**Aprendizado por reforço profundo para navegação visual
semântica com memória**

Iury Batista de Andrade Santos

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Iury Batista de Andrade Santos

Aprendizado por reforço profundo para navegação visual semântica com memória

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Prof.^aDr.^a Roseli Aparecida Francelin Romero

USP – São Carlos
Dezembro de 2020

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

S237a Santos, Iury Batista de Andrade
Aprendizado por reforço profundo para navegação
visual semântica com memória / Iury Batista de
Andrade Santos; orientadora Roseli Aparecida
Francelin Romero. -- São Carlos, 2020.
120 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2020.

1. Navegação de robôs móveis. 2. Aprendizagem
profunda. 3. Aprendizado por reforço profundo. 4.
Aprendizado de robôs. I. Romero, Roseli Aparecida
Francelin, orient. II. Título.

Iury Batista de Andrade Santos

**Deep reinforcement learning for visual semantic navigation
with memory**

Dissertation 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 Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof.^aDr.^a Roseli Aparecida Francelin Romero

**USP – São Carlos
December 2020**

*Este trabalho é dedicado aos meus pais,
Dora e Batista.*

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais, Maria Auxiliadora e José Batista, e a meu irmão, Ikaro Batista, pela fundação e apoio, essenciais para que fosse possível a realização desta jornada, e constante fonte de inspiração e motivação.

A minha orientadora, Prof.^aDr.^aRoseli Aparecida Francelin Romero, por ter acreditado neste trabalho e guiado-me ao longo desta jornada, tornado-a mais leve e, não raramente, trazendo palavras de calma e segurança quando se fizeram tão necessárias.

Aos membros da minha banca de qualificação, Prof. Dr. Fernando Santos Osório e Prof. Dr. Valdir Grassi Junior, pelas contribuições e críticas que permitiram a melhoria e aperfeiçoamento deste trabalho, sendo de valor inestimável.

Aos meus amigos de laboratório, pesquisa e caminhada, em especial, José Pedro e Felipe Padula, pelas conversas, as serias e as descontraídas, ajudas, incentivo e companheirismo. A Gean Pereira pelas acaloradas discordâncias, mas valorosas contribuições e partilha de ideias. A todos os companheiros que fazem parte do Laboratório de Aprendizado de Robôs (LAR) e do grupo de Computação Bioinspirada (Biocom), que compuseram um ambiente rico, divertido e prestativo.

A todas minhas amigadas, as novas do ICMC e USP, e as velhas e distantes, mas imensamente importantes e sempre próximas ao espírito. O caminho é sempre mais fácil quando há companhia e partilha.

A todos aqueles que trabalham no Instituto de Ciências Matemáticas e de Computação, em todos setores e funções, pelo trabalho de excelência realizado, sempre prestativo e gentil.

Agradeço as agências de fomento, CAPES, CPNq e FAPESP, pelo essencial financiamento, total e parcial, e suporte para a realização desta pesquisa e da ciência nacional.

A todos os professores e cientistas deste país comprometidos com a educação, avanço e progresso da nossa sociedade e das pessoas que a compõem, que persistem de maneira hercúlea a todas as tempestades e intempéries. São fontes de inspiração, orgulho e esperança.

Por fim, agradeço enormemente a todo o Povo brasileiro e a Universidade Pública, pelo imprescindível serviço prestado ao bem da nossa sociedade, não alcançado sem constante luta e bravura.

*“Começamos como povos errantes,
e ainda o somos.”
(Carl Sagan)*

RESUMO

SANTOS, I. B. DE A. **Aprendizado por reforço profundo para navegação visual semântica com memória**. 2020. 120 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

A navegação de robôs móveis é um tema amplamente estudado ao longo de décadas, sendo esta uma atividade crucial para a inserção de robôs em diversos cenários. Contudo, ambientes complexos e mutáveis, como interiores de residências, ainda apresentam desafios a serem superados, sendo objetos de estudo em diversos trabalhos que adotam abordagens com visão computacional sem usar mapas topológicos ou métricos. Este trabalho propõe uma arquitetura para navegação de robôs móveis visando a busca objeto-direcionada em ambientes internos de residências, utilizando métodos de visão computacional e informações semânticas com memória. A arquitetura proposta tem capacidade de generalizar por meio de conhecimento *a priori* dos objetos detectados em cenas e também reforçar relacionamentos por meio de experiências passadas, em uma abordagem de navegação baseada em aprendizado. Para tanto, são adotados os seguintes modelos de aprendizado de máquina: redes neurais convolucionais, redes neurais de grafos, redes neurais recorrentes e aprendizado por reforço profundo, em uma abordagem objeto-direcionada. Esta arquitetura foi treinada em diversos ambientes domésticos, adotando-se um ambiente simulado fotorrealista. Esta arquitetura foi avaliada por meio de análise qualitativa, com execuções de episódios do agente no ambiente simulado de maneira visual, e análise quantitativa, usando medidas como taxas de sucesso e taxas de sucesso ponderadas pela trajetória. Políticas aprendidas por meio da arquitetura proposta foram comparadas com agentes de política randômica, com agentes utilizando apenas aprendizado por reforço e, por fim, com agentes com políticas de navegação semântica sem memória. Os experimentos realizados mostraram um comportamento mais exploratório da arquitetura proposta em relação a outros sem memória, obtendo maiores taxas de êxito nas tarefas em ambas as métricas. Quando exposto a cenários mais restritos, e por conseguinte de maior dificuldade, as políticas aprendidas por estes modelos apresentaram os melhores resultados, com menor queda no desempenho quando comparados as execuções menos restritivas e com os demais modelos. Desta forma, o modelo proposto apresentou resultados consistentes de melhoria nas políticas aprendidas pelos agentes, resultando em comportamentos bem sucedidos para tarefas de busca objeto-direcionada em ambientes internos de residências.

Palavras-chave: Navegação Visual, Navegação Semântica, Aprendizagem Profunda, Aprendizado por Reforço Profundo, Robótica.

ABSTRACT

SANTOS, I. B. DE A. **Deep reinforcement learning for visual semantic navigation with memory** . 2020. 120 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

The navigation of mobile robots is a subject vastly studied in the last decades, being a crucial task for the insertion of robots in diverse scenarios. However, complex and changeable environments, as indoors of houses, still shows challengers to be transpassed, being an object of study in several works that adopts approaches as computer vision without topological or metric maps. This work proposes an architecture for the navigation of mobile robots aiming target-object search in indoor ambiances of houses, using computer vision methods and semantic information with memory. The proposed architecture can generalize through *a priori* acknowledgment of detect objects in scenes and reinforce relationships over experiences of the past, in a learning-based navigation approach. Therefore, the following models of machine learning will be adopted: neural convolutional networks, graph neural networks, recurrent neural networks and deep reinforcement learning, in a target-object approach. This architecture has trained in several domestic ambiances, adopting a photo-realistic simulated environment. The architecture was evaluated through qualitative analysis, executing episodes of the agent in the simulated environment with visual insight, and quantitative analysis, adopting metrics like success rate and success rate weighted by path length. Policies learn by the proposed architecture were compared with agents using random policies, agents using only reinforcement learning, and, finally, agents with navigation semantic policies without memory. The experiments performed showed a more exploratory behavior of the proposed architecture when compared with the non-memory approaches. reaching better success rates in the tasks for both metrics. When exposed to restrict scenarios, consequently being of greater difficulty, the policies learn by such models demonstrated better results, with a lower decrease in its performance when compared with less restrictive executions and other models. Thus, the proposed model presented consistent results with better policies learn by the agents, resulting in behaviors more successful in the task of target-object search in indoor-home environments.

Keywords: Visual Navigation, Semantic Navigation, Deep Learning, Deep Reinforcement Learning, Robotics.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação da arquitetura da rede LeNet	34
Figura 2 – Detalhe dos campos receptivos locais e compartilhamento de pesos presente em uma ConvNet	36
Figura 3 – Modelos arquiteturais de uma VGGNet com 19 camadas, de uma rede plana com 34 camadas e de uma ResNet com 34 camadas	37
Figura 4 – Detalhe da estrutura de um bloco de aprendizado residual	38
Figura 5 – Representação de uma RNN. A esquerda, uma representação recursiva, e a direita, como um grafo desmembrado	39
Figura 6 – Modelo de célula de uma <i>Vanilla</i> LSTM	41
Figura 7 – Representação de uma GNN	44
Figura 8 – Modelo arquitetural proposto por Zhu <i>et al.</i> (2017b)	62
Figura 9 – Modelo arquitetural proposto por Nguyen <i>et al.</i> (2018)	63
Figura 10 – Modelo arquitetural proposto por Mousavian <i>et al.</i> (2018)	66
Figura 11 – Modelo arquitetural proposto por Yang <i>et al.</i> (2019)	68
Figura 12 – Arquitetura base de Yang <i>et al.</i> (2019), sem utilização de memória, e nossa arquitetura proposta, com módulo de memória	80
Figura 13 – Procedimento para construção do grafo relacional objeto-objeto a ser computado pela GCN	83
Figura 14 – Ilustração da execução da GCN dado um grafo inicial como entrada e tendo ao final uma codificação das relações entre nós-objetos	84
Figura 15 – Cenas dos ambientes disponíveis para simulação no ambiente simulado AI2-THOR: sala de estar, cozinha, quarto e banheiro	88
Figura 16 – Gráficos de média total de recompensa, tamanho de trajetória e disparo de ação encerrar, agrupado por modelo, durante o treinamento	98
Figura 17 – Imagem da captura inicial obtida pelo agente na posição de início e da visão do ambiente do ponto de vista do topo. Na visão do topo, o agente é marcado pelo círculo azul, enquanto objetos-alvo válidos são marcados como círculos verdes	103
Figura 18 – Comparativo entre modelos no ambiente sala de estar e objeto-alvo cadeira	104
Figura 19 – Comparativo entre modelos no ambiente quarto e objeto-alvo livro	105
Figura 20 – Comparativo entre modelos no ambiente banheiro e objeto-alvo toalha	105
Figura 21 – Comparativo entre modelos no ambiente cozinha e objeto-alvo panela	106

Figura 22 – Comparativo entre modelos no ambiente quarto e objeto-alvo telefone
celular 107

LISTA DE ALGORITMOS

Algoritmo 1 – Algoritmo Q-Learning	50
Algoritmo 2 – Algoritmo REINFORCE com <i>baseline</i>	52
Algoritmo 3 – Algoritmo Actor-Critic	52

LISTA DE TABELAS

Tabela 1	– Tabela comparativa entre trabalhos relacionados envolvendo navegação	73
Tabela 2	– Lista de objetos disponíveis como objetos-alvo de acordo com as categorias de ambientes	94
Tabela 3	– Tabela de resultados com médias para STT e TS para todos os modelos. Os resultados são apresentados para “todos” os casos e casos com a restrição $L \geq 5$. Em parenteses é informado o desvio padrão	100

LISTA DE ABREVIATURAS E SIGLAS

A3C	<i>Asynchronous advantage actor-critic</i>
aGCN	<i>Attentional Graph Convolutional Network</i>
AP	Aprendizagem Profunda
AR	Aprendizado por Reforço
ARP	Aprendizado por Reforço Profundo
CEC	<i>Constant Error Carousel</i>
ConvNets	<i>Convolutional Neural Networks</i>
DQN	<i>deep Q-Network</i>
GCN	<i>Graph Convolutional Network</i>
GNN	<i>Graph Neural Network</i>
GRU	<i>Gated Recurrent Unit</i>
IL	<i>Imitation Learning</i>
ILSVRC	<i>ImageNet Large Scale Visual Recognition Challenge</i>
LSTM	<i>Long Short-Term Memory</i>
MLP	<i>Multilayer Perceptrons</i>
MS COCO	<i>Microsoft Common Objects in COntext</i>
PDM	Processo de Decisão de Markov
RePN	<i>Relation Proposal Network</i>
ResNet	<i>Residual Network</i>
RNAs	Redes Neurais Artificiais
RNNs	<i>Recurrent Neural Networks</i>
SR	<i>Successor representation</i>
STT	Sucesso ponderado pelo Tamanho da Trajetória
TS	Taxa de Sucesso

LISTA DE SÍMBOLOS

$\mathbb{R}^S$ — Vetor de dimensão- S do nó n

$v_\pi(s)$ — Função valor

π^* — Política ótima

$q_*(s,a)$ — Função ação-valor ótima

SUMÁRIO

1	INTRODUÇÃO	27
1.1	Organização do trabalho	30
2	FUNDAMENTAÇÃO TEÓRICA	33
2.1	Redes neurais profundas	33
2.1.1	<i>Redes neurais convolucionais</i>	34
2.1.2	<i>Rede Residual</i>	35
2.1.3	<i>Transferência de aprendizado</i>	38
2.1.4	<i>Redes neurais recorrentes</i>	39
2.1.5	<i>Long Short-Term Memory</i>	41
2.1.6	<i>Gated Recurrent Unit</i>	43
2.2	Redes neurais de grafo	43
2.2.1	<i>Rede Convolucional de Grafo</i>	46
2.3	Aprendizado por reforço	47
2.3.1	<i>Q-Learning</i>	49
2.3.2	<i>Sarsa</i>	50
2.3.3	<i>Métodos de Política por Gradiente</i>	50
2.3.4	<i>Actor-Critic</i>	51
2.4	Aprendizado por reforço profundo	52
2.5	Considerações finais	54
3	TRABALHOS RELACIONADOS	57
3.1	Relações visuais e compreensão semântica	57
3.2	Navegação visual	59
3.3	Navegação Semântica	64
3.4	Comparativo e Discussão	71
3.5	Considerações Finais	74
4	PESQUISA PROPOSTA	77
4.1	Navegação visual semântica com memória	77
4.1.1	<i>Módulo Visual</i>	81
4.1.2	<i>Módulo Semântico</i>	82
4.1.3	<i>Módulo de Memória</i>	84

4.1.4	<i>Módulo de Aprendizado</i>	85
4.2	<i>Ambientes Simulados</i>	86
4.3	<i>Técnicas e recursos</i>	89
4.4	<i>Definição da tarefa</i>	89
4.5	<i>Métricas de avaliação</i>	90
4.6	<i>Considerações finais</i>	90
5	EXPERIMENTOS E RESULTADOS	93
5.1	<i>Experimentos</i>	93
5.1.1	<i>Configuração do ambiente</i>	94
5.1.2	<i>Configurações de treinamento</i>	95
5.2	<i>Resultados e Discussão</i>	96
5.2.1	<i>Análise Quantitativa</i>	99
5.2.2	<i>Análise Qualitativa</i>	101
5.2.3	<i>Discussão dos resultados obtidos</i>	107
5.3	<i>Considerações finais</i>	109
6	CONCLUSÃO	111
	REFERÊNCIAS	115

INTRODUÇÃO

Uma das atividades essenciais para robôs sociais e domésticos é a navegação em ambientes complexos, sendo amplamente estudada na área de robótica ao longo das últimas décadas. Métodos utilizando diversas técnicas e sensores têm sido propostos com objetivo de lidar com cenários complexos e incertos impostos pela realidade, como ambientes mutáveis e imprevisíveis, compreensão em alto nível na interação com humanos e outros agentes, planejamento de tarefas elaboradas e interpretações de informações contextuais implícitas.

No que se refere ao acesso a informações sobre a composição do ambiente, pode-se dividir a área de navegação por métodos que utilizam uma representação do ambiente de maneira explícita, pela utilização de mapas topológicos ou mapas métricos ([FUENTES-PACHECO; RUIZ-ASCENCIO; RENDÓN-MANCHA, 2015](#); [ELFES, 1987](#); [ELFES, 1989](#); [GUPTA *et al.*, 2017](#); [CHIANG *et al.*, 2019](#)), e métodos livres de representação explícitas, como em [Lenser e Veloso \(2003\)](#), [Remazeilles, Chaumette e Gros \(2004\)](#), [Zhu *et al.* \(2017b\)](#). Estes últimos vêm sendo investigados em trabalhos recentes, buscando-se dotar agentes robóticos com a capacidade de navegar em ambientes variados, mutáveis ou incapazes de serem mapeados, adotando-se abordagens como navegação baseada em aprendizado atrelada a métodos de aprendizado por reforço profundo.

Assim como em relação a forma de representação mapas, a navegação de robôs móveis pode se diferenciar nos tipos e conjuntos de sensores utilizados por estes como, por exemplo, sensores laser, sonares e câmeras. Devido ao sucesso de modelos de redes neurais profundas aplicados a imagens, a navegação visual tem sido abordada na literatura recente e vários métodos tem sido propostos, como em [Gupta *et al.* \(2017\)](#), [Zhu *et al.* \(2017b\)](#), [Lobos-Tsunekawa, Leiva e Solar \(2018\)](#) e [Nguyen *et al.* \(2018\)](#).

A navegação visual pode ser entendida como modelos de navegação que utilizam como entrada informações provenientes de sensores visuais, sendo esta a fonte principal

ou única de informação sensorial do modelo. Em trabalhos recentes, tais métodos se valem de avanços na área de aprendizado de máquina, como redes neurais convolucionais para extrair características, reconhecer objetos e interpretar cenas a partir de imagens.

Além disso, avanços proporcionados por simuladores são de grande valia para as abordagens utilizando navegação visual. Simuladores como MINOS (SAVVA *et al.*, 2017), AI Habitat (SAVVA* *et al.*, 2019) e AI2-THOR (KOLVE *et al.*, 2017) fornecem cenários fotorrealistas de ambientes internos de residências, com riqueza de informações sobre objetos da cena, como classes, segmentações, profundidade e estado atual. Os ambientes simulados também permitem que trabalhos recentes utilizem técnicas como aprendizado por reforço profundo, objetivando aprender políticas por meio de experiências nestes ambientes, obtendo-se modelos mais generalistas e flexíveis.

Contudo, ainda são poucos os trabalhos que exploram a utilização de informações semânticas sobre o ambiente para auxiliar em tarefas de navegação. A análise semântica tem por objetivo entender como objetos se relacionam, a forma como são representados de maneira a caracterizá-los e os significados emergentes destas representações. Por exemplo, inferir que em um contexto de uma cozinha doméstica, próximo a uma cafeteira, possam ser encontradas xícaras, ou que próximo a uma geladeira pode-se frequentemente encontrar um fogão. Tais relações não se encontram explicitamente presentes no ambiente, requerendo assim que informações adicionais estejam ao alcance do modelo para que este possa tirar proveito dessas relações.

Uma das formas de se obter tal compreensão é por meio da vivência e aprendizado das relações e recorrências entre objetos. Este aprendizado pode ser obtido por meio da utilização dos métodos já citados de aprendizado por reforço profundo e ambientes simulados, aliados a métodos que proporcionem uma base de conhecimento prévia. Trabalhos como Mousavian *et al.* (2018) e Yang *et al.* (2019) utilizam informações semânticas para aprender relações ou representações de objetos no ambiente, almejando maior compreensão da cena e seu estado atual.

De maneira ilustrativa, pode-se considerar a forma como humanos locomovem-se em ambientes em busca de objetos, podendo utilizar tanto informações visuais sobre objetos e características do ambiente, bem como informações sobre relacionamentos e contexto semântico. Mesmo que em uma casa antes nunca visitada, é possível inferir que, quando em busca de um objeto como uma “faca”, considera-se mais provável obter-se sucesso procurando esta na cozinha. Ainda, pode-se considerar que tal objeto tem maiores chances de estar disposto em gavetas de armários, bancadas ou mesas. Essas informações podem ser desenvolvidas implicitamente, por meio de experiências passadas em ambientes semelhantes e relações observadas nestes contextos, e auxiliando a obtenção de resultados positivos mesmo quando o objeto não é inicialmente visível.

Em um contexto de navegação por agentes robóticos em ambientes internos de

residências, almejamos obter comportamentos semelhantes na tarefa de localizar objetos em uma cena, valendo-se de informações visuais sobre o ambiente e contexto semântico da relação entre objetos, em uma abordagem de navegação baseada em aprendizado. Definindo-se como um problema objeto-direcionado, é possível incumbir um agente robótico da tarefa de localizar um objeto qualquer como, por exemplo, um “telefone celular”, dado um cenário contextual de um ambiente do tipo quarto. Neste, baseando-se em imagens capturadas do ambiente, o agente deve locomover-se em busca do objeto-alvo, não tendo nenhuma representação como mapas topológicos ou métricos como informação, e inferindo como objetos visualizados na cena se relacionam. Esta composição é justificada pela característica não-estática do objeto-alvo, podendo estar disposto no ambiente em diversas localizações. Assim sendo, informações fixas sobre a posição dos objetos-alvo não são de interesse na formulação do problema, dispensando os mapas anteriormente citados. A observação de uma “mesa”, “bancada” ou outro aparelho eletrônico na cena podem indicar ao agente uma maior probabilidade de proximidade ao objeto-alvo, fazendo com que aquele se direcione para estas regiões e possa encontrar o “telefone celular” mesmo sem ter observado este no início da busca.

Deste modo, no presente trabalho estamos interessados em tornar um agente robótico capaz de navegar em ambientes internos de residências em uma abordagem objeto-direcionada. Usando visão por meio capturas de imagens do ambiente, posteriormente processadas por modelos de redes neurais convolucionais, e obtendo informações semânticas de contexto utilizando conhecimentos *a priori* e redes neurais de grafos, um agente deve aprender políticas de navegação eficientes em uma abordagem de navegação baseada em aprendizado, adotando técnicas de aprendizado por reforço profundo.

Nesta dissertação, uma arquitetura de navegação visual semântica com memória é proposta. Buscamos propor uma arquitetura de navegação capaz de aprender uma política de navegação eficiente, em uma abordagem de navegação baseada em aprendizado, não utilizando informações posicionais e, desta forma, dispensando o uso de mapas explícitos dos cenários. A arquitetura utiliza navegação visual para sensoriamento no ambiente, utilizando como entrada capturas da cena, posteriormente processadas por técnicas de redes neurais convolucionais. Não obstante, o modelo é acrescido com informações contextuais semânticas por meio do uso de redes neurais de grafos, inferindo como objetos se relacionam e enriquecendo o modelo com interpretações destas relações. Para o aprendizado das políticas de navegação, adotamos aprendizado por reforço profundo atrelado ao uso de ambientes simulados fotorrealistas.

Partindo da arquitetura base proposta em [Yang et al. \(2019\)](#), propomos realizar um trabalho de caráter incremental, estendendo o modelo base com a adição de um módulo de memória, valendo-se de redes neurais recorrentes. A hipótese investigada neste trabalho é que a navegação visual semântica, em um contexto de navegação baseada em

aprendizado valendo-se de aprendizado por reforço profundo, pode se beneficiar pelo uso de mecanismos de memória, enriquecidas pela memorização de experiências vivenciadas, considerando informações de recorrências e relações entre objetos, execuções e resultados obtidos em explorações anteriores.

Para tanto, diferentes arquiteturas de redes neurais recorrentes, como *Gated Recurrent Unit* (GRU) e *Long Short-Term Memory* (LSTM), foram investigadas e analisadas, sendo comparadas entre si e também com modelos desprovidos de memória, por meio da realização de testes qualitativos e quantitativos, utilizando métricas adotadas por trabalhos correlatos.

Almeja-se uma arquitetura de navegação visual semântica utilizando métodos de visão computacional, memória e compreensão semântica para possibilitar que um robô navegue de maneira inteligente em ambientes complexos, utilizando técnicas de aprendizagem profunda, combinados com métodos de aprendizado por reforço profundo em ambientes simulados.

As principais contribuições do nosso trabalho são a extensão de modelos arquiteturais com a inclusão de mecanismos de memória em abordagens de navegação visual semântica baseada em aprendizado, comparativos do nosso trabalho com outros sem a presença de memória e, por fim, análise comparativa entre os modelos de redes recorrentes GRU e LSTM em um problema de navegação complexo utilizando informação visual e semântica.

1.1 Organização do trabalho

No [Capítulo 2](#) são apresentadas as fundamentações teóricas pertinentes as principais técnicas e métodos relacionados a este trabalho. Redes neurais artificiais são discutidas, em específico sua concepção por arquiteturas profundas, com ênfase em suas aplicações e modelos relacionados a visão e processamento de dados por meio de grafos. Também são discutidos modelos de redes neurais capazes de lidar com dados sequenciais, possuindo característica de memorização. Além disso, métodos de aprendizado por reforço, algoritmos e sua abordagem conjunta com estruturas de aprendizagem profundas são apresentados.

No [Capítulo 3](#), alguns dos principais trabalhos relacionados a esta proposta são descritos. Os trabalhos são organizados em seções, de acordo com sua maior adesão a determinadas áreas ou aplicações, sendo interpretações de cenas, navegação visual e navegação semântica. Um comparativo entre estes trabalhos é apresentado, discutindo-se pontos de interesse e levantadas algumas lacunas existentes.

O [Capítulo 4](#) é dedicado a apresentação da proposta desta Dissertação. Neste

capítulo, descreve-se a lacuna existente a ser explorada e o que se espera obter a partir do desenvolvimento da proposta. A arquitetura proposta é descrita em detalhes em seus componentes, divididos em módulo visual, semântico, de memória e aprendizado. Alguns ambientes de simulação disponíveis atualmente são também apresentados e é justificada a opção pelo simulador selecionado. Técnicas e recursos que serão utilizados são discutidos, sendo seguidos pelo detalhamento da tarefa a ser executada pelo agente e as métricas adotadas para avaliação.

No [Capítulo 5](#) são descritos os experimentos realizados, com detalhes sobre a configuração do ambiente, recursos computacionais utilizados e configurações de treinamento. Ainda, são apresentados e discutidos os resultados obtidos, por meio de análises quantitativas e qualitativas, com gráficos coletados durante o treinamento dos diversos modelos considerados, tabela comparativa avaliando as métricas adotadas e resultados qualitativos da execução do agente no ambiente simulado de maneira visual. São realizadas observações sobre o comportamento dos modelos arquiteturais e suas limitações.

Por fim, no [Capítulo 6](#) são realizadas as conclusões deste trabalho, apresentando uma sumarização do conteúdo apresentado, dos resultados obtidos e perspectivas de trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos fundamentais a respeito de métodos e técnicas essenciais para a compreensão dos trabalhos relacionados e do presente trabalho. Inicia-se com uma descrição sobre redes neurais profundas e, de maneira detalhada, sobre redes neurais convolucionais, redes neurais recorrentes e redes neurais de grafos. Posteriormente, serão apresentados conceitos sobre aprendizado por reforço, seus algoritmos clássicos e sua utilização conjuntamente com redes neurais profundas, campo conhecido por aprendizado por reforço profundo.

2.1 Redes neurais profundas

As Redes Neurais Artificiais (RNAs) são modelos computacionais pertencentes a área de aprendizado de máquina. Essas redes surgiram por meio da inspiração biológica do funcionamento dos neurônios cerebrais e a maneira como estes processam informações e realizam o aprendizado. O conceito inicial de neurônio artificial foi proposto por [McCulloch e Pitts \(1943\)](#) como um modelo de circuitos elétricos manualmente modulados, tendo melhorias subsequentes realizadas por diversos pesquisadores, como [Hebb \(1949\)](#), com a primeira regra para ajuste automático dos pesos dos neurônios, [Rosenblatt \(1958\)](#) pela proposição da estrutura do *Perceptron*.

Os *Perceptrons* eram limitados a resolução de problemas linearmente separáveis, levando ao desenvolvimento posterior das redes de múltiplas camadas, como as *Multilayer Perceptrons* (MLP), estruturas compostas por n *perceptrons*, podendo estar distribuídos em diversas camadas, e com capacidade para lidar com problemas não-linearmente separáveis.

Desde então, as redes neurais artificiais vêm desenvolvendo-se juntamente com a expansão dos dados disponíveis para treinamento destas e com o aumento de recursos

e poder computacional. Esses desenvolvimentos foram propulsores para um novo campo de aplicações e modelos de redes neurais profundas, com alta quantidade de camadas e neurônios, o qual denominou-se Aprendizagem Profunda (AP) (do inglês *Deep Learning*).

Alguns modelos de AP se tornaram populares pelo seu bom desempenho e alta aplicabilidade em problemas reais, como é o caso das redes neurais convolucionais (do inglês *Convolutional Neural Networks* (ConvNets)), as quais lidam com dados topológicos como imagens, vídeos e áudio, e redes neurais recorrentes (do inglês *Recurrent Neural Networks* (RNNs)), utilizadas com dados sequenciais para tarefas de previsão. As ConvNets e RNNs são descritas mais detalhadamente a seguir.

2.1.1 Redes neurais convolucionais

As ConvNets são redes neurais convolucionais propostas por [LeCun et al. \(1998\)](#). Inicialmente, com o modelo LeNet, demonstrado na [Figura 1](#), foi utilizada para reconhecimento de escrita manual de símbolos numéricos em documentos. As ConvNets receberam notoriedade recente por resultados como os de [Krizhevsky, Sutskever e Hinton \(2012\)](#) na competição ILSVRC, para classificação de objetos presentes em imagens, alcançando o então estado da arte.

Figura 1 – Representação da arquitetura da rede LeNet

Fonte: Adaptada de [LeCun et al. \(1998\)](#).

As ConvNets são assim chamadas por aplicarem em suas camadas a operação de convolução, dada pela [Equação 2.1](#):

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n) K(m, n) \quad (2.1)$$

onde **I** e **S** são, respectivamente, uma imagem de entrada e de saída, e **K** representa um filtro bi-dimensional ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)). O $*$ é utilizado para indicar a operação de convolução.

Diversos filtros podem ser aplicados por camada convolucional, e representam os pesos dos neurônios dessas camadas. Assim como nas MLPs, os neurônios também possuem um valor de viés (bias) e uma função de ativação.

As ConvNets têm como uma de suas principais características o aprendizado hierárquico de representações. No caso de imagens, por exemplo, as primeiras camadas são responsáveis pelo aprendizado de padrões básicos e simples, como linhas horizontais, verticais, contraste entre cores e diferentes texturas. A medida que avançamos para as camadas mais profundas, a rede aprende padrões mais elaborados como formas geométricas, faces, objetos, dentre outros.

Algumas características são comuns as ConvNets: os campos receptivos locais, o compartilhamento de pesos e a subamostragem espacial (LECUN *et al.*, 1998). Os campos receptivos locais são caracterizados por neurônios possuírem conectividade apenas com uma determinada região da camada anterior, seja essa os dados de entrada ou uma camada de neurônios. Os campos receptivos locais proporcionam, como citado anteriormente, a extração de características específicas para cada local, que são posteriormente combinadas pelas camadas seguintes, resultando em representações mais complexas. As saídas dos neurônios em cada camada resultam nos mapas de características, e todos os neurônios pertencentes espacialmente a um mesmo mapa de características compartilham do mesmo conjunto de pesos. Por fim, os resultados dos mapas de características podem ser posteriormente sumarizados por meio de subamostragens espaciais realizadas pelas camadas de *pooling*. As camadas de *pooling* são aplicadas para reduzir a dimensionalidade do mapa de características e, conjuntamente a isto, reduzir a sensibilidade do modelo a distorções e ruídos, preservando as informações que são mais relevantes e provendo maior generalidade (LECUN *et al.*, 1998; GOODFELLOW; BENGIO; COURVILLE, 2016). Na Figura 2, é ilustrado o funcionamento dos campos receptivos locais e do compartilhamento de pesos.

Diferentes arquiteturas de ConvNets foram propostas ao longo dos últimos anos, com destaque para a VGGNet (SIMONYAN; ZISSERMAN, 2014), GoogleLeNet/Inception (SZEGEDY *et al.*, 2015) e a Rede Residual (em inglês, *Residual Network* (ResNet)) (HE *et al.*, 2016). Esta última é discutida em detalhes a seguir.

2.1.2 Rede Residual

A Rede Residual (ResNet) foi proposta por He *et al.* (2016). Assim como outras arquiteturas que a antecederam, a ResNet é constituída por diversas camadas convolucionais e possui um mecanismo de aprendizado semelhante. As ResNets se diferenciam dos demais modelos de redes de AP pela utilização de representações residuais e conexões, como uma espécie de atalho, entre suas camadas convolucionais. Adicionalmente, as camadas convolucionais das ResNets têm como resultado um mapeamento residual dado

Figura 2 – Detalhe dos campos receptivos locais e compartilhamento de pesos presente em uma ConvNet

Fonte: Adaptada de [Goodfellow, Bengio e Courville \(2016\)](#).

pela [Equação 2.2](#):

$$y = \mathcal{F}(x, \{W_i\}) + x. \quad (2.2)$$

onde y é a saída e $\mathcal{F}(x, \{W_i\})$ representa o mapeamento residual a ser aprendido, o qual pode aplicar uma função do tipo ReLU, e x é um mapeamento residual realizado previamente e obtido por meio da conexão atalho. Na [Figura 3](#), pode-se visualizar três modelos arquiteturais comparados lado a lado: uma VGGNet com 19 camadas, uma rede plana com 34 camadas e por fim, uma rede residual, também com 34 camadas. Nesta última pode-se perceber a existência de outras conexões, que não as tradicionais observadas em outras arquiteturas de redes AP. Tais conexões características das redes residuais são as conexões atalhos e, no modelo demonstrado, ocorrem de maneira intervalar entre camadas. Na [Figura 4](#) é ilustrado este mecanismo em detalhes.

A motivação para a proposição das ResNets se deve a observação de que a adição de mais camadas em redes AP não estava resultando em melhores resultados do que seus modelos mais rasos, e até mesmo tendo desempenho pior em alguns casos. Atribuiu-se tais observações ao problema do desaparecimento dos gradientes (*vanishing gradients*), ocasionado pelas sucessivas multiplicações ao longo das camadas durante as correções realizadas pelo processo de *backpropagation*, tornando o gradiente infinitamente pequeno e saturando o aprendizado ou degradando-o rapidamente. Os autores alegam que, por meio das conexões atalho e a utilização de mapeamentos identidade, o problema da degradação do gradiente seria atenuado, acarretando que o modelo aprenda uma função identidade

Figura 3 – Modelos arquiteturais de uma VGGNet com 19 camadas, de uma rede plana com 34 camadas e de uma ResNet com 34 camadas

Fonte: Adaptada de [He et al. \(2016\)](#).

Figura 4 – Detalhe da estrutura de um bloco de aprendizado residual

Fonte: Adaptada de [He et al. \(2016\)](#).

que não apresenta erro maior que sua camada anterior.

Em 2015, a ResNet alcançou as primeiras colocações nas competições *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) ([RUSSAKOVSKY et al., 2015](#)) e *Microsoft Common Objects in COntext* (MS COCO) ([LIN et al., 2014](#)), nos desafios de detecção e localização. Desde então, essa arquitetura vêm sendo aplicada em diversos trabalhos para classificação e reconhecimento de objetos.

2.1.3 Transferência de aprendizado

Os bons resultados obtidos por meio das redes profundas são alcançados, não raramente, por meio de grandes bases de dados e extensas e custosas (em tempo e em recursos computacionais) rotinas de treinamento. Como forma de minimizar tais problemas, comumente se realiza transferência de aprendizado, onde os pesos de uma nova rede são iniciados a partir dos valores obtidos pelo processo de treinamento concluído anteriormente em uma rede de arquitetura semelhante. Esta técnica pode ser aplicada de formas diferentes como, por exemplo, com todo o retreino de todos os pesos a partir dos valores pré-treinados; com o congelamento apenas de algumas camadas enquanto outras são totalmente substituídas por pesos com valores randômicos a serem treinados; ou ainda de maneira onde pesos de algumas camadas sejam congelados enquanto outras camadas tenham seus pesos refinados por meio do treino a partir dos pesos pré-treinados. Desta forma, um treinamento posterior pode ser realizado visando apenas um ajuste fino dos pesos para o problema atual.

Esta técnica é bastante adotada com ConvNets, onde costuma-se realizar a transferência de aprendizado de um treinamento anterior, geralmente congelando os pesos das

camadas iniciais (responsáveis por características mais simples e genéricas, de fácil aprendizado) e refinando apenas os pesos das camadas finais, alinhando-os ao problema em questão.

Os modelos de redes neurais apresentados anteriormente, como as MLPs e Conv-Nets, não possuem estruturas capazes de realizar o aprendizado considerando informações que são correlacionadas de maneira temporal. Para lidar com dados sequenciais podem ser utilizadas RNNs, descritas a seguir.

2.1.4 Redes neurais recorrentes

As Redes Neurais Recorrentes, em sua estrutura mais simples, podem ser comparadas a uma rede MLP que possuem ligações cíclicas em sua estrutura, permitindo assim que, em princípio, uma RNN possa mapear toda a história de entradas processadas, utilizando informações passadas anteriormente para prever saídas futuras (GRAVES, 2012).

Na Figura 5 é mostrada a representação de uma RNN de maneira recursiva ou como um grafo desmembrado. O modelo recebe um vetor x como entrada, o qual é parametrizado por um conjunto de pesos W_{xh} para uma unidade h . Um segundo conjunto de pesos W_{hh} parametriza relações de recorrência entre unidades h , e por fim, um terceiro conjunto de pesos W_{oh} parametriza os resultados da unidade h para uma saída y .

Na representação em forma de grafo desmembrado é possível visualizar mais claramente a característica de processamento temporal da RNN. A entrada x pode também ser genericamente representada como uma estrutura de dados sequenciais, x^t , sendo t o valor de tempo compreendido entre 1 e τ . Observa-se também a mesma relação temporal entre o resultado da unidade h . Portanto, uma saída y^t é resultado da computação em h^t , dada uma entrada atual x^t e o resultado de uma computação anterior h^{t-1} .

Figura 5 – Representação de uma RNN. A esquerda, uma representação recursiva, e a direita, como um grafo desmembrado

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

O cálculo resultante da unidade h_t é dado pela [Equação 2.3](#):

$$h^t = f(W_{hh}h^{t-1} + W_{xh}x^t) \quad (2.3)$$

onde f é uma função de ativação desejada, e W_{hh} e W_{xh} pesos para o estado anterior h_{t-1} e entrada atual x_t , respectivamente ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)).

Um problema enfrentado pelas arquiteturas de RNNs mais simples, a exemplo da demonstrada anteriormente, é a dificuldade em guardar informações durante longos períodos. Ao realizar a correção dos pesos, observa-se uma tendência destes valores a desaparecerem ou explodirem, de maneira exponencial, após a repetida execução deste procedimento, problema conhecido como *vanishing gradients* ([GRAVES, 2012](#)). O resultado do desaparecimento ou explosão destes valores é a perda de informações, principalmente aquelas apresentadas no início das sequências, prejudicando assim o treinamento da rede. Uma maneira de minimizar o problema no que se refere a explosão é realizar um corte heurístico dos valores de ajuste dado um limiar. No entanto, o problema de desaparecimento com a adoção apenas desta medida pode persistir.

Formas de lidar com memórias de longo prazo foram sugeridas, atacando principalmente o problema do desaparecimento. Estas abordagens propuseram novas maneiras de conectar neurônios, criando atrasos temporais, ou novas unidades para o armazenamento e controle do fluxo de informações. A criação de atrasos temporais é realizada estabelecendo conexões entre unidades com maiores intervalos temporais. Normalmente, uma unidade no tempo t se conecta a outra com um atraso temporal de $t + 1$. Aumentando-se esse intervalo é possível influenciar unidades no tempo presente com informações de um passado distante. Essa abordagem foi proposta por [Lin et al. \(1996\)](#).

Outra medida é a adoção de unidades *leaky* ou constantes de tempo, propostas por [Mozier \(1992\)](#). As unidades *leaky* funcionam aplicando auto-conexões lineares para uma medida $\mu^{(t)}$, que é atualizada como demonstrado na [Equação 2.4](#). Quando α é um valor aproximado de 1, informações passadas são mais influentes no resultado de $\mu^{(t)}$, enquanto que dada a ocorrência inversa, isto é, α ser um valor mais próximo de 0, influências passadas passam a ser rapidamente descartadas. Assim como a aplicação de atrasos temporais, o uso das unidades *leaky* também permitem o acesso a valores do passado, sendo uma abordagem que oferece ajustes de maneira mais flexível e suave por meio do controle do valor α .

$$\mu^{(t)} = \alpha\mu^{(t-1)} + (1 - \alpha)v^{(t)} \quad (2.4)$$

Atualmente, os modelos de RNNs LSTM, traduzido livremente como Memória Longa de Curto-Prazo, e GRU, estão dentre aqueles que apresentam maior eficiência, lidando bem com problemas como o desaparecimento ([GOODFELLOW; BENGIO; COUR-](#)

(CEC), e uma função de ativação de saída (GOODFELLOW; BENGIO; COURVILLE, 2016). Cada unidade *gate* possui uma unidade multiplicativa. O bloco LSTM, assim como ocorre em outros modelos de RNN, possui dados de entrada e de saída, os quais são distribuídos para todo o bloco. Informações de dentro e de fora do bloco são coletadas pelas três unidades multiplicativas, que controlam o fluxo destas. O fluxo de informações no bloco LSTM ocorre em duas direções: informações dos *gates* que fluem destes em direção a CEC, sendo reguladas por meio das unidades multiplicativas; e informações que fluem do CEC para os *gates*, essas sendo acessadas diretamente por meio de conexões chamadas de *peepholes*.

As unidades *input gate* e *output gate* são responsáveis pelo controle da informação de entrada e saída do bloco LSTM, respectivamente. A unidade *forget gate* realiza o controle do fluxo de informações interiores do bloco LSTM. As fórmulas para cada um dos *gates* e de saída do bloco para a célula são apresentadas na Equação 2.5:

$$\begin{aligned}
 \vec{z}^t &= W_z x^t + R_z y^{t-1} + b_z \\
 z^t &= g(\vec{z}^t) && \text{entrada do bloco} \\
 \vec{i}^t &= W_i x^t + R_i y^{t-1} + b_i \\
 i^t &= \sigma(\vec{i}^t) && \text{input gate} \\
 \vec{f}^t &= W_f x^t + R_f y^{t-1} + b_f \\
 f^t &= \sigma(\vec{f}^t) && \text{forget gate} \\
 c^t &= z^t \odot i^t + c^{t-1} \odot f^t && \text{célula} \\
 \vec{o}^t &= W_o x^t + R_o y^{t-1} + b_o \\
 o^t &= \sigma(\vec{o}^t) && \text{output gate} \\
 y^t &= h(c^t) \odot o^t && \text{saída do bloco}
 \end{aligned} \tag{2.5}$$

onde W_z , W_i , W_f e W_o são os pesos das unidades, R_z , R_i , R_f e R_o os pesos das recorrências, e b_z , b_i , b_f e b_o os pesos dos bias.

Adicionalmente, temos que σ , g e h são funções de ativação não-lineares, sendo σ função logística sigmoide, e g e h funções tangente hiperbólica. Multiplicações entre vetores são simbolizados por $\odot$ (GREFF *et al.*, 2016). Nestas equações foram suprimidas as conexões do tipo *peephole* para fins de simplificação.

É válido de novo mencionar que os valores de pesos e das unidades multiplicativas são aprendidos ao longo do treinamento da rede. Sendo assim, o modelo elimina complexidade ocasionada pelo uso de hiper-parâmetros.

2.1.6 Gated Recurrent Unit

Proposta por [Cho et al. \(2014\)](#), a GRU tem estrutura e comportamento semelhantes a rede LSTM, utilizando *gates* para controlar o fluxo de informações sendo propagados pela rede. Contudo, as GRU apresentam uma estrutura mais simples de serem computadas e implementadas, a medida que o controle do fluxo de informações é realizado apenas por dois *gates*, respectivamente, *reset* e *update gates*.

Considerando o processo de ativação para a j -*enésima* camada escondida, o *reset gate* é computado como:

$$r_j = \sigma([W_r x]_j + [U_r h_{<t-1>}]_j) \quad (2.6)$$

onde σ é uma função logística sigmoide, $[\cdot]_j$ representa o j -*enésimo* elemento de um vetor, x e $h_{<t-1>}$, respectivamente, entrada atual e computação da camada oculta anterior. Por fim, W_r e U_r são matrizes de pesos a serem aprendidas ([CHO et al., 2014](#)). O *update gate* é definido de maneira similar, como na equação a seguir:

$$z_j = \sigma([W_z x]_j + [U_z h_{<t-1>}]_j) \quad (2.7)$$

Finalmente, a ativação h_j é computada como:

$$h_j^{<t>} = z_j h_j^{<t-1>} + (1 - z_j) \tilde{h}_j^{<t>} \quad (2.8)$$

onde

$$\tilde{h}_j^{<t>} = \phi([W x]_j + [U(r \odot h_{<t-1>})]_j). \quad (2.9)$$

Com as formulações acima, a medida que o *reset gate* se aproxima do valor 0, o estado oculto anterior é forçosamente esquecido, sendo considerado apenas a entrada atual. O *update gate* atua controlando o quanto o fluxo de informações do estado anterior é considerado durante a execução atual. Em suma, unidades que tendem a ter memórias de curto-prazo apresentam *reset gates* mais acionados que aquelas que apresentam comportamento mais voltado ao longo prazo ([CHO et al., 2014](#); [CHUNG et al., 2014](#)).

2.2 Redes neurais de grafo

Métodos de aprendizado de máquina, como redes neurais artificiais, frequentemente são utilizados para aprender funções que generalizem informações a partir de exemplos presentes em conjuntos de dados estruturados. Em muitos casos, esses dados podem

ser representativamente estruturados por meio de grafos, como árvores, grafos cíclicos ou acíclicos.

A rede neural de grafo (em inglês *Graph Neural Network* (GNN)) foi proposta em Gori, Monfardini e Scarselli (2005) e posteriormente em Scarselli *et al.* (2008) no formato de arquiteturas de redes recorrentes, tendo por objetivo a concepção de uma rede capaz de lidar com grafos de maneira a preservar suas informações topológicas e utilizar um algoritmo de aprendizado. Diferentemente de outros métodos de aprendizado de máquina que, ao lidarem com grafos, submetem os dados a etapas de préprocessamento, que realizam transformações onde informações importantes (e.g., relações topológicas) podem ser perdidas, as GNNs são capazes de lidar com tais dados sem efetuar procedimentos de préprocessamento. Adicionalmente, a dispensa desses procedimentos evita o condicionamento dos dados as abordagens respectivamente adotadas. O aprendizado realizado por uma GNN objetiva aprender um estado $x_n \in \mathbb{R}^s$, contendo toda informação sobre a vizinhança para um dado nó n . A Figura 7 apresenta uma representação gráfica dos nós e suas arestas.

Figura 7 – Representação de uma GNN

Fonte: Adaptada de Scarselli *et al.* (2008).

A representação do conceito no estado x_n é utilizada para produzir uma saída o_n , sendo essa uma decisão a respeito do conceito de n como, por exemplo, sua classe. Podemos aprender o estado x_n por meio de uma função paramétrica f , a qual chamamos de função de transição local, e a saída o_n por meio de uma função de saída local g . Assim

sendo, x_n e o_n podem ser definidos de acordo com a [Equação 2.10](#):

$$\begin{aligned} x_n &= f(l_n, l_{co[n]}, x_{ne[n]}, l_{ne[n]}) \\ o_n &= g(x_n, l_n) \end{aligned} \quad (2.10)$$

onde l_n , $l_{co[n]}$, $x_{ne[n]}$ e $l_{ne[n]}$ são, respectivamente, características de n , as características de suas arestas, os estados dos nós-vizinhos de n , e as características dos nós em sua vizinhança ([SCARSELLI et al., 2008](#)). Como definida anteriormente, as funções f e g podem ser parametrizadas por um conjunto de pesos dependentes a um determinado nó n .

Também é possível representar f e g de modo que todos os nós compartilhem de uma mesma implementação. Para tanto, definimos a representação empilhada de todos os estados, todas as saídas, todas as classes, e todas as classes de nós como sendo x , o , l , l_N , respectivamente, bem como uma função de transição global F e uma função de saída global G . Reescrevendo a [Equação 2.10](#), temos ([SCARSELLI et al., 2008](#)):

$$\begin{aligned} x &= F(x, l) \\ o &= G(x, l_N). \end{aligned} \quad (2.11)$$

O procedimento iterativo adotado para a computação do estado é como apresentado na [Equação 2.12](#), na qual $x^{(t)}$ refere-se a iteração no tempo t de x . O sistema converge com velocidade exponencial para uma solução a partir de um valor inicial qualquer $x^{(0)}$. Logo, $x^{(t)}$ é o estado por meio da atualização da função paramétrica F , a qual é realizada utilizando gradiente descendente, de maneira a minimizar uma taxa de erro e de uma função de custo quadrática dada na [Equação 2.13](#):

$$x^{(t+1)} = F(x^t, l) \quad (2.12)$$

$$e = \sum_{i=1}^p (t_i - o_i)^2 \quad (2.13)$$

onde p é o número de nós supervisionados do conjunto de dados onde é conhecida sua classe.

As GNNs possuem aplicações em diversas áreas, como análise de imagens, descrição de cenas, redes de interações proteína-proteína, aplicações em ciências naturais e ciências sociais (como redes sociais), grafos de conhecimento, entre outros ([WU et al., 2020](#); [YANG et al., 2018](#); [YAO et al., 2018](#); [SÜNDERHAUF, 2019](#)). Desde sua proposição, essas redes vêm sendo exploradas por diferentes técnicas de aprendizado de máquina e modelos de RNAs. Uma variação de GNN de interesse para o presente trabalho é a Rede Convolutiva de Grafo (em inglês *Graph Convolutional Network* (GCN)).

2.2.1 Rede Convolutacional de Grafo

A GCN foi proposta por Kipf e Welling (2016). Neste trabalho, os autores consideraram uma GCN como um modelo de rede neural de grafo com $f(X, A)$, provendo motivações teóricas e seguindo uma regra de propagação em camadas como segue:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \quad (2.14)$$

onde, $\tilde{A} = A + I_N$ é a matriz adjacente de um grafo não direcional acrescido de auto-ligações, I_N a matriz identidade, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ e $W^{(l)}$ uma matriz de pesos de uma determinada camada. Uma função ativação é representada por $\sigma(\cdot)$, a exemplo de $ReLU(\cdot) = \max(0, \cdot)$. Por fim, $H^{(l)} \in \mathbb{R}^{N \times D}$ é a matriz de ativações na camada l , onde $H^{(0)} = X$. Os autores utilizam-se de aproximações, descritas ao longo do trabalho, para promover maior eficiência ao modelo e permitir a computação de grafos de maior magnitude, como aproximações de primeira ordem de filtros espectrais localizados em grafos.

As convoluções espectrais em grafos são definidas como multiplicações de um sinal $x \in \mathbb{R}^N$ com um filtro $g_\theta = \text{diag}(\theta)$ parametrizado por $\theta \in \mathbb{R}^N$ em um domínio de Fourier, tal como:

$$g_\theta \star x = U g_\theta U^\top x \quad (2.15)$$

sendo U uma matriz de autovetores de um grafo Laplaciano normalizado $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$, com a matriz diagonal de autovalores Λ e $U^\top x$ sendo a transformada de Fourier do grafo x e g_θ como uma função dos autovalores de L . Contudo, a computação da Equação 2.15 é computacionalmente custosa, tendo complexidade $\mathcal{O}(N^2)$, e cuja decomposição dos autovalores torna-se proibitiva para grafos grandes. Os autores então adotam medidas sugeridas em Hammond, Vandergheynst e Gribonval (2011), com a utilização de polinômios de Chebyshev, obtendo-se ao final a Equação 2.16

$$g_{\theta'} \star x \approx \sum_{k=0}^K \theta'_k T_k(\tilde{L}) x \quad (2.16)$$

em que $\tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$, sendo $\lambda_{\max}$ o maior autovalor de L e K sendo a máxima distância dos nós em relação ao nó central. Desta forma, obteve-se complexidade $\mathcal{O}(|\mathcal{E}|)$, linear em relação ao número de vértices.

A partir da Equação 2.16 é possível construir uma rede neural de grafos utilizando convoluções em suas camadas. Pode-se assim, considerando $\lambda_{\max} = 2$ e apenas um conjunto de parâmetros livres compartilhados por toda a arquitetura, simplificar a Equação 2.16 para:

$$g_{\theta} \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right). \quad (2.17)$$

Tendo um sinal $X \in \mathbb{R}^{N \times C}$ com C canais de entrada e F filtros ou mapas de características, pode-se generalizar como:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \quad (2.18)$$

onde $\Theta \in \mathbb{R}^{C \times F}$ é uma matriz de filtros paramétricos e $Z \in \mathbb{R}^{N \times F}$. A complexidade final então torna-se $\mathcal{O}(|\mathcal{E}|FC)$, podendo $\tilde{A} \tilde{D}^{-\frac{1}{2}}$ ser eficientemente calculado como o produto de uma matriz esparsa com uma matriz densa.

Finalmente, a partir do desenvolvimento demonstrado acima podemos condensar os conceitos apresentados no formato da [Equação 2.19](#), tendo um conjunto de dados X e uma matriz de adjacência simétrica A . Nesta, $W^{(0)} \in \mathbb{R}^{C \times H}$ representa uma matriz de pesos para uma camada escondida com H mapas de características, e $W^{(1)} \in \mathbb{R}^{H \times F}$ um conjunto de pesos na última camada escondida para a camada de saída, sendo aplicado *softmax* ao final e função de ativação ReLU na camada intermediária.

$$Z = f(X, A) = \text{softmax} \left(\hat{A} \text{ReLU} \left(\hat{A} X W^{(0)} \right) W^{(1)} \right) \quad (2.19)$$

2.3 Aprendizado por reforço

O Aprendizado por Reforço (AR) (do inglês *Reinforcement Learning*) é uma abordagem de aprendizado por meio de interações em um dado ambiente, as quais são direcionadas a um objetivo. Um problema de AR pode ser genericamente descrito como um agente que em determinado tempo t , recebe um estado s_t representativo do ambiente e que pertence a um conjunto de estados $\mathcal{S}$, no qual efetua uma ação a_t pertencente a um conjunto de ações $\mathcal{A}(s)$. A cada ação tomada, o agente recebe uma recompensa $R_{(t+1)}$, bem como o estado subsequente s_{t+1} de acordo com a dinâmica do ambiente. O objetivo do agente é maximizar o valor acumulado de R obtido ao longo do aprendizado ([SUTTON; BARTO, 2018](#)).

Na configuração apresentada anteriormente, um problema de AR pode ser matematicamente formulado como um Processo de Decisão de Markov (PDM) finito, sendo uma formulação clássica para tomadas de decisões sequenciais. Em consequência disso, pode-se estabelecer que problemas de aprendizado consequentes possuam seus conjuntos de estados, ações e recompensas pertencentes a um número finito de elementos ([SUTTON; BARTO, 2018](#); [RUSSELL; NORVIG, 2016](#)).

Ainda, pode-se considerar que, a partir de um estado atual s_t , todo o universo de possíveis valores para s_t e R_t está presente em s_{t-1} e a_{t-1} . Partindo desta premissa, s_t e R_t são definidos exclusivamente pelo estado e ação imediatamente anterior, contendo estes toda informação relevante a respeito das interações agente-ambiente passadas, consistindo assim em uma propriedade de Markov. Pode-se assumir que todos os valores possíveis de s_t e R_t estejam definidos em uma distribuição de probabilidade discreta dependente apenas de s_{t-1} e a_{t-1} , de acordo com a [Equação 2.20](#) (SUTTON; BARTO, 2018; RUSSELL; NORVIG, 2016):

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\} \quad (2.20)$$

Como dito anteriormente, num estado s_t o agente pode executar uma ação a_t pertencente a um conjunto $\mathcal{A}(s)$. Essa ação é tomada a partir de uma política $\pi(a_t|s_t)$. Uma política é definida como um mapeamento de probabilidades de selecionar uma determinada ação a dado um estado s , buscando estimar o quão bom é o resultado proveniente dessa execução, ao que podemos nos referir como função de valor (também conhecidas como *Value Function*). Uma função de valor $v_\pi(s)$ é o retorno esperado ao se executar uma política π a partir de s em diante, podendo ser definida como na [Equação 2.21](#) nos casos de PDMs, onde R é o valor de recompensa e γ é um valor de desconto.

$$v_\pi(s) \doteq \mathbb{E}[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \quad (2.21)$$

Valendo-se da propriedade recursiva que a [Equação 2.21](#) pode apresentar dada sua aplicação em problemas de AR, podemos formulá-la como uma Equação de *Bellman* 2.22. A Equação de *Bellman* representa a relação existente entre um estado e seus sucessores, realizando um balanceamento entre todas as possibilidades e atribuindo pesos respectivos a probabilidade de cada uma delas ocorrerem.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma v_\pi(s')] , \text{ para todo } s \in \mathcal{S} \quad (2.22)$$

Como anteriormente citado, um método de AR tem como objetivo a maximização do valor R acumulado ao longo de sua execução. Uma política deve, desta forma, perseguir tal objetivo, onde uma política ótima π^* é equivalente a $\max_\pi v_\pi(s)$. Conjuntamente, pode-se adotar a representação de uma função ação-valor $q_\pi(s, a)$ que define o valor de retorno esperado dada a tomada de uma ação a em um estado s e seguindo-se posteriormente a política π . Tal como para a função de valor, uma função ação-valor ótima $q_*(s, a)$ é definida como aquela que resulta no valor de retorno de recompensas máximo (SUTTON;

BARTO, 2018). Podemos definir $v_*(s)$ e $q_*(a, s)$ como, respetivamente, as Equação 2.23 e Equação 2.24.

$$v_*(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')] \quad (2.23)$$

$$q_*(a, s) = \max_a \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \quad (2.24)$$

Algoritmos de AR podem utilizar a política de maneira *On-policy* ou *Off-policy*. As abordagens se diferenciam pela forma como uma política é utilizada e atualizada, de maneira conjunta ou não. No caso dos algoritmos de AR que utilizam uma política *On-policy*, as ações são tomadas de acordo com uma política π , ao mesmo tempo que essa é atualizada por meio de q_π . Um algoritmo *On-policy* clássico é o algoritmo *Sarsa*. Já em abordagens que adotam uma política *Off-policy*, a política utilizada para tomada de ações se diferencia da política aplicada para o aprendizado. Um algoritmo clássico com esta abordagem é o *Q-Learning*.

2.3.1 Q-Learning

O algoritmo *Q-Learning* foi proposto por Watkins e Dayan (1992). Neste algoritmo, uma função ação-valor Q é utilizada para seleção da ação, dado um determinado estado seguindo uma política atual. No entanto, sua atualização se vale de uma política ótima, aproximando assim sua função ação-valor Q de uma função ação-valor ótima q_* . A atualização da função $Q(S_t, A_t)$ é definida como segue:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (2.25)$$

Desta forma, a convergência do algoritmo em direção a q_* é garantida desde que todos os pares estado-ação sejam visitados (o que também é requerido para outras abordagens) (SUTTON; BARTO, 2018). Como citado anteriormente, por utilizar uma política gulosa $\max_a Q(S_{t+1}, a)$ durante o aprendizado, enquanto adota uma política $Q(S_t, A_t)$ para a real seleção da ação tomada, tem-se um comportamento de maneira *Off-policy*. O método *Q-Learning* segue o fluxo apresentado no Algoritmo 1 (SUTTON; BARTO, 2018).

O algoritmo *Q-Learning* recebe como parâmetro um valor α , referente ao tamanho do passo, pertencente ao intervalo $(0, 1]$. Também recebe um pequeno valor $\epsilon > 0$. Esse último representa uma medida exploratória para o algoritmo, permitindo que uma determinada quantidade de ações sejam selecionadas aleatoriamente e não de acordo com a política π . Ocasionalmente, opções menos vantajosas de imediato são exploradas. Tais ações podem levar a descobertas que acabam por maximizar a recompensa posteriormente.

Algoritmo 1 – Algoritmo Q-Learning

```

1: procedimento Q-LEARNING( $\alpha, \epsilon$ )      ▷ Valores de tamanho do passo e  $\epsilon$ -greedy
2:
3:   Inicializa  $Q(s, a)$  para todos os  $s \in \mathcal{S}^+$  e  $a \in \mathcal{A}(s)$  de maneira arbitrária, com exceção
   do estado terminal, onde  $Q(\text{terminal}, \cdot) = 0$ 
4:   para cada episódio faça
5:     Inicialize  $S$ 
6:     enquanto  $S$  não for um estado terminal faça
7:       Escolha  $A$  de  $S$  de acordo com a política derivada de  $Q$  (incluindo  $\epsilon$ -greedy)
8:       Execute ação  $A$  e observe  $R$  e  $S'$ 
9:        $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
10:       $S \leftarrow S'$ 
11:    fim enquanto
12:  fim para
13: fim procedimento

```

A questão da exploração/exploração do ambiente é uma dilema presente na área de AR (RUSSELL; NORVIG, 2016).

2.3.2 Sarsa

Outro algoritmo clássico de AR é o Sarsa, introduzido por Rummery e Niranjan (1994). O algoritmo Sarsa é semelhante ao algoritmo *Q-Learning*, diferenciando principalmente em relação a forma como realiza a atualização da sua função ação-valor. Enquanto o *Q-Learning* adota uma abordagem *Off-policy*, o Sarsa a faz por meio da abordagem *On-policy*, sendo $Q(S, A)$ atualizado da seguinte maneira:

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)] \quad (2.26)$$

Como pode ser notado, o algoritmo Sarsa utiliza sua política atual π tanto para escolha da ação a ser tomada como para a atualização por meio da estimativa de q_π . Assim como para o algoritmo *Q-Learning*, o algoritmo do Sarsa também possui os parâmetros α e ϵ , com as mesmas funções.

2.3.3 Métodos de Política por Gradiente

Ambos os algoritmos apresentados anteriormente, em sua implementação clássica, baseiam-se num mapeamento de seus estados e possíveis ações a serem executados através das chamadas *Lookup Tables*. A medida que ambientes mais complexos e desafiadores são investigados, tal mapeamento exaustivo pode se mostrar inviável e até mesmo impossível. É neste cenário que se encontram os problemas do mundo real, o que se converte em um desafio para a aplicação de técnicas de AR.

Uma maneira de contornar o problema da limitada capacidade de representação por meio das *Lookup Tables* pode ser pela utilização de funções de aproximação parametrizadas. Essas funções devem ser acessíveis ao modelo de AR e serem capazes de suprir a informação necessária em vistas do conjunto de estados e ações existentes, podendo ser manualmente definidas ou modeladas de maneira automatizada por meio de técnicas de aprendizado de máquina, como RNAs (SUTTON; BARTO, 2018; RUSSELL; NORVIG, 2016). Neste cenário, uma política π é estabelecida por meio de um vetor de valores parametrizados θ . Desta forma, podemos definir $\pi(a|s, \theta) = Pr(A_t = a|S_t = s, \theta_t = \theta)$ como a probabilidade de uma política π tomar a ação a em um determinado estado em um tempo t de acordo com o parâmetro θ (SUTTON; BARTO, 2018). Os valores do parâmetro de política θ são ajustados observando uma medida de desempenho por meio do gradiente ascendente.

Uma classe de algoritmos chamados de *REINFORCE* (WILLIAMS, 1992) são utilizados para aplicações baseadas em abordagens como as executadas por RNAs, onde uma rede tem seus pesos W ajustados em consideração de um valor de reforço r recebido ao final de uma iteração. Esses possuem uma forma particular visualizada na Equação 2.27:

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij} \quad (2.27)$$

onde α_{ij} é uma taxa de aprendizado, b_{ij} é uma valor *baseline* de reforço, e $e_{ij} = \partial \ln g_i / \partial w_{ij}$.

Uma característica interessante dos algoritmos *REINFORCE* é que cada incremento é proporcional a multiplicação do retorno obtido e o gradiente das probabilidades de executar a ação de fato efetuada, dividido pela probabilidade de tomar esta ação. A atualização aumenta o vetor de parâmetros de maneira proporcional ao retorno, e inversamente proporcional a probabilidade da respectiva ação. Esse comportamento estabelece um balanço entre mover-se em direção a ações que geram maior retorno e entre ações que possuem maiores chances de serem selecionadas repetidamente (SUTTON; BARTO, 2018). No Algoritmo 2, é apresentada uma implementação do algoritmo *REINFORCE* com *baseline* segundo Sutton e Barto (2018).

2.3.4 Actor-Critic

O algoritmo *REINFORCE* como apresentado anteriormente apresenta a desvantagem de lidar com retornos resultantes do cálculo de episódios inteiros, o que torna-o lento e inconveniente de ser implementado para problemas contínuos. Algoritmos como *Actor-Critic* aplicam a regra de atualização do *REINFORCE* de maneira modificada, introduzindo recompensas obtidas passo-a-passo, por meio de um chamado *Critic*.

O *Critic* funciona como um *baseline* dentro da função *REINFORCE*, com o detalhe de ser calculado por meio de uma função estado-valor. Esta modificação permite que o al-

Algoritmo 2 – Algoritmo REINFORCE com *baseline*

```

1: procedimento REINFORCE( $\alpha^\theta > 0, \alpha^w > 0$ )    ▷ Valores de tamanho do passo
2:
3:   Inicializa o parâmetro de política  $\theta$  e os pesos de valores-estados  $w$ 
4:   para cada episódio faça
5:     Gere um episódio  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$ , seguindo  $\pi(\cdot|\cdot, \theta)$ 
6:     para cada passo do episódio  $t = 0, 1, \dots, T - 1$  faça
7:        $G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$ 
8:        $\partial \leftarrow G - \hat{v}(S_t, w)$ 
9:        $w \leftarrow w + \alpha^w \partial \nabla \hat{v}(S_t, w)$ 
10:       $\theta \leftarrow \theta + \alpha^\theta \gamma^t \partial \nabla \ln \pi(A_t|S_t, \theta)$ 
11:    fim para
12:  fim para
13: fim procedimento

```

goritmo *Actor-Critic* seja incremental e totalmente *online*, recebendo informações quanto a seu estado, ações e recompensa a medida que as mesmas ocorrem. Uma implementação de acordo com Sutton e Barto (2018) pode ser vista no Algoritmo 3.

Algoritmo 3 – Algoritmo Actor-Critic

```

1: procedimento ACTOR-CRITIC( $\alpha^\theta > 0, \alpha^w > 0$ )    ▷ Valores de tamanho do passo
2:
3:   Inicializa o parâmetro de política  $\theta$  e os pesos de valores-estados  $w$ 
4:   para cada episódio faça
5:     Inicializa  $S$ 
6:      $I \leftarrow 1$ 
7:     enquanto  $S$  não é terminal faça
8:        $A \sim \pi(\cdot|S, \theta)$ 
9:       Execute ação  $A$ , observe  $S', R$ 
10:       $\partial \leftarrow R + \gamma \hat{v}(S', w) - \hat{v}(S, w)$ 
11:       $w \leftarrow w + \alpha^w \partial \nabla \hat{v}(S, w)$ 
12:       $\theta \leftarrow \theta + \alpha^\theta I \partial \nabla \ln \pi(A|S, \theta)$ 
13:       $I \leftarrow \gamma I$ 
14:       $S \leftarrow S'$ 
15:    fim enquanto
16:  fim para
17: fim procedimento

```

2.4 Aprendizado por reforço profundo

Em diversos algoritmos de AR, como o *Q-Learning* e o Sarsa, em sua abordagem clássica utilizam tabelas responsáveis pelo mapeamento de conjunto de estados e ações possíveis do problema de aprendizado. O mapeamento do conjunto de estados e ações

em um problema de AR utilizando uma estratégia tabular impõe algumas limitações ao modelo.

A medida que um problema apresenta maior complexidade, esse procedimento pode ser de imensa dificuldade ou até mesmo intratável computacionalmente. Uma alternativa ao uso de tabelas de estados-ações é o uso de funções paramétricas ajustáveis, as quais poderiam aproximar a representação de um estado e suas ações. O algoritmo *Actor-Critic* faz uso dessa abordagem, utilizando um conjunto de pesos paramétricos, como demonstrado no [Algoritmo 3](#).

Como anteriormente exposto, tais funções paramétricas podem ser aproximadas por meio do uso de RNAs. O uso de funções paramétricas tipicamente prevê um número de pesos menor do que o número de possíveis estados do problema. Desta maneira, cada peso acaba por representar um conjunto de estados maior que apenas um único, e consequentemente seu ajuste acaba por generalizar para vários outros estados. A capacidade de generalização decorrente dessa característica torna o modelo mais poderoso, bem como de maior dificuldade para o seu aprendizado ([SUTTON; BARTO, 2018](#)).

Mais recentemente, devido aos avanços obtidos por meio do uso de redes neurais profundas, como já citado na [Seção 2.1](#), essas redes vêm sendo utilizadas conjuntamente com métodos de AR. A rede *deep Q-Network* (DQN), proposta por [Mnih et al. \(2015\)](#), é uma abordagem que une o algoritmo *Q-Learning* com o uso de *ConvNets*. A DQN foi treinada para jogar diversos jogos em um emulador da plataforma Atari 2600, obtendo resultados equivalentes ou maiores que o de humanos em diversos jogos. Na DQN, imagens do jogo são utilizadas como entradas para uma ConvNet, que retorna como saída uma ação a ser tomada dentro de um conjunto possível de ações. A ação é avaliada de acordo com o algoritmo *Q-Learning*, e utilizada para o treinamento da ConvNet. A utilização de algoritmos de aprendizado profundo, como ConvNets, juntamente com métodos de AR, passou a ser chamada de Aprendizado por Reforço Profundo (ARP) (do inglês, *Deep Reinforcement Learning*).

Outro notável resultado obtido utilizando ARP foi alcançado pelo trabalho de [Silver et al. \(2016\)](#), sendo aplicado para jogar o jogo de tabuleiro *Go*. Neste trabalho, os autores adotaram árvores de busca de Monte Carlo, as quais foram incrementadas por meio de políticas que buscavam prever movimentos humanos. Para as funções de política e valor do referido trabalho, foi utilizado redes neurais profundas. Uma ConvNet recebia imagens de um tabuleiro de dimensões 19×19 para construir representações de posições. Três políticas foram treinadas neste trabalho. A primeira trata-se de uma política p_σ treinada valendo-se de movimentos humanos de maneira supervisionada; a segunda de uma política rápida p_π , capaz de rapidamente selecionar amostras de ações durante *rollouts*; e por fim, uma política p_ρ , que aprimora a política treinada por meio de aprendizado supervisionado. A política p_ρ é a política de AR e é otimizada de acordo com os resultados finais da partida,

aprendendo assim o real objetivo destas. Conjuntamente, uma rede de valor v_θ que prevê o vencedor das partidas de AR foi treinada. Foram obtidas taxas de 99,8% de vitória contra outros programas de *Go*. Quando jogou contra o vencedor europeu de *Go*, o placar obtido foi de 5×0 , sendo esta a primeira vez que um programa de computador derrotou um jogador profissional de *Go* em um tabuleiro de tamanho completo.

Outros algoritmos clássicos de AR foram propostos recentemente utilizando redes neurais profundas, a exemplo do algoritmo *Actor-Critic* (ver Algoritmo 3), com a proposição do algoritmo *Asynchronous advantage actor-critic* (A3C) por Mnih *et al.* (2016), que utiliza redes neurais profundas para aprendizado de suas funções de política e valor, de maneira paralelizada.

Atualmente, abordagens utilizando aprendizado profundo vêm sendo aplicadas não apenas na área de jogos, mas também em robótica, como controle de braços robóticos, movimentação de robôs, aprendizado de comportamentos, entre outros, obtendo resultados promissores.

2.5 Considerações finais

Foram apresentados ao longo deste capítulo métodos de aprendizado de máquina utilizando modelos de redes neurais artificiais e suas diversas variantes. Com resultados expressivos na resolução de problemas desafiadores, arquiteturas como ConvNets, LSTMs e GRUs se tornaram escolhas padrão para lidar com dados topológicos e sequenciais. Não obstante, GNNs e sua variante convolucional, a GCNs, surgem como uma alternativa para problemas de agrupamento e quando a manutenção de informações topológicas desempenham um papel de interesse.

A aplicação das redes neurais profundas como funções paramétricas em modelos de AR também vêm demonstrando bons resultados em problemas deste campo. Algoritmos como *Q-Learning*, SARSA e *Actor-Critic* foram abordados e descritos, conjuntamente com suas características de comportamento *on-policy* e *off-policy*. Versões destes algoritmos valendo-se de redes neurais profundas foram aplicados com êxito para resolução de problemas complexos. Métodos de políticas por gradiente decorrentes de classes de algoritmo *REINFORCE* permitiram tais abordagens utilizando funções parametrizadas ao invés de métodos tabulares. Essa tendência da aplicação conjunta de redes profundas em técnicas de AR passou a ser reconhecida como ARP.

Avanços com o uso de ARP foram comentados dada realizações de trabalhos recentes, obtendo resultados relevantes em jogos virtuais e de tabuleiro. Ainda, o algoritmo A3C, uma variação paralelizada e com AP do algoritmo *Actor-Critic*, bastante utilizada em diversos trabalhos, foi apresentada e seu pseudocódigo demonstrado.

As técnicas apresentadas ao longo deste capítulo são utilizadas no presente trabalho. Arquiteturas de ConvNets permitem o entendimento da imagem por meio do reconhecimento de objetos presentes na cena, GCNs são utilizadas para obter correlações semânticas entre os objetos observados, e redes neurais com memórias são utilizadas para interpretar todo o contexto extraído anteriormente. Conjuntamente, métodos de ARP como A3C se valem do contexto provido pela rede de memória para, a partir desta, indicarem a ação a ser tomada dentro do conjunto de ações possíveis.

TRABALHOS RELACIONADOS

Neste capítulo são apresentados, discutidos e comparados trabalhos que se relacionam com a presente proposta em seus objetivos, temática, técnicas e conceitos. Os trabalhos são organizados de acordo com sua área de maior aderência dentre os conceitos que serão explorados neste trabalho. Apesar de estarem separados em diversas seções, alguns trabalhos podem compartilhar pontos de intersecção entre os tópicos definidos. Inicialmente, serão apresentados trabalhos que lidam com relações visuais e relações semânticas entre objetos. Em seguida, trabalhos utilizando visão para navegação de robôs móveis serão discutidos. Trabalhos que exploram a navegação de robôs móveis utilizando navegação visual conjuntamente com navegação semântica também serão considerados. Finalmente, um comparativo entre esses trabalhos e uma breve discussão sobre os mais pertinentes a proposta deste projeto serão apresentados.

3.1 Relações visuais e compreensão semântica

Nesta seção são discutidos trabalhos que exploram as relações visuais e semânticas entre objetos presentes em uma imagem, tendo como objetivo interpretação de cenas, detecção de objetos, criação de grafos relacionais e geração de descrições de imagens em linguagem natural.

Em [Yang et al. \(2018\)](#) foi proposto o *framework Graph R-CNN* para geração de grafos contextuais de uma cena capaz de prover informações a respeito de objetos e seus relacionamentos. Esse *framework* pode ser descrito em três estágios principais: a) extração de nós-objetos ;b) definição e poda de arestas entre nós; e c) integralização e aperfeiçoamento de contexto do grafo. Foram propostas duas redes para composição do *framework*. A primeira delas é a *Relation Proposal Network* (RePN), com função de ranqueamento de relação entre nós (objetos) e a segunda, *Attentional Graph Convolutional Network* (aGCN), designada a propagar informações contextuais de alto nível através do

grafo, atualizando os nós e seus relacionamentos baseado na vizinhança dos mesmos.

Inicialmente, o *framework* realiza a detecção de objetos na cena utilizando uma ConvNet do tipo *Fast* R-CNN. O resultado obtido por meio da *Fast* R-CNN é um número n de nós, os quais representam as regiões identificadas pela rede. O grafo gerado considerando-se relações entre todos os nós pode ter ligações na ordem de $O(n^2)$, muitas das quais podem não representar interações entre objetos no mundo real.

Devido a imensa dimensionalidade que tais grafos podem obter, tornando-os por vezes intratáveis, procedimentos de poda são realizados, comumente sendo aplicados métodos heurísticos como amostragem randômica, o qual está sujeito a realizar poda de relações importantes.

A RePN visa realizar o procedimento de poda por meio da avaliação de relações entre os objetos, buscando não sacrificar a eficiência do modelo e executando apenas podas necessárias. Posteriormente ao procedimento de poda, o modelo proposto aGCN é utilizado para atualizar e ponderar as relações entre os nós, aprendendo pesos distintos para tais relações, mecanismo ao qual os autores chamaram de atenção.

Vários testes foram realizados e obtiveram resultados superiores em todas as métricas avaliadas comparados a outros modelos, demonstrando a melhor representação dos objetos em imagens e suas relações por meio da adoção das GNNs, tendo como resultado grafos representativos com nós relacionando-se de maneira coerente e sumarizada.

Em Yao *et al.* (2018) também foram investigadas as relações semânticas e espaciais inerentes entre objetos, mas com o intuito de descrever imagens por meio de sentenças em linguagem natural. Como realizado em outros trabalhos semelhantes, foram utilizadas algumas técnicas de visão computacional para detecção de objetos e definição de regiões, adotando-se ConvNets do tipo R-CNN. Adicionalmente, GCN conjuntamente com LSTMs foram adotadas como forma de valer-se ainda mais das relações semânticas e espaciais.

A base de dados *Visual Genome* (KRISHNA *et al.*, 2017) foi utilizada para a construção do grafo semântico preditivo de relações, onde cada vértice do grafo representa uma região detectada e cada adjacência representa uma relação entre duas regiões, sendo este um grafo direcionado. As regiões para as quais foram aprendidas relações são posteriormente utilizadas como entrada para uma LSTM para geração da sentença.

Os resultados obtidos superaram outras abordagens aplicadas para o problema de descrição de imagens, como arquiteturas que utilizam ConvNets com LSTMs, bem como ConvNets com LSTMs e atenção, sem a utilização de GNNs. Além do comparativo utilizando diversas métricas utilizadas, resultados qualitativos foram exibidos por meio da comparação de descrições geradas.

Desta forma, os autores demonstraram que a utilização de GNNs e LSTMs conseguiram superar outras abordagens nas métricas avaliadas, obtendo-se êxito em enriquecer

o modelo com mais informações semânticas a respeito dos objetos e seus relacionamentos, gerando sentenças mais precisas e descritivas como consequência.

Um modelo de rede foi proposto em [Li et al. \(2017\)](#), utilizando GNNs para reconhecimento de situações em imagens. Diversos trabalhos vêm sendo realizados nos últimos anos buscando compreender a cena apresentada em uma imagem, de maneira a não apenas descrever objetos, mas ter informações suficientes para um maior entendimento e interpretação.

Para tanto, os modelos buscam identificar relações semânticas e contextuais entre objetos presentes na cena. Trabalhos citados neste artigo compartilham do uso de ConvNets para o reconhecimento de elementos presentes na imagem, comumente seguindo para a utilização de RNNs. Foi substituído o uso de RNNs por uma arquitetura de GNN provida de uma estrutura de *gatings* de funções de recorrência, chamada de *Gated Graph Neural Network* (GGNN). Por meio da GGNN, a estrutura de grafo foi explorada, que intrinsecamente representa relações entre nós, os quais são os objetos da imagem e suas relações.

Os resultados obtidos foram comparados com técnicas utilizando a estrutura padrão de ConveNets + RNNs, demonstrando que este trabalho apresentou resultados de $\sim 4,5\%$ de acurácia superiores quando consideradas as métricas que avaliam o contexto geral.

3.2 Navegação visual

Trabalhos que utilizam métodos de visão para navegação de robôs móveis são apresentados nesta seção. Alguns modelos combinam os métodos de visão com etapas de planejamento posteriores, enquanto outros utilizam mecanismos de memória ou processamento de linguagem natural para execução de comandos.

A utilização de navegação visual e instruções apresentadas por meio de linguagem natural foi abordada em [Anderson et al. \(2018b\)](#). Neste trabalho, instruções são informadas como sentenças em linguagem natural, as quais devem ser interpretadas e executadas por um robô em um ambiente. O robô navega em tal cenário utilizando visão, partindo de um ponto inicial e devendo locomover-se a um ponto objetivo.

A interpretação de comandos em linguagem natural é um campo de estudo de relevância atual e que vêm sendo beneficiado pelo sucesso do uso de LSTMs no processamento de cadeias sequenciais. Duas redes LSTMs foram utilizadas em [Anderson et al. \(2018b\)](#), com a finalidade de interpretar os comandos dados ao agente, aplicadas conjuntamente com uma ConvNet, do tipo ResNet, para lidar com imagens durante a navegação visual.

O simulador *Matterport3D* foi utilizado. Este simulador é construído utilizando

imagens da captura de ambientes reais, permitindo a execução de ações de locomoção e movimentação de câmera por um agente robótico simulado.

O modelo foi testado em ambientes não vistos durante a fase de treinamento, buscando aferir a sua capacidade de generalização. Nos testes foram realizadas comparações entre o caminho ótimo (sendo este o caminho mais curto), caminhada randômica no ambiente, e entre as duas estratégias de aprendizado adotadas pelo autor. Adicionalmente, o modelo foi comparado com trajetórias realizadas por humanos. Os resultados apresentados mostraram que os fatores analisados (taxa de erro, taxa de sucesso, entre outros) no conjunto de validação, tanto para ambientes conhecidos como desconhecidos, mantiveram-se aproximados quando submetido ao conjunto de teste (apenas com ambientes desconhecidos).

Em [Bansal et al. \(2019\)](#), técnicas de controle ótimo e navegação visual foram combinadas, almejando que um robô navegue em ambientes desconhecidos com trajetórias mais precisas e suaves.

Dois módulos foram propostos: o módulo de percepção e o módulo de planejamento. O módulo de percepção utilizou ConvNets para extrair conhecimento do ambiente por meio de uma imagem dada como entrada para a rede. Por meio desta, o módulo de percepção indica qual direcionamento deve ser adotado para o cálculo da rota do robô na próxima etapa, denominada por *waypoint*. O *waypoint* é utilizado pelo módulo de planejamento, que adota um *linear feedback controller*. Este controlador utiliza as informações do *waypoint* e as velocidades lineares e angulares do robô para planejar uma rota a ser executada durante um período de tempo x , determinado. Após feito isso, uma nova imagem deve ser capturada e o procedimento deve ser realizado novamente.

A base de imagens *Stanford large-scale 3D Indoor Spaces dataset* ([ARMENI et al., 2016](#)) foi utilizada para treinamento, por meio de aprendizado supervisionado. Após o treinamento, foram realizados testes com um robô real. O modelo proposto foi comparado com o método estado da arte E2E.

O modelo proposto apresentou um desempenho superior ao E2E, com maiores taxas de sucesso, trajetórias com menos oscilação, menor velocidade durante execução da trajetória (positivamente pontuado nos resultados) e menor tempo para conclusão das tarefas.

Além dos resultados apresentados acima, os autores comentam sobre o aprendizado semântico realizado pelo modelo. Em situações onde o robô se encontrava em uma sala e deveria se movimentar para outro local, observou-se que o mesmo correlacionava a transição de um ambiente para outro por meio das portas, procurando por estas no ambiente e se direcionando até elas, após encontrá-las.

A navegação visual foi combinada com técnicas de processamento de imagens,

memória e ARP para realizar a tarefa de navegação no problema de futebol com robôs humanoides bípedes, como pode ser visto em Lobos-Tsunekawa, Leiva e Solar (2018).

Dispondo do sensor de câmera presente nos robôs humanoides (robôs do modelo NAO), a abordagem de navegação visual foi adotada. Para tanto, utilizou-se uma rede ConvNet, conjuntamente com métodos de ARP (método DDPG) para o aprendizado das estratégias dos robôs. Para o treinamento foi utilizado um ambiente simulado, tendo o sistema desenvolvido sido posteriormente aplicado em ambiente real. Devido a imposição do problema de lacuna de realidade entre o ambiente simulado e o ambiente real, técnicas de segmentação de imagens foram usadas em imagens de ambos ambientes. Além das técnicas de processamento de imagens, ConvNets e ARP utilizadas, uma rede LSTM também foi adotada para inserir nos robôs mecanismos de memória sobre o ambiente, permitindo que os mesmos recordassem de cenas visualizadas anteriormente, as quais poderiam ser uma possível estratégia na ocorrência de um futuro bloqueio no caminho atual, bem como se tornarem mais sensíveis a possíveis movimentos dos demais jogadores e delimitações do campo.

O uso de LSTMs melhorou o desempenho durante a aprendizagem do modelo. Adicionalmente, o processamento das imagens possibilitou que o modelo aprendido em ambiente simulado fosse transferido de maneira satisfatória para o ambiente real. Finalmente, foi possível observar que os robôs recordaram de furos em barreiras ou bloqueios durante as execuções, mudando suas rotas e refazendo caminhos que possibilitassem o término da tarefa, resultados consequentes do uso de LSTMs.

No trabalho de Zhu *et al.* (2017b), um modelo de navegação baseado em visão e utilizando ARP foi proposto. A abordagem sugerida buscou lidar com um dos problemas enfrentados pela adoção de ARP, referente ao necessário retreino do modelo, dada a mudança dos objetivos do problema. Nesse trabalho, ao invés de definir os objetivos a serem aprendidos por meio da implantação desses nos parâmetros do modelo, os autores se valeram da definição do objetivo por meio da entrada do modelo. A essa abordagem foi dado o nome de navegação visual orientada ao alvo (traduzido livremente do inglês *Target-Driven visual navigation*). Na Figura 8 é mostrado um esquema do modelo proposto.

O modelo de ARP teve por objetivo o aprendizado de uma política estocástica π , a qual recebeu duas entradas: uma representação do atual estado do ambiente (captura da cena atual do robô) e uma representação do objetivo (imagem do local-alvo ao qual o robô deve chegar), s_t e g , respectivamente. Foi obtida por conseguinte uma distribuição de probabilidades sobre o espaço de ações $\pi(s_t, g)$. Para entradas do modelo foram utilizadas capturas do ambiente e uma imagem da cena objetivo, a primeira como representação do atual estado e a segunda como representação do objetivo. Como abordagem de ARP adotada utilizou-se redes *actor-critic* siamesas profundas. Como modelo foi adotado o algoritmo A3C (MNIH *et al.*, 2016). No entanto, ao realizarem o treinamento em paralelo,

Figura 8 – Modelo arquitetural proposto por [Zhu et al. \(2017b\)](#)

Fonte: Adaptada de [Zhu et al. \(2017b\)](#).

foram treinadas em cada instância uma cena objetivo diferente, atualizando as camadas específicas para cada cena, de acordo com as tarefas desta, enquanto que os respectivos gradientes das execuções eram acumulados, atualizando as redes siamesas para todos os alvos posteriormente.

Os experimentos foram executados, primeiramente, no *framework* de simulação AI2-THOR ([KOLVE et al., 2017](#)). Este provê um ambiente simulado fotorrealista, isto é, que possui cenários e objetos em 3D com composição de cores, formas e texturas próximas ao visualizado em ambientes e objetos reais, onde um agente pode executar ações de mover-se, realizar movimentações de câmera e interagir com diversos objetos (por exemplo, pegar e mover objetos, abrir armários e geladeiras, entre outros). Testes comparativos foram efetuados entre o modelo proposto e as seguintes abordagens: navegação randômica, caminho mais curto, *One-Step Q*, A3C com 1 *thread* e A3C com 4 *threads*. O modelo proposto foi dividido em duas abordagens: ramo único e final. O primeiro diz respeito a execução do modelo utilizando redes siamesas sem camadas de cenas específicas, enquanto o segundo refere-se ao modelo completo, com camadas para cenas específicas.

Os resultados foram mensurados por meio do tamanho dos trajetos percorridos pelo robô. O modelo proposto superou todas as abordagens em sua versão final, com exceção ao método caminho mais curto, o qual serve como comparativo de melhor resultado e utiliza uma representação do mapa não disponível aos outros métodos.

Posteriormente, o modelo treinado em ambiente simulado foi aplicado em ambiente real. Observou-se o desempenho deste dada três configurações: um treinamento completo no ambiente real, um treinamento parcial apenas nas camadas específicas enquanto manteve-se a camada genérica congelada, e finalmente, com o treinamento das camadas específicas, enquanto que para a camada genérica realizou-se apenas um ajuste

fino, usando transferência de aprendizado.

Foi percebido que a utilização de transferência de aprendizado do ambiente simulado para o ambiente real resultou em convergência do modelo 44% mais rápida que o modelo treinado do zero, evidenciando o valor de simulações para o treinamento de robôs.

O trabalho proposto por [Nguyen et al. \(2018\)](#) buscou realizar a navegação em um ambiente por meio de visão, aplicando redes neurais profundas e ARP. O trabalho pode ser visto como um incremento sobre a arquitetura proposta por [Zhu et al. \(2017b\)](#), compartilhando diversos aspectos.

De maneira similar a [Zhu et al. \(2017b\)](#), a arquitetura apresentada em [Nguyen et al. \(2018\)](#) (ver [Figura 9](#)) utilizou ConvNets do modelo ResNet para reconhecimento de objetos em um ambiente simulado. Conjuntamente a imagens capturadas do ambiente, foi tida como entrada uma imagem-alvo. O aprendizado da navegação foi realizado por meio da utilização de métodos de ARP, em específico do algoritmo A3C ([Mnih et al., 2016](#)). Contudo, algumas modificações foram realizadas na arquitetura proposta por [Zhu et al. \(2017b\)](#), como a incorporação de *checking points* e uma rede LSTM para memorização de ações passadas. Adicionalmente, a imagem-alvo utilizada pela rede em [Nguyen et al. \(2018\)](#) é apenas a imagem do objeto específico a ser encontrado, e não de toda a cena como em [Zhu et al. \(2017b\)](#).

Figura 9 – Modelo arquitetural proposto por [Nguyen et al. \(2018\)](#)

Fonte: Adaptada de [Nguyen et al. \(2018\)](#).

Um dos principais pontos do trabalho de [Nguyen et al. \(2018\)](#) foi a utilização de *checking points*. A ideia dos *checking points* foi prover ao modelo determinadas localizações que facilitem ao agente encontrar um objeto-alvo. Esses *checking points* foram selecionados

no ambiente, sendo definidos em posições estratégicas onde diversos objetos podem ser visualizados.

Também foi incorporada ao modelo uma rede LSTM (definido no modelo como *Auxiliary Network*), a qual recebeu informações de ações passadas executadas pelo agente, objetivando a memorização de trajetórias de ações, auxiliando o agente na exploração do ambiente utilizando *checking points* e evitando repetição de estados enquanto busca pelo objeto-alvo.

Os testes apresentados nesse trabalho demonstram que o aprendizado teve convergência mais rápida por meio do uso dos *checking points* e da rede LSTM.

3.3 Navegação Semântica

Nesta seção, são apresentados trabalhos que se valem de conceitos semânticos para navegação de robôs móveis. Os modelos buscam extrair características semânticas entre objetos presentes em uma cena para locomover-se em um ambiente.

Em [Sünderhauf \(2019\)](#), uma rede do modelo GCN foi utilizada para realizar a navegação semântica de um agente. O trabalho explorou formas de navegação ricas em semântica que se utilizam de mapeamento semântico centrado em objetos utilizando técnicas recentes com SLAM. Nestas abordagens, objetos são reconhecidos conjuntamente com o mapeamento e demarcados espacialmente.

Diferentemente dos trabalhos apresentados acima, um mapa topológico foi utilizado neste trabalho, no formato de um grafo, e acessível a rede de política do agente. A composição do grafo foi formada por nós que indicam posicionamentos e marcadores que indicam objetos estáticos no ambiente. Os nós de posicionamento se relacionam com outros nós de posicionamento, indicando uma movimentação possível entre estes; os marcadores de objetos se conectam aos nós de posicionamento, indicando que aquele objeto pode ser acessível e visível a partir daquela posição.

Neste trabalho, os autores buscam encontrar objetos não-estáticos como, por exemplo, controles remoto, xícaras, entre outros. Estes objetos não estão inclusos no conjunto de objetos que compõem os marcadores descritos acima. Contudo, os objetos alvos podem possuir relações com os objetos-estáticos mapeados no ambiente.

Uma GCN foi utilizada como rede de política, treinada por meio de aprendizado por reforço. A GCN buscou aprender uma política $\pi(\mathcal{G}, C_{target})$, objetivando encontrar um objeto-alvo C_{target} em um determinado ambiente representado por meio de um grafo $\mathcal{G}$. Para tanto, esta rede deve aprender as relações probabilísticas presentes nestes ambientes entre objetos estáticos e objetos-alvo não estáticos, indicando quais nós de posição devem ser navegados até se chegar ao objeto-alvo.

O autor realizou o treinamento de 200 agentes, utilizando ambientes gerados por meio de regras probabilísticas definidas pelos autores. Tais regras definiam a disposição dos nós-posições nos grafos e a categoria de cômodo ao qual pertenciam. Posteriormente, marcadores representando objetos estatísticos também foram dispostos no ambiente. Os objetos-alvo, contudo, não apareciam diretamente no mapa, sendo distribuídos próximo a objetos com os quais se relacionavam de acordo com uma probabilidade.

Os autores avaliaram o modelo com dois *baselines*: um modelo que executa ações randômicas; e um modelo com total acesso ao modelo probabilístico que controla onde os objetos-alvo encontram-se dispostos no ambiente. Os autores relataram resultados próximos ao do modelo Oráculo, obtendo elevadas taxas de sucesso e trajetórias de tamanho curto. Também foi indicado por aqueles que o modelo conseguiu manter generalização para ambientes e objetos não apresentados anteriormente.

Em [Zhu et al. \(2017a\)](#) foi proposto um modelo para planejamento e execução de tarefas utilizando navegação visual e informações semânticas, ao qual os autores intitularam de planejamento visual semântico. Esse consiste na tarefa de prever uma série de ações semânticas que levam o agente de um determinado estado atual para outro desejado. Os problemas foram tratados em diferentes níveis de complexidade, alguns dos quais incluíram navegar no ambiente, identificar objetos e cenas e interagir com elementos. Para tanto, foram utilizadas abordagens como aprendizado por imitação (do inglês, *Imitation Learning* (IL)) e representação sucessora (em inglês, *Successor representation* (SR)), além da utilização de métodos de AP, como redes neurais profundas e ARP.

Utilizando IL, um modelo de rede foi treinada, por meio de uma abordagem supervisionada, com base em um planejamento de trajetória ótimo. Em segundo momento, o aprendizado com IL pôde ser refinado com base no ARP. Como método do tipo ARP foi adotado o algoritmo A3C. A utilização de IL permitiu evitar o problema das recompensas esparsas, as quais dificultam o aprendizado por métodos de ARP. O treinamento foi realizado em ambiente simulado adotando-se o *framework* AI2-THOR.

Para avaliar o modelo foi feita uma comparação com as seguintes abordagens de navegação e tomada de ações: abordagem randômica, randômica em um universo de ações válidas, algoritmo A3C puro, CLS-MLP (algoritmo A3C para treinamento de um classificador *softmax*) e CLS-MLP com LSTM. Os modelos propostos foram variados entre utilização de SR e IL, sem treinamento posterior com ARP, e utilização de SR e IL com treinamento posterior por meio de ARP.

Os resultados demonstraram que o modelo utilizando SR e IL e SR, IL e ARP obtiveram, de maneira geral, execução das tarefas com uma média de tamanho do episódio inferior as outras abordagens comparadas. As taxas de sucesso mantiveram-se equivalentes aos modelo CLS-MLP e CLS-MLP com LSTM.

Outro resultando apontado é um comparativo entre as taxas de sucesso obtidas, tanto para episódio como para ações, em relação ao número de iterações de treinamento para quando o modelo é refinado a partir da utilização de SR ou quando toda a rede é retreinada. Foi demonstrado que, quando o modelo apenas refinado foi possível obter maiores taxas de sucesso com o mesmo valor de iterações de treinamento.

Representação visual e informações sobre o contexto para realizar a navegação em ambientes reais, sem a utilização de mapas, foi proposta em Mousavian *et al.* (2018). A arquitetura do modelo proposto é composta por ConvNets voltadas a detecção de objetos e segmentação de imagens, além de uma rede LSTM.

Na Figura 10 pode ser observado o modelo proposto. Inicialmente, realizaram-se sobre a imagem de entrada procedimentos para detecção de objetos e segmentação. Características foram extraídas e concatenadas com uma imagem de profundidade, posteriormente sendo utilizadas como entrada para uma rede ConvNet do modelo ResNet. O modelo seguiu com a utilização de camadas totalmente conectadas que recebiam os resultados dos processamentos anteriores e o objetivo repassado como um vetor *one-hot*, projetando um vetor combinado de tamanho 128. Uma rede LSTM foi então utilizada, recebendo o vetor combinado por meio das camadas totalmente conectadas e informações sobre a ação realizada previamente e seu indicador de sucesso. Por fim, os resultados foram processados por mais uma última camada totalmente conectada, tendo como saída uma previsão de custo $v(o, a, c)$ de selecionar uma determinada ação no estado atual.

Figura 10 – Modelo arquitetural proposto por Mousavian *et al.* (2018)

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

As redes descritas acima para detecção de objetos e segmentação das imagens são utilizadas como uma abordagem para sumarizar as informações contidas na imagem de entrada não processada. Por meio dessa abordagem, o ambiente pode ser descrito de maneira simplificada, mantendo informações relevantes sobre objetos e suas características, e possibilitando a melhora dos resultados quando submetido a ambientes nunca antes visitados.

O modelo foi treinado utilizando bases de dados de imagens do interior de residên-

cias, com ambientes diversificados e enriquecidas com informações (reais ou sintéticas) a respeito das categorias de objetos, segmentações desses, *bounding-boxes*, entre outras.

Os resultados foram avaliados observando a taxa de sucesso obtido na conclusão do trajeto. Foi possível observar que melhores valores foram obtidos por meio do uso de imagens segmentadas e com detecção de objetos, em especial quando o modelo foi submetido a novos ambientes, melhorando sua capacidade de generalização. Adicionalmente, a utilização de dados sintéticos conjuntamente a dados reais no treinamento melhorou o desempenho do modelo. Avaliações também foram realizadas com o modelo sem o uso da rede LSTM. Os autores observaram que utilizando memória obteve-se maiores taxas de sucesso.

A utilização de dados sintéticos, processamento de imagens de maneira a obter contextualização semântica através de detectores e segmentações, bem como o uso de memória sobre ações e resultados tomados anteriormente, demonstraram melhores resultados, de maneira especial quando aplicado sobre novos ambientes nunca antes visitados, obtendo-se melhor generalização.

Em [Yang et al. \(2019\)](#) foi proposto um modelo de navegação semântica utilizando visão computacional. Nesse trabalho, informações semânticas *a priori* do ambiente foram utilizadas para auxiliar no processo de navegação, buscando uma abordagem capaz de maior generalização. Para tanto, o modelo proposto valeu-se de diversas técnicas de aprendizado de máquina, como ConvNets, GCN e ARP.

A arquitetura é mostrada na [Figura 11](#). Em um primeiro momento, o modelo foi composto por uma ConvNet do tipo ResNet, vetores de palavras (em inglês, *word embeddings*) e uma GCN. A ResNet foi utilizada como rede visual, extraindo objetos presentes em uma imagem capturada pelo agente no ambiente. Como saída foi retornado um vetor de características de 512 dimensões. Um *word embedding* foi adotado com fins de expandir os termos atrelados a um determinado objeto-alvo (repassado em formato textual) com palavras relacionadas (por exemplo, “abajur” pode designar uma alternativa a “lâmpada”), por meio de um vetor de palavras de 100 dimensões. Finalmente, uma GCN foi usada para construir um grafo de conhecimento por meio de uma composição do atual estado da rede visual e da rede semântica. A saída da GCN pode ser compreendida como uma codificação das relações semânticas a serem utilizadas posteriormente na arquitetura.

Em um segundo momento, as informações da rede visual e da rede semântica foram concatenados com a saída do grafo de conhecimento, criando um vetor unificado de características. Esse vetor foi utilizado como entrada para o método de ARP. Nesta etapa foi utilizado pelos autores o modelo *Actor-Critic* por meio do algoritmo A3C. Uma ação para o estado atual é selecionada e executada no ambiente de acordo com a entrada (vetor unificado de características). O conjunto de ações permitidos ao agente foram “Mover-se adiante”, “Move-se para trás”, “Rotacionar à direita” e “Rotacionar à esquerda” e “Parar”.

Figura 11 – Modelo arquitetural proposto por Yang *et al.* (2019)

Fonte: Adaptada de Yang *et al.* (2019).

A GCN foi inicialmente construída utilizando a base de dados *Visual Genome* (KRISHNA *et al.*, 2017) para construir o grafo de conhecimento *a priori*, limitando os objetos dessa base à apenas aqueles presentes no ambiente de simulação e que eram visíveis nesse. Durante a execução, esse grafo de conhecimento foi atualizado a partir da cena atual, utilizando o resultado da computação da ResNet.

Um dos principais pontos explorados foi a utilização de um grafo de conhecimento pelo qual conhecimentos semânticos do ambiente podem ser obtidos *a priori*, o que ajudou o modelo a generalizar melhor para ambientes e objetos nunca vistos antes, segundo os autores.

O modelo apresentado na Figura 11 foi treinado e testado no *framework* de simulação AI2-THOR (KOLVE *et al.*, 2017). Os ambientes presentes no simulador e adotados pelos autores pertenciam a 4 categorias: sala de estar, cozinha, quarto e banheiro. Para cada categoria de ambiente, diversos cenários com variações em texturas, decoração e planta do ambiente são possíveis, com um total de 30 modelos de cenários para cada ambiente. Os cenários foram divididos em conjunto de treino, validação e teste, sendo 20, 5 e 5 cenários para cada conjunto, respectivamente. Os testes foram realizados considerando as seguintes possibilidades: objetos conhecidos e cenas visitadas; objetos desconhecidos e cenas visitadas; objetos conhecidos e cenas não visitadas; e objetos não conhecidos e cenas não visitadas. Para comparativo de resultados, para cada possibilidade de testes fora executada uma política de navegação randômica, uma política utilizando apenas o algoritmo A3C puro e, por fim, o modelo proposto pelos autores.

Os autores limitaram-se a fazer experimentos em busca de objetos que fossem

visíveis no ambiente. Ao todo, os ambientes simulados contam com 87 objetos comuns, dentre os quais 53 possuíam visibilidade em cenários gerados randomicamente. Desta forma, os autores não consideraram objetos como, por exemplo, colheres, que podem ser inclusas dentro de gavetas durante a geração randômica, podendo nem sempre estarem visíveis.

Neste trabalho, o encerramento do episódio poderia ocorrer de duas maneiras: uma onde o robô executava o término do episódio, por meio de uma ação “parar”, presumindo-se que teria o agente visualizado o objeto-alvo (mais desafiador); e quando o ambiente realizava a notificação de término, observando que o agente encontrava-se com o objeto em seu campo de visão, porém não havia executado a ação de término (menos desafiador).

Os resultados obtidos foram avaliados sob duas métricas: taxa de sucesso e taxa de sucesso ponderada pelo tamanho da trajetória, esta última uma medida de eficiência. Adicionalmente, os modelos foram avaliados considerando tanto o treinamento com término do agente, como o treinamento com término do ambiente. Os autores indicaram melhor desempenho para o modelo proposto, alegando desempenho superior em quase todos os cenários e ambientes, para ambas as métricas e condições de término. Dentre os resultados, o de menor expressão é aquele onde robô se encontra em cenários da categoria de ambientes banheiro, onde as taxas de sucesso entre todos os algoritmos são equivalentes. Isso se justifica devido ao banheiro ser um ambiente pequeno. É válido de nota pontuar uma expressiva diferença nos resultados apresentados quando permitido o término do episódio pelo ambiente (cenário mais simples) e com o término realizado exclusivamente pelo agente (mais desafiador), este último com valores menores.

Os autores treinaram modelos separados para cada categoria de ambiente, selecionando para avaliação em testes o modelo de melhor desempenho dentre os treinados. Para mensurar a generalização dos modelos entre ambientes, os autores utilizavam um modelo treinado em uma categoria de ambiente e avaliaram as métricas quando aplicando-o em outra categoria. Por exemplo, um modelo treinado na categoria de cozinha era aplicado para testes em um ambiente do tipo sala de estar. Os autores apresentam uma tabela de resultados considerando o cenário com término possível pelo ambiente, isto é, mais simples, porém não realizam uma discussão a respeito dos valores. Ainda, os autores relatam que, quando considerando o término do episódio realizado apenas pelo agente robótico, os valores obtidos foram próximos do randômico. Treinamento de um único modelo em todos os ambientes foi efetuado, não tendo resultados expostos no trabalho, porém tendo sido alegado pelos autores desempenho inferior aos modelos treinados separadamente. Por fim, em [Yang et al. \(2019\)](#) não foi explorada a utilização de memória, não considerando as relações temporais recorrentes e resultados entre objetos correlatos e episódios bem sucedidos.

Em [Wortsman et al. \(2019\)](#) foi proposto um modelo de navegação visual semân-

tica utilizando uma abordagem de aprendizado contínuo por meio da aplicação de meta-aprendizado conjuntamente com aprendizado por reforço. Neste trabalho, um agente desempenhou a atividade de encontrar objetos-alvo no ambiente utilizando visão. Assim como em outros trabalhos anteriores com navegação baseada em aprendizado, neste trabalho o agente robótico também não possuiu acesso a nenhum mapa topológico ou métrico do ambiente.

De maneira semelhante a [Yang et al. \(2019\)](#), em [Wortsman et al. \(2019\)](#) também utilizou-se uma ConvNet para o reconhecimento de objetos no ambiente, adotando uma rede do modelo ResNet. Ainda, o modelo contou com a utilização de *word embeddings*, também em vista de enriquecer as compreensões a respeito dos objetos-alvo. O modelo de [Wortsman et al. \(2019\)](#) apresentou ainda uma rede de memória, com o modelo LSTM, e, atrelado ao aprendizado por reforço com agentes A3C, foi adotada a abordagem de meta-aprendizado.

Os autores adotaram o ambiente simulado AI2-THOR para realizar o treinamento e testes dos modelos, tendo quatro ambientes do interiores de residências, sendo: cozinha, sala de estar, quarto e banheiro. O agente pôde executar as ações de “Mover-se adiante”, “Rotacionar à direita”, “Rotacionar à esquerda”, “Olhar para cima”, “Olhar para baixo” e “Encerrar”.

Os autores treinaram um único modelo para todos os tipos de cômodos, e realizam avaliações comparativas com os *baselines*: agente randômico, vizinho mais próximo (comparativo entre observações visuais observadas e cenas com o objeto procurado presente) e agentes não-adaptativos, sendo estes os modelos utilizando A3C-puro e também o modelo de [Yang et al. \(2019\)](#).

Foram utilizados 20 cenários para treino, 5 para validação e 5 para testes. Os autores relataram resultados superiores para o seu modelo com meta-aprendizado quando comparados aos demais modelos nas métricas observadas. Ainda, os autores afirmam que o agente apresentou melhor comportamento quando posto em situações mais suscetíveis a erros, como ao colidir ou encontrar-se cercado por um ou mais objetos.

Contudo, é válido de nota destacar que o modelo de [Yang et al. \(2019\)](#) utilizado nas análises comparativas foi acrescido de uma rede LSTM, diferente do modelo originalmente proposto. Entretanto, nenhuma análise foi realizada a respeito de possíveis mudanças de comportamento e desempenho do modelo alterado frente ao modelo original, visto que o trabalho de [Wortsman et al. \(2019\)](#) focou na análise comparativa com o seu modelo utilizando meta-aprendizado.

3.4 Comparativo e Discussão

Os trabalhos apresentados na primeira seção deste capítulo foram focados na interpretação de cenas buscando extrair informações a respeito do relacionamento entre objetos. Alguns trabalhos tiveram por objetivo a construção de grafos de relacionamentos, enquanto outros almejavam a criação de descrições das cenas em linguagens natural. Notou-se que os trabalhos desenvolvidos recentemente estão estendendo técnicas anteriores, baseadas na utilização de ConvNets e redes LSTM, com a adição de redes GNNs.

Os demais trabalhos apresentados nesse capítulo se encontram relacionados com navegação de robôs móveis, utilização de visão computacional e, em alguns casos, de informações semânticas.

Em [Anderson et al. \(2018b\)](#) e [Bansal et al. \(2019\)](#) foram demonstrados os benefícios do uso de ambientes simulados fotorrealistas para navegação de robôs móveis, sendo esta uma maneira efetiva de refinar o modelo e acelerar o processo de aprendizagem. Contudo, esses trabalhos se limitaram ao uso exclusivo de abordagens atreladas ao aprendizado supervisionando, não explorando a aplicação de ARP, por exemplo.

Em [Lobos-Tsunekawa, Leiva e Solar \(2018\)](#) diversas técnicas foram exploradas além da visão computacional com ConvNets. Mecanismos de memorização foram aplicados com sucesso por meio de LSTM, permitindo aos robôs relembrarem rotas observadas anteriormente, bem como aprendizado por meio de ARP. A lacuna de realidade ocasionada pelo simulador não-fotorrealista utilizado foi contornada por meio da segmentação de imagens. O trabalho apresentou resultados interessantes, sendo aplicado posteriormente em robôs reais, porém possui um escopo reduzido em vistas do ambiente de menor complexidade em relação aos demais trabalhos de navegação visual.

Os trabalhos apresentados em [Zhu et al. \(2017b\)](#) e [Nguyen et al. \(2018\)](#) compartilham diversos aspectos, como o ambiente de simulação AI2-THOR e a arquitetura utilizada. Contudo, em [Nguyen et al. \(2018\)](#) apenas a imagem do objeto específico a ser encontrado é passada como objetivo para o modelo, e não toda a cena como ocorre em [Zhu et al. \(2017b\)](#), permitindo assim que o seu modelo seja mais específico quanto ao alvo enquanto torna-se mais generalista quanto ao ambiente. Adicionalmente, em [Nguyen et al. \(2018\)](#) foram selecionados pontos no mapa para referência do modelo na busca pelos objetos, os quais foram memorizados por meio de LSTMs ao longo do treinamento, resultando na convergência mais rápida do modelo. Porém, os *checking points* foram definidos, não sendo aprendidos pelo modelo de maneira a avaliar a importância durante os resultados na aprendizagem.

[Sünderhauf \(2019\)](#) aplicou redes GCN para planejar a navegação a partir de um grafo representando um mapa topológico do ambiente. Neste trabalho, mapas foram gerados, tendo nós-posições e marcadores de objetos estáticos distribuídos, construindo mapas

topológicos de interiores de residências. O objetivo era navegar por entre os nós-posições até um nó onde fosse possível acessar um objeto-alvo não estático no ambiente. Tais objetos não estáticos eram relacionados aos objetos estáticos, possuindo assim relações semânticas contextuais entre si. Diferentemente de outros trabalhos apresentados, a GCN utilizada em [Sünderhauf \(2019\)](#) foi aplicada diretamente em um grafo representando um mapa do ambiente, enquanto outros trabalhos discutidos não possuíam nenhuma representação topológica do ambiente. Os autores alegaram resultados próximos do modelo ótimo nomeado de Oráculo, com acesso ao modelo probabilístico com o qual o mapa foi gerado.

Em [Zhu et al. \(2017a\)](#) foram adicionados IL e SR aos modelos de visão e ARP, buscando não apenas realizar a navegação no ambiente, mas construir um planejamento sobre cadeias maiores de comandos e tarefas a serem executados. O uso de IL permitiu ao modelo lidar com o problema de recompensas esparsas, melhorando a convergência deste quando realizado ajuste fino durante o aprendizado com ARP.

Em [Mousavian et al. \(2018\)](#), assim como realizado em [Lobos-Tsunekawa, Leiva e Solar \(2018\)](#), segmentação das imagens do ambiente foram realizadas como forma de sumarizar informações. Enquanto no último o objetivo foi a redução da lacuna de realidade, em [Mousavian et al. \(2018\)](#) buscou-se, adicionalmente a isto, utilizar apenas informações mais características dos objetos e criar um contexto semântico. Apesar do uso de LSTM, esta não foi utilizada para memorizar correlações entre objetos, mas apenas ações anteriores e os seus resultados. Também não foi utilizado nenhum ambiente simulado ou método de AR.

O trabalho mais relacionado ao plano de pesquisa proposto nesta dissertação é o realizado em [Yang et al. \(2019\)](#). Neste trabalho, ConvNets e ARP foram acrescidos do uso de GCN, buscando partir de conhecimentos de relações entre um objeto-alvo e outros objetos presentes no ambiente. A medida que o agente navega pelo ambiente, o grafo construído com conhecimentos *a priori* é constantemente atualizado, valendo-se de informações como vetor representativo de palavras e a cena atualmente visualizada pelo agente. Foi utilizado um ambiente simulado fotorrealista, e resultados demonstraram que o modelo conseguiu generalizar para ambientes e objetos não vistos durante o treinamento. Diferentemente da GCN utilizada em [Sünderhauf \(2019\)](#), em [Yang et al. \(2019\)](#) não foi utilizado nenhuma mapa topológico ou métrico do ambiente, sendo treinado em uma abordagem de navegação baseada em aprendizado. Contudo, mecanismos de memória não foram explorados neste trabalho como realizado em outros, uma lacuna com interessantes resultados quando investigada em outros trabalhos com o acréscimo de recursos de memorização.

Por fim, em [Wortsman et al. \(2019\)](#) foi utilizado redes ConvNets, ARP e meta-aprendizado na tarefa de navegação em ambientes internos de residências. Tal como outros

trabalhos, uma abordagem de navegação baseada em aprendizado foi adotada, não sendo disponibilizado ao agente nenhum mapa topológico. Ainda, a navegação foi realizada utilizando visão e tendo objetos-alvo definidos de maneira semelhante ao trabalho de [Yang et al. \(2019\)](#). A arquitetura também utilizava uma rede LSTM para incluir mecanismos de memória ao agente. Os autores relataram bons resultados neste trabalho, comparando o mesmo com modelos randômicos e um modelo de [Yang et al. \(2019\)](#) acrescido com uma LSTM. Contudo, tal modificação não foi explorada ou discutida no trabalho, a medida que os autores se ocuparam em apresentar o modelo proposto com meta-aprendizado.

Tal conjunto de trabalhos são sumarizados e apresentados na [Tabela 1](#). Os tópicos comparados foram o uso de ConvNets, uso de redes de memória (LSTMs ou GRUs), aplicação de ARP, utilização de GNNs, e execução dos modelos em ambientes simulados e reais. Foi incluído nesta tabela o modelo arquitetural proposto neste trabalho.

Tabela 1 – Tabela comparativa entre trabalhos relacionados envolvendo navegação

Trabalho	ConvNets	Memória	ARP	GNN	Ambiente Simulado	Ambiente Real
Anderson et al. (2018b)	✓ (ResNet)	✓	×	×	✓ (Matterport3D)	×
Bansal et al. (2019)	✓ (ResNet)	×	×	×	✓ (Stanford)	✓
Lobos-Tsunekawa, Leiva e Solar (2018)	✓ (SegNet)	✓	✓ (DDPG)	×	✓ (SimRobot)	✓
Zhu et al. (2017b)	✓ (ResNet)	×	✓ (A3C)	×	✓ (AI2-THOR)	✓
Nguyen et al. (2018)	✓ (ResNet)	✓	✓ (A3C)	×	✓ (AI2-THOR)	×
Zhu et al. (2017a)	✓ (ResNet)	×	✓ (A3C)	×	✓ (AI2-THOR)	×
Mousavian et al. (2018)	✓ (ResNet)	✓	×	×	✓ (SUNCG)	×
Yang et al. (2019)	✓ (ResNet)	×	✓ (A3C)	✓ (GCN)	✓ (AI2-THOR)	×
Wortsman et al. (2019)	✓ (ResNet)	✓	✓ (A3C)	×	✓ (AI2-THOR)	×
Santos I. B. A.; Romero (2020)	✓ (ResNet)	✓	✓ (A3C)	✓ (GCN)	✓ (AI2-THOR)	×

É possível visualizar que todos os trabalhos comparados utilizaram ConvNets como método para tratar as imagens do ambiente. O modelo de ConvNet utilizado na maioria dos trabalhos foi o modelo ResNet (variando seu número de camadas a depender do trabalho), sendo a única exceção o trabalho de [Lobos-Tsunekawa, Leiva e Solar \(2018\)](#), que adotou uma rede específica para as imagens resultantes do processo de segmentação. Já o uso redes de memória foi adotado em apenas metade dos trabalhos, quando os autores buscaram integrar aos modelos redes do tipo LSTM.

A maioria dos trabalhos aplicou ARP durante o aprendizado da navegação, em sua maioria adotando o algoritmo A3C. É válido destacar que o algoritmo DDPG internamente também faz uso do algoritmo A3C. Uma GNN foi utilizado apenas por [Yang et al. \(2019\)](#), demonstrando ser esta uma técnica ainda pouco explorada para a atividade de navegação, a despeito dos resultados de melhora na compreensão da relação entre objetos alegada em diversos trabalhos.

A respeito dos ambientes, todos os trabalhos foram realizados em ambiente simulado, sendo 1 no ambiente *Matterport3D*, 1 no ambiente Stanford, 1 no ambiente *SimRobot*, 1 no ambiente SUNCG e 5 no ambiente AI2-THOR. Alguns trabalhos não utilizaram propriamente um ambiente simulado, mas realizaram o treinamento diretamente em imagens

de bases de dados. Uma minoria de trabalhos também tiveram testes realizados em ambientes reais, onde todos os 3 fizeram uso conjuntamente com ambiente simulado.

O presente trabalho busca explorar a inserção de mecanismos de memória na arquitetura proposta por [Yang et al. \(2019\)](#) por meio de redes como LSTM e GRUs. É esperado que o modelo aprendido consiga inferir relações temporais entre objetos que possuam correlações. Ainda, fora expandido o conjunto de ações possíveis de serem executadas pelo agente robótico em relação ao trabalho de [Yang et al. \(2019\)](#). O modelo é detalhadamente descrito no capítulo a seguir.

3.5 Considerações Finais

Neste capítulo foram apresentados diversos trabalhos relacionados a conceitos e temáticas que são de interesse para a pesquisa deste trabalho, organizados por suas áreas e aplicações de maior afinidade. Inicialmente, trabalhos que buscaram realizar a detecção de objetos, suas relações e a interpretação da cena geral observada foram apresentados, utilizando métodos de AP como ConvNets, redes LSTMs e redes GNN. A interpretação de cenas, de maneira aprofundada e observando relações semânticas, é de grande interesse para este trabalho, de maneira que tais relações são importantes para a construção do modelo sugerido.

Posteriormente, foram explorados trabalhos que realizaram a atividade de navegação de robôs móveis. Os trabalhos apresentados utilizavam informação visual capturada por meio de câmeras, sejam apenas RGB ou RGB-D, como principal sensor para a execução de suas atividades. As imagens foram utilizadas conjuntamente com ConvNets para a detecção de objetos e reconhecimento de cenas. Alguns trabalhos também utilizaram memória por meio de LSTMs buscando melhores resultados, tanto na aplicação direta do aprendizado da navegação ou para aprendizado de planejamento de atividades.

Finalmente, informações semânticas *a priori* e constantemente atualizadas durante a execução da navegação foram exploradas em trabalhos recentes. A construção do conhecimento semântico nestes trabalhos foi realizada pela aplicação de procedimentos auxiliares nas imagens, como segmentação, ou por meio de GNNs para o enriquecimento com grafos de conhecimento.

Trabalhos recentes também aplicaram ARP para realizar navegação baseada em aprendizado, com agentes efetuando buscas objeto-direcionadas. Em diversos desses modelos foi dispensado a utilização de um mapa topológico ou métrico, sendo aprendido um mapa implícito do ambiente a medida que o agente explorava e aprendia sobre este último.

No presente trabalho explorar-se a lacuna do uso de redes de memória, LSTM e GRU, em modelos de navegação baseado em aprendizado, conjuntamente com a adoção

de redes GCN para contextualização semântica da cena. Investigou-se a possibilidade de a inclusão de memória desempenhar papel positivo em permitir um aprendizado mais exploratório e capaz de encontrar o objeto-alvo. A arquitetura proposta é apresentada e detalhada no capítulo a seguir.

PESQUISA PROPOSTA

Neste capítulo é apresentada a arquitetura proposta para navegação visual de um robô móvel em ambientes residenciais. O diferencial desta arquitetura é que ela inclui um módulo de memória ao modelo proposto por em [Yang *et al.* \(2019\)](#). Inicialmente, são descritas as motivações do modelo de navegação visual semântica com memória, apresentada sua arquitetura, seu funcionamento e o detalhamento dos seus módulos componentes. Em seguida, serão apresentados os ambientes de simulação disponíveis e justificado o ambiente selecionado para a execução do trabalho. As técnicas e recursos a serem utilizados serão explicitados. Por fim, é definida formalmente a tarefa a ser executada pelo agente robótico, as métricas de avaliação adotadas neste trabalho, seguidas pelas considerações finais.

4.1 Navegação visual semântica com memória

Nesta seção, uma arquitetura para realizar a navegação de agentes robóticos no contexto de ambientes internos de residências é proposta. Para tanto, técnicas de navegação visual, navegação baseada em aprendizado e redes neurais profundas são adotadas, motivados por avanços expressivos em técnicas de aprendizado de máquina, apresentadas no [Capítulo 2](#), e os resultados recentes descritos em trabalhos como os apresentados no [Capítulo 3](#).

Partindo da arquitetura proposta, objetivamos permitir a um robô móvel navegar em ambientes complexos do interior de residências em busca de objetos-alvo comuns em tais contextos, de maneira inteligente, utilizando conhecimento semântico das relações entre objetos observados e valendo-se de técnicas de visão computacional para compreender a cena ao seu redor. A navegação em ambientes complexos e mutáveis é uma tarefa desafiadora devido a variabilidade e diversidade inerentes a estes cenários. Nesta proposta busca-se obter um modelo de conhecimento flexível, que possa ser adotado em diversas ca-

tegorias de ambientes e que não utilize representações como mapas topológicos e métricos, ou utilização de sensores avançados, limitando-nos a adoção apenas de câmeras RGB.

Para ilustrar melhor a tarefa que estamos propondo a resolver, consideremos a situação hipotética onde um agente robótico encontra-se em uma cozinha e recebe um comando onde lhe seja definido a tarefa de encontrar neste ambiente um objeto-alvo da classe “micro-ondas”. O objeto-alvo é repassado em formato textual, podendo ser extraído a partir de uma sentença linguística como “Por favor, direcione-se até o micro-ondas”. Neste trabalho, o foco não está nos procedimentos para obtenção do objeto-alvo, como extração textual e outros, de maneira que partimos do momento em que o mesmo já é disponibilizado no formato esperado ao agente, isto é, o objeto-alvo “micro-ondas”.

O agente deverá iniciar sua execução no ambiente em busca do objeto-alvo, de acordo com as ações disponíveis e explorando o ambiente à medida que navega pelo ambiente. Não possuindo acesso a mapas topológicos, o agente deve guiar suas ações de acordo com observações visuais e entendimento semântico contextual da cena. A partir de imagens RGB capturadas do ambiente, objetos podem ser reconhecidos e inferências relacionais semânticas serem consideradas. Humanos inferem a possível presença de um “micro-ondas” como estando, de maneira mais provável, disposto em armários e sendo avisado, frequentemente, próximo a outros objetos correlacionados, como fogões e geladeiras. Aqui, nosso agente robótico deve comportar-se de modo similar, valendo-se do seu reconhecimento visual, informações semânticas *a priori* do seu modelo semântico e, por fim, pela recorrência e memória de ocorrências passadas onde tais correlações se demonstraram verdadeiras.

Em resumo, mesmo que partindo de um local qualquer, e sem acesso a nenhum mapa topológico ou métrico que descreva o ambiente, objetivamos que o nosso agente robótico possa encontrar um objeto-alvo qualquer, por meio de observações e da exploração do ambiente, traçando relações entre o objeto-alvo e outros objetos detectados em cena (por exemplo, “micro-ondas são dispostos em armários, próximos a fogões e geladeiras”), intensificando tais relações por meio de experiências passadas e dirigindo-se de maneira mais inteligente e eficiente ao seu objeto-alvo em novas ocasiões, localizando-o com sucesso.

Para o desenvolvimento da arquitetura proposta utilizou-se abordagens promissoras exploradas na literatura recente, as quais têm seu papel bem definido na arquitetura e são detalhadas ao longo deste capítulo. Algumas dessas abordagens são navegação visual, exploração de informações semânticas e contextuais, navegação baseada em aprendizado e redes de memória. A utilização de visão na área de robótica é estabelecida e aplicada de longa data. Contudo, os avanços em ConvNets recentemente alcançados vêm impulsionando a aplicação de tais modelos para navegação visual de robôs. Não obstante, arquiteturas de redes neurais profundas aplicadas em um contexto de ARP possibilitam

adotar uma abordagem de navegação baseada em aprendizado, onde um agente robótico aprende a navegar em um ambiente por meio de experiências, construindo uma política de navegação e um modelo de conhecimento. Além disso, a navegação baseada em aprendizado adotada neste trabalho dispensa o uso de informações como mapas topológicos ou mapas métricos, isto é, o agente robótico não possui informações sobre a composição como mapas explícitos do ambiente por meio de representações estruturais como grafos ou outras, sendo esta compreensão implicitamente desenvolvida durante o aprendizado.

Mais recente é o uso de grafos para construir compreensões semânticas e percepção de contexto utilizando GNNs e variantes, como GCNs. Relações entre objetos e contexto semântico do ambiente ao redor é de grande valor para construir agentes mais inteligentes, partindo-se de conhecimentos *a priori* e constantemente atualizando-se a compreensão do agente a partir de novas observações. Tal abordagem é explorada neste trabalho objetivando prover maior compreensão e domínio sobre o ambiente no qual o agente está inserido.

Finalmente, a arquitetura proposta neste trabalho busca utilizar memória no processo de navegação. A partir da arquitetura proposta em [Yang et al. \(2019\)](#), propomos expandi-lá por meio da inserção de mecanismos e redes de memória e investigar o impacto das mudanças ocasionadas neste modelo. Para fins de simplicidade, nos referiremos as arquiteturas de redes neurais GRU e LSTM como mecanismos e redes de memória.

A arquitetura proposta pode ser dividida nos seguintes módulos: módulo visual, semântico, de memória (nossa contribuição) e de aprendizagem. Na [Figura 12b](#) é ilustrado o modelo da arquitetura proposta e os referidos módulos. Também é apresentada a arquitetura base proveniente da proposta de [Yang et al. \(2019\)](#) para fins comparativos.

A navegação visual é iniciada pelo robô a partir da definição do objeto-alvo a ser encontrado no ambiente. O objeto-alvo é processado com fins de se obter um *word embedding* composto por objetos de significado próximo ou de valor contextual semelhante. Por exemplo, “lâmpada” e “abajur” podem ser utilizadas de maneira intercambiada à medida que podem ser compreendidas como o mesmo objeto em diversos contextos, podendo assim ter seus significados unificados. A cada passo dado pelo agente, uma captura atual do ambiente é processada por uma rede ConvNet para detecção de objetos. Tanto o *word embedding* quanto a saída da ConvNet são mapeados para novos vetores distintos, separadamente. Tais vetores são utilizados em dois momentos: primeiramente como entrada para o grafo GCN presente no módulo semântico (apenas o vetor saída da rede ConvNet), e em um segundo momento como parte do vetor unificado, repassado para a rede de memória (ambos vetores, tanto da ConvNet como do *word embedding*). Detalhes sobre esse processo são descritos nas [Subseção 4.1.1](#) e [Subseção 4.1.2](#).

O Módulo Semântico é constituído pela rede de grafo GCN, que recebe como entrada *word embeddings* preprocessados, descrevendo todos os objetos explorados no

Figura 12 – Arquitetura base de Yang *et al.* (2019), sem utilização de memória, e nossa arquitetura proposta, com módulo de memória

(a) Arquitetura base proposta em Yang *et al.* (2019)

Fonte: Adaptada de Yang *et al.* (2019).

(b) Arquitetura proposta para navegação semântica com memória

Fonte: Elaborada pelo autor.

modelo, e a saída mapeada da ConvNet. Esses vetores são utilizados pela GCN para codificar relações semânticas entre o conjunto total de objetos e os objetos detectados em cena. Assim como ocorre com a rede ConvNet, a cada passo do agente no ambiente, a rede de grafo GCN processará informações novas providas do módulo de visão, tendo como saída um vetor de 512-d que codifica as relações atuais.

No final do processamento, três vetores são obtidos: a) o vetor obtido diretamente do módulo de visão; e b) dois vetores provenientes do módulo semântico: um vetor de palavras para o objeto-alvo (*word embedding*) e outro vetor resultado da GCN. Esses vetores serão concatenados formando um único vetor, que será entrada para o módulo de memória.

O Módulo de Memória receberá o vetor unificado, que será processado por uma

rede de memória do tipo LSTM ou GRU. Desta forma, informações relativas aos objetos detectados em cena, representação de palavras relacionadas ao objeto-alvo e relações entre objetos visualizados e objeto-alvo são processados por uma rede de memória, levando em consideração todo contexto atual conjuntamente com as execuções e contextos vivenciados em passos anteriores. A saída deste módulo é um vetor de 512-d, o qual é recebido pelo método de ARP adotado, finalmente definindo uma ação a ser tomada pelo agente. Os módulos acima são descritos em detalhes a seguir.

4.1.1 Módulo Visual

Neste trabalho é adotada a abordagem de navegação utilizando-se métodos de visão computacional. O Módulo Visual tem por objetivo extrair informações das imagens do ambiente capturadas pelo agente, detectando objetos presentes em cena. Como comentado nos [Capítulo 2](#) e [Capítulo 3](#), redes neurais profundas como ConvNets vem sendo amplamente utilizados nos últimos anos com tais objetivos, obtendo bons resultados e sendo adotadas em diversos trabalhos.

A utilização de métodos de visão computacional para a atividade de navegação permite ao modelo fazer uso de câmeras como sensores, dispensando o uso de aparatos complexos ou custosos, ao mesmo tempo que oferecem uma enorme riqueza de informações sobre o ambiente, como detecção de objetos e suas dimensões. A detecção de objetos é de imensa importância para o trabalho aqui proposto, uma vez que adotamos uma abordagem objeto-direcionada, em busca de um objeto-alvo e inferimos contexto semântico a partir de relações entre objetos presentes nas cenas.

As informações da cena são extraídas adotando-se uma rede ConvNet do modelo ResNet-18. Este modelo é uma variante do modelo descrito na [Subseção 2.1.2](#) com um total de 18 camadas residuais. Originalmente, a rede ResNet-18 tem como saída um vetor de dimensão 1000-d, referente as 1000 classes de objetos presentes no conjunto de dados ImageNet ([RUSSAKOVSKY et al., 2015](#)).

Na presente arquitetura, o vetor resultante com 1000-d é mapeado utilizando-se operações convolucionais *pointwise* para um tensor de 512 mapas de características de tamanho 7×7 , o qual é novamente mapeado para 64 mapas de características de tamanho 7×7 . Finalmente, ele é utilizado para formação do vetor final que será a entrada para o módulo de memória. Para entrada da GCN é realizado um mapeamento utilizando-se uma camada totalmente conectada (fully connected network) do vetor final de 1000-d para um vetor de 512-d.

A ResNet-18 utilizada contém valores de pesos pre-treinados no conjunto de dados ImageNet ([RUSSAKOVSKY et al., 2015](#)), via transferência de aprendizado. Com exceção das duas últimas camadas da sua estrutura, todas as demais são congeladas, não tendo

assim os seus pesos modificados e retreinados durante a execução do agente.

4.1.2 Módulo Semântico

O Módulo Semântico é composto por dois elementos: um modelo para obter representações de palavras (*word embedding*) e uma rede de grafo GCN para extração de informações semânticas de relações entre objetos.

A representação de palavras é obtida no início da execução do agente, para todos os objetos considerados, nos quais estão inclusos os objetos-alvo, sendo computados uma única vez no início do processo utilizando o algoritmo GloVe (PENNINGTON; SOCHER; MANNING, 2014) com modelo pre-treinado, o qual nos retorna um vetor de palavras relacionadas. Por exemplo, para a palavra “Lâmpada”, um vetor como: {lâmpada, abajur, luminária, ...} seria retornado como resultado. Extraímos o vetor respectivo ao objeto-alvo a ser procurado pelo agente na execução atual a partir do conjunto de vetores computados para todos os objetos. É este vetor de palavras que é efetivamente utilizado pelo modelo proposto como representante da classe do objeto a ser encontrada no ambiente. Isto ocorre à medida que o objeto-alvo em cena pode ser considerado como quaisquer um destes objetos semanticamente semelhantes e pertencentes a essa representação de palavras. O vetor representativo de palavras tem dimensionalidade inicial de 300-d, sendo mapeado para um tensor de 64 mapas de características de tamanho 7×7 via uma camada convolucional *pointwise*. Este tensor é concatenado mais adiante para a formação do vetor unificado.

Ainda no módulo semântico, a GCN é responsável por prover-nos com uma representação das relações entre objetos. Neste trabalho, um total de 83 objetos podem ser encontrados no ambiente simulado, compondo-se assim um grafo com 83 nós-objetos. O modelo parte de um grafo de conhecimento *a priori*, construído por meio da base de dados *Visual Genome* (KRISHNA *et al.*, 2017), do qual obtemos uma matriz de adjacência entre objetos, como apresentada na Figura 13. Cada nó é populado com o vetor representativo de palavras do objeto representado em um nó e pela saída proveniente do módulo de visão. Enquanto o primeiro permanece imutável durante todo o processo e execuções do agente, a saída do módulo visual é atualizada a cada passo no ambiente. A saída do módulo de visão utilizado pela GCN é mapeada diretamente de 1000-d para um vetor de 512-d, não sendo a mesma representação enviada para a construção do vetor unificado. Por meio da constante atualização referente a saída do módulo de visão, GCN mantém-se atualizada da cena atual sendo observada pelo agente, recalculando-se as relações entre objetos.

O grafo final construído neste primeiro momento pode ser visualizado, de maneira ilustrativa, na Figura 13, que demonstra esse processo. Este grafo, populado com informações a respeito dos nós-objetos, das relações entre estes e da cena atualmente observada pelo agente, é utilizado como entrada para a rede GCN, que por meio da sua execução

Figura 13 – Procedimento para construção do grafo relacional objeto-objeto a ser computado pela GCN. Enquanto a matriz adjacente de objetos e a matriz com *word embeddings* é constante e computada uma única vez no início do processo, as informações providas do módulo de visão por meio da rede ConvNet são atualizadas a cada passo do agente no ambiente

Fonte: Elaborada pelo autor.

oferece como saída um vetor de valores que codifica as relações entre os nós-objeto. Esse segundo processo é ilustrado na [Figura 14](#). A rede GCN utilizada é composta por 3 camadas, que são seguidas por operações do tipo ReLu. A saída é um vetor de 83-d, respectivo aos 83 nós-objeto (que estão sendo considerados no nosso experimento), sendo em seguida mapeada para um vetor de 512-d.

O uso de uma rede GCN é de interesse pelo enriquecimento que este modelo de rede pode oferecer por meio da análise de contexto semântico de relações entre objetos. Por exemplo, a existência de uma cafeteira na cena pode significar a possível presença de uma caneca ou xícara. Esse conhecimento semântico entre os objetos e suas relações confere ao modelo informações interessantes, permitindo que o mesmo possa, por exemplo, especular sobre a presença de objetos mesmos que não tenham sido visualizados na cena corrente.

Vale observar que o grafo relacional entre objetos não tem correlações quaisquer com um mapa topológico ou métrico do ambiente que, como informado ao longo deste trabalho, não é utilizado no nosso modelo. Assim, nenhuma informação posicional sobre os objetos é extraída com esse grafo, o qual apenas infere como objetos se relacionam de maneira codificada no vetor de 512-d.

A implicação de um módulo semântico é de um maior potencial de generalização do modelo, seja pela expansão contextual provida pelo *word embedding* na diversificação

Figura 14 – Ilustração da execução da GCN dado um grafo inicial como entrada e tendo ao final uma codificação das relações entre nós-objetos

Fonte: Elaborada pelo autor.

dos diferentes usos linguísticos para um mesmo objeto, seja pelo enriquecimento semântico das relações entre objetos, não limitando o modelo unicamente às representações visuais do objeto-alvo e ampliando seu conhecimento sobre o contexto da cena.

Finalmente, após os processamentos descritos anteriormente, criamos um vetor conjunto das saídas provenientes dos módulos visual e semântico. Desta forma, os tensores de $64 \times 7 \times 7$ do módulo visual e do mapeamento do *word embedding* para o objeto-alvo são juntamente mapeados, uma última vez, para um único tensor de $64 \times 7 \times 7$. Então, este tensor é concatenado com o vetor de 512-d saída da GCN, resultando por fim em um único vetor concatenado de 3648-d, que é representação final de todos os processos operados pelo módulo visual e semântico. Este vetor por sua vez será fornecido como entrada para o módulo de memória, descrito a seguir.

4.1.3 Módulo de Memória

O Módulo de Memória tem por objetivo estabelecer e sedimentar conhecimentos obtidos por meio da exploração e experimentação no ambiente. Ele possui um aliado importante que é o Módulo de Aprendizado (detalhado na próxima subseção). Este módulo é sensível a informações sequenciais e recorrências, então sua finalidade é realizar o aprendizado de correlações entre cenas, objetos e resultados obtidos pela tomada de ações do agente. Na literatura, diversos trabalhos utilizam-se de mecanismos de memória para aprimorar modelos baseados em experiências passadas e ambientes onde não é possível ter-se uma observação completa.

Partindo de arquiteturas de navegação semântica já exploradas em outros trabalhos, a inclusão de um módulo de memória parece um caminho natural e de interesse, como é observado em diversos outros modelos que demonstraram melhoria dos resultados por meio da utilização de redes de memória. Contudo, comumente pode ser observada

a adoção de redes do modelo LSTM por padrão, sem análises comparativas com outras arquiteturas. Neste trabalho, exploramos tanto o uso da rede LSTM quanto GRU na constituição do Módulo de Memória. Em ambas opções, as redes foram definidas de maneira idêntica. Assim, o Módulo de Memória recebe como entrada o vetor final, resultante da concatenação das representações provenientes dos módulos anteriores, sendo de tamanho 3648-d. As redes de memória possuem então uma única camada oculta de 512 neurônios, e como saída um vetor de 512-d. Para fins práticos, este último vetor pode ser considerado, de maneira geral, como o espaço de estados do agente, bem como representação interna do estado do agente durante os episódios. Neste momento, a representação do agente proveniente do Módulo de Memória é repassada para o Módulo de Aprendizado para a escolha da ação a ser tomada pelo agente.

Em suma, o Módulo de Memória recebe informações sobre o estado atual do ambiente e considera conjuntamente ações e resultados obtidos em execuções passadas, observadas no ambiente pelos resultados do Módulo de Aprendizado. Entende-se que o Módulo de Memória seja modelado de acordo com as observações atuais de objetos e suas relações, em vistas das decisões tomadas em ocorrências passadas e seus resultados. Espera-se que a rede aprenda e reforce relações frequentes, dando maior peso a estas para a seleção de suas ações.

4.1.4 Módulo de Aprendizado

O Módulo de Aprendizado tem por função, dada todas as informações providas pelos módulos anteriormente descritos, decidir, executar e avaliar ações tomadas em um ambiente inicialmente desconhecido. Neste módulo será utilizado ARP (como descrito na [Seção 2.4](#)), adotando-se o algoritmo A3C ([MNIH et al., 2016](#)). A adoção de ARP tem a finalidade de aprender uma política de navegação por meio da exploração de um ambiente, executando ações de acordo com observações do estado atual.

A adoção de ARP prevê que uma política inicial se direcione para uma política ótima ao longo de repetidas execuções e exploração do ambiente. Os resultados recentes do uso de ARP em diversos trabalhos demonstram resultados promissores, capacitando agentes robóticos, simulados ou reais, a execução de atividades complexas. E o mais importante, sem a necessidade de definições quaisquer sobre o que deve ser realizado ou modelagem de políticas por humanos. Conjuntamente a isto, o surgimento de ambientes simulados fotorrealistas vêm possibilitando que o aprendizado obtido por métodos de ARP sejam transferidos com facilidade para ambientes reais, requerendo ocasionalmente apenas refinamentos.

O algoritmo A3C adotado neste trabalho segue o caminho realizado em trabalhos relacionados e recentes da área. Por meio deste algoritmo, o aprendizado pode ser paralelizado, permitindo maior rapidez na obtenção dos resultados ou na aprendizagem

concorrente de políticas específicas com objetivos distintos.

Neste trabalho, os métodos *Actor* e o *Critic* do A3C são definidos igualmente como camadas lineares, recebendo como entrada vetores de 512-d (obtida do Módulo de Memória), e tendo como saída um vetor referente as ações do agente, para o *Actor*, e um valor resultado da avaliação para o *Critic*.

Um agente é inserido em um ambiente simulado fotorrealista, não possuindo qualquer informação como mapas do ambiente (por exemplo, mapas topológicos ou métricos), recebendo observações pelo uso de uma câmera RGB, e podendo executar um conjunto discreto de ações. Um conjunto de 6 ações são possíveis de serem desempenhadas pelo agente. São elas: “mover-se adiante”, “rotacionar à direita”, “rotacionar à esquerda”, “Olhar para cima”, “Olhar para baixo” e “encerrar”. Este conjunto de ações é igual ao conjunto utilizado em [Yang et al. \(2019\)](#), exceto pelas ações “Olhar para cima” e “Olhar para baixo”. Todas as ações podem ser acionadas em qualquer estado do ambiente, não havendo um filtro que impeça o agente de selecionar apenas ações possíveis de serem efetivamente executadas em um determinado estado. Adicionalmente, nenhum procedimento para desvio de obstáculos é executado pelo agente, seja por rotinas no ambiente ou por modelagem na função de recompensa. A função de recompensa é detalhada na [Subseção 5.1.1](#).

4.2 Ambientes Simulados

A utilização de ARP como método para aprendizado impõe uma ampla exploração do ambiente até que resultados satisfatórios sejam obtidos, tornando-o de difícil aplicação diretamente em ambientes reais. Isto se deve ao fato que ambientes reais requerem preparação, procedimentos de recarga de robôs, sujeitados a danos em iterações iniciais, entre outros. Desta forma, é comum a utilização de ambientes simulados para aprendizagem utilizando AR ou ARP.

Um problema frequente na utilização de ambientes simulados é a lacuna de realidade (*reality gap*) quando se busca transferir e executar o conhecimento obtido em ambientes reais. Isso ocorre devido a maior complexidade existente em ambientes reais dada sua dinâmica, comportamentos físicos e sensores que possuem leituras ruidosas. Especificamente no caso de modelos que utilizam visão como sensor, como é o caso deste trabalho, simuladores podem apresentar imagens demasiadamente artificiais e que não são fidedignas aos ambientes reais.

Nos últimos anos, diversos esforços vêm sendo realizados na construção de simuladores de ambientes fotorrealistas. Tais simuladores se valem de *engines* comumente utilizadas por jogos, imagens reais dos ambientes ou texturização realista. Para este trabalho foram investigados simuladores fotorrealistas com ambientes domésticos. Os simuladores estudados foram os simuladores MINOS ([SAVVA et al., 2017](#)), AI Habitat ([SAVVA* et](#)

al., 2019) e AI2-THOR (KOLVE *et al.*, 2017).

O simulador MINOS é construído utilizando imagens das bases de dados SUNCG (SONG *et al.*, 2017) e *Matterport3D* (CHANG *et al.*, 2017), contendo 45.000 modelos tridimensionais. Ele permite estabelecer objetivos por meio de pontos no ambiente, objetos ou cômodos, onde o agente pode dispor de sensores de visão, profundidade, toque (sensibilidade a forças de contato) e segmentação semântica. No entanto, os ambientes não podem ser modificados, bem como não há simulações de física ou possibilidade de manipulação de objetos no ambiente.

O AI Habitat é um simulador de alto nível, composto pelos módulos Habitat-Sim e Habitat-API. O Habitat-Sim é responsável pela construção de um ambiente de simulação utilizando as base de dados SUNCG (SONG *et al.*, 2017), *Matterport3D* (CHANG *et al.*, 2017) e Gibson (XIA *et al.*, 2018). As bases são utilizadas para a construção do ambiente com imagens reais ou sintéticas das respectivas bases, permitindo neste a configuração de agentes e sensores. Adicionalmente, o módulo Habitat-API apresenta uma interface de alto nível para o desenvolvimento de aplicações, contento algoritmos de ARP embutidos, permitindo a seleção de tarefas, e configuração de treinamento por abordagens como IL e ARP, além de métricas de avaliação inclusas. De maneira similar ao MINOS, o AI Habitat não possui simulação de física ou interação do agente com objetos no ambiente, também não sendo possível a modificação da distribuição de objetos no mesmo.

Finalmente, o simulador AI2-THOR possui um ambiente simulado fotorrealista, utilizando como motor a *engine Unity*, sendo possível selecionar entre 120 cenários distribuídas em 4 ambientes diferentes (sala de estar, cozinha, quarto e banheiro), como apresentado na Figura 15. As categorias de ambientes possuem cenários com tamanhos distintos. De maneira geral, os ambientes da cozinha e da sala de estar apresentam cenários maiores que os dos ambientes quarto e banheiro. Contudo, há variações de tamanho internas a cada categoria, com cenários de tamanhos variantes. Um diferencial do ambiente AI2-THOR é que os objetos presentes nele são passíveis de realocação no ambiente e interações com o agente. Por exemplo, portas de prateleiras, geladeiras e micro-ondas podem ser abertas e fechadas, bem como diversos objetos podem ser movidos pelo agente, como xícaras, frutas, entre outros.

Também são simuladas propriedades físicas do ambiente e materiais dos objetos, que possuem massa, podem ter sua temperatura modificada, bem como seu estado contextual (alguns alimentos podem ser modificados de crus para cozidos). Assim como o AI Habitat, o simulador AI2-THOR possui uma interface de desenvolvimento em alto nível para comunicação com o ambiente simulado. Contudo, diferente dos demais simuladores comentados anteriormente, o AI2-THOR não permite a passagem de um agente robótico de um ambiente para outro, isto é, não há conexões entre ambientes distintos que possam ser navegados de maneira contínua como, por exemplo, partindo da cozinha o agente

Figura 15 – Cenas dos ambientes disponíveis para simulação no ambiente simulado AI2-THOR: sala de estar, cozinha, quarto e banheiro

(a) Ambiente da categoria sala

Fonte: Elaborada pelo autor.

(b) Ambiente da categoria cozinha

Fonte: Elaborada pelo autor.

(c) Ambiente da categoria quarto

Fonte: Elaborada pelo autor.

(d) Ambiente da categoria banheiro

Fonte: Elaborada pelo autor.

mover-se, passando por portas ou outras passagens, para um ambiente como sala de estar ou qualquer outro.

Neste trabalho o simulador AI2-THOR será adotado, uma que vez que este possui amplos recursos, diversidade de ambientes, simulação fotorrealista, API para desenvolvimento de alto nível para linguagem *Python* e com o fornecimento de uma vasta gama de informações sobre o ambiente. Adicionalmente, o AI2-THOR tem sido o ambiente de simulação selecionado pela maioria dos trabalhos relacionados apresentados no [Capítulo 3](#), possibilitando assim uma análise comparativa destes trabalhos.

4.3 Técnicas e recursos

Este projeto foi desenvolvido utilizando a linguagem de programação *Python 3.5*, uma linguagem de alto nível, de fácil desenvolvimento e manutenção, tendo disponível bibliotecas e *frameworks* de relevância na área de Aprendizado de Máquina e AR.

O *framework* *PyTorch* (PASZKE *et al.*, 2017) foi adotado para o desenvolvimento e aplicação de técnicas de AP e ARP. O *PyTorch* é amplamente utilizado pela comunidade, podendo ser utilizado com a linguagem *Python* e permitindo implementação de algoritmos, sejam em baixo ou alto nível. Os métodos e algoritmos para implementação de redes ConvNets, GCNs, LSTMs, GRUs e do ambiente de aprendizado com ARP foram implementados utilizando este *framework*.

Como dito na Seção 4.2, foi utilizado o ambiente de simulação AI2-THOR, que também possui API para linguagem *Python* e pode ser conjuntamente utilizado com o *PyTorch*.

Utilizamos como base para o nosso desenvolvimento o código disponibilizado em Wortsman *et al.* (2019), uma vez que este código possui uma reimplementação do trabalho de Yang *et al.* (2019).

4.4 Definição da tarefa

A definição de episódios, tarefas, cenários e objeto-alvo neste trabalho seguiu a definição apresentada em Yang *et al.* (2019) e Wortsman *et al.* (2019). Formalmente, um episódio ε é composto por um conjunto de passos e em um tempo t , onde $\varepsilon = \{e_1, e_2, \dots, e_t\}$. Em cada episódio, um cenário de um conjunto de cenários $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ e um objeto-alvo do conjunto de classes de objetos-alvo $\mathcal{O} = \{o_1, o_2, \dots, o_m\}$ são escolhidos para compor uma tarefa $\tau = (S, o, p)$, sendo p a posição inicial do agente.

A cada passo e no tempo t , o agente seleciona uma ação a do conjunto de ações $\mathcal{A} = \{\text{Mover-se adiante, Rotacionar à direita, Rotacionar à esquerda, Olhar para cima, Olhar para baixo, Encerrar}\}$. O episódio é terminado quando o agente aciona uma ação de término, ou quando o número máximo de passos por episódio é atingido. Caso o objeto-alvo encontre-se visível e suficientemente próximo do agente (definido como 1 metro de distância) quando a ação de término é executada, o episódio é concluído como bem sucedido. Caso contrário, o episódio é concluído como falho.

A dificuldade de uma tarefa pode variar de acordo com o objeto-alvo, cenário e posição inicial do agente. Tarefas com objetos-alvo menores, em locais com baixa visibilidade (obstrução por outros objetos, por exemplo) ou distantes da posição inicial do agente podem aumentar o nível de dificuldade da tarefa. Ainda, a depender da posição inicial do agente, que é escolhida de maneira aleatória, em diversos episódios o objeto-alvo não

estará de imediato no campo de visão do agente, requerendo vários passos e exploração para que aquele possa ser visualizado nas imagens capturadas do ambiente.

4.5 Métricas de avaliação

A arquitetura proposta foi avaliada por meio de comparação com outros métodos e modelos, como realizado em trabalhos semelhantes (YANG *et al.*, 2019; WORTSMAN *et al.*, 2019). As análises são compostas por observações quantitativas e qualitativas.

Em relação aos resultados quantitativos, avaliamos o total da recompensa obtida pelo agente ao decorrer do treino e gráficos referentes ao tamanho das trajetórias para as diversas modalidades de arquiteturas analisadas. Também são realizados testes comparativos entre os modelos: agente com política randômica; agente utilizando A3C puro; arquitetura base semelhante a proposta por Yang *et al.* (2019); e as variações da arquitetura proposta com inclusão do módulo de memória. Não são realizados comparativos diretos com métodos de planejamento de trajetória clássicos.

As métricas para este último conjunto de avaliações quantitativas seguem aquelas adotadas em outros trabalhos, sendo elas: a Taxa de Sucesso (TS) e o Sucesso ponderado pelo Tamanho da Trajetória (STT), propostas em Anderson *et al.* (2018a). A TS refere-se a quantidade de vezes que um agente navegou com êxito até o objeto-alvo. STT, por sua vez, considera a taxa de sucesso levando em conta o tamanho da trajetória percorrida pelo agente dado o ponto inicial até o ponto onde se encontra o objeto-alvo, definida como $\frac{1}{N} \sum_{i=1}^N S_i \frac{l_i}{\max(p_i, l_i)}$, onde N é o número total de episódios, S_i é um indicador binário de sucesso, e p_i e l_i são o tamanho da trajetória e o tamanho ótimo da trajetória no episódio i , respectivamente. Desta forma, a STT atua como uma métrica que avalia a eficiência do agente ao percorrer a trajetória.

Para cada métrica considerou-se 2 cenários: a) cenário onde todos os episódios de teste são considerados, sem nenhuma restrição; e b) cenário no qual se encontram apenas os episódios com tamanho do caminho $L \geq 5$. Este último cenário se apresenta como mais desafiador, a medida que desconsidera episódios demasiadamente curtos e ocasionalmente vantajosos para casos onde o agente se encontra próximo ao objeto-alvo.

Por fim, os resultados qualitativos são realizados por meio da execução dos modelos no ambiente simulado, podendo esta ser visualizada. Os resultados das análises, tanto qualitativas como quantitativas, são apresentadas no Capítulo 5.

4.6 Considerações finais

Como descrito ao longo deste capítulo, neste trabalho propusemos uma arquitetura de navegação visual semântica para robôs móveis. Para sua implementação foram utili-

zados métodos de visão computacional como redes ConvNets, grafos de conhecimentos por meio de GCNs e memória. Para implementação do mecanismo de memória valeu-se de redes de memória, como LSTM e GRU, e aprendizado de políticas de navegação utilizando ARP. Foram descritos vários ambientes de simulação fotorrealistas utilizados em trabalhos recentes, dentre os quais escolheu-se o ambiente AI2-THOR para a execução deste projeto.

A utilização de um ambiente simulado fotorrealista permite o treinamento do modelo de maneira rápida, segura e eficiente, tendo sido relatado em alguns trabalhos relacionados como possibilitador de algum nível de transferência do aprendizado obtido em ambiente simulado para ambientes reais.

O trabalho foi realizado utilizando recursos de software estabelecidos em seus segmentos e adotados pela comunidade de pesquisa.

A avaliação do modelo foi realizada utilizando métricas semelhantes a de outros trabalhos e de pertinência para o trabalho atual, como taxa de sucesso e taxa de sucesso ponderada pela trajetória percorrida, permitindo comparações do modelo proposto com outros modelos já presentes em trabalhos relacionados.

EXPERIMENTOS E RESULTADOS

Neste capítulo serão apresentados os experimentos realizados, sendo detalhados os recursos computacionais adotados, cenários de execução, métodos e variáveis utilizadas. Em seguida, os resultados obtidos serão submetidos a avaliação segundo as métricas adotadas e já descritas, por meio de gráficos e tabelas, além de exibições do modelo no ambiente simulado para análise qualitativa. Finalmente, serão discutidos os resultados e feitas análises sobre comportamentos dos modelos, partindo de métodos *baselines* e terminando com comparações com a influência da inclusão de memória.

5.1 Experimentos

Os experimentos realizados buscaram explorar os efeitos da inclusão do Módulo de Memória na arquitetura tomada como referência [Yang et al. \(2019\)](#), sendo executado em três possibilidades de modificações na arquitetura:

- o modelo mais próximo ao proposto originalmente por [Yang et al. \(2019\)](#), utilizando unicamente uma MLP, sendo a saída desta a representação do estado interno do agente;
- adotando como Módulo de Memória uma rede do tipo GRU;
- adotando como Módulo de Memória uma rede do tipo LSTM.

Efetivamente, a primeira variação é baseada no modelo original de [Yang et al. \(2019\)](#), não possuindo nenhum mecanismo de memória, enquanto as duas últimas são alteradas e estendidas pelo nosso trabalho com a adição de redes de memória.

Independente da arquitetura adotada, todos os modelos foram treinados com a mesma configuração do ambiente, recursos computacionais, funções de recompensa para os agentes de ARP, bem como avaliados sobre as mesmas métricas.

5.1.1 Configuração do ambiente

Como abordado no [Capítulo 4](#), o modelo proposto é direcionado para navegação em ambientes internos de residências. Neste trabalho foi utilizado o ambiente simulado AI2-THOR, o qual provê representações fotorrealistas de tais ambientes. Cada uma das quatro categorias do ambiente (sala de estar, cozinha, quarto e banheiro) possuem 30 cenários diferentes, as quais variam em sua planta baixa, decorações (pisos, paredes, cores), conjunto de objetos presentes no ambiente e, por fim, na disposição destes objetos. Tais conjuntos de cenários foram divididos em 20 para treino, 5 para validação e 5 para teste.

Ao todo são utilizados 83 objetos para a construção do grafo relacional utilizado pela GCN no módulo semântico. Destes, 44 objetos únicos podiam ser selecionados como objeto-alvo. Contudo, alguns objetos podem ser visualizados no contexto de ambientes diferentes, repetindo-se em diferentes categorias como, por exemplo, “Vaso de Plantas” e “Caixa”, que podem aparecer tanto em quartos como em cozinhas, sendo possível como objeto-alvo durante as execuções de ambos ambientes.

Assim, considerando as repetições, ao todo são 55 objetos, sendo 18 para os cenários da categoria cozinha, 10 para categoria sala de estar, 18 para quartos e 9 para banheiros. A [Tabela 2](#) contém os objetos e seus respectivos ambientes. Vale ressaltar que as diferentes categorias possuem variação no tamanho das cenários (por exemplo, cozinhas são, de maneira geral, maiores que banheiros), além de variações de tamanho entre os cenários de uma mesma categoria.

Tabela 2 – Lista de objetos disponíveis como objetos-alvo de acordo com as categorias de ambientes

Ambiente	Objetos
Cozinha	<i>Geladeira, Máquina de Café, Tigela, Vaso de Planta, Botão de Fogão, Pia, Tampo de Mesa, Batata, Pão, Tomate, Faca, Vasilha, Faca de Manteiga, Alface, Panela, Queimador de Fogão, Prato, Esponja</i>
Sala de estar	<i>Televisão, Caixa, Vaso de Planta, Cadeira, Tampo de Mesa, Roupa, Jornal, Molho de chaves, Vaso D’água, Cabide de pintura</i>
Quarto	<i>Vaso de Planta, Lâmpada, Livro, Alarme, Pintura, Telefone Celular, Interruptor, Vela, Tampo de Mesa, Estátua, Cartão de Crédito, Molho de Chaves, Tigela, Caneta, Caixa, Lápis, Persianas, Laptop</i>
Banheiro	<i>Pia, Papel Higiênico, Pote de Sabão, Interruptor, Vaso de Spray, Pintura, Vela, Suporte de Toalha, Relógio</i>

Para cada episódio, o objeto-alvo foi selecionado aleatoriamente sobre um conjunto de objetos-alvo possíveis. Este conjunto é composto por objetos contidos no grupo da respectiva categoria de ambiente do cenário, e garantida a existência de algum objeto visível ao agente neste cenário, não requerendo necessidade de alguma manipulação adicional (isto é, abrir gabinetes, geladeiras ou mover objetos). Em suma, em todos os episódios o agente procura por um objeto-alvo existente e visível, não ocorrendo episódios onde não haja nenhum exemplar do objeto-alvo procurado, permitindo assim que um episódio possa ter um término bem sucedido.

O conjunto $\mathcal{A} = \{\text{Mover-se adiante}, \text{Rotacionar à esquerda}, \text{Rotacionar à di-}$

reita, Olhar para baixo, Olhar para cima, Encerrar} define as ações que podem ser tomadas pelo agente no ambiente. Com exceção da ação **Encerrar**, as ações são reconhecidas e definidas pelo ambiente simulado de maneira discretizada, sendo passadas via API como comandos de alto nível, e executadas pelo simulador de maneira predefinida. A ação **Encerrar** não é inata do ambiente simulado, atuando como um sinal de encerramento do episódio a partir do agente e tratado no código desenvolvido.

A recompensa dada ao agente durante o treinamento foi estabelecida de maneira que a cada passo o agente recebesse uma punição no valor de -0,1. Punições para cada passo no ambiente são comumente aplicadas em trabalhos com temática semelhante, onde objetiva-se o aprendizado de trajetórias mais curtas. O agente pode acionar a ação **Encerrar** a qualquer momento na execução do episódio, terminando-o. Caso o agente se encontre, no momento do disparo da ação de encerramento, em uma distância do objeto igual ou inferior ao limite estabelecido (1 metro de distância), e este também esteja visível ao agente, o episódio é encerrado como bem-sucedido e o agente recebe uma recompensa positiva de 5. Caso contrário, se o agente não estiver próximo o suficiente do objeto ou o mesmo não estiver visível no seu campo de visão, o episódio termina com "falha" e o agente não recebe nenhuma recompensa positiva, tendo apenas o acúmulo das recompensas negativas dadas a cada passo. Outra possibilidade é a de o agente não acionar a ação de **Encerrar** em nenhum momento. Neste caso, o episódio é encerrado automaticamente ao atingir o limite máximo de passos permitidos, sendo de 30 passos durante o treinamento, recebendo o valor acumulado de punições no episódio e definindo este como um caso de "falha".

5.1.2 Configurações de treinamento

Todos os modelos foram treinados em um servidor dedicado com as seguintes configurações: processador Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz com 24 cores e 2 NVIDIA Tesla K20m GPU com 4743MiB de memória.

Foram treinados quatro configurações de modelos, sendo elas: a) modelo utilizando apenas o algoritmo A3C puro; b) modelo arquitetural com inclusão de GCN e MLP (GCN-MLP), semelhante ao proposto por [Yang et al. \(2019\)](#) e base para a arquitetura proposta; c) modelo da arquitetura proposta neste trabalho, com inclusão de Módulo de Memória, utilizando rede do tipo GRU (GCN-GRU); e d) modelo de arquitetura proposta neste trabalho, com inclusão de Módulo de Memória, utilizando rede do tipo LSTM (GCN-LSTM).

Para cada variação de arquitetura foram treinados 5 modelos distintos, em ambiente como definido na [Subseção 5.1.1](#) e tarefas como definidas na [Seção 4.4](#), totalizando ao final do treinamento 20 modelos (5 A3C, 5 GCN-MLP, 5 GCN-GRU e 5 GCN-LSTM). A inicialização de pesos foi feita de modo aleatório para cada bateria de treino e foi utilizado

a mesma semente (a mesma *seed*) para todos modelos durante uma mesma rodada para efeitos de comparação. Em [Yang et al. \(2019\)](#), foram treinados modelos distintos para cada categoria de ambiente, tendo sido comentado sobre o treinamento de um modelo único, porém não foram demonstrados resultados expressivos e nem detalhes. Em nosso trabalho, adotamos uma abordagem mais próxima da aplicada por [Wortsman et al. \(2019\)](#), onde um modelo geral foi treinado em todas as categorias de ambientes, isto é, durante cada execução de treino o agente era exposto a todos os ambientes, aprendendo ao final um único modelo por execução. A outra abordagem adotada em trabalhos correlacionados foi, a cada execução, serem treinados diferentes modelos para ambientes específicos, tendo-se ao final n modelos por execução, especializados cada qual em uma única categoria de ambiente. Adotamos a primeira abordagem devido ao fato de ser de maior interesse, a medida que apresenta-se mais desafiadora e propõe uma rede com maior poder de generalização. Não obstante, o treinamento de modelos especialistas para cada ambiente demanda maiores recursos computacionais e de tempo, sendo limitantes para o nosso trabalho.

Em suma, para cada arquitetura (A3C, GCN-MLP, GCN-GRU e GCN-LSTM) modelos foram treinados sendo expostos a todas as categorias de ambientes, tendo-se ao final um único modelo genérico. Os pesos era iniciados utilizando um mesmo valor como semente para todas as arquiteturas. Ao todo, foram realizadas 5 baterias de treino como descritas acima, alterando-se apenas o valor da semente entre as baterias. Desta forma, ao final do treino de todas as baterias, fora obtido 5 modelos genéricos de cada arquitetura treinado.

Para o treinamento do modelo, utilizamos como método de otimização o Adam ([KINGMA; BA, 2014](#)), com taxa de aprendizado fixada em 1^{-4} em todas as rodadas. Os modelos foram treinados até 3,5 milhões de épocas. A princípio, alguns treinos foram realizados até 6 milhões de épocas, porém não se verificou maiores mudanças a partir de 3,5 milhões de épocas, onde este último foi definido como máximo e utilizado para os modelos válidos. Foram utilizados 8 agentes durante os treinamentos, de maneira paralelizada.

5.2 Resultados e Discussão

Durante o treinamento, dados foram coletados para possibilitar a observação durante o processo e, adicionalmente, posterior análise como forma de avaliação quantitativa. Para a geração dos gráficos, os treinamentos foram agrupados pelo tipo de modelo arquitetural. Consideramos apenas os gráficos com os modelos GCN-MLP, GCN-GRU e GCN-LSTM, onde a GCN-MLP é estabelecida como *baseline*.

A [Figura 16a](#) exibe o gráfico de recompensa média acumulada ao longo dos episódios ao decorrer das épocas de treino. Podemos observar diferenças consideráveis entre os modelos comparados, não apenas quanto ao máximo de recompensa obtido, umas das

principais métricas de acompanhamento de treinamento de modelos de ARP, bem como do momento de início de convergência e variações. A GCN-MLP tem seu momento de maior crescimento no intervalo próximo entre 300 e 600 mil épocas, mantendo-se estável até o término do treinamento com 3,5 milhões de épocas, obtendo ao fim um valor médio de 2,33.

O modelo GCN-GRU apresenta período de crescimento na obtenção de recompensa mais tardio, não concentrando-se em uma única região. O modelo utilizando GRU como memória ainda apresentou grande variação durante os treinamentos, com execuções onde o modelo manteve-se em cerca 0,5 até próximo as 3 milhões de épocas. Na média, a GCN-GRU teve obtenção de recompensa final de 2,45, próximo ao obtido pela GCN-MLP. Entretanto, em alguns episódios foi obtido valores de recompensas mais altos, chegando até discretamente acima de 3,0 em seu máximo.

Finalmente, a GCN-LSTM demonstrou-se o modelo com maior acumulo de recompensa, alcançando valor médio de 3,98. Não obstante, este modelo tem o início de sua convergência mais precoce que os demais, apresentando uma taxa elevada até cerca de 1 milhão de épocas, e se mantendo em crescimento até o fim do treinamento. Além disso, é possível perceber que a GCN-LSTM apresentou desempenho estável entre os treinos, tendo pouca variação e com evolução previsível.

Outra informação verificada durante o treinamento e de interessante análise é o tamanho médio das trajetórias percorridas durante os episódios, apresentado no gráfico da [Figura 16b](#). Diferentemente do visualizado na [Figura 16a](#), neste as médias dos modelos GCN-MLP e GCN-GRU apresentam maior distância, sendo de 6,69 e 8,44, respectivamente. A GCN-LSTM alcança o maior valor, chegando a 9,79 passos dados em média por episódio. Assim como no gráfico da [Figura 16a](#), a GCN-MLP e a GCN-LSTM apresentam comportamentos estáveis, enquanto a GCN-GRU varia significativamente durante o treinamento. É possível observar que o aumento da trajetória no início do treinamento coincide com o crescimento no valor de recompensa obtido. Tal correlação se deve ao crescimento da trajetória estar relacionado com maior exploração do ambiente, além de permitir encontrar objetos-alvo mesmo quando o agente não é inicializado nas suas proximidades.

A GCN-GRU tem sua média final de tamanho de trajetória maior que a GCN-MLP, mesmo tendo obtido média de recompensa semelhante. Não obstante, nos melhores casos de treino, a GCN-GRU obteve valores próximos aos obtidos pela GCN-LSTM, percorrendo caminhos tão longos quanto esta última. A diferença ocasionada pela possibilidade de percorrer trajetórias maiores é um comportamento mais desejável por parte dos agentes, evitando termos precoces do episódio. Essa característica é ainda mais necessária aos modelos quando o ambiente pertence as categorias sala de estar e cozinha, os quais geralmente apresentam tamanho maior que os demais. Ainda, a necessidade de observações outras e não apenas o acompanhamento das recompensas obtidas é ressaltada, permitindo

Figura 16 – Gráficos de média total de recompensa, tamanho de trajetória e disparo de ação encerrar, agrupado por modelo, durante o treinamento

(a) Recompensa total média

(b) Tamanho da trajetória média

(c) Disparo médio da ação Encerrar

Fonte: Elaborada pelo autor.

melhor entendimento sobre os resultados qualitativos apresentados adiante neste capítulo.

Na Figura 16c exibimos o gráfico referente ao disparo da ação **Encerrar** realizadas pelo agente. Como explicado anteriormente, a ação **Encerrar** é acionada pelo agente como forma de terminar o episódio. Uma política ótima é de tal maneira que o disparo desta ação seja sempre ocasionada pela constatação que o objeto-alvo tenha sido encontrado, esteja próximo ao agente e visível. Contudo, episódios ditos como falhas são ocasionados pelo disparo equivocado desta ação pelo agente, ou ainda pela ausência do disparo, levando ao término do episódio pelo alcance do limite máximo de passos no ambiente. Podemos ob-

servar que no início do treinamento, independente do modelo, todos os agentes terminam seus episódios por meio da ação de término. Contudo, ocasionalmente os agentes iniciam um comportamento de não mais executarem a ação **Encerrar**, prosseguindo no ambiente até o término pelo limite máximo. Tal comportamento leva os agentes a uma inevitável exploração do ambiente, mesmo que sem retorno de recompensa positiva ao fim. É neste momento que podemos observar também uma correlação com os demais gráficos apresentados, onde os agentes aumentam o tamanho de suas trajetórias e começam a acumular maiores recompensas.

Posteriormente, os agentes apresentam aumento na frequência de acionamentos de encerramento, isto é, os agentes voltam a não terem mais os seus episódios compulsoriamente encerrados pelo limite máximo de passos, mas sim por escolha da ação **Encerrar** de acordo com sua política. No caso da GCN-LSTM, o modelo mantém-se aproximadamente com trajetórias do mesmo tamanho, enquanto a GCN-MLP apresenta uma queda no tamanho de suas trajetórias. Com o desenvolvimento de uma melhor política ao longo do treinamento, os agentes alteram a característica do seu disparo de ação **Encerrar**, de um disparo precoce e incerto, no início, para disparos mais confiantes de que aquela ação seria a melhor, ao final.

Contudo, é possível observar que, mesmo após retornar a disparar a ação de **Encerrar** neste segundo momento, a ocorrência de terminos precoces dos episódios continua atingindo o modelo GCN-MLP, comprometendo seu desempenho em cenários onde os objetos-alvo encontram-se distantes da posição inicial do agente. Por fim, apesar da instabilidade, a GCN-GRU, de maneira geral, apresenta ao decorrer de todo o treino uma menor queda da média dos disparo das ações de **Encerrar**, ao fim apresentando uma taxa média inferior aos outros modelos. Todavia, a GCN-GRU apresenta tamanho médio de trajetória superior a GCN-MLP, tendo um resultado intermediário entre esta e a GCN-LSTM.

5.2.1 Análise Quantitativa

Para a análise quantitativa vários testes também foram realizados por meio da utilização das métricas apresentadas na [Seção 4.5](#). Para esta análise, os modelos de conhecimento foram avaliados nos cenários de teste. A avaliação considerou 250 execuções para cada categoria de ambiente, variando entre os cenários do conjunto de teste e os objetos-alvo. Ao todo, cada modelo foi executado e avaliado quanto a Taxa de Sucesso (TS) e ao Sucesso ponderado pelo Tamanho da Trajetória (STT) em 1000 tarefas diferentes (como definido na [Seção 4.4](#)). Nesta análise foram consideradas as seguintes estratégias: a) modelo randômico; b) modelo com algoritmo A3C puro; c) GCN-MLP; d) GCN-GRU; e e) GCN-LSTM.

Como também citado na [Seção 4.5](#), duas formas de avaliação foram consideradas.

A primeira considerando todos os episódios sem nenhuma imposição de tamanho de trajetória mínimo; e uma segunda forma, onde apenas tarefas que cumprissem a restrição de possuir $L \geq 5$, isto é, o caminho até o objeto-alvo devia ter, ao menos, uma trajetória de 5 passos. Esta última, devido a restrição imposta, apresenta maior dificuldade à medida que é requerido ao agente ter aprendido uma política mais exploratória e capaz de encontrar objetos-alvo mais distantes. A [Tabela 3](#) exibe os resultados para todos os modelos avaliados em ambos os casos de avaliação considerados. São apresentadas as médias de todos os modelos treinados e, em parenteses, o desvio padrão.

Tabela 3 – Tabela de resultados com médias para STT e TS para todos os modelos. Os resultados são apresentados para “todos” os casos e casos com a restrição $L \geq 5$. Em parenteses é informado o desvio padrão

	Todos		$L \geq 5$	
Modelo	STT	TS	STT	TS
Randômico	4.35	10.20	0.03	0.18
A3C	$6.38_{\pm(0.25)}$	$14.76_{\pm(1.32)}$	$0.06_{\pm(0.22)}$	$1.94_{\pm(0.77)}$
GCN-MLP	$7.29_{\pm(0.42)}$	$15.38_{\pm(0.73)}$	$1.68_{\pm(0.17)}$	$3.95_{\pm(0.17)}$
GCN-GRU	$8.16_{\pm(1.20)}$	$18.64_{\pm(4.88)}$	$3.08_{\pm(1.57)}$	$6.10_{\pm(3.51)}$
GCN-LSTM	$11.62_{\pm(0.88)}$	$26.32_{\pm(2.18)}$	$6.84_{\pm(1.38)}$	$12.93_{\pm(2.75)}$

Considerando primeiramente o cenário “Todos”, podemos perceber que o agente com modelo de políticas randômicas apresentou os piores resultados em ambas as métricas. Com a inclusão de métodos de ARP (A3C), seguidos pela adição de GCN (GCN-MLP) e, finalmente, redes de memória GRU e LSTM (GCN-GRU e GCN-LSTM), é possível observar um aumento gradual dos valores de avaliação. Todos os modelos apresentaram melhores resultados na avaliação unicamente da TS se comparados com a STT. Ainda avaliando para o cenário “Todos”, a GCN-LSTM apresenta os melhores resultados para STT e TS, com 11,62 e 26,32, respectivamente, seguida pela GCN-GRU com 8,16 para STT e 18,64 para TS, ambas superando a GCN-MLP, com STT de 7,29 e TR 15,38.

O cenário com restrição onde $L \geq 5$, assim como o cenário “Todos”, apresentou os piores resultados para o agente com políticas randômicas, gradualmente tendo incrementos a medida que os modelos eram acrescidos com ARP, GCN e redes de memória. Novamente, os melhores resultados foram apresentados pela GCN-LSTM, com STT de 6,84 e TS de 12,93, seguido pela GCN-GRU com 3,08 e 6,10, respectivamente, para STT e TS, superando assim a arquitetura sem memória GCN-MLP.

É possível observar uma queda em todos os modelos quando comparados aos valores obtidos no cenário “Todos”. Como já anteriormente pontuado, a restrição neste cenário, descartando casos onde $L < 5$, impõe maior dificuldade aos agentes, a medida que estes não se beneficiam de ocorrências onde o objeto-alvo encontrava-se próximo ao ponto inicial do agente. Desta forma, os modelos precisam aprender políticas que possibilitam ao

agente uma maior capacidade exploratória, retornando melhores resultados para modelos que evitam termos precoces.

Todavia, podemos constatar que os modelos acrescido de memórias sofreram uma redução inferior na avaliação de suas métricas, principalmente em relação a STT. A GCN-LSTM, por exemplo, apresenta STT de 11,62 e 6,84 para o cenário “Todos” e $L \geq 5$, respectivamente, uma queda de aproximadamente 41,0%. A GCN-GRU tem queda de 62,2%, de 8,16 para 3,08 entre os cenários. A GCN-MLP, sem mecanismo de memória, tem a maior redução entre os modelos com GCN, de 7,29 para 1,68 para STT, cerca de 76,9%.

Desta forma, podemos verificar que os modelos propostos neste trabalho com mecanismos de memória, em especial a GCN-LSTM, sofreram menor impacto em seus resultados no cenário mais desafiador, apresentando menores quedas de desempenho, fator de grande importância para aplicações em ambientes reais.

Além disso, é possível perceber que a GCN-GRU sofre grande variação, como comentando anteriormente nas análises dos gráficos da [Figura 16](#). Mesmo considerando o melhor valor para GCN-GRU comparativamente com o pior resultado da GCN-LSTM, observamos que a última obteve melhores resultados, fundamentando sua melhor posição nos resultados quantitativos apresentados na [Tabela 3](#). A instabilidade apresentada pela GCN-GRU, apesar dos seus bons resultados em alguns cenários, como poderá ser visualizado nos resultados qualitativos, gera incerteza durante o treinamento de arquiteturas utilizando esse modelo.

5.2.2 Análise Qualitativa

Uma análise dos resultados qualitativos fora realizada permitindo aferir visualmente o comportamento dos agentes de acordo com as políticas de cada modelo. Para tanto, foi selecionado o modelo treinado com melhor desempenho dentre as arquiteturas. Consideramos apenas os modelos GCN-MLP, GCN-GRU e GCN-LSTM para esta análise. O limite máximo de passos permitido aos agentes foi acrescido durante os testes, sendo definido como 200 passos, independente da categoria do ambiente. A [Figura 17](#) exibe as imagens, para cada teste executado, da primeira captura obtida pelo agente na posição inicial do episódio e a visão do ambiente do ponto de vista do topo. Nesta última, o agente encontra-se identificado pelo círculo azul, com o seu campo de visão representando por um triângulo branco translúcido. Os possíveis objetos-alvo dispostos no ambiente foram marcados com círculos verdes. Reforçamos que apenas a imagem do ponto de vista do agente é acessível aos modelos durante o treino e teste, isto é, a imagem do ambiente visto do topo não é acessível para o agente e foi gerada apenas para fins de visualização e demonstração da organização do ambiente e do posicionamento do agente e dos objetos-alvo. É possível observar que não há nenhuma regra estabelecida quanto ao agente ser disposto

em posições iniciais com o objeto-alvo já presente em seu campo de visão, assim como não há nenhuma regra que impeça este evento, a medida que a posição inicial do agente é selecionada aleatoriamente. Os ambientes e objetos-alvo, respectivamente, são: Sala de estar e cadeira; quarto e livro; banheiro e toalha; cozinha e panela; e quarto e telefone celular.

Em todos testes apresentados a seguir, são exibidos os últimos 5 passos dados pelos agentes dos respectivos modelos, juntamente com as ações. As ações são exibidas abreviadas, por motivos de limitação de espaço, como segue: Mover-se adiante (M. A.), Rotacionar à direita (R. D.), Rotacionar à esquerda (R. E.), Olhar para baixo (O. B.), Olhar para cima (O. C) e Encerrar (E.). A ação indicada no tempo t é a selecionada a partir da representação de estado naquele momento, sendo efetivamente executada em $t + 1$, isto é, a imagem posterior é resultado da ação definida no momento anterior. O último quadro é colorido de acordo com o resultado da execução, onde o fundo em verde representa um episódio bem sucedido, enquanto quadros com fundo em vermelho indicam episódio com "falha".

Na [Figura 18](#) é exibido um caso para um ambiente da categoria sala de estar, tendo como objeto-alvo uma "Cadeira". Neste cenário, os agentes com modelos GCN-GRU e GCN-LSTM obtiveram êxito em identificar visualmente o objeto-alvo, tendo terminado o episódio com o disparo da ação **Encerrar**, em posição próxima o suficiente do objeto, isto é, tendo concluído a tarefa com êxito. No entanto, o agente com o modelo GCN-MLP não obteve êxito, permanecendo praticamente durante todo o episódio na posição inicial, realizando ações de movimento circulares no próprio eixo. Nesta execução, a GCN-GRU e a GCN-LSTM realizaram trajetórias com 23 e 27 passos, respectivamente. Ambos os modelos executaram durante quase todo o episódio apenas passos necessários, direcionando-se corretamente até o alvo, com melhor desempenho do modelo GCN-GRU. A GCN-MLP deu 14 passos, tendo estes, como dito anteriormente, não direcionando o agente a uma conclusão do episódio bem sucedida.

A tarefa da [Figura 19](#) tem como ambiente um quarto e objeto-alvo um "Livro" (em azul, localizado sobre a cadeira). Neste cenário, tanto a GCN-GRU quanto a GCN-MLP falharam para concluir a tarefa. Ambos modelos tem uma cama como objeto predominantemente visível na última captura da cena. A GCN-GRU executou diversas ações de **Mover-se adiante**, de maneira a tentar atravessar a cama. Ao todo, o agente GCN-GRU deu 200 passos, não tendo disparado em nenhum momento a ação de **Encerrar**, atingindo o limite máximo de passos permitidos. A GCN-MLP, em direção oposta, acionou a ação de **Encerrar** precocemente, tendo efetuado apenas 5 passos. O único agente a obter êxito neste cenário foi aquele com políticas providas pela GCN-LSTM, totalizando 17 passos, direcionando-se efetivamente ao objeto-alvo e ocasionalmente explorando o ambiente, encerrando com êxito ao visualizar o livro posicionado sobre a cadeira.

Figura 17 – Imagem da captura inicial obtida pelo agente na posição de início e da visão do ambiente do ponto de vista do topo. Na visão do topo, o agente é marcado pelo círculo azul, enquanto objetos-alvo válidos são marcados como círculos verdes

Fonte: Elaborada pelo autor.

Figura 18 – Comparativo de trajetórias dos agentes com modelo GCN-LSTM, GCN-GRU e GCN-MLP. A categoria de ambiente para este episódio foi sala de estar, tendo “Cadeira” como objeto-alvo. GCN-LSTM e GCN-GRU tiveram episódios concluídos com sucesso, enquanto a GCN-MLP falhou

Fonte: Elaborada pelo autor.

A categoria banheiro foi explorada na [Figura 20](#), com objeto-alvo “Toalha”. Dois objetos-alvo válidos podiam ser encontrados neste ambiente, relativamente próximos um do outro: uma toalha branca, presa a um suporte na parede, entre o armário e a banheira; e uma toalha em um suporte na parede, ao lado do armário no lado oposto a banheira. A GCN-MLP falhou neste episódio, tendo o agente efetuado movimentos de rotação e se movido, a partir do momento $t = -3$, olhando para o teto do recinto. O encerramento do episódio deu-se nesta mesma direção, não tendo em sua visão nenhum exemplar do objeto-alvo. Novamente, ambos os modelos com memória obtiveram êxito, tendo cada qual encontrado um exemplar diferente da toalha. A GCN-GRU concluiu o episódio com uma trajetória de tamanho 7, enquanto a GCN-LSTM terminou o episódio após 21 passos.

O objeto-alvo “Panela” foi definido para o ambiente cozinha, demonstrado na [Figura 21](#). Neste caso, todas as redes obtiveram êxito, encontrando o objeto-alvo localizado sobre o fogão. Diferentemente dos demais exemplos demonstrados até então, a GCN-MLP obteve o melhor resultado, percorrendo o menor caminho até o encerramento do episódio, um total de 11 passos. A GCN-GRU, com 20 passos, teve o segundo melhor resultado e a GCN-LSTM, por fim, a trajetória mais longa, com 55 passos. Nesta tarefa, a panela encontra-se relativamente próxima aos agentes, apesar de não estar visível desde o início e necessitar de algum grau de observação exploratória. Adicionalmente, o ambiente

Figura 19 – Comparativo de trajetórias dos agentes com modelo GCN-LSTM, GCN-GRU e GCN-MLP. A categoria de ambiente para este episódio foi quarto, tendo “Livro” como objeto-alvo. Apenas a GCN-LSTM concluiu o episódio com sucesso, enquanto a GCN-GRU e GCN-MLP falharam

Fonte: Elaborada pelo autor.

Figura 20 – Comparativo de trajetórias dos agentes com modelo GCN-LSTM, GCN-GRU e GCN-MLP. A categoria de ambiente para este episódio foi banheiro, tendo “Toalha” como objeto-alvo. GCN-LSTM e GCN-GRU tiveram episódios concluídos com sucesso, enquanto a GCN-MLP falhou

Fonte: Elaborada pelo autor.

Figura 21 – Comparativo de trajetórias dos agentes com modelo GCN-LSTM, GCN-GRU e GCN-MLP. A categoria de ambiente para este episódio foi cozinha, tendo “Panela” como objeto-alvo. Todos os modelos foram exitosos

Fonte: Elaborada pelo autor.

deste cenário apresentava um tamanho consideravelmente menor que a regra geral para ambientes do tipo cozinha, o que pode justificar o bom desempenho da GCN-MLP. De maneira geral, é possível perceber, como discutido em relação ao gráfico da [Figura 16b](#), bem como os demais exemplos qualitativos exibidos anteriormente, que a GCN-LSTM costuma percorrer trajetórias mais longas, explorando o ambiente mais extensivamente, até mesmo quando explorações menores seriam suficientes.

Finalmente, o último cenário explorado foi a categoria quarto, tendo como objeto-alvo um aparelho “Telefone Celular” (localizado sobre a escrivaninha, entre o livro e o *laptop*), apresentado na [Figura 22](#). Apesar do ambiente pequeno, neste cenário foi selecionado um objeto mais desafiador pelo seu tamanho reduzido. Além de pequeno, o objeto encontra-se ocultado por uma cadeira, não sendo visível ou facilmente identificado sem alguma exploração do ambiente e ações de aproximação pelo agente. A GCN-LSTM não obteve êxito neste episódio, tendo executado uma trajetória longa, 87 passos, explorando os arredores, porém sem direcionar-se ao objeto-alvo. A GCN-GRU concluiu a tarefa de maneira bem sucedida, com um total de 65 passos dados. Finalmente, o melhor resultado foi obtido com a GCN-MLP, tendo efetuado apenas 9 passos, direcionados efetivamente em direção ao objeto-alvo.

Figura 22 – Comparativo de trajetórias dos agentes com modelo GCN-LSTM, GCN-GRU e GCN-MLP. A categoria de ambiente para este episódio foi quarto, tendo “Telefone celular” como objeto-alvo. GCN-GRU e GCN-MLP obtiveram sucesso. Apenas a GCN-LSTM falhou

Fonte: Elaborada pelo autor.

5.2.3 Discussão dos resultados obtidos

De maneira geral, os resultados apresentados partiram desde análises realizadas durante o treinamento dos modelos, até testes quantitativos e qualitativos. Os gráficos apresentados na [Figura 16](#), coletados durante o treino, mostraram-se correlacionados com comportamentos dos modelos treinados e posteriormente submetidos aos casos de testes, permitindo avaliar e observar a evolução de comportamentos nos agentes, como disparo da ação de encerramento e tendências no tamanho de trajetórias.

Os testes quantitativos avaliaram a robustez das arquiteturas avaliadas, nos diversos cenários e com dificuldades diferentes. As métricas de TS e STT permitiram avaliar o quanto os agentes foram bem sucedidos e, ainda, o quão eficiente foram suas trajetórias, sendo métricas utilizadas em outros trabalhos relacionados e ajudando em análises comparativas com estes. Por meio destas métricas pôde ser observado que os modelos arquiteturais desenvolvidos neste trabalho e estendidos com o módulo de memória, GCN-GRU e GCN-LSTM, apresentaram comportamento mais robustos, mantendo melhores resultados quando expostos a casos de teste de maior dificuldade, com objetos-alvos mais distantes e requeirindo maior exploração do ambiente. Não obstante, esses modelos apresentaram os melhores resultados em ambos cenários avaliados, como demonstrado na [Tabela 3](#), superando o modelo sem memória GCN-MLP.

Nos testes qualitativos foi possível observar como os modelos arquiteturais estendidos com memória alcançaram sucesso em mais casos, geralmente percorrendo trajetórias mais longas e, conseqüentemente, explorando mais o ambiente. Esta observação é condizente com os gráficos da [Figura 16b](#) de tamanho de trajetórias e com a maior manutenção dos resultados para casos com a condição $L \geq 5$, apresentados na [Tabela 3](#). Foi possível ainda observar as diferenças entre os modelos de maneira visual e intuitiva, para diferentes ambientes e objetos-alvo. Os testes qualitativos apresentados neste trabalho podem ser visualizados com sua execução completa por meio do vídeo disponibilizado no endereço <https://youtu.be/HCjpsOsZOpo>.

Independente do modelo avaliado, algumas limitações foram observadas, como a dificuldade dos modelos em assimilar limitações físicas do ambiente. Em diversos momentos das avaliações qualitativas os agente tentaram transpassar objetos, insistindo em ações sem sucesso (por exemplo, mover-se adiante estando obstruído por um objeto ou parede). Baseando-se apenas em uma câmera RGB comum, e não possuindo a modelagem de sua recompensa de maneira a considerar colisões, tal limitação não é estranha aos resultados, sendo um tópico de exploração e aprimoramento nos modelos. Além disso, foi possível observar que em alguns episódios as redes GCN-GRU e GCN-LSTM desempenharam trajetórias demasiadamente longas, afastando-se de um desempenho ótimo. Exemplos deste comportamento é a GCN-GRU na execução exibida na [Figura 19](#), que atingiu o limite máximo de 200 passos (e terminando sem sucesso) e a GCN-LSTM na [Figura 22](#), que percorreu 87 passos, sem sucesso, enquanto a GCN-MLP terminou o episódio exitosamente com apenas 9 passos. Desta forma, uma investigação maior entre exploração e conclusão dos episódios deve ser investigada, de maneira a tornar mais eficiente as redes com memória, sem apresentar os términos precoces recorrentes da arquitetura sem memória, GCN-MLP.

Finalmente, as redes com inclusão de Mecanismos de Memória conseguiram execuções bem sucedidas em todas as categorias de ambientes, alternando resultados com êxitos entre GCN-GRU e GCN-LSTM, e ambas nos exemplos das [Figura 18](#), [Figura 20](#) e [Figura 21](#). Os diversos resultados explicitados ao longo deste capítulo demonstram efeitos positivos nas políticas aprendidas pelos modelos acrescidos com memória, acarretando melhores execuções dos agentes no ambiente, maior exploração do cenário, evitando términos precoces e com êxitos em mais casos. Somado a isso, a investigação explorando memória com redes do tipo GRU e LSTM demonstrou que, apesar de corriqueiramente esta última ser escolhida por padrão, as GRUs podem obter resultados semelhantes as LSTMs, tendo uma arquitetura mais simples e, em muitos problemas, requerendo recursos computacionais inferiores. Em diversos trabalhos, este tipo de análise foi realizada com problemas de geração de sentenças linguísticas, enquanto em nosso trabalho conseguimos demonstrar que as mesmas ideias podem ser utilizadas em outro contexto, isto é, demonstramos a aplicação destas para ambientes simulados objetivando navegação e baseando-se

em imagens e contexto semântico.

Ao longo do capítulo foi detalhado o ambiente de treinamento e testes adotado neste trabalho. Foram especificados as categorias de ambientes e os objetos-alvo possíveis para cada ambiente, e como as rotinas de treinamentos foram executadas. A análise dos modelos treinados foi realizada com testes quantitativos e qualitativos. Em relação ao primeiro, gráficos foram apresentados, demonstrando como os agentes se comportaram ao longo do treinamento, possibilitando a visualização do desenvolvimento de sua política e evolução dos comportamentos emergentes. Tais gráficos foram posteriormente correlacionados com os resultados obtidos pelas métricas de avaliação TS e STT, adotadas em trabalhos relacionados e permitindo mensurar a taxa de sucesso e eficiência em tamanho de trajetória dos agentes. Foram submetidos a estas análises um agente com política randômica, um agente com apenas o algoritmo A3C puro, e como *baseline* o modelo proposto em Yang *et al.* (2019), conjuntamente com as alterações propostas em nosso trabalho, com os modelos acrescidos com o módulo de memória GCN-GRU e GCN-LSTM.

As métricas TS e STT foram calculadas em duas modelagens de problemas: considerando todas as execuções e uma outra, limitando apenas a episódios onde a distância do agente aos objetos fosse superior a 5 passos, considerado de maior dificuldade. Foi demonstrado que os modelos com memória tiveram os melhores resultados entre todos os modelos. Ainda, a GCN-LSTM apresentou a menor queda de desempenho dentre os modelos em relação aos valores de STT no cenário mais desafiador.

Por fim, os modelos foram observados em testes qualitativos, sendo possível visualizar o comportamento dos agentes nos ambientes, com diversos objetos-alvos e em diferentes cenários, seguindo as políticas aprendidas para os modelos GCN-MLP, GCN-GRU e GCN-LSTM. Novamente os resultados obtidos nas análises anteriores foram reafirmados, onde os modelos com memória obtiveram êxito em episódios onde a GCN-MLP falhou. Não obstante, as características mais exploratórias dos modelos GCN-GRU e GCN-LSTM se fizeram valer em ambientes maiores. O código-fonte deste trabalho encontra-se disponível no endereço <<https://github.com/IuryBAS/NavVisualSemanticMemory>>.

5.3 Considerações finais

Neste capítulo, foram descritos sobre os experimentos realizados, sendo detalhadas as configurações de treinamento e de ambiente, bem como os recursos computacionais utilizados neste trabalho. Foi apresentado o conjunto de objetos-alvo adotados durante as execuções, as categorias de ambientes do simulador, o conjunto de ações possíveis ao agente de ARP e explicitada a definição da função de recompensa.

Os experimentos foram analisados, tendo seus resultados apresentados e discutidos ao longo do capítulo. As avaliações quantitativas foram apresentadas por meio de gráficos

para os modelos de navegação semântica com e sem memória, e tabelas com valores obtidos para as métricas STT e TS para agentes randômicos, modelo A3C puro, GCN-MLP, GCN-GRU e GCN-LSTM. Os resultados apresentaram interessantes informações sobre a curva de aprendizado dos modelos, conjuntamente com melhores resultados obtidos pelas arquiteturas acrescidas de memória, sendo esta a principal contribuição deste trabalho.

Finalmente, foram apresentados e discutidos resultados qualitativos, com mosaicos visuais de imagens da execução dos modelos treinados no ambiente simulado para as diferentes categorias de ambientes e com diversidade de objetos-alvo, permitindo uma visualização intuitiva sobre o comportamento dos agentes na prática. Não obstante, algumas limitações foram apresentadas, sendo campo para estudos e aperfeiçoamentos futuros.

CONCLUSÃO

A robótica vêm se expandindo em nossa sociedade em ambientes sociais, hospitalares e domésticos, não mais limitada a aplicações estritamente industriais e em ambientes controlados. Neste novo âmbito, os agentes robóticos precisam lidar com ambientes mais complexos, de características mutáveis e, por vezes, imprevisível. Tais desafios exigem que técnicas como compreensão, percepção, manipulação e navegação sejam aprimoradas e potencializadas.

O presente trabalho foi desenvolvido no escopo da navegação de agentes robóticos e neste contexto um modelo arquitetural de navegação visual semântica com memória foi proposto, visando a tarefa de busca de objetos em ambientes internos de residências. As abordagens de navegação visual e navegação baseada em aprendizado foram adotadas, tomando-se vantagem dos recentes avanços obtidos em análise de cenas e detecção de objetos com ConvNets e métodos de aprendizado com ARP. Não obstante, conhecimento semântico sobre o contexto do ambiente foi obtido pela utilização de GNNs, enriquecendo o modelo por meio da análise de informações sobre relações entre objetos visualizados na cena. Finalmente, mecanismos de memória foram considerados neste trabalho, em uma abordagem incremental, de maneira a expandir um modelo existente e investigar as implicações do uso de redes recorrentes, como GRU e LSTM, incorporadas à navegação visual semântica.

Resultados foram avaliados sob a perspectiva de análises qualitativas e quantitativas. Em relação as primeiras, execuções do agente utilizando um modelo de conhecimento treinado foram realizadas em cenários de testes considerando os seguintes ambientes: sala de estar, cozinha, quarto e banheiro. Observou-se que os modelos acrescidos com o Módulo de Memória proposto realizavam maior análise exploratória do ambiente, evitando termos precoces e sendo capazes de percorrer trajetórias mais longas, dado que os objetos-alvo se encontravam distantes do ponto origem, com execuções bem sucedidas tanto pelo mo-

delo GCN-GRU quanto pelo modelo GCN-LSTM.

Em relação aos resultados quantitativos, foram demonstrados gráficos de recompensa, tamanho de trajetória e disparo de ação de término, gerados ao longo do processo de treinamento dos modelos para a GCN-MLP, GCN-GRU e GCN-LSTM. Observou-se que a GCN-GRU apresentou maior oscilação e instabilidade durante o treinamento, opondo-se a GCN-MLP e GCN-LSTM que apresentaram comportamento com menor variação. Foi explorada a relação existente entre os diferentes gráficos e observou-se aumento no acúmulo das recompensas obtidas à medida que o agente percorria trajetórias mais longas. Ainda, foi possível observar que a exploração do ambiente atingiu seu limite máximo de passos permitidos, uma vez que não foi disparada a ação de término do agente. Todos os modelos apresentaram maior período de convergência em ocorrências de características semelhantes, tendo o modelo GCN-LSTM obtido os melhores resultados considerando a média de recompensas.

Ainda em relação aos resultados qualitativos, métricas adotadas em trabalhos semelhantes foram utilizadas para avaliar as taxas de sucesso do agente e taxas de sucesso ponderadas pelo tamanho da trajetória, esta última verificando a eficácia dos agentes. Resultados comparativos foram apresentados tanto considerando um agente randômico, um agente utilizando o algoritmo A3C puro, um agente com modelo sem utilização de memória, GCN-MLP e, finalmente, os modelos acrescidos de memória GCN-GRU e GCN-LSTM. Todos modelos foram avaliados considerando-se um cenário onde todos os episódios foram considerados válidos e um cenário mais desafiador, restringido a episódios com $L \geq 5$. Em todas análises realizadas, os modelos providos com mecanismos de memória apresentaram resultados superiores aos modelos sem memória, com melhores resultados para o modelo GCN-LSTM. Apesar de todos os modelos apresentarem queda no desempenho quando submetidos as configurações restritas aos episódios com $L \geq 5$, foi possível observar que os modelos com memória, e em especial a GCN-LSTM, sofreram menores quedas em seu desempenho, tanto se comparadas com a configuração de análise mais simples, como se comparadas com os outros modelos avaliados.

Em resumo, concluímos que a inclusão de mecanismos de memória, expandindo o modelo tomado como base, apresentou melhores resultados qualitativos e quantitativos quando comparado com modelos similares sem o uso de mecanismos de memória, demonstrando o importante papel que essas redes podem desempenhar na aprendizagem de políticas em modelos de navegação visual baseadas em aprendizado. Esta contribuição favorece a construção de um modelo de conhecimento sobre o ambiente considerando experiências passadas e recorrências vivenciadas pelos agentes.

Apesar dos resultados obtidos com a inclusão de memória ao modelo de navegação visual semântica, o desempenho apresentado ainda é insuficiente para sua aplicação em ambientes reais, garantindo segurança e integridade física ao agente robótico e demais

agentes possivelmente presentes. Destacamos, por exemplo, a não concepção pelos modelos explorados neste trabalho de limitações físicas do ambiente, tendo estes, em diversas ocasiões, tentado transpor ou atravessar objetos. Tal resultado foi esperado uma vez que não existe uma modelagem de punições aos agentes em casos de colisões, e nem é empregado algum mecanismo de desvio de obstáculos, por isso é pertinente ressaltar a existência de tal lacuna nos modelos. Além disso, as trajetórias percorridas pelos agentes apresentam comportamento por vezes demasiadamente repetitivos, tendo os agentes postergado o disparo de ações de término mesmo quando o objeto-alvo havia sido avistado.

A realização presente trabalho propiciou os seguintes resultados:

- a escrita e submissão do artigo intitulado: "Deep Reinforcement Learning for Visual Semantic Navigation with Memory" que foi aceito para apresentação e publicação nos Anais do IEEE LARS-SBR 2020.
- Participação na Competição Brasileira de Robótica na Categoria: @HomeCBR: na CBR-2019 e CBR-2020.

Trabalhos futuros incluem, primeiramente, a melhora na execução das trajetórias pelos agentes, à medida que os modelos apresentam uma queda significativa na qualidade de seus resultados quando submetidos a configurações de cenários mais desafiadoras e com objetos-alvo mais distantes. Um ponto pertinente seria investigar maneiras de tornar os agentes menos exitantes em terminar episódios quando o objeto-alvo já tenha sido visualizado em condições que cumprem o necessário para um término bem sucedido. Possibilidades incluem prover esses modelos com informações que possibilitem uma maior certeza por parte dos agentes como, por exemplo, adoção de *data augmentation*, assim como alterações e exploração da função de recompensa adotada.

Um segundo ponto a ser considerado refere-se as relações entre objetos exploradas neste trabalho, que poderiam ser expandidas para noções mais abstratas e abrangendo maiores características do ambiente. Assim sendo, grafos contextuais não apenas objeto-objeto, mas objeto-ambiente, ambiente-ambiente, entre outros, são possíveis de serem adotados. Nestes casos, são requeridas noções semânticas mais abstratas e gerais para execução de navegações, tendo como ponto de partida do agente qualquer ambiente de uma residência.

Uma terceira lacuna que também pode ser investigada é a navegação entre ambientes. Devido a limitações do ambiente simulado, não foi possível ao agente transitar entre ambientes distintos, sendo obrigatoriamente posicionado no ambiente ao qual o objeto-alvo pertencia. Navegação entre ambientes é uma importante habilidade para uma navegação em ambientes reais, propondo um cenário mais desafiador e complexo de compreensão e planejamento. Não obstante, o planejamento e execução de rotinas mais

complexas, expandindo-se as ações possíveis ao agente, tomando vantagens de recursos disponibilizados pelos ambientes simulados também poderiam ser consideradas.

Finalmente, a transferência do modelo treinado em ambiente simulado para agente robótico em ambientes reais é de imenso interesse para trabalhos futuros, bem como a melhora do desempenho dos modelos propostos. A aplicação em ambiente real é certamente alvo de pesquisas nesta área, uma vez que permite avaliar a robustez dos modelos propostos que utilizam imagens de cenas com ideias conceituais.

REFERÊNCIAS

ANDERSON, P.; CHANG, A.; CHAPLOT, D. S.; DOSOVITSKIY, A.; GUPTA, S.; KOLTUN, V.; KOSECKA, J.; MALIK, J.; MOTTAGHI, R.; SAVVA, M. *et al.* On evaluation of embodied navigation agents. **arXiv preprint arXiv:1807.06757**, 2018. Citado na página 90.

ANDERSON, P.; WU, Q.; TENEY, D.; BRUCE, J.; JOHNSON, M.; SÜNDERHAUF, N.; REID, I.; GOULD, S.; HENGEL, A. van den. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 3674–3683. Citado nas páginas 59, 71 e 73.

ARMENI, I.; SENER, O.; ZAMIR, A. R.; JIANG, H.; BRILAKIS, I.; FISCHER, M.; SAVARESE, S. 3d semantic parsing of large-scale indoor spaces. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. Citado na página 60.

BANSAL, S.; TOLANI, V.; GUPTA, S.; MALIK, J.; TOMLIN, C. Combining optimal control and learning for visual navigation in novel environments. **arXiv preprint arXiv:1903.02531**, 2019. Citado nas páginas 60, 71 e 73.

CHANG, A.; DAI, A.; FUNKHOUSER, T.; HALBER, M.; NIESSNER, M.; SAVVA, M.; SONG, S.; ZENG, A.; ZHANG, Y. Matterport3d: Learning from rgb-d data in indoor environments. **International Conference on 3D Vision (3DV)**, 2017. Citado na página 87.

CHIANG, H.-T. L.; FAUST, A.; FISER, M.; FRANCIS, A. Learning navigation behaviors end-to-end with autorl. **IEEE Robotics and Automation Letters**, IEEE, v. 4, n. 2, p. 2007–2014, 2019. Citado na página 27.

CHO, K.; MERRIËNBOER, B. V.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014. Citado na página 43.

CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**, 2014. Citado na página 43.

ELFES, A. Sonar-based real-world mapping and navigation. **IEEE Journal on Robotics and Automation**, IEEE, v. 3, n. 3, p. 249–265, 1987. Citado na página 27.

_____. Using occupancy grids for mobile robot perception and navigation. **Computer**, IEEE, v. 22, n. 6, p. 46–57, 1989. Citado na página 27.

- FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial Intelligence Review**, Springer, v. 43, n. 1, p. 55–81, 2015. Citado na página 27.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>. Citado nas páginas 34, 35, 36, 39, 40, 41 e 42.
- GORI, M.; MONFARDINI, G.; SCARSELLI, F. A new model for learning in graph domains. In: **Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005**. [S.l.: s.n.], 2005. v. 2, p. 729–734 vol. 2. Citado na página 44.
- GRAVES, A. Supervised sequence labelling. In: **Supervised sequence labelling with recurrent neural networks**. [S.l.]: Springer, 2012. p. 5–13. Citado nas páginas 39, 40 e 41.
- GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. Lstm: A search space odyssey. **IEEE transactions on neural networks and learning systems**, IEEE, v. 28, n. 10, p. 2222–2232, 2016. Citado nas páginas 41 e 42.
- GUPTA, S.; DAVIDSON, J.; LEVINE, S.; SUKTHANKAR, R.; MALIK, J. Cognitive mapping and planning for visual navigation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 2616–2625. Citado na página 27.
- HAMMOND, D. K.; VANDERGHEYNST, P.; GRIBONVAL, R. Wavelets on graphs via spectral graph theory. **Applied and Computational Harmonic Analysis**, Elsevier, v. 30, n. 2, p. 129–150, 2011. Citado na página 46.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. Citado nas páginas 35, 37 e 38.
- HEBB, D. The organization of behavior. a neuropsychological theory. John Wiley, 1949. Citado na página 33.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 41.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014. Citado na página 96.
- KIPF, T. N.; WELING, M. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016. Citado na página 46.
- KOLVE, E.; MOTTAGHI, R.; GORDON, D.; ZHU, Y.; GUPTA, A.; FARHADI, A. Ai2thor: An interactive 3d environment for visual ai. **arXiv preprint arXiv:1712.05474**, 2017. Citado nas páginas 28, 62, 68 e 87.
- KRISHNA, R.; ZHU, Y.; GROTH, O.; JOHNSON, J.; HATA, K.; KRAVITZ, J.; CHEN, S.; KALANTIDIS, Y.; LI, L.-J.; SHAMMA, D. A. *et al.* Visual genome: Connecting language and vision using crowdsourced dense image annotations. **International Journal of Computer Vision**, Springer, v. 123, n. 1, p. 32–73, 2017. Citado nas páginas 58, 68 e 82.

- 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 na página [34](#).
- 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 nas páginas [34](#) e [35](#).
- LENSER, S.; VELOSO, M. Visual sonar: Fast obstacle avoidance using monocular vision. In: IEEE. **Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)**(Cat. No. 03CH37453). [S.l.], 2003. v. 1, p. 886–891. Citado na página [27](#).
- LI, R.; TAPASWI, M.; LIAO, R.; JIA, J.; URTASUN, R.; FIDLER, S. Situation recognition with graph neural networks. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2017. p. 4173–4182. Citado na página [59](#).
- LIN, T.; HORNE, B. G.; TIÑO, P.; GILES, C. L. Learning long-term dependencies is not as difficult with narx networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1996. p. 577–583. Citado na página [40](#).
- LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755. Citado na página [38](#).
- LOBOS-TSUNEKAWA, K.; LEIVA, F.; SOLAR, J. Ruiz-del. Visual navigation for biped humanoid robots using deep reinforcement learning. **IEEE Robotics and Automation Letters**, IEEE, v. 3, n. 4, p. 3247–3254, 2018. Citado nas páginas [27](#), [61](#), [71](#), [72](#) e [73](#).
- 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 [33](#).
- MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In: **International conference on machine learning**. [S.l.: s.n.], 2016. p. 1928–1937. Citado nas páginas [54](#), [61](#), [63](#) e [85](#).
- 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 na página [53](#).
- MOUSAVIAN, A.; TOSHEV, A.; FISER, M.; KOSECKA, J.; WAHID, A.; DAVIDSON, J. Visual representations for semantic target driven navigation. **arXiv preprint arXiv:1805.06066**, 2018. Citado nas páginas [15](#), [28](#), [66](#), [72](#) e [73](#).
- MOZER, M. C. Induction of multiscale temporal structure. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1992. p. 275–282. Citado na página [40](#).

- NGUYEN, D.-V.; VUONG, T.-L.; KIEU, H.-D.; PHAM, L.; LE, T.-H. Vision memory for target object navigation using deep reinforcement learning: An empirical study. In: **IEEE. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. [S.l.], 2018. p. 3267–3273. Citado nas páginas [15](#), [27](#), [63](#), [71](#) e [73](#).
- PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in pytorch. 2017. Citado na página [89](#).
- PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543. Citado na página [82](#).
- REMAZEILLES, A.; CHAUMETTE, F.; GROS, P. Robot motion control from a visual memory. In: **IEEE. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004**. [S.l.], 2004. v. 5, p. 4695–4700. Citado na página [27](#).
- 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 [33](#).
- RUMMERY, G. A.; NIRANJAN, M. **On-line Q-learning using connectionist systems**. [S.l.: s.n.], 1994. v. 37. Citado na página [50](#).
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. Image-Net Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015. Citado nas páginas [38](#) e [81](#).
- RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016. Citado nas páginas [47](#), [48](#), [50](#) e [51](#).
- SANTOS I. B. A.; ROMERO, R. A. F. Deep reinforcement learning for visual semantic navigation with memory. In: **Proceedings of the IEEE Latin American Robotics Symposium / Brazilian Symposium of Robotics**. [S.l.: s.n.], 2020. p. 1–6. Citado na página [73](#).
- SAVVA, M.; CHANG, A. X.; DOSOVITSKIY, A.; FUNKHOUSER, T.; KOLTUN, V. MINOS: Multimodal indoor simulator for navigation in complex environments. **arXiv:1712.03931**, 2017. Citado nas páginas [28](#) e [86](#).
- SAVVA*, M.; KADIAN*, A.; MAKSYMETS*, O.; ZHAO, Y.; WIJMANS, E.; JAIN, B.; STRAUB, J.; LIU, J.; KOLTUN, V.; MALIK, J.; PARIKH, D.; BATRA, D. Habitat: A Platform for Embodied AI Research. **arXiv preprint arXiv:1904.01201**, 2019. Citado nas páginas [28](#) e [87](#).
- SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. The graph neural network model. **IEEE Transactions on Neural Networks**, IEEE, v. 20, n. 1, p. 61–80, 2008. Citado nas páginas [44](#) e [45](#).

- SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; DRIESSCHE, G. V. D.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVA, V.; LANCTOT, M. *et al.* Mastering the game of go with deep neural networks and tree search. **nature**, Nature Publishing Group, v. 529, n. 7587, p. 484, 2016. Citado na página 53.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014. Citado na página 35.
- SONG, S.; YU, F.; ZENG, A.; CHANG, A. X.; SAVVA, M.; FUNKHOUSER, T. Semantic scene completion from a single depth image. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 1746–1754. Citado na página 87.
- SÜNDERHAUF, N. Where are the keys?—learning object-centric navigation policies on semantic maps with graph convolutional networks. **arXiv preprint arXiv:1909.07376**, 2019. Citado nas páginas 45, 64, 71 e 72.
- SUTTON, R.; BARTO, A. **Reinforcement Learning: An Introduction**. MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262039246. Disponível em: <<https://books.google.com.br/books?id=6DKPtQEACAAJ>>. Citado nas páginas 47, 48, 49, 51, 52 e 53.
- SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. Disponível em: <<http://arxiv.org/abs/1409.4842>>. Citado na página 35.
- WATKINS, C. J.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 279–292, 1992. Citado na página 49.
- 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 51.
- WORTSMAN, M.; EHSANI, K.; RASTEGARI, M.; FARHADI, A.; MOTTAGHI, R. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 6750–6759. Citado nas páginas 69, 70, 72, 73, 89, 90 e 96.
- WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C.; PHILIP, S. Y. A comprehensive survey on graph neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, 2020. Citado na página 45.
- XIA, F.; ZAMIR, A. R.; HE, Z.-Y.; SAX, A.; MALIK, J.; SAVARESE, S. Gibson Env: real-world perception for embodied agents. In: **IEEE Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on**. [S.l.], 2018. Citado na página 87.
- YANG, J.; LU, J.; LEE, S.; BATRA, D.; PARIKH, D. Graph r-cnn for scene graph generation. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 670–685. Citado nas páginas 45 e 57.

YANG, W.; WANG, X.; FARHADI, A.; GUPTA, A.; MOTTAGHI, R. **Visual Semantic Navigation using Scene Priors**. 2019. Disponível em: <<https://openreview.net/forum?id=HJeRkh05Km>>. Citado nas páginas 15, 28, 29, 67, 68, 69, 70, 72, 73, 74, 77, 79, 80, 86, 89, 90, 93, 95, 96 e 109.

YAO, T.; PAN, Y.; LI, Y.; MEI, T. Exploring visual relationship for image captioning. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 684–699. Citado nas páginas 45 e 58.

ZHU, Y.; GORDON, D.; KOLVE, E.; FOX, D.; FEI-FEI, L.; GUPTA, A.; MOTTAGHI, R.; FARHADI, A. Visual semantic planning using deep successor representations. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2017. p. 483–492. Citado nas páginas 65, 72 e 73.

ZHU, Y.; MOTTAGHI, R.; KOLVE, E.; LIM, J. J.; GUPTA, A.; FEI-FEI, L.; FARHADI, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: **IEEE. 2017 IEEE international conference on robotics and automation (ICRA)**. [S.l.], 2017. p. 3357–3364. Citado nas páginas 15, 27, 61, 62, 63, 71 e 73.