “Sem Rosto” para indicar quando um rosto não é visível ou não há um humano na visão do robô. Desta forma, o vetor de entrada para o estado $S_{em}$ é de tamanho 4.

Na Tabela 3 são compilados os valores dos principais parâmetros utilizados para treinamento da SocialDQN. O valor das recompensas são os mesmos utilizados na Tabela 2, apresentados anteriormente. O parâmetro *taxa de aprendizagem*, que define o grau de ajuste dos pesos da rede, foi definido para $25^{-5}$. Além disso, é utilizado o tamanho 50 mil para *buffer* de

Tabela 3 – Parâmetros usados na SocialDQN.

| Parâmetros | Valores |
|---|---|
| Total de Episódios | 15.000 |
| Max. Interações | 25 |
| N.º Agentes (simuladores) | 10 |
| Ações | 4 |
| Imagens | 8 |
| Tamanho das Imagens | 198×198 |
| Sinais Sociais | 4 |
| Tamanho do *buffer* de repetição | 50.000 |
| Tamanho do lote de experiências | 64 |
| Taxa de aprendizagem | 25e-5 |
| Velocidade da Simulação | 1 |

repetição e 64 para os lotes de experiências recuperadas durante o treinamento.

O simulador foi configurado com um agente humano e um robô por simulação. A cada episódio, o simulador é reiniciado, com o humano reposicionado em um local aleatório e a recompensa cumulativa zerada. Além disso, a emoção do humano é sorteada, podendo assumir uma emoção positiva, negativa ou até mesmo neutra. Esta característica é imutável durante todo o episódio, variada novamente no próximo episódio.

Para agilizar o treinamento da rede, foram definidas 10 instâncias do SimDRLSR para coleta de informações no ambiente. Estas instâncias compartilham entre si a mesma rede, *buffer* de repetição e parâmetros, ou seja, há apenas uma representação da SocialDQN, que gerencia todas as execuções do simulador.

No Capítulo 7 serão apresentados uma série de testes e validações envolvendo a SocialDQN, simulador SimDRLSR e robô real. Desta forma, alguns dos parâmetros de treinamento podem sofrer alterações para cada caso. Contudo, estas especificidades serão relatadas, mas no geral, os experimentos seguem os parâmetros apresentados.

## 6.7 Características adicionais

A SocialDQN foi desenvolvida na linguagem de programação Python 3.8, utilizando PyTorch para modelar e treinar a DQN, e várias outras bibliotecas para auxiliar no desenvolvimento da arquitetura proposta. O código está disponível no repositório GitHub através do link <github.com/JPedroRBelo/SocialDQN>, sob a licença GNU GPL v3.0, que permite a comunidade acesse ao código, podendo modificar, publicar e compartilhar o código livremente, o que possibilita replicar os resultados deste trabalho.

Além do código da SocialDQN, também foi disponibilizado o algoritmo MDQN em Python, denominado PyMDQN <https://github.com/JPedroRBelo/pyMDQN/>. Este código foi utilizado para a validação dos primeiros protótipos do simulador SimDRLSR. Entretanto,

atualmente é possível configurar a SocialDQN com os parâmetros da MDQN, dispensando a utilização da PyMDQN na execução de experimentos.

Tanto a SocialDQN quanto o simulador SimDRLSR foram desenvolvidos, validados e testados utilizando um computador de mesa com processador Intel®Core™ i7-10700K CPU @ 3.80GHz × 16, com memória RAM de 62,7 GiB, placa de vídeo NVIDIA GeForce RTX 3090 Turbo, e sistema operacional Ubuntu 20.04.4 LTS.

## 6.8 Considerações finais

Neste capítulo foi apresentada a arquitetura de aprendizado por reforço profundo e robótica social, SocialDQN, que agrega a utilização de sinais sociais provenientes dos humanos para interação social. Nesta tese, os sinais sociais adotados envolvem a emoção humana, para compor os estados do ambiente, e também o foco de atenção humano, para auxiliar na modelagem da recompensa.

Foi apresentada a arquitetura geral da SocialDQN, bem como as ações, estados, recompensas e rede de treinamento da mesma. Além disso, os ambientes de atuação foram brevemente discutidos, que envolvem o simulador SimDRLSR e o robô Pepper. Este último é utilizado nesta tese para a validação do sistema no ambiente real.

O objetivo geral da SocialDQN é oferecer comportamentos socialmente aceitáveis a partir da utilização de sinais sociais. Desta forma, a seguir, serão apresentados uma série de experimentos e testes visando validar esta e outras questões, tais como a capacidade de acumular recompensas e análise do comportamento do agente diante diferentes emoções.

CAPÍTULO

7

# EXPERIMENTOS E VALIDAÇÕES

Neste Capítulo, serão apresentados os resultados de experimentos realizados com um robô social simulado treinado a partir da SocialDQN e da MDQN. Inicialmente, é utilizado o simulador proposto nesta tese, o SimDRLSR, para a validação, comparação e análise das arquiteturas mencionadas, visando verificar o potencial da SocialDQN. Posteriormente, a validação é realizada no robô Pepper interagindo com pessoas em um ambiente real. Todo o treinamento necessário em ambos os casos foi feito usando o simulador proposto.

Desta forma, antes de validar as arquiteturas, é necessário verificar o potencial do simulador SimDRLSR em oferecer um ambiente para treinamento de sistemas de aprendizado por reforço para robótica social. Esta tarefa se deu com auxílio da MDQN, com base nos parâmetros e configurações descritas em Qureshi *et al.* (2016). Os resultados deste teste foram de grande importância para definir paradigmas de treinamento para a SocialDQN, bem como para refinar e evoluir o simulador.

Após a validação inicial mencionada, a SocialDQN foi posta à prova na simulação. Para fins de comparação, a MDQN também foi re-treinada, onde foram analisadas as recompensas cumulativas obtidas por ambas as arquiteturas. Essa métrica é comumente utilizada em RL para verificar a capacidade do agente em maximizar o sucesso de atuação no ambiente. Para esta fase de validação, foi utilizada uma versão mais simplificada da SocialDQN, visando equiparar os valores de recompensa com a MDQN. A partir desta comparação, foi possível observar que o sistema proposto se sobressaiu sobre a outra arquitetura, resultado que permitiu evoluir e avançar com a pesquisa.

Na terceira fase de testes, a SocialDQN foi treinada novamente, utilizando recompensas adicionais e configurações que otimizam a obtenção de recompensa cumulativa. O objetivo desta tarefa foi verificar o comportamento da arquitetura durante o treinamento, observando a execução das ações conforme a emoção detectada no ambiente.

Na quarta fase, foi realizada uma validação após o treinamento da SocialDQN, onde

foram coletadas centenas de interações para análise. Além disso, este aprendizado é posto à prova em relação a uma política de seleção de ações aleatória.

Na quinta fase de validação, a SocialDQN é comparada novamente com a MDQN em simulação, contudo o foco foi verificar a aptidão do sistema do ponto de vista social. Para isto, foi desenvolvida uma ferramenta onde voluntários podem visualizar um cenário, composto por uma sequência de 8 imagens, a emoção detectada e a ação selecionada pelo agente. Desta forma, alguns voluntários participantes deveriam dizer se as ações selecionadas pelo robô eram socialmente aceitas ou não para os cenários apresentados. A partir dos resultados, foi possível verificar que a SocialDQN sobressai à MDQN em relação ao comportamento social.

Por fim, o modelo treinado a partir da SocialDQN no simulador SimDRLSR foi transferido para o robô Pepper para interação com pessoas em ambiente real. Da mesma forma que no experimento anterior, voluntários analisaram as interações do agente e responderam se as ações tomadas por eles são socialmente aceitas ou não;

Os experimentos mencionados serão apresentados detalhadamente a seguir.

## 7.1 Validação do simulador SimDRLSR através da MDQN

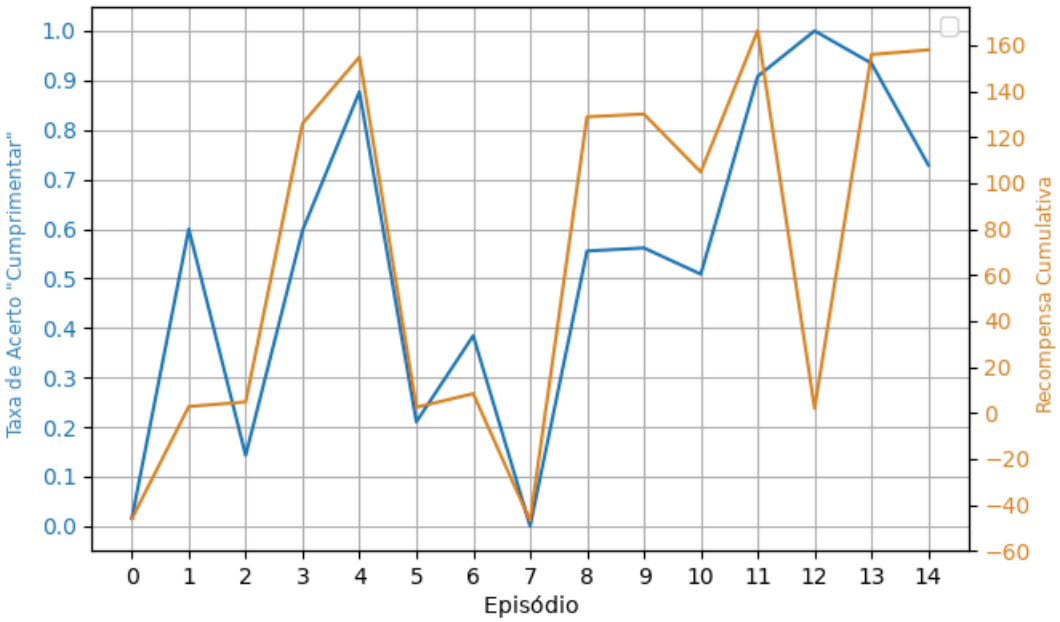
O desenvolvimento do simulador SimDRLSR é uma das contribuições mais relevantes desta tese, uma vez que ele contribui diretamente para a área de robótica social, propiciando o treinamento de robôs que necessitam interagir inúmeras vezes com humanos para que o aprendizado ocorra. Não menos importante, o simulador permite que protótipos e sistemas inteligentes possam ser testados durante o processo de desenvolvimento, sem a necessidade de recorrer a sistemas e voluntários reais para interação com o agente. Desta forma, o simulador ampara tanto o desenvolvimento da SocialDQN quanto à sua validação.

Para verificar e validar o simulador desenvolvido, foi necessário testá-lo antes de prosseguir o desenvolvimento da arquitetura DRL proposta. Para isto, a MDQN de Qureshi *et al.* (2016) foi utilizada. Entretanto, os *frameworks*, bibliotecas e linguagens adotadas pelos autores estão desatualizadas, tais como linguagem Lua e Toch, sendo que muitas delas não são compatíveis com sistemas operacionais e arquiteturas de hardware recentes. Dadas estas limitações, a MDQN teve que ser reescrita na linguagem de programação Python 3.8, com auxílio do *framework* PyTorch para modelagem de redes profundas. Tal adaptação serviu como alicerce para o desenvolvimento da SocialDQN.

Uma das características da MDQN é a divisão entre fase de coleta de dados e fase de treinamento, visando priorizar a interação com humanos sem a interrupção para treinamento do sistema. Esta configuração foi adotada originalmente em Qureshi *et al.* (2016) e reaplicada em Qureshi *et al.* (2018), trabalhos nos quais o robô interagia por volta de 4 horas por dia (etapa de coleta), durante 14 dias (14 episódios). Ao final do dia, o agente era treinado e o aprendizado

era aplicado na coleta de dados do dia seguinte. A cada dia eram coletadas por volta de 2 mil interações com o ambiente, totalizando 28 mil interações ao final do treinamento.

Figura 34 – Recompensa cumulativa e taxa de acerto durante o treinamento da MDQN no simulador SimDRLSR por 14 episódios




Fonte: Elaborada pelo autor.

De tal forma, a MDQN foi configurada com as condições propostas nos trabalhos citados e treinada no simulador SimDRLSR. Os resultados relacionados ao treinamento da arquitetura em questão são apresentados na Figura 34, onde é possível visualizar a acurácia da ação "cumprimentar" e a recompensa cumulativa ao longo dos episódios.

Com base nas duas métricas utilizadas, é possível observar que o agente aprendeu durante o treinamento. Foi possível verificar que o SimDRLSR conseguiu prover um ambiente de treinamento de agentes baseados em RL, em especial em DRL. Além disso, foi possível avaliar a rede MDQN e como o agente se comporta a partir de interações simples com humanos simulados.

Contudo, apesar de o sistema ter aprendido a se comportar no ambiente, ambas as curvas não apresentaram uma estabilidade ao final do treinamento. Possivelmente, este comportamento se dá pela rede ter sido treinada por apenas 14 episódios, quantidade ínfima para garantir certa estabilidade no treinamento. Na literatura, algoritmos de RL são submetidos a milhares (ou milhões) de interações com o ambiente, visando maximizar a exploração de estados e refinar a função de valor (estado-ação).

A fim de suprir esta necessidade e compensar os poucos episódios de treinamento, Qureshi *et al.* (2016) configuram cada episódio para coletar cerca de 2 mil interações. Como

apresentado anteriormente, esta abordagem visa evitar o treinamento do sistema durante a fase de interação com humanos, dando prioridade na coleta de informações do ambiente para treinamento posterior. O aprendizado do agente durante uma interação pode causar uma latência na comunicação socialmente desconfortável ou inaceitável. Entretanto, este problema somente ocorre quando o aprendizado é realizado em ambientes reais. No caso desta tese, as condições de simulação permitem controlar variáveis e parâmetros de treinamento e execução, bem como definir a finalização de episódios de acordo com certas condições.

A partir dos resultados obtidos, foi possível direcionar o refinamento do simulador SimDRLSR e o desenvolvimento da SocialDQN. Na próxima seção, tanto a MDQN quanto a SocialDQN são treinadas em simulação por 15 mil episódios, visando demonstrar o efeito de aumentar o número de episódios de treino e realizar uma comparação direta entre as arquiteturas.

## 7.2 Comparação entre SocialDQN e MDQN

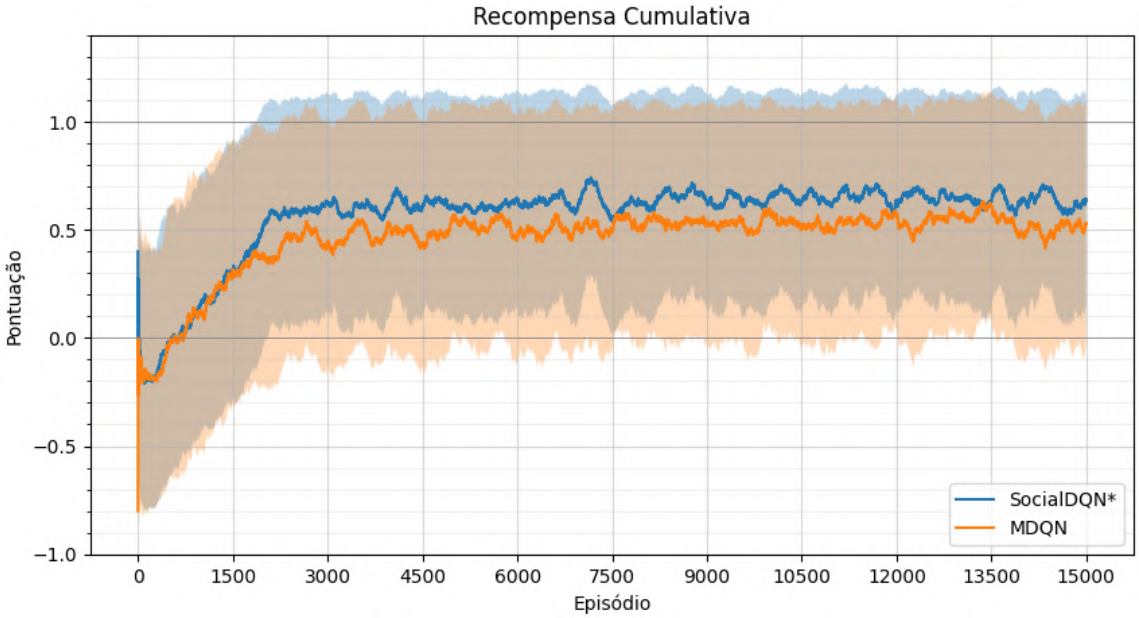
Nesta fase de testes, a SocialDQN foi configurada para refletir os parâmetros e características do modelo MDQN (QURESHI *et al.*, 2016) visando uma comparação direta com a arquitetura proposta.

A métrica de comparação adotada foi a recompensa cumulativa, utilizada na maioria dos trabalhos em RL para medir o sucesso do aprendizado destes sistemas. Para isto, é razoável esperar que os valores de recompensa para ambas arquiteturas, SocialDQN e MDQN, sejam compatíveis. Desta forma, foram utilizados valores neutros para a execução das ações “esperar”, “olhar” e “acenar”. Para a ação “cumprimentar”, o agente recebe o valor ‘-0.1’ em caso de fracasso e uma recompensa igual a ‘1’, caso contrário.

A MDQN se difere da SocialDQN, basicamente em relação às informações do estado do ambiente utilizadas. Enquanto a primeira utiliza duas sequências de 8 imagens, uma em escala de tons de cinza e outra de profundidade, a segunda utiliza somente uma sequência de 8 imagens e um vetor contendo sinais sociais (emoções) detectados no ambiente. A partir dos parâmetros de treinamento descritos anteriormente, as arquiteturas MDQN e SocialDQN foram treinadas utilizando o simulador SimDRLSR. Vale a pena ressaltar que os avatares humanos foram configurados na simulação para expressarem emoções através da face, onde cada grupo de emoção (positiva, negativa e neutra) influência no comportamento dos mesmos.

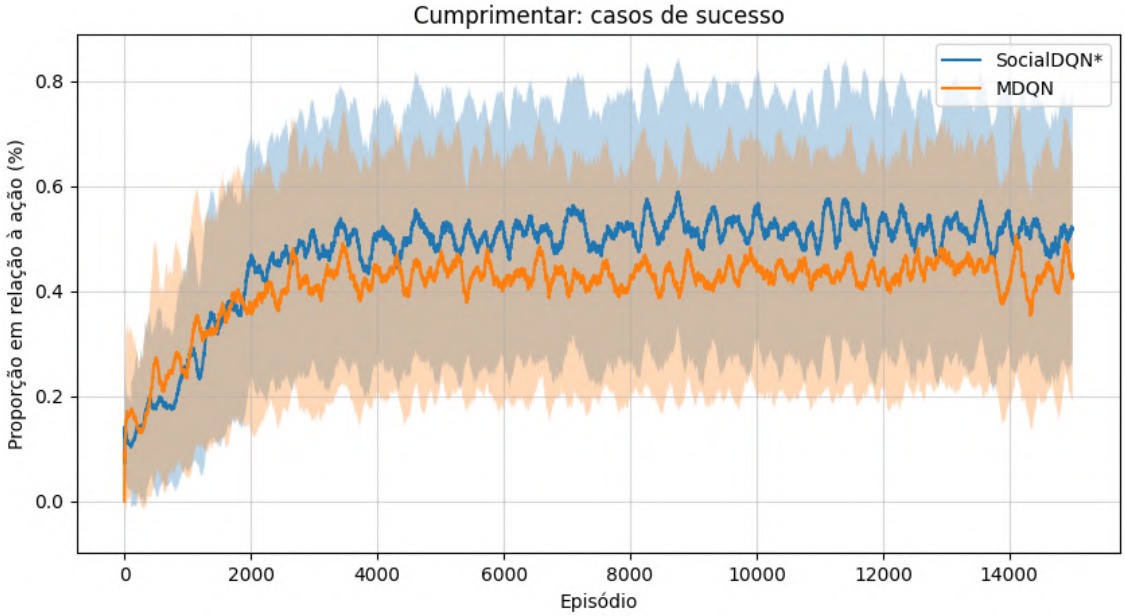
Durante os treinos, foi possível observar o comportamento das arquiteturas através das médias das recompensas cumulativas obtidas, representadas pelos gráficos ilustrados na Figura 35. No gráfico, fica evidente que a versão simplificada da SocialDQN se sobressai à MDQN ao decorrer de 15 mil episódios de treinamento. Durante este processo, o agente pode ser punido (recompensa negativa) ou recompensado (recompensa positiva) conforme a falha ou sucesso, respectivamente, na execução do aperto de mão com um humano. Analisando a média de acertos nesta ação, é possível confirmar que o agente SocialDQN se sobressai sobre

Figura 35 – Recompensa cumulativa entre SocialDQN e MDQN durante 15 mil episódios de treinamento. *A versão utilizada da SocialDQN foi limitada para ser possível fazer uma comparação direta com a MDQN.




Fonte: Elaborada pelo autor.

Figura 36 – Proporção na execução com sucesso da ação “Cumprimentar”




Fonte: Elaborada pelo autor.

a MDQN no que se refere à interação de cumprimentar. Este fato pode ser visto na Figura 36, onde é ilustrada a média móvel de sucesso na execução da ação em questão.

Em ambas as arquiteturas, foi observado que o agente executou a ação “acenar” demasia-

damente, mesmo quando não havia pessoas na cena. Na simulação, esta ação permite chamar a atenção do humano quando ambos estão distantes. Desta forma, é possível que ele tenha aprendido que acenar em qualquer situação (tirando casos onde a pessoa está próxima o bastante para um aperto de mão) é preferível que "esperar" ou "olhar" dado que, na prática, na ação de aceno o agente também olha para o humano. Contudo, o gesto de acenar pode ser considerado não usual do ponto de visual social. Além disso, esta ação exige a operação de motores adicionais, em relação às outras ações, o que pode trazer desgastes dos mesmos e consumo adicional de bateria. Deste modo, a adição de recompensa para esta ação pode trazer benefícios sociais e técnicos para o robô.

Como mencionado, a SocialDQN teve seus parâmetros simplificados para ser possível fazer uma comparação direta com a MDQN. Na Seção 7.5, será apresentada uma análise entre os dois modelos do ponto de vista social. Diferente do teste realizado nesta seção, a comparação a seguir considera a última versão da SocialDQN, com parâmetros otimizados e dotada de mecanismos mais direcionados para detecção de sinais sociais, em especial a identificação do foco de atenção humano, este último utilizado para mapear a recompensa na ação "acenar", como mencionado anteriormente.

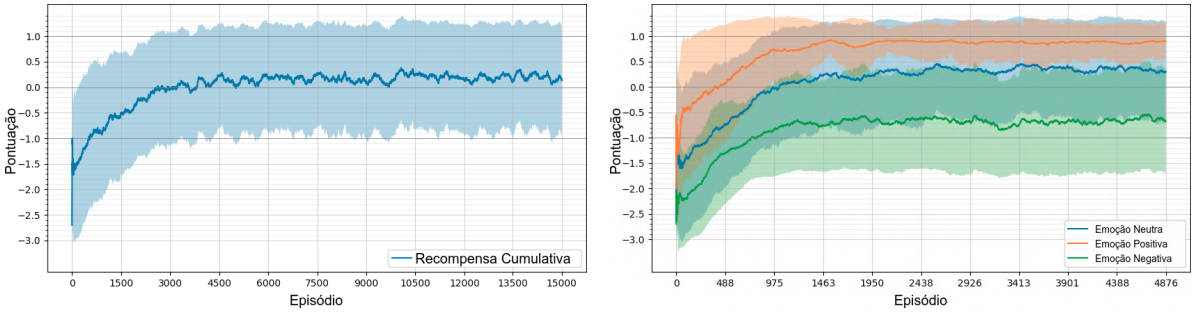
## 7.3 Aprendizado da SocialDQN durante o treinamento

A partir do experimento anterior, foi possível verificar indícios de que a utilização de sinais sociais podem trazer benefícios à interação. No caso anterior, o sinal utilizado foi a emoção humana, mapeada como forma de estado do ambiente. Outro sinal social analisado pela SocialDQN é o foco de atenção do humano, porém este sinal é utilizado para mapear a recompensa do agente ao executar a ação "acenar".

Como apresentado na Seção 6.6, quando o agente acena com o braço direito, é verificado se algum humano está com foco de atenção (direção do olhar) voltado para o robô. Em caso positivo, é gerada uma recompensa neutra, em caso de fracasso o agente recebe uma "punição" de '-0.1'. Adicionalmente, a punição do agente ao falhar em cumprimentar um humano foi configurada para '-0.2'. Esta configuração parte do princípio que falhar em cumprimentar é menos aceitável socialmente que errar em acenar. Demais parâmetros de configuração e treinamento podem ser consultadas na Seção 6.6.

Na Figura 37a pode-se visualizar a curva do aprendizado do agente durante o treinamento da rede ao longo de 15.000 episódios. Neste gráfico, é mostrada a média móvel (em azul escuro) e os desvios-padrão (em azul claro). Note que, a recompensa máxima durante um episódio é de '1', pois no caso ideal, o episódio termina quando o agente executa o *aperto de mão* com sucesso, recebendo o valor máximo de recompensa. Em cada episódio o humano apresenta uma emoção aleatória. Esta emoção afeta na forma como o robô interage com o humano, podendo reduzir a recompensa cumulativa.

Figura 37 – Média móvel da recompensa cumulativa durante o treinamento da SocialDQN




(a) Episódios não agrupados

(b) Episódios agrupados por emoção

Fonte: Elaborada pelo autor.

No segundo gráfico, na Figura 37b, são apresentadas as mesmas recompensas cumulativas do gráfico anterior, contudo, os episódios são separados pelas emoções predominantes em cada episódio. No simulador SimDRLSR, cada tipo de emoção influencia diferentemente o comportamento do humano. Note que o robô tem maior chance de sucesso na interação quando o humano está com uma emoção positiva (linha superior, laranja) do que quando configurado com a emoção negativa (linha inferior, verde). Vale notar também que a curva da recompensa associada a emoção negativa tende a subir com o tempo, indicando que mesmo que o robô não consiga executar a ação "cumprimentar" com sucesso, ele consegue maximizar a recompensa para esta situação, ou seja, ele consegue minimizar eventuais punições. Na emoção neutra (linha intermediária, em azul), o comportamento do agente fica entre as outras duas situações, se aproximando ligeiramente da emoção positiva.

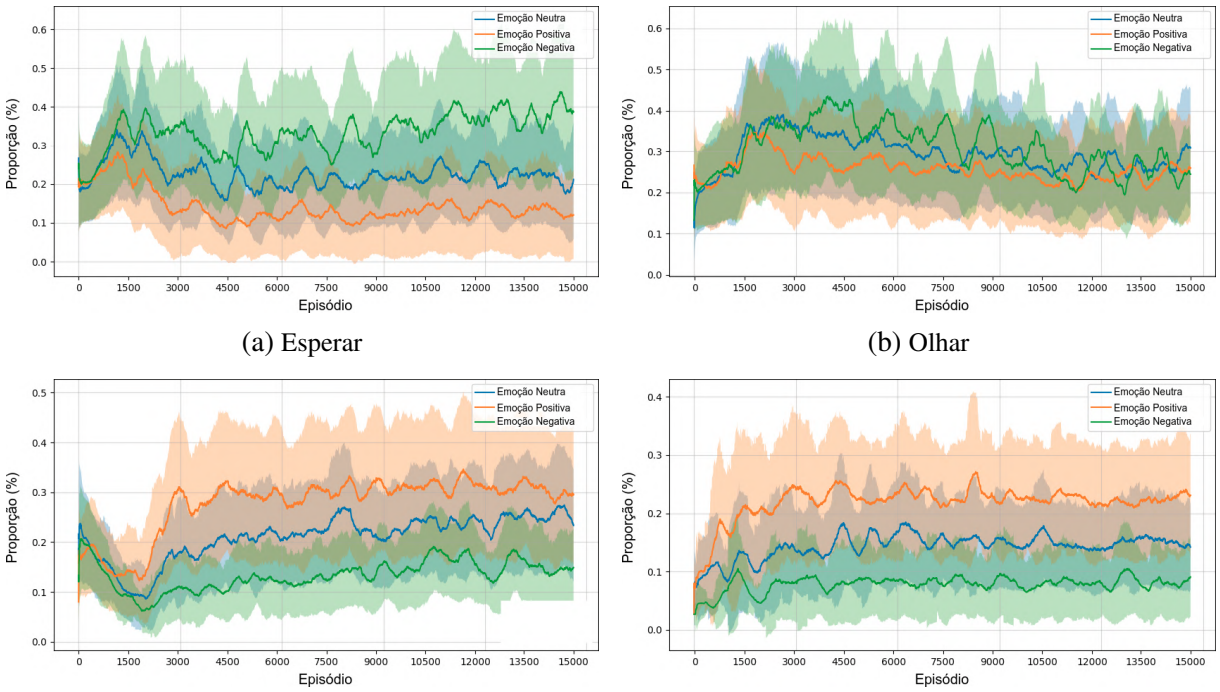
Na Figura 38 são apresentadas as proporções das ações executadas durante o treinamento da SocialDQN, com os episódios agrupados pela emoção predominantes em cada um deles. Para emoções negativas, o agente tende a executar mais as ações "esperar" (Figura 38a) e "olhar" (Figura 38b). Provavelmente, isto acontece porque o agente evita interagir com o humano a fim de reduzir punições. Já para emoções positivas, o agente tende a executar as outras duas ações, "acenar" (Figura 38c) e "cumprimentar" (Figura 38d), pois ele aprende que nesta condição há mais chance do humano responder às ações dele. Assim como na análise anterior, o comportamento do robô se mantém entre as outras duas situações quando a emoção do humano é neutra.

Nesta fase de validação o objetivo foi observar o aprendizado do agente durante o treinamento. A seguir, serão apresentados resultados sob a perspectiva pós-treino.

## 7.4 Validação quantitativa após o treinamento

Para validar o aprendizado do robô e verificar o comportamento do mesmo, foram realizadas várias execuções no ambiente simulado após o treinamento da rede. Para isto, são utilizadas duas políticas para a seleção de ações, a política gananciosa (explotação) e a aleatória

Figura 38 – Média móvel com a proporção da execução de cada uma das ações, agrupadas pela emoção predominante em cada episódio




(a) Esperar (b) Olhar

(c) Acenar (d) Cumprimentar

Fonte: Elaborada pelo autor.

(exploração).

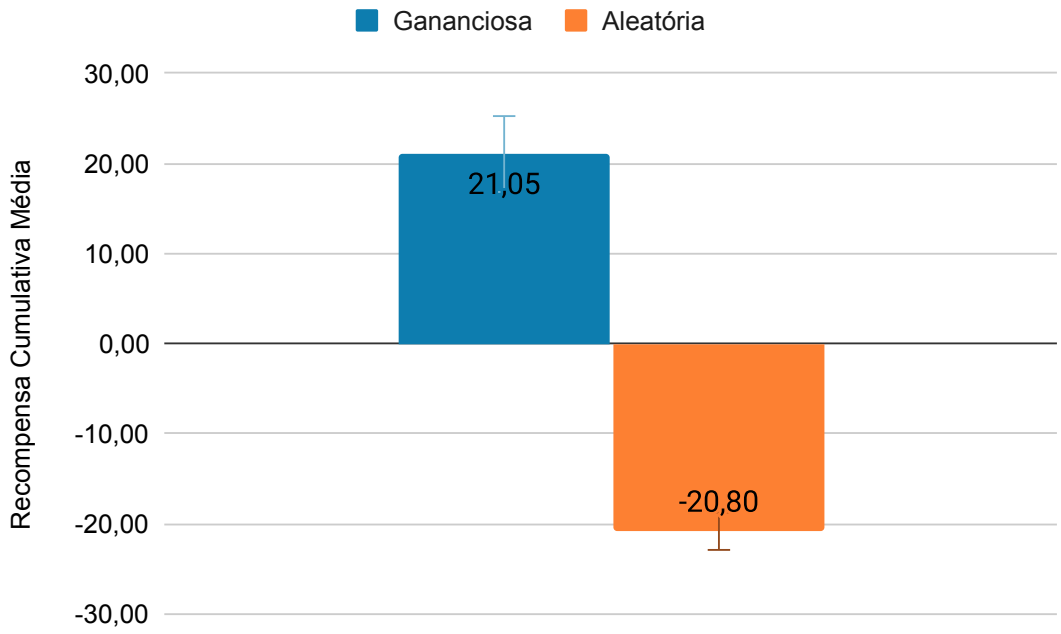
Na política aleatória, o agente desconsidera o aprendizado pelo modelo DQN, selecionando aleatoriamente uma das 4 ações disponíveis independente do estado do ambiente. Já na política gananciosa, a SocialDQN seleciona a ação conforme a rede DQN treinada por 15 mil episódios.

Neste experimento, são efetuadas 5 rodadas de 500 interações para cada uma das políticas a serem analisadas. O simulador é reiniciado quando um estado terminal é atingido, buscando uma aleatoriedade na emoção e na posição do humano simulado. Contudo, é desconsiderada a recompensa negativa no caso de um episódio finalizar sem a execução bem sucedida da ação "cumprimentar". No treinamento, o agente recebia o valor '-1' caso não obtivesse sucesso em cumprimentar, incentivando o mesmo a interagir com pessoas.

Na Figura 39 é apresentada a "Recompensa Cumulativa Média" da SocialDQN através das duas abordagens, gananciosa e aleatória. Analisando estes resultados, fica evidente que o agente aprendeu a interagir no ambiente visando maximizar a recompensa. Na política onde o agente põe o aprendizado à prova (gananciosa), foi possível adquirir uma recompensa total média de $21.05 \pm 4.22$, já a abordagem aleatória uma média de $-20.8 \pm 1.53$.

Note que, a política aleatória finaliza com uma pontuação de recompensa cumulativa

Figura 39 – Recompensa Cumulativa Média da SocialDQN a partir de 5 rodadas de 500 interações, utilizando políticas gananciosa (modelo treinado) e aleatória



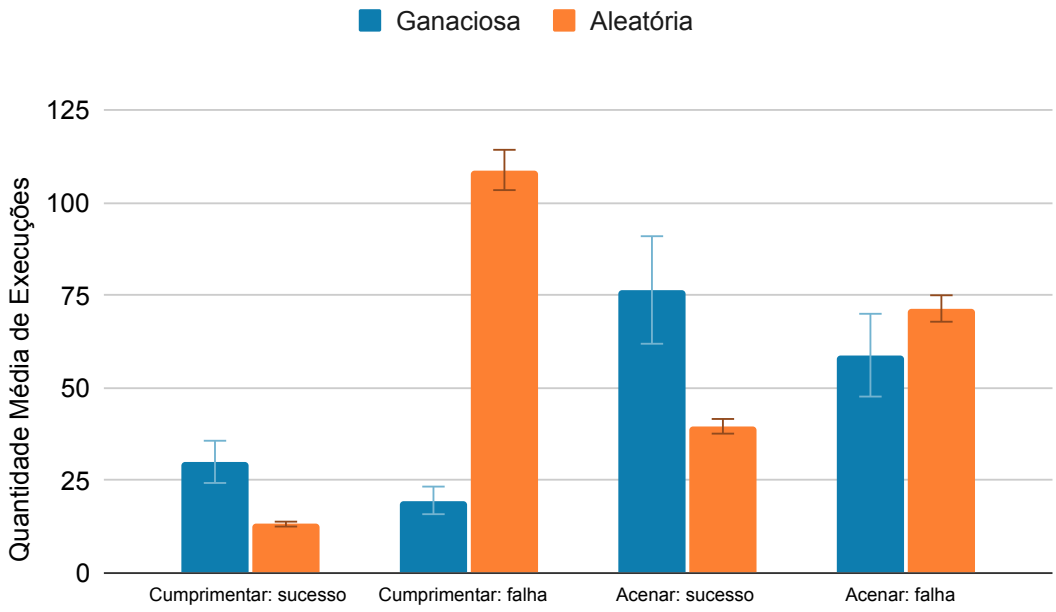

Fonte: Elaborada pelo autor.

negativa. Isto acontece devido a utilização de recompensas negativas, conforme apresentado na Tabela 2. No aprendizado por reforço, o agente aprende, de certa forma, a executar uma sequência de ações que o levem a um estado que maximize sua recompensa. Na política aleatória, tal sequência pode ocorrer por coincidência, mas com baixa probabilidade, dada as possíveis combinações de ações e situações. Desta forma, o agente acaba somando recompensas negativas, totalizando um saldo que pode ficar abaixo de '0'.

Como as ações "Cumprimentar" e "Acenar" possuem recompensas atreladas a elas, é possível analisar o sucesso na execução delas independentemente das demais ações. Na Figura 40 é possível observar que na política gananciosa tem mais sucesso na execução das ações em relação à política aleatória. É interessante também notar que a frequência na execução destas ações, principalmente "cumprimentar" é bastante reduzida na política gananciosa em relação a aleatória. Isto indica que o robô tende a ser cauteloso, executando esta ação somente quando ele avalia ser oportuno.

Nos experimentos apresentados, foram avaliadas as ações do robô em um aspecto quantitativo, com foco na recompensa cumulativa, na execução das ações e no efeito das emoções no comportamento do robô (Seção 7.3). Adicionalmente, é pertinente avaliar como as ações são vistas no aspecto social e como o sistema proposto se sobressai em relação à MDQN neste sentido. Desta forma, a seguir serão apresentados experimentos envolvendo o julgamento das

Figura 40 – Média de quantidade de vezes que as ações “cumprimentar” e “acenar” foram executadas pelo robô com *sucesso* e *falha* durante 5 rodadas de 500 interações




Fonte: Elaborada pelo autor.

ações do agente por parte de voluntários.

## 7.5 Validação social entre SocialDQN e MDQN

Como apresentado na Seção 1.1, o objetivo geral desta Tese envolve contribuir para que robôs se comportem de forma socialmente aceita através de sinais sociais humanos. Para isto, foi necessário avaliar a SocialDQN sob uma perspectiva de aceitação social, visando avaliar tanto esta questão quanto verificar o efeito da utilização de sinais sociais. Para isto, a MDQN é utilizada como *baseline* para verificar estes pontos, dado que a mesma não leva em conta, diretamente, sinais sociais humanos.

Diferente da comparação realizada na Seção 7.2, esta validação se dá após o treinamento de ambas as arquiteturas, não sendo considerado as recompensas cumulativas. Desta forma, é possível utilizar valores de recompensa distintos, priorizando a otimização de ambos os sistemas independentemente.

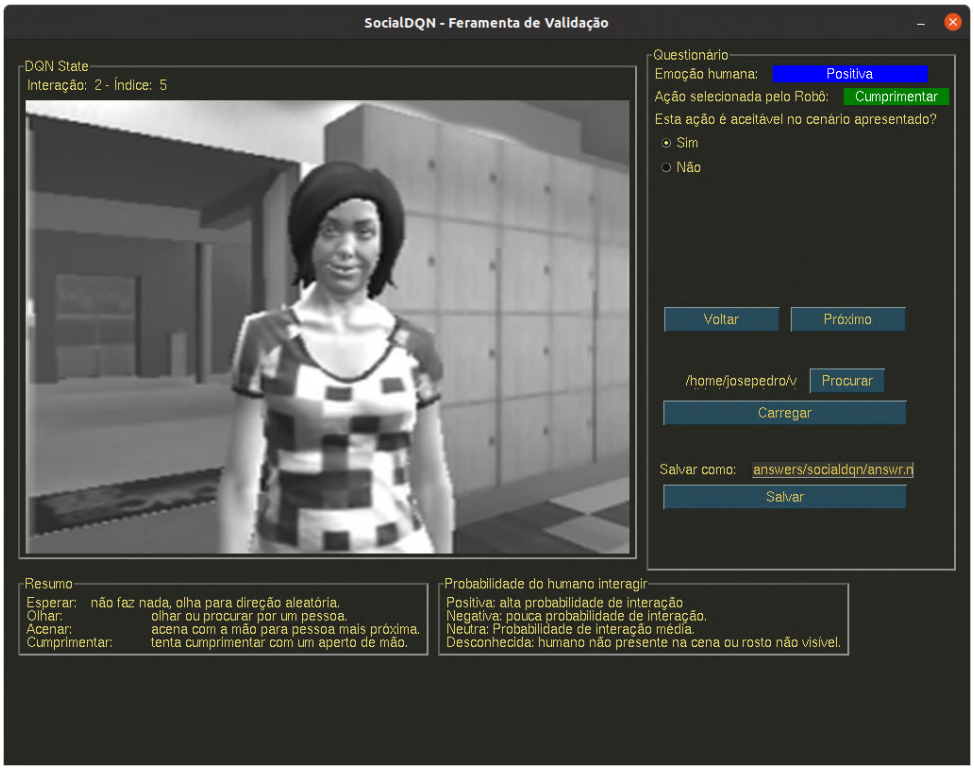
Para esta validação, foi desenvolvida uma interface gráfica para auxiliar na coleta de respostas de voluntários reais. Antes de apresentar a validação e os resultados obtidos nessa fase de testes, a seguir a ferramenta em questão será apresentada.

### 7.5.1 Ferramenta para validação do comportamento social do agente

A ferramenta de validação desenvolvida, tem como foco apresentar interações do robô com o ambiente para que voluntários avaliem as ações do agente. Cada interação é composta por oito imagens em sequência, capturadas do ponto de vista do robô. Além disso, é apresentada a emoção humana detectada e a ação executada pelo robô em determinada interação.

A ferramenta também apresenta um questionamento ao usuário, que deve inserir no sistema respostas baseadas em sua opinião em relação às informações do cenário apresentado. O sistema tem também como objetivo padronizar e salvar as respostas inseridas pelo usuário para posterior avaliação.

Figura 41 – Captura da tela da ferramenta para validação qualitativa.




Fonte: Elaborada pelo autor.

Na Figura 41 é ilustrada uma captura da tela da ferramenta desenvolvida. Nesta imagem, é possível visualizar uma das interações do robô com o ambiente, apresentada ao usuário uma imagem (capturada pelo robô) com uma pessoa, além das informações de que a pessoa está com emoção "positiva" (feliz) e que a ação tomada pelo robô foi de cumprimentar o humano presente. A partir disso, o usuário deve informar se a ação executada pelo robô é "socialmente aceitável" ou não. Em caso negativo, o voluntário deve selecionar qual ação seria mais aceita naquele cenário. A partir disso, o usuário avança para a próxima interação e, ao finalizar o preenchimento, efetua o salvamento de suas respostas. O arquivo gerado nesta operação é utilizado para análise estatística.

A ferramenta foi desenvolvida em Python 3.8, com auxílio de bibliotecas para a implementação de interfaces gráficas. A mesma foi disponibilizada para a comunidade científica em <https://github.com/JPedroRBelo/validation_tool_socialdqn>. No Vídeo, hospedado em <https://youtu.be/GJzXoStAFzI>, é possível visualizar uma demonstração que exemplifica as interações dos voluntários com a ferramenta em questão. Os experimentos, realizados com auxílio dessa aplicação, são apresentados a seguir.

### 7.5.2 Validação social com voluntários analisando SocialDQN e MDQN

Utilizando a ferramenta descrita anteriormente, foram recrutados 12 voluntários para analisar o comportamento social do robô através da SocialDQN e MDQN.

Assim, foram coletadas 100 interações do robô com o ambiente simulado após o treinamento de ambas as arquiteturas. Como mencionado anteriormente, cada interação é composta por uma sequência de 8 imagens em escala de cinza, a emoção detectada no ambiente e a ação executada pelo robô. Apesar da MDQN não utilizar a informação da emoção para tomada de decisão, ela foi capturada de modo a ser utilizada na avaliação.

A ferramenta de validação foi configurada utilizando as interações coletadas. Posteriormente, os voluntários foram instruídos sobre os elementos da interface, sobre as ações do robô e o efeito das emoções sobre o comportamento humano. Após esta breve instrução, cada uma das interações foram apresentadas aos voluntários para analisarem os cenários em questão.

Os dados apresentados na Tabela Tabela 4 são resultantes da análise estatística das tabelas verdade, consequentes das respostas dos voluntários. Como houve escolhas distintas na seleção das ações, foi considerada a resposta (ação) mais "votada" para cada cenário. Além da comparação com a MDQN, também foram analisados os resultados usando um modelo seguindo uma política aleatória.

Tabela 4 – Resultados obtidos através da matrizes de confusão, calculadas a partir das respostas dos voluntários sobre as ações do agente. Foram avaliadas as arquiteturas SocialDQN, MDQN e uma política de seleção de ações aleatória. Nesta tabela: Acc, Prec, Rec, F1 e Cumpr. representam as abreviações das medidas: Acurácia, Precisão, Recall, F1-Score e Cumprimentar, respectivamente. Em negrito são ressaltados os melhores valores médios obtidos entre as 3 modalidades para cada uma das métricas.

| **Classe** | **SocialDQN (%)** | | | | **MDQN (%)** | | | | **Política Aleatória (%)** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | Prec | Rec | F1 | Acc | Prec | Rec | F1 | Acc | Prec | Rec | F1 |
| Esperar | 96,00 | 92,59 | 92,59 | 92,59 | 77,00 | 100,00 | 23,33 | 37,84 | 55,00 | 15,38 | 14,81 | 15,09 |
| Olhar | 89,00 | 100,00 | 65,62 | 79,25 | 76,00 | 100,00 | 14,29 | 25,00 | 56,00 | 29,63 | 24,24 | 26,67 |
| Acenar | 85,00 | 59,26 | 80,00 | 68,09 | 52,00 | 38,16 | 96,67 | 54,72 | 65,00 | 21,74 | 22,73 | 22,22 |
| Cumpr. | 94,00 | 80,00 | 95,24 | 86,96 | 99,00 | 92,31 | 100,00 | 96,00 | 72,00 | 29,17 | 38,89 | 33,33 |
| **Média** | **91,00** | **82,96** | **83,36** | **81,72** | 76,00 | 82,62 | 58,57 | 53,39 | 62,00 | 23,98 | 25,17 | 24,33 |

Na tabela é possível constatar que, segundo os voluntários, a SocialDQN produz ações mais socialmente aceitas que a MDQN e a política aleatória, com 91% de acurácia, 82,96% de precisão, 83,36% de *recall* e 81,72% de *F1-score* nas médias entre as ações. Vale observar que a ação "acenar" teve pontuação inferior às demais, com 85,00% de acurácia e 59,26% de precisão.

Ao analisar os cenários, foi verificado que o robô aprendeu a realizar essa ação mesmo quando uma pessoa se afasta. Os voluntários avaliaram que esta ação dificilmente é aceitável quando não há nenhuma ninguém na cena ou uma pessoa está de costas.

Já em relação à MDQN, esta apresentou resultados mais satisfatórios em razão à ação "Cumprimentar". Isto mostra que a rede se especializou em detectar o momento correto para efetuar tal ação, apesar de não ter ido tão bem nos demais comportamentos. Diferente da MDQN, na SocialDQN o agente considera também as emoções do humano. De fato, no simulador o comportamento do agente é afetado diretamente pela emoção do mesmo, sendo que emoções negativas fazem com que o avatar tenha pouca probabilidade de interação com o robô. Em relação a isto, foi observado que na SocialDQN o robô prioriza as demais ações quando um humano está triste (emoção negativa), evitando executar a ação de aperto de mão (cumprimentar), dado que nestas condições a probabilidade do humano corresponder o gesto é reduzida.

As ações "esperar" e "olhar" tiveram as melhores avaliações na SocialDQN. No geral, é possível observar que os voluntários selecionaram essas ações quando um humano estava de costas ou não estava presente na cena. Na MDQN foi observado que o agente havia aprendido a acenar mesmo quando não havia ninguém no cenário, realizando as ações "esperar" e "olhar" em poucos casos. Este comportamento se torna mais evidente na validação do simulador SimDRLSR, descrito na Seção 7.1, onde a arquitetura MDQN foi submetida a poucos episódios de treinamento (como sugerido em (QURESHI *et al.*, 2016)). A melhora neste comportamento se dá graças a adoção de milhares de episódios para treinamento, ficando evidente ao analisar a estabilidade do aprendizado das redes MDQN e SocialDQN, como apresentado anteriormente.

Por fim, a partir dos resultados obtidos, pode-se concluir que o uso de sinais sociais no processo de aprendizagem é benéfico à interação no que diz respeito à aceitação social dos comportamentos do agente robótico. A próxima etapa de validação tem como intuito verificar se o comportamento socialmente aceitável da SocialDQN se mantém no ambiente real, através da atuação do robô Pepper interagindo com pessoas. Além disso, esta tarefa permite avaliar eventuais limitações na transferência de conhecimento do agente treinado em simulação para o ambiente real, o que permite avaliar a capacidade do simulador SimDRLSR em prover um ambiente para treinamento de agente robóticos sociais.

## 7.6 Validação social em ambiente real

Para a validação com pessoas reais, foi utilizado o robô físico Softbank Pepper, o qual serviu como base para o robô simulado. Este agente foi posicionado no saguão principal da biblioteca Achille Bassi, do ICMC-USP, com o papel de analisar o ambiente e atuar conforme o conhecimento adquirido em simulação. Para estas tarefas, foi necessário implementar e utilizar recursos adicionais.

### 7.6.1 Configurações adicionais pré-validação

Alguns dos recursos mencionados foram abordados na Subseção 6.5.2, como reconhecimento de emoções faciais, arquitetura de comunicação com o robô, protocolo de mensagens, etc. Estes módulos dizem respeito ao nó de processamento da SocialDQN. Além disso, foram utilizadas as implementações de atuação e captura de imagens por parte do robô Pepper, disponibilizadas por Qureshi *et al.* (2016);

Em Qureshi *et al.* (2016), o robô Pepper foi dotado de uma luva para a detecção da ação de cumprimentar. Diferente desta abordagem, no experimento que será descrito a seguir, a detecção desta ação foi dada através do sensor existente no dorso da mão do agente. Entretanto, esta informação é utilizada somente para que o robô faça o movimento de apertar a mão do humano após o toque. Isso porque na fase de validação não há necessidade de retornar uma recompensa para o agente. Desta forma, não houve a necessidade de mapear a detecção do foco de atenção humano a fim de mapear a recompensa da ação "acenar".

Antes de validar a SocialDQN com pessoas reais, além do desenvolvimento das funções mencionadas, foram executados alguns procedimentos, tais como readequação do ambiente do simulador, treinamento do modelo, transferência de conhecimento, testes no robô real e retreinamento com novos parâmetros conforme o necessário.

Recentemente, alguns móveis e estruturas do saguão principal da biblioteca física foram realocados. Desta forma, o ambiente simulado foi reconfigurado, com objetivo de que o robô consiga reproduzir seu aprendizado baseado no comportamento humano, descartando eventuais discrepâncias entre ambientes. Desta forma, o modelo teve que ser retreinado.

A partir do treino do modelo, foram realizados pequenos testes envolvendo o robô real no laboratório LAR. No início dos testes, foi observado que o agente não estava se comportando conforme os testes anteriores. Após configurar a recompensa de "Falha no Episódio" para '0' (originalmente configurado como '-1'), o sistema começou a se portar de forma mais consistente com os resultados em simulação. A tabela de recompensas atualizada para o experimento em questão é apresentada na Tabela 5.

Tabela 5 – Valores de recompensas para a validação em ambiente real

| Recompensa | Valor |
|---|---|
| Sucesso "Cumprimentar" | 1 |
| Falha "Cumprimentar" | -0,2 |
| Sucesso "Acenar" | 0 |
| Falha "Acenar" | -0.1 |
| Neutra | 0 |
| Falha no episódio | 0 |

Nota: A recompensa "Neutra" diz respeito as ações "esperar" e "olhar".

Nota: A falha no episódio ocorre quando o episódio se encerra sem sucesso na ação "cumprimentar".

Na próxima seção, é abordada a atuação do robô Pepper na biblioteca do ICMC, bem como serão apresentadas as configurações e caraterísticas da interação.

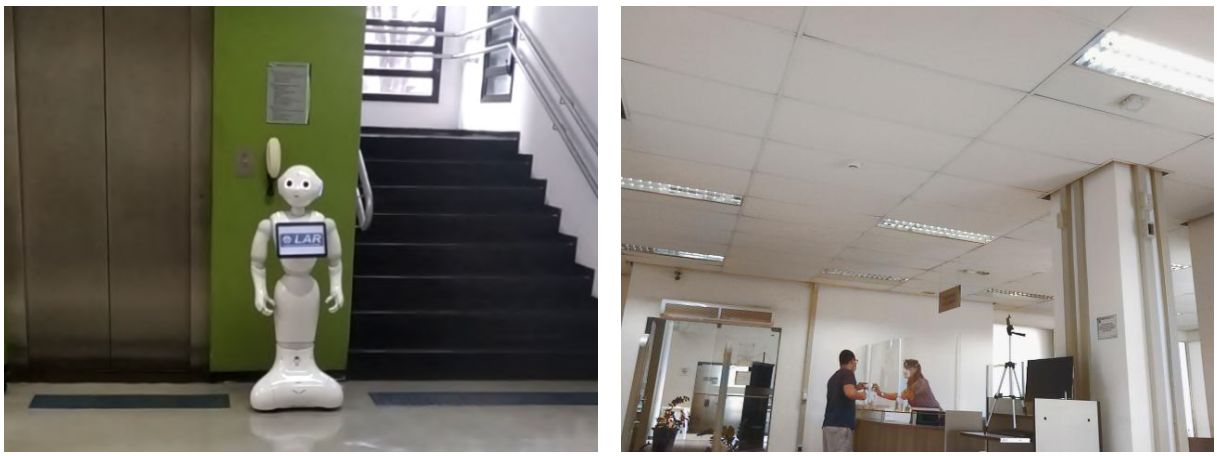
### 7.6.2 Atuação no ambiente real

Como descrito anteriormente, esta tarefa consistiu em posicionar o robô Pepper no saguão principal da biblioteca do ICMC-USP para que ele interagisse com pessoas reais. O robô ficou disposto no local por 4 dias, durante 1 hora por dia. Ao longo das interações, foram armazenadas as imagens, emoções reconhecidas e ações executadas pelo robô, visando a validação destas interações por voluntários, assim como foi realizado com os resultados em simulação.

Para execução da SocialDQN foi utilizado um laptop, que se comunica com o robô Pepper através de um roteador Wi-Fi 5GHz. A utilização de tal frequência permite a transmissão de dados massivos e com uma latência menor do que em frequências mais convencionais.

O agente foi posicionado entre o acesso para o elevador e a escadaria da biblioteca, locais onde pessoas transitam com frequência. Além disso, o local possui uma visão ampla do local, permitindo o agente visualizar a movimentação e aproximação de pessoas, bem como uma boa configuração para captura e análise da face para o reconhecimento de emoções. Na Figura 42, é possível visualizar o ambiente sob dois ângulos, o primeiro representa a visão de quem entra na biblioteca (Figura 42a) e o segundo na perspectiva da visão do robô (Figura 42b).

Figura 42 – Ambiente de atuação



(a) Agente e ambiente (b) Ambiente do ponto de vista do robô

Fonte: Elaborada pelo autor.

A captura das imagens segue o padrão utilizado em simulação, com o agente capturando cerca de 8 imagens por segundo. O envio destes dados para o laptop e o processamento destes dados pode levar até 3 segundos, dado que o modelo de detecção de faces e emoções adotado prioriza a acurácia de acerto em detrimento ao tempo de processamento. A fase de classificação

por parte da SocialDQN é irrisório, levando alguns milissegundos para decidir a ação a ser tomada.

Em relação ao tempo de atuação, existe uma variância entre cada uma das quatro possíveis ações. Os comportamentos de "esperar" e "olhar" possuem uma duração de 3 segundos, enquanto a ação de "acenar" tem um tempo de 4 segundos. A ação "cumprimentar" difere das demais, pois ela depende que algum humano corresponda este gesto. Nesta ação, o robô projeta sua mão indicando que gostaria de cumprimentar o humano. O agente aguarda por cerca de 4 segundos por uma resposta humana e em caso de sucesso ele executa, por mais 2 segundos, o gesto de apertar a mão e logo em seguida retorna os atuadores para a posição inicial. Caso o humano corresponda logo no início do gesto, o tempo de espera pode ser menor. Ao todo, esta ação pode durar cerca de 6 segundos.

Com intuito de permitir que o humano tenha tempo de perceber e atuar após uma ação do robô, a captura de novas imagens do ambiente se dá após 1 segundo da última ação executada. Desta forma, todo o ciclo de captura, processamento e atuação leva em torno de 8 a 11 segundos, dependendo da ação. Entretanto, o foco deste experimento é verificar a aceitação social das ações do robô, dado que o módulo a ser avaliado é a SocialDQN. O tempo de execução das ações, latência de comunicação, processamento e reconhecimento de emoções são restrições das técnicas disponíveis na literatura.

No vídeo em <https://youtu.be/Dp_HjqvFskw> é possível visualizar uma interação com humano, onde o robô observa a pessoa após de executar a ação "olhar" e logo após ele acena para a mesma. O humano se aproxima, o agente repete o gesto e, por fim, executa um aperto de mão. Nesta interação, o robô detectou que o humano estava com uma emoção "positiva". No vídeo em <https://youtube.com/shorts/Nu1yAUdjEMQ> é exibida a interação do robô na perspectiva do humano. Da mesma forma que no primeiro exemplo, nesse vídeo o humano se aproxima gradualmente do robô e finaliza com um aperto de mão. Note que, enquanto o humano não se aproxima o bastante para tentar um aperto de mão, o agente insiste na ação "acenar".

Como mencionado, o agente interage com diversas pessoas durante alguns dias. Com isto, foram capturadas informações destas interações. A seguir, os resultados obtidos através da interação do robô com o ambiente real são apresentados sob uma perspectiva social.

### 7.6.3 Validação social com voluntários

Ao todo, foram coletadas cerca de 2 mil interações com o ambiente, envolvendo pessoas ou não, buscando obter a maior gama de situações possíveis. Como várias destes cenários foram ambíguos, foi realizada uma redução no conjunto de dados, sendo selecionadas 150 interações para que voluntários pudessem avaliar o comportamento do robô no cenário real.

Foi utilizada a ferramenta de validação descrita anteriormente na Subseção 7.5.1, mas desta vez o *script* foi populado com as 150 interações capturadas no ambiente real. Foram

recrutados 12 voluntários para interagir com a ferramenta, os quais responderam se a ação executada pelo robô era socialmente aceita em relação ao cenário apresentado (sequência de 8 imagens e emoção detectada). A partir disto, as respostas foram analisadas, resultados calculados e compilados através da Tabela 6.

Tabela 6 – Pontuações calculadas a partir da matriz de confusão calculada a partir de questionário com voluntários. Nesta tabela: Acc, Prec, Rec, F1 e Cumpr. representam as abreviações das medidas: Acurácia, Precisão, Recall, F1-Score e Cumpr., respectivamente.

| **Classe** | **Pepper com SocialDQN (%)** | | | |
|---|---|---|---|---|
| | Acc | Prec | Rec | F1 |
| Esperar | 99,33 | 87,50 | 100,00 | 93,33 |
| Olhar | 97,33 | 100,00 | 88,24 | 93,75 |
| Acenar | 88,00 | 69,64 | 97,50 | 81,25 |
| Cumpr. | 90,00 | 98,21 | 79,71 | 88,00 |
| **Média** | 93,67 | 88,84 | 91,36 | 89,08 |

Fonte: Elaborada pelo autor.

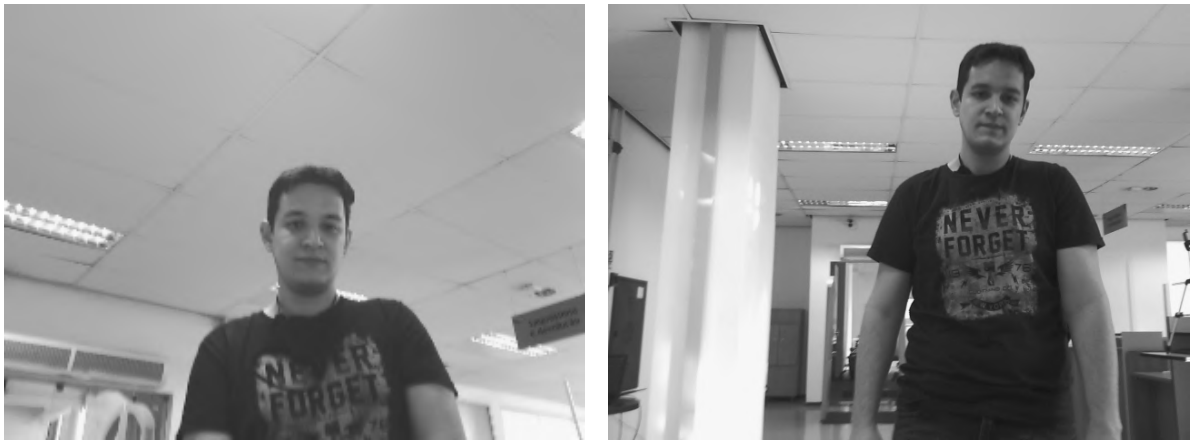
Os resultados indicam que os voluntários consideram que a SocialDQN, no ambiente real, performou as ações de forma socialmente aceita com 93,67% de acurácia, 88,64% de precisão, 91,36% de *recall* e 89,08% de *F1-score*. É interessante notar que estes resultados apresentados superam até mesmo os obtidos pela própria SocialDQN em simulação, ilustrados anteriormente na Tabela 4.

Sobre o comportamento do agente e a percepção dos voluntários sobre isso, é possível levantar alguns pontos. Assim como nos resultados anteriores, foi observado que o robô aprendeu a associar emoções a certos comportamentos. Por exemplo, quando uma pessoa está com a emoção negativa e próxima ao robô, o mesmo evita cumprimentar o humano, prevendo que o mesmo não irá corresponder o gesto.

Em algumas situações, onde pessoas estavam próximas ao robô Na Figura 43, são apresentados duas situações onde pessoas estão próximas ao robô e que alguns avaliadores julgaram que o comportamento do agente deveria ser de "cumprimentar". Contudo, na Figura 43b, a emoção o robô acena para o humano. Isto porque na Figura 43a, o humano apresenta uma emoção de valência positiva, já na Figura 43b a emoção é detectada como negativa. Houveram também voluntários que julgaram que o segundo caso é correto, ou seja, avaliaram que o robô não deveria cumprimentar o humano em caso dele estar triste, com raiva, com medo, ou outra emoção negativa. Quando perguntado ao voluntário do porquê desta resposta (após a aplicação do questionário, para não influenciar nas demais respostas), o mesmo justificou que, se estivesse na posição da pessoa ilustrada nas imagens, ele não cumprimentaria o robô caso ele estivesse triste, com medo ou com raiva.

Os resultados, obtidos nesta fase de validação, se mostram muito satisfatórios e confirmam que a arquitetura SocialDQN consegue prover um comportamento socialmente mais aceito através do uso de sinais sociais, em especial, por meio do uso de emoções extraídas da

Figura 43 – Situações onde humano está próximo e o agente seleciona diferentes ações para emoções distintas.



(a) Emoção detectada: “positiva”.
Ação do agente: “cumprimentar”.

(b) Emoção detectada: “negativa”.
Ação do agente: “acenar”.

Fonte: Elaborada pelo autor.

face humana. Além disto, a validação no ambiente real mostra que o simulador SimDRLSR é robusto o bastante para oferecer um ambiente para treinamento de sistemas de robótica social com paradigma de autoaprendizagem que exigem milhares de interações com humanos, sem exigir um pós-treinamento no ambiente de atuação.

## 7.7 Discussão sobre os experimentos e resultados

Nas seções anteriores, foram apresentados diversos experimentos, cada qual com um objetivo de avaliação. Foram avaliados o simulador SimDRLSR, o treinamento e validação da SocialDQN, bem como a comparação desta com a MDQN, e por último foi apresentada a execução do agente em um ambiente real.

O simulador SimDRLSR foi validado através da MDQN, se mostrando capaz de fornecer um ambiente para fase de desenvolvimento, treinamento e validação de agentes sociais através de aprendizagem por reforço. Este potencial fica ainda mais evidente durante a demonstração da SocialDQN, especialmente na última fase de validação, a qual verificou o comportamento do agente no ambiente real. Apesar desta fase não envolver o simulador diretamente, todo o conhecimento aplicado pelo robô real, através da SocialDQN, foi baseado no conhecimento adquirido no simulador. Esta constatação permite comprovar que o simulador SimDRLSR é uma alternativa viável à utilização de robôs sociais reais durante para treinamento de sistemas inteligentes.

Em relação à MDQN, foram inicialmente respeitados os parâmetros e configurações utilizadas em Qureshi *et al.* (2016), principalmente na abordagem de treinar o agente por 14 episódios, com milhares de passos em cada um. Contudo, isto causa uma instabilidade

no treinamento, dado que há pouca margem para o agente colocar em prática o aprendizado adquirido e corrigir eventuais sobre-ajustes. Segundo os autores da MDQN, a estratégia é evitar que o robô encerre uma interação linear com humanos para efetuar o treinamento, o que é socialmente inaceitável. Além disso, a interação com humanos reais pode ser lenta, exigindo vários dias ou semanas para que o agente aprenda um padrão comportamental consistente. Contudo, quando são utilizados simuladores, estas limitações já não fazem muito sentido.

Desta forma, graças ao desenvolvimento e utilização do simulador SimDRLSR, a concepção da SocialDQN teve como base o paradigma de efetuar o treinamento por milhares de episódios, com cada episódio tendo poucos passos de interação. Os resultados apresentados anteriormente mostraram que tal abordagem traz estabilidade no treinamento de ambas as arquitetura.

Além da sequência de imagens em escala de cinza, a MDQN recorre a 8 imagens de profundidade. A primeiro momento é possível imaginar que estas informações podem ser importantes para que o sistema infira quando uma pessoa está próxima do robô. Contudo, essas conclusões podem ser obtidas através das imagens em escala de cinza. Por exemplo, é intuitivo dizer que uma pessoa está próxima à câmera se sua forma ocupa boa parte da imagem. Além disso, a captura e transferência de imagens entre nós de processamento pode trazer latência na interação. Nos testes epíricos realizados em simulação, notou-se que de fato há um atraso maior no ciclo de atuação, fazendo com que o robô tivesse menos oportunidades de interação com o humano simulado. Contudo, ainda é necessário avaliar estes efeitos, bem como medir a latência de comunicação nesses casos. De toda forma, como mostrado nos experimentos comparativos, a utilização de sinais sociais (SocialDQN) é mais beneficial a interação do que a adoção de imagens de profundidade (MDQN).

Em todos os testes que envolveram a SocialDQN, foi possível observar um grande potencial desta arquitetura, principalmente em relação à MDQN, onde as comparações realizadas mostraram que a arquitetura proposta se sobressai tanto nos testes quantitativos quanto ao nível de comportamento social.

No experimento que foca no treinamento da SocialDQN, foram analisados os comportamento do agente em relação às emoções detectadas no ambiente. Neste, pôde-se verificar que a abordagem de se utilizar sinais sociais, em específico a emoção humana, permite que o agente seja flexível em relação às ações executadas. Apesar do experimento incentivar que o robô execute a ação "cumprimentar" (recompensa máxima), em um ambiente social é possível que um humano não tenha interesse em interagir com o robô naquele momento, desta forma é importante que o robô identifique estas situações e execute ações que minimizem as punições. Um exemplo disso é quando o humano está triste (emoção negativa), que no simulador utilizado esta emoção está associada a maior probabilidade ignorar o robô.

Ainda no experimento em questão, é possível verificar que, mesmo que dada ação dê uma recompensa positiva e imediata ao agente, esta ação pode levá-lo a um estado não tenha

uma recompensa global (cumulativa) máxima. Esta é uma das vantagens do aprendizado por reforço, a capacidade de mapear e consultar a relação entre ações e estados, permitindo o agente prever os próximos estados, bem como a recompensa final estimada.

A validação realizada após o treinamento da SocialDQN, teve como objetivo verificar a capacidade do agente em maximizar a recompensa usando duas políticas distintas. A política gananciosa representa o conhecimento adquirido durante o treinamento e se sobressai sobre a política aleatória. O aprendizado é demonstrado através da comparação das duas políticas verificando o sucesso nas ações "cumprimentar" e "acenar", onde a política gananciosa executa com sucesso ambas as ações tanto em relação à outra política quanto aos casos de falha. Vale ressaltar que "cumprimentar" depende da execução de outras ações, pois esta necessita que o humano esteja em condições ideias para a execução bem sucedida de um aperto de mão. Para avaliação das outras ações ("esperar" e "olhar") seria necessário o mapeamento de recompensas adicionais associadas a elas. Futuramente, serão modeladas recompensas para refinar a execução e avaliação destas ações.

Na fase de validação social, o foco foi verificar a aceitação social do comportamento do agente SocialDQN perante voluntários. Os resultados neste experimento foram bastante satisfatórios, principalmente porque a arquitetura proposta se mostra mais aceitável do que a MDQN. Como observado nos experimentos, ambas as arquiteturas executaram muito bem a ação "cumprimentar", contudo, a MDQN não performa tão bem as demais ações, como, por exemplo, acenando sem nem haver pessoas no ambiente de atuação. A SocialDQN aprendeu a executar esta ação satisfatoriamente, dando prioridade ao comportamento de "esperar" e "olhar" quando não há ninguém na cena ou o agente reconhecer que não é pertinente. Além das emoções, a adição do sinal social "foco de atenção", atrelado à recompensa após o robô acenar, contribuiu para o comportamento do agente, dado que o mesmo era punido quando o mesmo acenava e nenhum humano direcionava o olhar para ele.

Outro ponto importante é que o sistema aprendeu a se portar conforme os comportamentos modelados através do simulador SimDRLSR, dado através de tabelas de probabilidade. Estas tabelas são um esforço inicial para modelar estes comportamentos e é necessário ainda refinar estes comportamentos para refletir o comportamento real humano. Além disso, as probabilidades fazem com que o humano possa se comportar diferentemente, mesmo que o robô execute a mesma ação na mesma situação. Por exemplo, se o robô acena para um humano que está feliz, este humano tem 90% de chances de se mover ou olhar para o robô, tendo 10% de chances de ignorá-lo. Ou seja, mesmo que o robô execute uma ação avaliada correta, o simulador tem um grau de aleatoriedade que influência nos resultados dos testes de validação.

O último experimento realizado envolveu transferir o aprendizado do agente para o robô real. Além da importante constatação de que é possível realizar tal transferência, esta etapa possibilitou a validação do sistema através de comportamentos humanos reais. A análise realizada por voluntários mostrou que a SocialDQN é tão socialmente aceita no ambiente real quanto em

simulado. Entretanto, foi observado que os voluntários, muito das vezes, desconsideraram as emoção humana e consideravam somente a linguagem corporal e distância entre humano e robô para a avaliação. Durante a aplicação do questionário, foi pedido ao voluntário para tentar se colocar no lugar dos humanos apresentados nas imagens. Contudo, emular uma emoção de forma espontânea não é uma tarefa trivial, o que pode ter afetado na avaliação das ações em relação às emoções. Uma forma de contornar isto é que a própria pessoa que interagiu com o agente faça a avaliação das ações do mesmo. Estas questões, bem como a influência das emoções na IHR serão tratadas em trabalhos futuros. Independente disto, a SocialDQN permite a utilização de outros sinais sociais, não se limitando somente às emoções, possibilitando a combinação desta rede com outros modelos disponíveis na literatura para abstração de informações de interações sociais.

Uma limitação do sistema, ao aplicá-lo no ambiente real, diz respeito ao tempo de cada ciclo de interação. No ambiente da biblioteca foi observado que, muito das vezes, as pessoas transitavam rapidamente, não dando muito atenção ao robô, mesmo que este acene para as elas. Como relatado anteriormente, cada fase de interação do agente com ambiente possui uma duração de cerca de 8 segundos, tempo suficiente para que o humano saia da visão do robô ou se distancie o bastante para aquela ação executada não surtir efeito no humano. Um possível esforço seria dotar o agente com ações mais efusivas, com ciclos de atuação mais curtos, ou até mesmo a adoção de diversos modelos de RL para atuar em diferentes fases da interação ou tipos de engajamento, utilizando informações mais simples do ambiente a fim de tornar as ações mais efetivas e imediatas.

Apesar da limitação citada, o foco desta tese foi prover um ambiente de treinamento para que robôs sociais pudessem atuar de forma socialmente aceita. Este objetivo foi alcançado através da atuação do agente no ambiente e validação destas interações por voluntários, tanto ao nível de simulação quanto no ambiente real. De forma geral, em todos os experimentos, os resultados apontam que o robô aprendeu a interagir no ambiente adequadamente a maximizar as recompensas, evitando executar ações que levam a um estado desfavorável, além de aprender a executar ações de forma bastante aceitáveis em relação ao comportamento humano.

## 7.8 Considerações finais

Neste capítulo, foram apresentados os experimentos realizados, envolvendo tanto o simulador SimDRLSR quanto a arquitetura SocialDQN. Foram descritas também, as configurações adotadas para treinamento da arquitetura em simulação. O modelo multimodal MDQN também foi abordado, visando validar o simulador proposto e, posteriormente, compará-lo diretamente com a SocialDQN. Vários testes quantitativos foram realizados envolvendo a SocialDQN e o simulador SimDRLSR. Além disso, a arquitetura foi validada através de voluntários, os quais analisaram a aceitação social da arquitetura SocialDQN, atuando tanto em simulação quanto em

ambiente real através do robô Pepper.

O agente robótico, tanto simulado quanto real, tinha como foco a atuação na fase de pré-interação com humanos. Para isto, a SocialDQN ficou responsável por aprender a relação entre estados e ações. O primeiro é representado por uma sequência de imagens e um vetor com a informação da emoção humana detectada. Para validar o sistema, as emoções foram agrupadas nas classes "postiva", "negativa", "neutra" e "desconhecida". Já as ações, focaram em fazer o agente o acenar, cumprimentar, esperar e olhar para pessoas.

Os experimentos e resultados foram apresentados, analisados e discutidos ao longo do capítulo. Tanto os resultados quantitativos quanto os obtidos de análises sociais demostraram que o sistema proposto, com ajuda do simulador SimDRLSR, consegue prover um modelo para robô social priorizando comportamentos socialmente aceitos, utilizando sinais sociais, em especial a emoção humana extraída através da face.

CAPÍTULO

# 8

# CONCLUSÃO

Um sistema computacional para o treinamento, teste e validação de modelos para robôs sociais foi desenvolvido e validado em ambientes reais. Esse sistema é direcionado especificamente para agentes que precisam se comportar de forma socialmente aceita por humanos durante uma interação. Além disso, foram considerados sinais sociais como parte do mapeamento dos estados do ambiente.

O sistema em questão é composto por dois nós de processamento, a arquitetura SocialDQN (*Social Robotics Deep Q-Network*) e o simulador denominado SimDRLSR (*Simulator for Deep Reinforcement Learning and Social Robotics*). O primeiro consiste no aprendizado de comportamentos humanos interativos a partir de sinais sociais e sequências de imagens. O segundo oferece um ambiente simulado para atuação de um robô social para interação com humanos. Este simulador foi desenvolvido através do motor de jogos Unity 3D e consiste em uma extensão do *Robot House Simulator*, proposto por nós previamente.

A rede SocialDQN aprende graças à técnica *Deep Q-Network*, que considera dados sensoriais de alta dimensão e recompensas das ações do agente como referência. O comportamento humano é complexo, cheio de nuances e muita das vezes moldado pela cultura e padrões sociais. A utilização de imagens para a captura de tais comportamentos se torna viável com a utilização de redes profundas, capazes de aprender padrões que podem não ser tão óbvios para seres humanos. Contudo, existem sinais sociais que podem auxiliar na convergência da rede para comportamentos socialmente aceitáveis, como emoções humanas e foco de atenção. Desta forma, a rede SocialDQN se beneficia tanto da utilização de imagens quanto dos sinais sociais mencionados.

No âmbito de sistemas inteligentes, muitos trabalhos negligenciam aspectos e sinais sociais para interação com pessoas, em especial, abordagens com paradigma de autoaprendizagem. Neste sentido, a SocialDQN oferece um arcabouço para a utilização de sinais sociais diversos para regulação das ações do robô. Nesta tese, foi utilizada a emoção extraída da face

humana para compor a entrada de sinais sociais e para o treinamento da SocialDQN foi utilizado Deep Reinforcement Learning. Em virtude destas características, a rede SocialDQN representa a primeira contribuição desta tese de doutorado.

Na literatura, existem diversos simuladores voltados para DRL. No entanto, nenhum deles possui uma ferramenta para modelar humanos e seus comportamentos por meio de sinais sociais, sendo esta uma segunda contribuição desta tese.

A concepção e emprego do simulador SimDRSLR permitiu o desenvolvimento, treinamento, verificação e validação da arquitetura SocialDQN. O treinamento foi um dos pontos cruciais desta tese para o qual o simulador SimDRSLR foi de altíssima importância, dada a necessidade de estabilizar o aprendizado do modelo, dado através do treinamento de milhares de interações com o ambiente. Esta questão é fundamental na área de Aprendizado por Reforço, pois o mapeamento da relação entre estados e ações cresce exponencialmente com o número de ações e possíveis estados, necessário cada vez mais interações para refinar estes valores, que no nosso caso são mapeados através dos pesos da rede DQN.

O simulador SimDRLSR visa auxiliar na fase de desenvolvimento e teste de parâmetros de algoritmos para aprender comportamentos humanos. Inicialmente, planejou-se que o modelo fosse treinado no simulador e retreinado no ambiente real para que o robô se comportasse adequadamente. Contudo, os testes em ambiente real mostraram ser possível transferir diretamente o modelo para o robô sem a necessidade de fazer um ajuste fino ou treinamento adicional.

A rede MDQN (*Multimodal Deep Q-Network*), modelo multimodal para que robôs sociais atuem na fase de pré-interação com humanos, foi utilizada como alicerce de desenvolvimento de ambos os sistemas apresentados. Além disto, a MDQN serviu como *benchmark* de validação para a SocialDQN, tanto para verificar o potencial desta arquitetura em relação ao cálculo das recompensas acumuladas, quanto na avaliação do comportamento social. Durante o processo de treinamento, a rede MDQN se mostrou bastante instável, o que foi resolvido através do ajuste do paradigma de treino, dado principalmente pelo treinamento da arquitetura por milhares de episódios. Isto foi possível graças a capacidade do simulador SimDRLSR oferecer um ambiente de treinamento ágil, prático e controlado.

Tanto o simulador SimDRLSR quanto a arquitetura SocialDQN tiveram resultados muito satisfatórios e possibilitaram alcançar os objetivos predefinidos nesta tese. Além disso, a utilização de sinais sociais colaborou de forma efetiva para a atuação de agentes robóticos na fase de pré-interação com pessoas. Os testes para verificação da aceitação social do sistema apresentaram uma acurácia de 93,67% com o robô atuando no ambiente real e 91% em simulação. No último caso, a rede SocialDQN superou à MDQN em todas as métricas avaliadas.

É importante citar que tanto a rede SocialDQN quanto o simulador SimDRLSR são aderentes ao modelo de código aberto com a disponibilidade de versões por meio do repositório GitHub. Esta disposição incentiva a colaboração de outros pesquisadores na evolução da

arquitetura e também do simulador.

Os resultados obtidos durante esta tese geraram alguns artigos, aceitos e publicados em conferências importantes na área. Vários trabalhos são frutos diretos dos resultados alcançados, outros relacionam-se à evolução do RHS (simulador base do SimDRLSR) para a validação de uma ontologia e também estão relacionados a projetos envolvendo interações entre humanos e o robô Pepper. Estes trabalhos envolvendo o Pepper contribuíram tecnicamente em vários módulos de interação, comunicação com servidor externo, captura e processamento de imagens, entre vários outros. Os trabalhos em questão são:

- Belo e Romero (2021): BELO, José Pedro R.; ROMERO, Roseli AF. A Social Human-Robot Interaction Simulator for Reinforcement Learning Systems. In: **2021 20th International Conference on Advanced Robotics (ICAR)**. IEEE, 2021. p. 350-355.
- Azevedo *et al.* (2020): AZEVEDO, Helio et al. Use of Ontology to Model the Perception of the Environment by a Humanoid Robot. In: **ONTOBRAS**. 2020. p. 243-250.
- Belo, Sanches e Romero (2019): BELO, José Pedro Ribeiro; SANCHES, Felipe Padula; ROMERO, Roseli Aparecida Francelin. Facial recognition experiments on a robotic system using one-shot learning. In: **2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)**. IEEE, 2019. p. 67-73.
- Belo, Azevedo e Romero (2018): BELO, José Pedro Ribeiro; AZEVEDO, Helio; ROMERO, Roseli AF. Enhancements in a social robotic simulator for indoor environments. In: **2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)**. IEEE, 2018. p. 457-463.

Também foram publicados artigos em revistas de grande impacto na área:

- Belo *et al.* (2022): BELO, José Pedro R. et al. Deep Q-network for social robotics using emotional social signals. **Frontiers in Robotics and AI**, v. 9, 2022.
- Azevedo, Belo e Romero (2020): AZEVEDO, Helio; BELO, José Pedro R.; ROMERO, Roseli AF. Using ontology as a strategy for modeling the interface between the cognitive and robotic systems. **Journal of Intelligent & Robotic Systems**, v. 99, n. 3, p. 431-449, 2020.

Como trabalhos futuros, pretende-se adicionar mais ações à SocialDQN, visando agregar mais comportamentos para interação com humanos. Isso certamente implicará em definir outros tipos de recompensas visando estimar o sucesso da ação do ponto de vista social. Pretende-se também agregar memória à SocialDQN, usando redes recorrentes, tais como, LSTM, GRU

ou TCN, com intuito de fazer com que o robô aprenda a se comportar adequadamente sem a repetição de ações e comportamentos, o que pode causar desinteresse no humano ao longo da interação. Além disso, essa abordagem pode ser uma solução para redução na latência de cada ciclo de interação, atuando para que a rede consiga aprender a relacionar-se considerando a dependência temporal entre estados, dispensando a utilização da sequência de 8 imagens para a tomada de decisão.

Como já mencionado, o simulador SimDRLSR é uma importante contribuição desta tese, e sua evolução representa o amadurecimento para que a comunidade se beneficie de seus benefícios. Neste sentido, futuramente, as ações dos humanos simulados serão refinadas para refletir diferentes comportamentos sociais. Vale ressaltar que os comportamentos humanos são muitas vezes complexos e cheios de intenções dissimuladas que variam conforme o ambiente, a situação e a cultura.

Um outro ponto que pode ser investigado é adequar o simulador para que fique aderente ao *Unity Machine Learning Agents Toolkit* (ML-Agents) (JULIANI *et al.*, 2018). Este *toolkit* permite transformar jogos e simulações para ambientes de treinamento de agentes inteligentes. É possível, nativamente, utilizar implementações de algoritmos de última geração para IA, como aprendizado por reforço, aprendizado por imitação, neuro-evolução, etc.

Finalmente, através do sistema proposto, é possível aplicar robôs sociais em situações do dia a dia, interagindo com pessoas em locais públicos e privados, tais como recepções de bibliotecas, eventos, seminários, asilos, escolas, hospitais e vários outros ambientes, nos quais o robô deve apresentar comportamentos adequados para cada situação.

# REFERÊNCIAS

ADOBE. **Mixamo - Get animated. Animate 3D characters for games, film, and more.** 2022. Disponível em: <https://www.mixamo.com/>. Acesso em: 31 out. 2022. Citado na página 87.

AKALIN, N.; LOUTFI, A. Reinforcement learning approaches in social robotics. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 21, n. 4, p. 1292, 2021. Citado nas páginas 55, 56 e 65.

ANDRIELLA, A.; TORRAS, C.; ALENYÀ, G. Learning Robot Policies Using a High-Level Abstraction Persona-Behaviour Simulator. In: **2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)**. [S.l.: s.n.], 2019. p. 1–8. Citado nas páginas 57, 64 e 65.

AZEVEDO, H.; BELO, J. P. R.; ROMERO, R. A. Using ontology as a strategy for modeling the interface between the cognitive and robotic systems. **Journal of Intelligent & Robotic Systems**, Springer, v. 99, n. 3, p. 431–449, 2020. Citado na página 141.

AZEVEDO, H.; BELO, J. P. R.; SOUZA, I. E. de; ROMERO, R. A. F.; CARLOS-SP-BRAZIL, S. Use of ontology to model the perception of the environment by a humanoid robot. In: **ONTOBRAS**. [S.l.: s.n.], 2020. p. 243–250. Citado na página 141.

AZEVEDO, H.; ROMERO, R. A. F.; BELO, J. P. R. Reducing the gap between cognitive and robotic systems. In: **2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)**. [S.l.: s.n.], 2017. p. 1049–1054. Citado na página 71.

BAGHERI, E.; ROESLER, O.; CAO, H.-L.; VANDERBORGHT, B. A reinforcement learning based cognitive empathy framework for social robots. **International Journal of Social Robotics**, Springer, v. 13, n. 5, p. 1079–1093, 2021. Citado nas páginas 62 e 65.

BELLA, M.; HANINGTON, B. Universal methods of design. **Beverly, MA: Rockport Publishers**, p. 204, 2012. Citado na página 62.

BELO, J. P. R. **Um simulador para robótica social aplicado a ambientes internos**. Dissertação (Dissertação (Mestrado em Ciências de Computação e Matemática Computacional)) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2018. Citado nas páginas 71, 72 e 74.

BELO, J. P. R.; AZEVEDO, H.; RAMOS, J. J.; ROMERO, R. A. Deep q-network for social robotics using emotional social signals. **Frontiers in Robotics and AI**, Frontiers Media SA, v. 9, 2022. Citado na página 141.

BELO, J. P. R.; AZEVEDO, H.; ROMERO, R. A. F. Rhs simulator for robotic cognitive systems. In: **14rd Latin American Robotics Symposium - LARS'2017**. Curitiba, Brazil: IEEE, 2017. ISBN: 978-1-5090-3656-1. Citado na página 31.

____. Enhancements in a social robotic simulator for indoor environments. In: IEEE. **2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)**. [S.l.], 2018. p. 457–463. Citado nas páginas 31 e 141.

BELO, J. P. R.; ROMERO, R. A. F. A social human-robot interaction simulator for reinforcement learning systems. In: **2021 20th International Conference on Advanced Robotics (ICAR)**. [S.l.: s.n.], 2021. p. 350–355. Citado na página 141.

BELO, J. P. R.; SANCHES, F. P.; ROMERO, R. A. F. Facial recognition experiments on a robotic system using one-shot learning. In: **2019 Latin American Robotic Symposium, 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)**. [S.l.: s.n.], 2019. Citado na página 141.

BERTSEKAS, D. P.; BERTSEKAS, D. P.; BERTSEKAS, D. P.; BERTSEKAS, D. P. **Dynamic programming and optimal control**. [S.l.]: Athena scientific Belmont, MA, 1995. v. 1. Citado na página 39.

BLANCO, G. **Classificação de úlceras venosas dermatológicas para apoio a consultas por similaridade utilizando superpixels e aprendizado profundo**. Tese (Doutorado) — Universidade de São Paulo, 2019. Citado na página 46.

BRAGA, A. de P.; CARVALHO, A. C. P. de Leon Ferreira de; LUDERMIR, T. B. **Redes neurais artificiais: teoria e aplicações**. [S.l.]: LTC Editora Rio de Janeiro, Brazil:, 2007. Citado na página 44.

BREAZEAL, C. **Designing Sociable Robots**. Cambridge, MA, USA: MIT Press, 2002. ISBN 0262025108. Citado na página 35.

____. Emotion and sociable humanoid robots. **International journal of human-computer studies**, Elsevier, v. 59, n. 1-2, p. 119–155, 2003. Citado na página 59.

BREAZEAL, C.; DAUTENHAHN, K.; KANDA, T. Social robotics. In: **Springer handbook of robotics**. [S.l.]: Springer, 2016. p. 1935–1972. Citado nas páginas 29, 33, 35 e 49.

BROEKENS, J. Emotion and reinforcement: affective facial expressions facilitate robot learning. In: **Artifical intelligence for human computing**. [S.l.]: Springer, 2007. p. 113–132. Citado nas páginas 55, 64 e 65.

CABANAC, M. What is emotion? **Behavioural Processes**, v. 60, n. 2, p. 69–83, 2002. ISSN 0376-6357. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0376635702000785>. Citado na página 49.

CASTRO-GONZÁLEZ, Á.; MALFAZ, M.; GOROSTIZA, J. F.; SALICHS, M. A. Learning behaviors by an autonomous social robot with motivations. **Cybernetics and Systems**, Taylor & Francis, v. 45, n. 7, p. 568–598, 2014. Citado na página 55.

CHRISTENSEN, H. A roadmap for us robotics from internet to robotics, 2016 edn. **Sponsored by National Science Foundation & University of California, San Diego**, 2016. Citado na página 73.

CHURAMANI, N.; BARROS, P.; GUNES, H.; WERMTER, S. Affect-driven learning of robot behaviour for collaborative human-robot interactions. **Frontiers in Robotics and AI**, Frontiers, p. 20, 2022. Citado nas páginas 59 e 65.

CLARK-TURNER, M.; BEGUM, M. Deep reinforcement learning of abstract reasoning from demonstrations. In: **Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction**. [S.l.: s.n.], 2018. p. 160–168. Citado nas páginas 63 e 65.

DANTZER, R. The psychosomatic delusion. **Why the mind is not the source of all our ills**, Free Press, 1993. Citado nas páginas 29 e 49.

DARWIN, C.; PRODGER, P. **The expression of the emotions in man and animals**. [S.l.]: Oxford University Press, USA, 1998. Citado na página 50.

ECHEVERRIA, G. b.; LEMAIGNAN, S. b. c.; DEGROOTE, A. b.; LACROIX, S. b.; KARG, M.; KOCH, P.; LESIRE, C.; STINCKWICH, S. f. Simulating complex robotic scenarios with morse. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 7628 LNAI, p. 197–208, 2012. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84868031048&doi=10.1007%2f978-3-642-34327-8_20&partnerID=40&md5=67bff3a53ae2e2d35582679f473e2efe>. Citado na página 71.

EKMAN, P. Basic emotions. **Handbook of cognition and emotion**, v. 98, n. 45-60, p. 16, 1999. Citado nas páginas 49 e 50.

EKMAN, P.; FRIESEN, W. V. Constants across cultures in the face and emotion. **Journal of personality and social psychology**, American Psychological Association, v. 17, n. 2, p. 124, 1971. Citado nas páginas 30, 50, 51, 91 e 104.

EKMAN, P. E.; DAVIDSON, R. J. **The nature of emotion: Fundamental questions.** [S.l.]: Oxford University Press, 1994. Citado na página 51.

ELLIS, H. C.; MOORE, B. A. Mood and memory. **Handbook of cognition and emotion**, p. 193–210, 1999. Citado na página 49.

FONG, T.; NOURBAKHSH, I.; DAUTENHAHN, K. A survey of socially interactive robots. **Robotics and Autonomous Systems**, v. 42, n. 3-4, p. 143–166, 2003. Citado na página 29.

FRANÇOIS-LAVET, V.; HENDERSON, P.; ISLAM, R.; BELLEMARE, M. G.; PINEAU, J. *et al.* An introduction to deep reinforcement learning. **Foundations and Trends® in Machine Learning**, Now Publishers, Inc., v. 11, n. 3-4, p. 219–354, 2018. Citado na página 49.

GAO, Y.; YANG, F.; FRISK, M.; HEMANDEZ, D.; PETERS, C.; CASTELLANO, G. Learning socially appropriate robot approaching behavior toward groups using deep reinforcement learning. In: IEEE. **2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)**. [S.l.], 2019. p. 1–8. Citado nas páginas 64 e 66.

GLATT, R. **Knowledge reuse for deep reinforcement learning.** Tese (Doutorado) — Universidade de São Paulo, 2019. Citado na página 49.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 46.

GOODFELLOW, I. J.; ERHAN, D.; CARRIER, P. L.; COURVILLE, A.; MIRZA, M.; HAMNER, B.; CUKIERSKI, W.; TANG, Y.; THALER, D.; LEE, D.-H. *et al.* Challenges in representation learning: A report on three machine learning contests. In: SPRINGER. **International conference on neural information processing**. [S.l.], 2013. p. 117–124. Citado na página 76.

GOODRICH, M. A.; SCHULTZ, A. C. Human-robot interaction: a survey. **Foundations and trends in human-computer interaction**, Now Publishers Inc., v. 1, n. 3, p. 203–275, 2007. Citado na página 29.

GRAVES, A. Supervised sequence labelling. In: **Supervised sequence labelling with recurrent neural networks**. [S.l.]: Springer, 2012. p. 5–13. Citado na página 45.

GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. In: IEEE. **2013 IEEE international conference on acoustics, speech and signal processing**. [S.l.], 2013. p. 6645–6649. Citado na página 45.

GREENE, J. D. **Moral tribes: Emotion, reason, and the gap between us and them**. [S.l.]: Penguin, 2013. Citado na página 49.

HALL, E. T.; BIRDWHISTELL, R. L.; BOCK, B.; BOHANNAN, P.; JR, A. R. D.; DURBIN, M.; EDMONSON, M. S.; FISCHER, J.; HYMES, D.; KIMBALL, S. T. *et al.* Proxemics [and comments and replies]. **Current anthropology**, v. 9, n. 2/3, p. 83–108, 1968. Citado na página 64.

HAO, M.; CAO, W.; LIU, Z.; WU, M.; YUAN, Y. Emotion Regulation Based on Multi-objective Weighted Reinforcement Learning for Human-robot Interaction. In: **2019 12th Asian Control Conference (ASCC)**. [S.l.: s.n.], 2019. p. 1402–1406. Citado nas páginas 59, 64 e 65.

HAYKIN, S. **Neural Networks and Learning Machines, 3/E**. [S.l.]: Pearson Education India, 2010. Citado na página 44.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778. Citado nas páginas 45 e 47.

HINTON, G.; DENG, L.; YU, D.; DAHL, G.; MOHAMED, A.-r.; JAITLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; KINGSBURY, B. *et al.* Deep neural networks for acoustic modeling in speech recognition. **IEEE Signal processing magazine**, v. 29, 2012. Citado na página 45.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 45.

HUAMáN, A. **Cascade classifier**. 2022. Disponível em: <https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html>. Citado na página 77.

JULIANI, A.; BERGES, V.-P.; TENG, E.; COHEN, A.; HARPER, J.; ELION, C.; GOY, C.; GAO, Y.; HENRY, H.; MATTAR, M.; LANGE, D. **Unity: A General Platform for Intelligent Agents**. arXiv, 2018. Disponível em: <https://arxiv.org/abs/1809.02627>. Citado na página 142.

KESSELS, R. P.; MONTAGNE, B.; HENDRIKS, A. W.; PERRETT, D. I.; HAAN, E. H. de. Assessment of perception of morphed facial expressions using the emotion recognition task: Normative data from healthy participants aged 8–75. **Journal of neuropsychology**, Wiley Online Library, v. 8, n. 1, p. 75–93, 2014. Citado nas páginas 29, 33 e 49.

KIM, S.; ASADI, K.; LITTMAN, M.; KONIDARIS, G. Deepmellow: removing the need for a target network in deep q-learning. In: **Proceedings of the Twenty Eighth International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. Citado na página 109.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105. Citado nas páginas 45 e 47.

KUMRAI, T.; KORPELA, J.; MAEKAWA, T.; YU, Y.; KANAI, R. Human Activity Recognition with Deep Reinforcement Learning using the Camera of a Mobile Robot. In: **2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)**. [S.l.: s.n.], 2020. p. 1–10. Citado nas páginas 61 e 64.

KUO, P.-H.; HO, Y.-F.; LEE, K.-F.; TAI, L.-H.; LI, T.-H. Development of humanoid robot simulator for gait learning by using particle swarm optimization. In: . [s.n.], 2013. p. 2683–2688. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893606684&doi=10.1109%2fSMC.2013.457&partnerID=40&md5=13b9498b17bcea8289eac27dc4a570cb>. Citado na página 71.

LATHUILIèRE, S.; MASSé, B.; MESEJO, P.; HORAUD, R. Neural network based reinforcement learning for audio–visual gaze control in human–robot interaction. **Pattern Recognition Letters**, Elsevier BV, v. 118, p. 61–71, Feb 2019. ISSN 0167-8655. Disponível em: <http://dx.doi.org/10.1016/j.patrec.2018.05.023>. Citado nas páginas 60 e 64.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 45.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. *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 47.

LI, M.; XIE, L.; ZHANG, A.; REN, F. Reinforcement Emotion-Cognition System: A Teaching Words Task. **Computational Intelligence and Neuroscience**, Hindawi, v. 2019, 2019. Citado nas páginas 60 e 65.

LI, S.; DENG, W. Deep facial expression recognition: A survey. **IEEE transactions on affective computing**, IEEE, 2020. Citado na página 76.

LI, Y. Deep reinforcement learning. **CoRR**, abs/1810.06339, 2018. Disponível em: <http://arxiv.org/abs/1810.06339>. Citado nas páginas 39, 40, 41, 44, 48 e 49.

LIN, J.; ZHANG, Q.; GOMEZ, R.; NAKAMURA, K.; HE, B.; LI, G. Human social feedback for efficient interactive reinforcement agent learning. In: **2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)**. [S.l.: s.n.], 2020. p. 706–712. Citado nas páginas 61, 64 e 65.

LUGER, G. F.; STUBBLEFIELD, W. A. **Artificial intelligence and the design of expert systems**. [S.l.]: Benjamin-Cummings Publishing Co., Inc., 1990. Citado na página 44.

MAROTO-GÓMEZ, M.; GONZÁLEZ, R.; CASTRO-GONZÁLEZ, Á.; MALFAZ, M.; SALICHS, M. Á. Speeding-Up Action Learning in a Social Robot With Dyna-Q+: A Bioinspired Probabilistic Model Approach. **IEEE Access**, v. 9, p. 98381–98397, 2021. Citado nas páginas 58 e 65.

MATARIĆ, M. J.; ARKIN, R. C. **The robotics primer**. [S.l.]: Mit Press, 2007. Citado na página 55.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 44.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. *et al.* Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529, 2015. Citado nas páginas 47, 48, 67 e 101.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial: Tradução da 3a Edição**. [S.l.]: Elsevier Brasil, 2014. v. 1. Citado na página 37.

OATLEY, K. Emotion: Theories. Oxford University Press, 2000. Citado na página 49.

PAPAIOANNOU, I.; DONDRUP, C.; NOVIKOVA, J.; LEMON, O. Hybrid chat and task dialogue for more engaging HRI using reinforcement learning. In: **2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)**. [S.l.: s.n.], 2017. p. 593–598. Citado nas páginas 56 e 65.

Paul Ekman Group. **Universal Emotions. What are Emotions?** 2022. Disponível em: <https://www.paulekman.com/universal-emotions/>. Acesso em: 07 feb. 2022. Citado na página 91.

PERGHER, G. K.; GRASSI-OLIVEIRA, R.; ÁVILA, L. d.; STEIN, L. M. Memória, humor e emoção. **Revista de psiquiatria do Rio Grande do Sul**, SciELO Brasil, v. 28, n. 1, p. 61–68, 2006. Citado na página 49.

PLUTCHIK, R. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. **American Scientist**, Sigma Xi, The Scientific Research Society, v. 89, n. 4, p. 344–350, 2001. ISSN 00030996. Disponível em: <http://www.jstor.org/stable/27857503>. Citado nas páginas 50 e 51.

QURESHI, A. H.; NAKAMURA, Y.; YOSHIKAWA, Y.; ISHIGURO, H. Robot gains social intelligence through multimodal deep reinforcement learning. In: **2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)**. [S.l.: s.n.], 2016. p. 745–751. ISSN 2164-0580. Citado nas páginas 19, 30, 32, 63, 66, 67, 68, 69, 70, 81, 107, 112, 113, 117, 118, 119, 120, 129, 130 e 134.

____. Intrinsically motivated reinforcement learning for human–robot interaction in the real-world. **Neural Networks**, Elsevier, v. 107, p. 23–33, 2018. Citado nas páginas 63, 66, 67, 68, 69, 70, 71, 107 e 118.

RICH, C.; PONSLER, B.; HOLROYD, A.; SIDNER, C. L. Recognizing engagement in human-robot interaction. In: IEEE. **2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)**. [S.l.], 2010. p. 375–382. Citado na página 93.

RIEDMILLER, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In: SPRINGER. **European Conference on Machine Learning**. [S.l.], 2005. p. 317–328. Citado na página 48.

ROCHA, C. D. F. d. *et al.* Aplicação do algoritmo haar cascade em um sistema embarcado para detecção de ovos do mosquito aedes aegypti em palhetas de ovitrampas. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2018. Citado na página 77.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 44.

RUDOVIC, O.; PARK, H. W.; BUSCHE, J.; SCHULLER, B.; BREAZEAL, C.; PICARD, R. W. Personalized Estimation of Engagement From Videos Using Active Learning With Deep Reinforcement Learning. In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)**. [S.l.: s.n.], 2019. p. 217–226. Citado nas páginas 60, 65 e 93.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *et al.* Learning representations by back-propagating errors. **Cognitive modeling**, v. 5, n. 3, p. 1, 1988. Citado na página 44.

RUMMERY, G. A.; NIRANJAN, M. **On-Line Q-Learning Using Connectionist Systems**. [S.l.], 1994. Citado na página 40.

RUSSELL, J. A.; LEWICKA, M.; NIIT, T. A cross-cultural study of a circumplex model of affect. **Journal of personality and social psychology**, American Psychological Association, v. 57, n. 5, p. 848, 1989. Citado nas páginas 52 e 104.

SIDNER, C. L.; LEE, C.; KIDD, C. D.; LESH, N.; RICH, C. Explorations in engagement for humans and robots. **Artificial Intelligence**, Elsevier, v. 166, n. 1-2, p. 140–164, 2005. Citado na página 93.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014. Citado na página 47.

SPARC Robotics. Robotics 2020 multi-annual roadmap for robotics in europe. **SPARC Robotics, EU-Robotics AISBL, The Hauge, The Netherlands, accessed Feb**, v. 5, p. 2018, 2016. Citado nas páginas 29 e 30.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. [S.l.]: MIT press, 2018. Citado nas páginas 37, 38, 39, 40, 41, 42 e 88.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9. Citado nas páginas 45 e 47.

TESAURO, G. Td-gammon, a self-teaching backgammon program, achieves master-level play. **Neural computation**, MIT Press, v. 6, n. 2, p. 215–219, 1994. Citado na página 48.

TOZADORE, D. C.; VALENTINI, J. P. H.; RODRIGUES, V. H. S.; VENDRAMETO, F. M. L.; ZAVARIZZ, R. G.; ROMERO, R. A. F. Towards adaptation and personalization in task based on human-robot interaction. In: **2018 Latin American Robotics Symposium (LARS) and 2018 Brazilian Symposium on Robotics (SBR)**. [S.l.: s.n.], 2018. Citado nas páginas 104 e 113.

TUBB, N. A development path to success in neural computing. **Expert Systems Applications**, v. 9, n. 5, p. 5–9, 1993. Citado na página 44.

Unity Technologies. **Unity Platform**. 2022. Disponível em: <https://unity.com/products/unity-platform>. Acesso em: 05 feb. 2022. Citado nas páginas 31, 71 e 81.

VIOLA, P.; JONES, M. J. Robust real-time face detection. **International journal of computer vision**, Springer, v. 57, n. 2, p. 137–154, 2004. Citado na página 77.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. In: **Machine Learning**. [S.l.: s.n.], 1992. p. 279–292. Citado na página 40.

WIDROW, B. Generalization and information storage in networks of adaline neurons. self organizing systems. **Yovitz, MC, Jacobi, GT, and Goldstein, GD editors**, p. 435–461, 1962. Citado na página 44.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 229–256, 1992. Citado na página 42.

ZHANG, K.; ZHANG, Z.; LI, Z.; QIAO, Y. Joint face detection and alignment using multitask cascaded convolutional networks. **IEEE signal processing letters**, IEEE, v. 23, n. 10, p. 1499–1503, 2016. Citado na página 77.

ANEXO

A

# PÁGINAS INTERESSANTES NA INTERNET

**<https://sci-hub.se/>** Plataforma de acesso gratuito a milhões de artigos científicos e livros acadêmicos;

**<https://www.tablesgenerator.com/>** Ferramenta online para formatar e gerar tabelas para LaTeX;


ICMC
USP
SÃO CARLOS