

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Suane Pires Pinheiro da Silva

**Localização e Navegação de Robôs Móveis em Mapas Topológicos
utilizando Redes Neurais Convolucionais baseada em *Transfer
Learning***

Fortaleza – CE

10 de outubro de 2019

Suane Pires Pinheiro da Silva

**Localização e Navegação de Robôs Móveis em Mapas
Topológicos utilizando Redes Neurais Convolucionais
baseada em *Transfer Learning***

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Processamento de Imagens e Sinais

Orientador: Prof. Dr. Pedro Pedrosa Rebouças Filho

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

10 de outubro de 2019

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

D1111 da Silva, Suane Pires Pinheiro.
Localização e Navegação de Robôs Móveis em Mapas Topológicos utilizando Redes Neurais Convolucionais baseada em Transfer Learning / Suane Pires Pinheiro da Silva. - 2019.
86 f. : il. color.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2019.

Orientação: Prof. Dr. Pedro Pedrosa Rebouças Filho.

1. Robôs Móveis. 2. Redes Neurais Convolucionais. 3. Transfer Learning. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Suane Pires Pinheiro da Silva

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

**Orientador: Prof. Dr. Pedro Pedrosa Rebouças
Filho**

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. Antonio Carlos da Silva Barros
Universidade da Integração Internacional da Lusofonia
Afro-Brasileira (UNILAB)

**Prof. Dr. Carlos Maurício Jaborandy de Mattos
Dourado Junior**
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Fortaleza – CE
10 de outubro de 2019

Agradecimentos

À Deus, acima de tudo, por me guiar, me iluminar e me dar força e sabedoria para seguir em frente com os meus objetivos.

Aos meus pais, Solange Pires Pinheiro e José Lúcio da Silva, por me dar amor, confiança e estrutura para alcançar meus sonhos.

Ao meu marido, Diego Barros Pereira, pelo amor, compreensão e incentivo para que eu pudesse concluir esta importante etapa da minha vida.

Ao meu orientador, Prof. Dr. Pedro Pedrosa, pelos ensinamentos, confiança, paciência, dedicação e amizade depositados ao longo deste trabalho.

Aos professores e a todos os meus amigos e companheiros de pesquisa do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal do Ceará (PPGCC-IFCE) e, em especial, do Laboratório de Processamento de Imagens, Sinais e Computação Aplicada (LAPISCO), que contribuíram para o meu crescimento profissional e pessoal.

Ao Leandro Bezerra, ao Jefferson Almeida e à Elene Ohata, pela contribuição direta na construção deste trabalho.

Ao Instituto Federal de Educação, Ciências e Tecnologia do Ceará (IFCE) e a todos os profissionais que o compõem por proporcionar um ambiente adequado ao desenvolvimento intelectual e social dos alunos.

À Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP), pelo suporte financeiro.

Resumo

A Robótica Móvel é uma área de pesquisa crescente e desafiadora, com aplicações em diferentes tarefas. O interesse em robôs móveis autônomos e os sucessivos avanços na tecnologia tem aumentado de forma simultânea. Com isso, esses robôs tem sido largamente empregados em atividades civis e militares. Motivado por isso, esta dissertação apresenta uma abordagem para localização e navegação de robôs móveis utilizando Visão Computacional através do uso de Redes Neurais Convolucionais (*Convolutional Neural Network*, CNN) como extrator de característica, a partir da aplicação do conceito de *Transfer Learning*. O uso de *Transfer Learning* permite que um modelo de CNN pré-treinado seja reutilizado em um novo problema, ou seja, é possível empregar uma rede treinada em um outro banco de dados, normalmente maior, para solucionar uma problema diferente. Utilizar esse conceito traz vantagens como minimizar a ocorrência de questões como *overfitting*, diminuir o custo de treinamento, uma vez que CNNs muito profundas requerem *hardware* potente para serem treinadas, e eliminar o obstáculo em obter um conjunto de dados grande o suficiente para se treinar adequadamente uma CNN. O robô móvel realiza sua navegação em uma área do tipo interna, utilizando o recurso de mapeamento topológico e processando imagens provenientes de uma câmera de visão convencional de alta resolução e de uma câmera de visão omnidirecional. As imagens foram capturadas no ambiente real e, em seguida, no ambiente virtual simulado computacionalmente. Logo, são considerados quatro bancos de dados diferentes: imagens reais convencionais, imagens sintéticas convencionais, imagens reais omnidirecionais e imagens sintéticas omnidirecionais. Com o objetivo de evidenciar o alto desempenho da abordagem proposta foi realizada uma avaliação de combinações de arquiteturas de CNN com classificadores estabelecidos na literatura, dentre eles, Naive Bayes, Perceptron Múltiplas Camadas, k -Vizinhos Mais Próximos, Floresta Aleatória, e Máquina de Vetores de Suporte. As métricas Acurácia, *Recall*, *Precision* e F1-Score e parâmetros como tempo de processamento foram empregados para provar a confiabilidade e a eficácia da abordagem, uma vez que essas propriedades são fundamentais na análise de sistemas embarcados. Os resultados indicam que a combinação da arquitetura VGG19 com o classificador Naive Bayes, utilizando imagens reais com campo de visão convencional e imagens sintéticas com campo de visão omnidirecional, mostrou-se adequada para a tarefa de localização, considerando os valores atingidos nas métricas de avaliação, sendo em todas 100%, e o menor tempo na fase de teste, com 0,002 s. Logo, os resultados comprovam a relevância da abordagem proposta para a atividade de localização e, conseqüentemente, de navegação de robôs móveis, abrangendo sistemas de visão computacional do ponto de vista real e simulado (virtual).

Palavras-chave: Robôs Móveis, Redes Neurais Convolucionais, *Transfer Learning*.

Abstract

Mobile Robotics is a growing and challenging area of research, with applications in different tasks. Interest in autonomous mobile robots and successive advances in technology has increased simultaneously. As a result, these robots have been widely employed in civil and military activities. This dissertation presents an approach for localization and navigation of mobile robots using Computational Vision through the use of Convolutional Neural Networks (CNN) as feature extractor, from the application of the Transfer Learning concept. Using Transfer Learning allows a pre-trained CNN model to be reused for a new problem, meaning that you can employ a trained network in another, usually larger, database to solve a different problem. Using this concept has advantages such as minimizing issues such as overfitting, lowering the cost of training, since very deep CNNs require powerful hardware to be trained, and eliminating the obstacle to obtaining a large enough data set. to properly train a CNN. The mobile robot navigates within an internal type area using the topological mapping feature and processing images from a conventional high resolution vision camera and an omnidirectional vision camera. The images were captured in the real environment and then in the computationally simulated virtual environment. Therefore, four different databases are considered: conventional real images, conventional synthetic images, omnidirectional real images, and omnidirectional synthetic images. In order to highlight the high performance of the proposed approach, an evaluation of CNN architecture combinations with classifiers established in the literature was performed, among them Naive Bayes, Multilayer Perceptron, k -Nearest Neighbors, Random Forest, and Support Vector Machine. Accuracy, Recall, Precision, and F1-Score metrics and parameters such as processing time were employed to prove the reliability and effectiveness of the approach, as these properties are critical in the analysis of embedded systems. The results indicate that the combination of the VGG19 architecture with the Naive Bayes classifier, using conventional real field of view and omnidirectional field of view synthetic images, was adequate for the localization task, considering the values achieved in the evaluation metrics. , being in all 100%, and the shortest time in the test phase, with 0.002s. Thus, the results prove the relevance of the proposed approach to the localization activity and, consequently, of mobile robot navigation, covering real and simulated (virtual) computer vision systems.

Keywords: Mobile Robots, Convolutional Neural Networks, Transfer Learning.

Lista de ilustrações

Figura 1 – Categorização das tecnologias de posicionamento em ambientes internos.	19
Figura 2 – Modelo genérico de uma Rede Neural Convolutacional.	36
Figura 3 – Operação de convolução em imagem.	37
Figura 4 – A função de ativação da Unidade Linear Retificada (<i>Rectified Linear Unit</i> , ReLU).	38
Figura 5 – Demonstração do processo de <i>max pooling</i> .	39
Figura 6 – Em (a), a ilustração representa o fluxo de informações convencionais durante uma passagem direta de uma rede e, em (b), esse fluxo é interrompido (definido como zero) em algum neurônio, consistindo no processo chamado de <i>Dropout</i> .	40
Figura 7 – Arquitetura da Rede Neural Convolutacional LeNet-5.	42
Figura 8 – O campo receptivo efetivo de uma pilha de duas convoluções 3×3 .	44
Figura 9 – Duas ilustrações de arquitetura inicial: (a) Módulo <i>Inception</i> simples e (b) Módulo <i>Inception</i> com redução de dimensionalidade.	45
Figura 10 – Uma ilustração da arquitetura do Módulo Inception com convoluções assimétricas.	47
Figura 11 – Um exemplo da arquitetura de um Bloco Residual.	49
Figura 12 – Uma ilustração da arquitetura geral quando combinado Conexões Residuais com Módulo <i>Inception</i> .	50
Figura 13 – Ilustração da arquitetura de um Bloco Denso.	52
Figura 14 – Ilustração dos módulos da NASNet.	53
Figura 15 – Infograma da metodologia empregada na abordagem proposta.	55
Figura 16 – Robô móvel usado na abordagem proposta.	57
Figura 17 – Mapa topológico do ambiente interno (apartamento).	57
Figura 18 – Amostras de imagens da classe 5 (nó 4) para (a) ambiente real com câmera GoPro [®] , (b) ambiente real com câmera omnidirecional, (c) ambiente virtual com câmera GoPro [®] e (d) ambiente virtual com câmera omnidirecional.	59
Figura 19 – Ilustração das duas primeiras etapas de uma CNN. No primeiro, a imagem de entrada é submetida a uma sequência de camadas convolucionais e sub-amostragem (<i>pooling</i>). No segundo, os resultados da primeira etapa são redimensionados e concatenados em uma única lista, chamada de <i>deep features</i> .	62
Figura 20 – Ilustração do processo usado para transformar uma CNN em um extrator de características.	62

Lista de tabelas

Tabela 1 – Comparação entre ferramentas de localização em ambientes internos. Intensidade do sinal recebido (<i>received signal strength</i> - RSS). Tempo de retorno (<i>time of arrival</i> - ToA). Diferença de tempo de chegada (<i>time difference of arrival</i> - TDOA). Ângulo de chegada (<i>angle of arrival</i> - AoA).	20
Tabela 2 – Abordagens de localização topológica baseadas em visão computacional.	21
Tabela 3 – Especificações do robô móvel considerado.	56
Tabela 4 – Especificações dos bancos de imagem.	58
Tabela 5 – Número de atributos retornados por cada extrator de características. .	60
Tabela 6 – Matriz de confusão para um problema de classificação multiclasse. . .	64
Tabela 7 – Acurácia alcançada, no ambiente real , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	68
Tabela 8 – Recall alcançado, no ambiente real , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	69
Tabela 9 – Precision alcançado, no ambiente real , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	69
Tabela 10 – F1-Score alcançado, no ambiente real , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	70
Tabela 11 – Acurácia alcançada, no ambiente virtual , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	71
Tabela 12 – Recall alcançado, no ambiente virtual , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	71
Tabela 13 – Precision alcançado, no ambiente virtual , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	72
Tabela 14 – F1-Score alcançado, no ambiente virtual , pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro [®] e da câmera omnidirecional.	72

Tabela 15 – Tempo de treino , em segundos , para o banco de imagens convencionais no ambiente real e para o banco de imagens omnidirecionais no ambiente virtual	73
Tabela 16 – Tempo de teste , em milisegundos , para o banco de imagens convencionais no ambiente real e para o banco de imagens omnidirecionais no ambiente virtual	74
Tabela 17 – Tempo de extração para cada arquitetura CNN.	75

Sumário

1	INTRODUÇÃO	17
1.1	Trabalhos Relacionados	18
1.2	Objetivos	22
1.2.1	Objetivos Específicos	22
1.3	Produção Científica	23
1.4	Estrutura	24
2	MÉTODOS DE APRENDIZAGEM DE MÁQUINA	25
2.1	Naive Bayes	25
2.2	<i>k</i>-Vizinhos Mais Próximos	26
2.3	Máquina de Vetores de Suporte	27
2.3.1	Classificador de Hiperplano	27
2.3.2	Classificador de Margem Máxima	28
2.3.3	Classificador de Margem Suave	29
2.3.4	Classificador SVM	29
2.4	Floresta Aleatória	30
2.5	Perceptron Múltiplas Camadas	31
3	REDE NEURAL CONVOLUCIONAL	35
3.1	Estrutura de uma Rede Neural Convolutacional	35
3.1.1	Camada Convolutacional	36
3.1.2	Não-linearidade ReLU	38
3.1.3	Camada de <i>Pooling</i>	38
3.1.4	<i>Dropout</i>	40
3.1.5	Camada Totalmente Conectada	41
3.1.6	<i>Softmax</i>	41
3.2	Lenet-5	41
3.3	AlexNet	42
3.4	Arquiteturas VGG	43
3.4.1	Convoluções Fatoradas	43
3.5	Arquiteturas Inception	44
3.5.1	Módulo Inception	44
3.5.2	Camada <i>batch-normalization</i>	45
3.5.3	Convoluções Assimétricas	47
3.6	Arquiteturas ResNet	48
3.6.1	Bloco Residual	48

3.7	Arquiteturas Inception-ResNet	48
3.7.1	Módulo Inception com Conexões Residuais	49
3.8	Arquitetura Xception	49
3.8.1	Convoluções Separáveis por Profundidade	50
3.9	Arquitetura MobileNet	50
3.9.1	Multiplicadores de Largura	51
3.9.2	Multiplicadores de Resolução	51
3.10	Arquiteturas DenseNet	51
3.10.1	Dense Blocks	52
3.11	Arquiteturas NASNet	52
3.11.1	NASNet Search Space	53
4	METODOLOGIA	55
4.1	Mapa Topológico e Robô	56
4.2	Captura de Imagens	57
4.3	Extração de Características	59
4.3.1	Rede Neural Convolucional como Extrator de Características	60
4.3.1.1	Construção e Inicialização de Arquiteturas	60
4.3.1.2	Treinamento das Arquiteturas	60
4.3.1.3	Convertendo CNNs em Extratores de Características	61
4.4	Métodos de Classificação	63
4.5	Análise de Desempenho	63
4.5.1	Métricas de Avaliação	63
4.5.2	Validação Cruzada	65
5	RESULTADOS	67
5.1	Análise de Desempenho	67
5.1.1	Resultados no Ambiente Real	67
5.1.2	Resultados no Ambiente Virtual	70
5.2	Análise do Tempo de Processamento	73
5.2.1	Tempo de Treino	73
5.2.2	Tempo de Teste	74
5.3	Análise Geral dos Resultados	75
6	CONCLUSÃO	77
6.1	Trabalhos Futuros	78
	REFERÊNCIAS	79

1 Introdução

As aplicações para Robótica Móvel estão se tornando cada vez mais abrangentes, podendo ser encontradas em ambientes militares, industriais, agrícolas, domésticos, entre outros. Em Ravindran, Mills e Krishnan (2018), o uso de robôs móveis foi proposto para proteger a formação de um pelotão militar. Em Widyantara, Rivai e Purwanto (2018), um robô móvel olfativo foi empregado para localizar vazamentos de gás, poupando humanos de realizar essa tarefa. Enquanto em Łukowska et al. (2019), uma plataforma móvel foi desenvolvida para auxiliar na tarefa de amostragem de solo na agricultura e na redução de custos. Já em Ciabattoni et al. (2017), um robô móvel foi configurado para detecção de queda humana por meio de comunicação de voz com o usuário.

Visto os diversos propósitos que podem ser atendidos com o uso dessa tecnologia, entre os tópicos mais explorados relacionados aos robôs móveis está a tarefa de localização, pois trata-se de uma atividade primordial para que qualquer outra atividade atribuída ao robô possa ser viabilizada. Embora a prática dessa tarefa seja bastante explorada nesse campo de pesquisa, ela ainda continua a ser um procedimento complexo, pois envolve diversos recursos que devem funcionar adequadamente de modo simultâneo.

De acordo com o ambiente considerado para a navegação do robô, há técnicas propícias para área adotada, as quais podem ser aplicadas para a concepção do sistema de localização. Para ambientes externos, um dos métodos mais utilizados é o Sistema de Posicionamento Global (*Global Positioning System*, GPS), como em Kurdi, Dadykin e Elzein (2017). Porém o uso de GPS não é recomendado para ambientes internos, pois a sua precisão é dada em metros, o que não é favorável para a localização de robôs móveis em escala humana (DIOP et al., 2016). Além disso, o GPS não opera em ambientes fechados. Para ambientes internos, entre as ferramentas mais empregadas está o Wi-Fi, a Identificação por Radiofrequência (*Radio-Frequency Identification*, RFID), o Ultrassom e o Bluetooth (SONG; JIANG; HUANG, 2011a).

Ainda considerando ambientes internos, um dos mais recentes recursos empregados para localização de robôs móveis é o Reconhecimento por Imagem. Dentre os benefícios dessa técnica está a tolerância a ruídos sonoros, ao contrário do Ultrassom, não se limitar à distância, permitindo a abrangência de uma vasta área, em contrapartida ao Bluetooth, não necessitar de uma infraestrutura complexa, como exigido quando empregado Wi-Fi, além de não requerer modificações no ambiente, como demanda com o uso do RFID (SONG; JIANG; HUANG, 2011a).

Contudo, a localização de um robô móvel vai além do que apenas definir uma posição absoluta no espaço, para tanto é necessário também elaborar um mapa do ambiente

a ser navegado e determinar o ponto no mapa onde encontra-se o robô. Dentre as técnicas de mapeamento desenvolvidas estão a abordagem geométrica e a abordagem topológica. Em mapas geométricos, o espaço de navegação é descrito por um sistema de coordenadas globais, como em Jiang et al. (2017). Em mapas topológicos, o ambiente é retratado como um grafo, como em Chin et al. (2016).

As pesquisas mais recentes de navegação de robôs móveis baseados em visão computacional estão cada vez mais prósperas (WANG et al., 2012; LOWRY et al., 2016). Esse fato ocorre devido à percepção humana ser primordialmente visual. Além de tudo, compreender e utilizar esse tipo de informação é fundamental para proporcionar a comunicação homem-robô. Os atributos visuais de um local podem ser empregadas para estabelecer a sua categoria funcional (por exemplo, sala, quarto, cozinha e outros). Pontos de referência visuais também são comumente utilizados por seres humanos para organizar rotas e destacar lugares. Essas características motivam pesquisadores a aplicar a informação visual na tarefa de localização de robôs móveis.

Tendo em vista o conteúdo apresentado, este trabalho propõe uma nova abordagem para desenvolver um sistema de localização e navegação para robôs móveis por Visão Computacional, empregando Redes Neurais Convolucionais aliada ao conceito de *Transfer Learning*, o que permite que esse método seja aplicado como um extrator de características. Além disso, técnicas de Aprendizagem de Máquina foram aplicadas para a concepção do sistema de Visão Computacional proposto, visto a capacidade desses métodos de aprender padrões complexos e tomar decisões com base em um conjunto de dados. Os métodos de Aprendizagem de Máquina empregados foram Naive Bayes, k -Vizinhos Mais Próximos (k -Nearest Neighbors, kNN), Floresta Aleatória (*Random Forest*), Perceptron Múltiplas Camadas (*Multilayer Perceptron* - MLP) e Máquina de Vetores de Suporte (*Support Vector Machine*, SVM). O ambiente de navegação do veículo é do tipo interno e o mapeamento é realizado considerando a abordagem topológica.

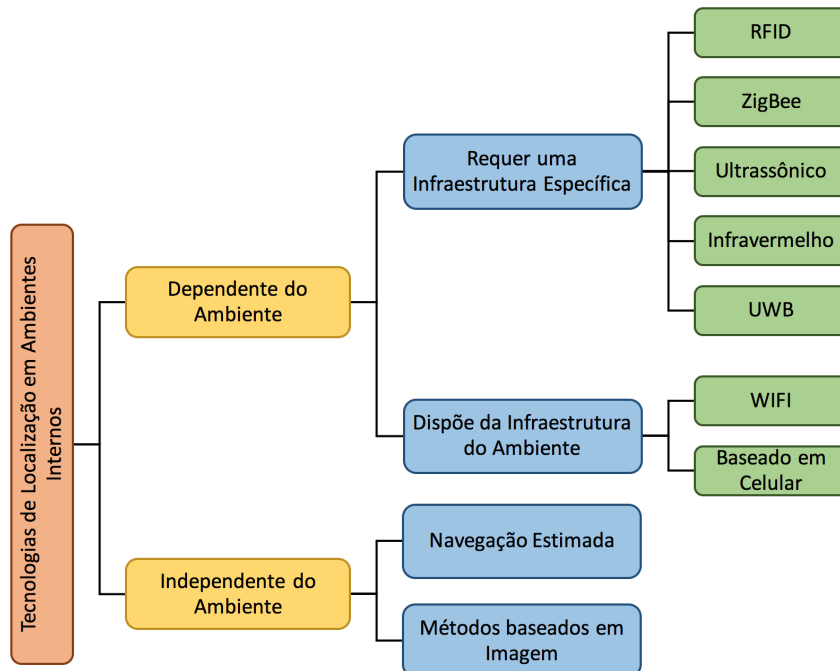
Para ressaltar a eficácia da abordagem, uma avaliação de desempenho e o tempo de processamento foram apresentados para a combinação de várias arquiteturas CNN com os métodos de classificação considerados, verificando o desempenho com bancos de imagens formados a partir de dois tipos de câmera (GoPro[®] e Omnidirecional) em dois ambientes diferentes (real e simulado) para um espaço do tipo apartamento.

1.1 Trabalhos Relacionados

A localização é uma habilidade crucial para a navegação. Logo, para efetuar tal tarefa, o robô precisa conhecer o seu posicionamento no ambiente (CASTELLANOS; TARDOS, 2012). Sendo assim, é interessante que o sistema de localização obtenha o maior número possível de informações do ambiente.

Na Robótica Móvel, há diversos padrões de sensores que podem ser empregados para localização, dentre eles estão sistemas baseados em laser, em ultrassom ou em visão computacional (DEAK; CURRAN; CONDELL, 2012). A Tabela 1 mostra as tecnologias direcionadas para posicionamento em ambiente internos agrupadas segundo a infraestrutura da aplicação que as adotam. Essa organização é realizada de acordo com Alarifi et al. (2016).

Figura 1 – Categorização das tecnologias de posicionamento em ambientes internos.



Fonte: (ALARIFI et al., 2016).

A Tabela 1 contrasta as vantagens e desvantagens dessas técnicas, além de mostrar alguns estudos referentes. Entre as vantagens da utilização de um sistema de localização baseado em visão computacional, destaca-se a facilidade de escalabilidade, sem aumento expressivo do custo. Além disso, esses sistemas não requerem mudanças no ambiente (MAUTZ; TILCH, 2011).

Mapas topológicos são úteis para a tarefa de localização de robôs móveis. Por serem compactos e simples, esses tipos de mapa exigem menos memória do computador e, conseqüentemente, torna o processo de navegação do robô mais rápido (CHENG; CHEN; LIU, 2015). Em Garcia-Fidalgo e Ortiz (2015b) é retratado um estudo sobre diversas abordagens de localização e mapeamento topológico usando visão computacional. Nesse levantamento, as abordagens são divididas em descritores globais, descritores locais e bolsa de palavras (*Bag-Of-Words* - BoW). A Tabela 2 apresenta os pontos positivos e negativos de cada abordagem, bem como algumas aplicações. A partir da Tabela 2, pode-se constatar várias contribuições da visão computacional na localização e navegação de robôs, confirmando a importância da pesquisa.

Tabela 1 – Comparação entre ferramentas de localização em ambientes internos. Intensidade do sinal recebido (*received signal strength* - RSS). Tempo de retorno (*time of arrival* - ToA). Diferença de tempo de chegada (*time difference of arrival* - TDOA). Ângulo de chegada (*angle of arrival* - AoA).

Ferramenta	Técnica Comum	Vantagem	Desvantagem	Sistema
Baseado em imagem	Reconhecimento de padrões	Não requer modificações no ambiente (DEAK; CURRAN; CONDELL, 2012); Alta escalabilidade sem expressivo aumento de custo (MAUTZ; TILCH, 2011). O hardware econômico e simples de instalar (CHOWDHURY et al., 2016).	Sujeito ao campo de visão; Algumas técnicas demandam um alto custo computacional (HARTMANN, 2011).	(DEBSKI et al., 2015) (LI, 2014) (RUBIO et al., 2016) (SON; KIM; SOHN, 2015). (CAMPOS; LOVISOLO; CAMPOS, 2014) (NOWICKI; SKRZYPCZYNSKI, 2015)
WLAN	RSS		Afetado por múltiplos caminhos e efeito de fading do sinal (CHOWDHURY et al., 2016).	(CAMPES; LOVISOLO; NOWICKI; SKRZYPCZYNSKI, 2015)
RFID	Proximidade; RSS	Apropriado para ambientes densos (DEAK; CURRAN; CONDELL, 2012).	Alto custo dos leitores e infraestrutura complexa (DEAK; CURRAN; CONDELL, 2012).	(CALDERONI et al., 2015) (MAH; GUEAIEB, 2011)
Bluetooth	Proximidade; RSS	Existente em equipamentos populares como <i>smartphones</i> (DEAK; CURRAN; CONDELL, 2012); Não requer campo de visão (SONG; JIANG; HUANG, 2011b).	É necessário que o usuário efetue uma varredura (DEAK; CURRAN; CONDELL, 2012); O limite de cobertura é limitado (MAINETTI; PATRONO; SERGI, 2014).	(LI, 2014); (KWIECEN et al., 2015).
Ultrassom	ToA, TDOA	Baixo custo; Não sofre interferências eletromagnéticas (DEAK; CURRAN; CONDELL, 2012).	Afetado por sons de alta frequência; Escalabilidade complexa (DEAK; CURRAN; CONDELL, 2012)	(KIM; CHOI, 2008) (YAYAN; YUCELI; YAZICI, 2015)
Infravermelho	Proximidade, Mudança de Fase Diferencial, AoA	Não sofre de interferências eletromagnéticas (CHOWDHURY et al., 2016).	Altos custos de manutenção e hardware (CHOWDHURY et al., 2016).	(SIOUTIS; TAN, 2014) (SCHINDLER; METZGER; STARNER, 2006)
Navegação estimada	RSS	Capacidade de trabalhar na mesma frequência de outros dispositivo (DEAK; CURRAN; CONDELL, 2012).	Baixa precisão (DEAK; CURRAN; CONDELL, 2012).	(JIMENEZ et al., 2009) (KOTHARI et al., 2012)
Baseado em celular	Rastreamento	Não requer hardware extra (ALARIFI et al., 2016).	Calcula uma posição aproximada (ALARIFI et al., 2016).	(VARSHAVSKY et al., 2007)

Fonte: Autor.

Tabela 2 – Abordagens de localização topológica baseadas em visão computacional.

	Vantagem	Desvantagem	Descritor	Referência
Descritor Global	Baixo custo de armazenamento e computacional (BONIN-FONT; ORTIZ; OLIVER, 2008).	Pouco robusto aos efeitos de oclusão e iluminação, reduzindo o poder de discriminação (BONIN-FONT; ORTIZ; OLIVER, 2008).	DP-FACT; Omni-gist; PCA	(BESSA et al., 2015) (LIU; SIEGWART, 2012)
Descritor Local	Alto poder de discriminação; São geralmente mais robustos às mudanças de escala, oclusão, iluminação e rotação (GARCIA-FIDALGO; ORTIZ, 2015a).	O custo computacional e as necessidades de armazenamento são maiores do que para descritores globais (FUENTES-PACHECO; ASCENCIO; RENDÓN-MANCHA, 2015).	SIFT; Wavelets; 3D-PIRF (SURF)	(JOHNS; YANG, 2011) (SAEDAN; LIM; ANG, 2007)
BoW	Adequado com elevado número de imagens para processar (FUENTES-PACHECO; ASCENCIO; RENDÓN-MANCHA, 2015).	Há presença de palavras com ruído, devido o modo de construção do vocabulário e a perda das relações espaciais entre as palavras; Requer uma quantidade de armazenamento moderada (GARCIA-FIDALGO; ORTIZ, 2015a)	FAST/BRIEF; SURF; SIFT	(GÁLVEZ-LÓPEZ; TARDÓS, 2012) (RANGANATHAN; DELLAERT, 2011)

Fonte: Autor.

Considerando a perspectiva tecnológica, este trabalho tem como contribuição a aplicação, para a tarefa de localização de robôs móveis, de Redes Neurais Convolucionais segundo o conceito de *Transfer Learning* juntamente com métodos de Aprendizagem de Máquina consolidados. Além disso, neste trabalho pretende-se avaliar diversas configurações para otimizar a atividade de localização e navegação de robôs móveis.

1.2 Objetivos

O objetivo principal desse trabalho é o desenvolvimento de um sistema de localização e, conseqüentemente, navegação para robôs móveis utilizando Redes Neurais Convolucionais baseadas no conceito de *Transfer Learning*.

1.2.1 Objetivos Específicos

De modo mais específico, este trabalho tem os seguintes objetivos:

- Analisar o uso de imagens obtidas a partir de uma câmera de alta resolução, em ambiente real e simulado, na tarefa de localização de robôs móveis;
- Analisar o uso de imagens obtidas a partir de uma câmera omnidirecional, em ambiente real e simulado, na tarefa de localização de robôs móveis;
- Utilizar técnicas de Visão Computacional para o reconhecimento de local interno;
- Explorar várias arquiteturas de CNN como extratores de características (com base na noção de *Transfer Learning*);
- Extrair atributos, a partir dos conjuntos de imagens considerados, usando 11 modelos diferentes de CNNs pré-treinadas, como: VGG16 (SIMONYAN; ZISSERMAN, 2014a), VGG19 (SIMONYAN; ZISSERMAN, 2014a), MobileNet (HOWARD et al., 2017a), InceptionV3 (SZEGEDY et al., 2016), Xception (CHOLLET, 2017), ResNet50 (HE et al., 2016), InceptionResNetV2 (SZEGEDY et al., 2017), NASNetMobile (ZOPH et al., 2017), DenseNet121, DenseNet169 e DenseNet201 (HUANG et al., 2017);
- Aplicar 5 métodos de classificação, como: Naive Bayes (THEODORIDIS; KOUTROUMBAS, 2008), *k*-Vizinhos Mais Próximos (COVER; HART, 1967), Floresta Aleatória (LIAW; WIENER et al., 2002), Perceptron Múltiplas Camadas (HAYKIN, 2008) e Máquina de Vetores de Suporte (VAPNIK, 1998);
- Desenvolver um sistema navegação e localização com mapas topológicos para robôs móveis usando CNN, incorporando o conceito de *Transfer Learning*;
- Analisar a relevância da abordagem proposta utilizando métrica de avaliação como Acurácia, *Recall*, *Precision* e F1-Score, além de parâmetros de tempo de processamento.

1.3 Produção Científica

Durante este projeto de mestrado, os seguintes artigos científicos foram aceitos e/ou publicados:

- (CAIP 2017) - A Novel Approach for Mobile Robot Localization in Topological Maps Using Classification with Reject Option from Structural Co-occurrence Matrix - 1º Autor - Qualis B1 (CC);
- (ICMLA 2017) - A Novel Approach for Robot Localization in Topological Maps using Classification with Reject Option from Structural Co-occurrence Matrix in Omnidirectional Images - 1º Autor - Qualis B1 (CC);
- (IJCNN 2018) - Localization of Mobile Robots with Topological Maps and Classification with Reject Option using Convolutional Neural Networks in Omnidirectional Images - 1º Autor - Qualis A1 (CC);
- (BRACIS 2018) - Robot Localization Using Classification with Reject Option from Structural Co-occurrence Matrix in Omnidirectional Images - 1º Autor - Qualis B2 (CC);
- (SIBGRAPI 2018) - An Approach to Navigation in Outdoor and Indoor Environments with Unmanned Aerial Vehicle Using Visual Topological Map - 2º Autor - Qualis B1 (CC);
- (CBMS 2018) - Lung Nodule Classification via Deep Transfer Learning in CT Lung Images - 3º Autor - Qualis B1 (CC);
- (IEEE Latin America Transactions 2018) - Higher-Order Statistics Applied to Machine Learning as an Approach to Identify Broken Rotor Bars in Induction Motors - 2º Autor - Qualis B2 (Eng. IV);
- (Neural Computing & Applications 2018) - Lung Nodule Malignancy Classification in Chest Computed Tomography Images using Transfer Learning and Convolutional Neural Networks - 4º Autor - Qualis B1 (CC);
- (Journal of Computational Science 2019) - Control of Singularity Trajectory Tracking for Robotic Manipulator by Genetic Algorithms - 2º Autor - Qualis B1 (CC);
- (Computer Networks 2019) - Deep Learning IoT System for Online Stroke Detection in Skull Computed Tomography Images - 2º Autor - Qualis A1 (CC);
- (FGCS 2019) - A New Approach for Mobile Robot Localization Based on an Online IoT System - 2º Autor - Qualis A2 (CC);
- (IJCNN 2019) - Evaluation of Heart Disease Diagnosis Approach using ECG Images - 5º Autor - Qualis A1 (CC);
- (BRACIS 2019) - A New Approach to Navigation of Unmanned Aerial Vehicle using Deep Transfer Learning - 1º Autor - Qualis B2 (CC);
- (SIBGRAPI 2019) - An Approach to Classify Chronic Kidney Diseases using

Scintigraphy Images - 4º Autor - Qualis B1 (CC);

- (SIBGRAPI 2019) - A New Strategy for Mobile Robots Localization based on Omnidirectional Sonar Images and Machine Learning - 3º Autor - Qualis B1 (CC).

1.4 Estrutura

Além do capítulo de Introdução (Seção 1), esta dissertação apresenta outros 5 capítulos. O capítulo 2 apresenta métodos de aprendizado de máquina consolidados na literatura, os quais foram usados para avaliar a abordagem proposta. O capítulo 3 traz as principais arquiteturas de Redes Neurais Convolucionais e seus conceitos fundamentais. O capítulo 4 descreve a metodologia utilizada nesta pesquisa, explicando as etapas de processamento do banco de dados, elaborando a abordagem proposta e sua respectiva avaliação. O capítulo 5 apresenta os resultados obtidos e discussões, com as imagens reais e sintéticas utilizando os dois tipos de câmera. Finalmente, o capítulo 6 contém as conclusões do estudo e as propostas para trabalhos futuros.

2 Métodos de Aprendizagem de Máquina

Neste capítulo, os classificadores empregados nesta dissertação são apresentados: Naive Bayes (Seção 2.1), k -Vizinhos Mais Próximos (Seção 2.2), Floresta Aleatória (Seção 2.4), Máquina de Vetores de Suporte (Seção 2.3) e Perceptron Múltiplas Camadas (Seção 2.5).

2.1 Naive Bayes

Em Aprendizagem de Máquina, há um conjunto de classificadores probabilísticos simples conhecidos como Naive Bayes. Com base no teorema de Bayes, esses classificadores assumem independência condicional de classe entre os atributos. Em outras palavras, um classificador Naive Bayes presume que, para uma dada classe, a probabilidade de presença (ou ausência) de uma determinada característica não está relacionada à presença de qualquer outra característica (THEODORIDIS; KOUTROUMBAS, 2008). De modo mais formal, o teorema de Bayes afirma a seguinte relação:

$$P(c|f_1, \dots, f_k) = \frac{P(c) \cdot P(f_1, \dots, f_k|c)}{P(f_1, \dots, f_k)}, \quad (2.1)$$

onde c é a variável de classe, $P(c)$ pode ser interpretada como a frequência relativa da classe no conjunto de treinamento (Probabilidade *a Priori*), k é o número de atributos e de f_1 a f_k são os elementos vetoriais de atributos dependentes. Com base na proposição da independência, pode-se afirmar que:

$$P(f_i|c, f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_k) = P(f_i|c). \quad (2.2)$$

Assim, o teorema de Bayes pode ser reconfigurado como:

$$P(c|f_1, \dots, f_k) = \frac{P(c) \cdot \prod_{i=1}^k P(f_i|c)}{P(f_1, \dots, f_k)}. \quad (2.3)$$

No entanto, como $P(f_1, \dots, f_k)$ é constante para todas as classes possíveis, a Equação 2.3 pode ser ainda mais simplificada, utilizando a regra de decisão de Máxima Probabilidade *a Posteriori* (MAP), a qual é descrita matematicamente como:

$$P(c|f_1, \dots, f_k) \propto P(c) \cdot \prod_{i=1}^k P(f_i|c) \quad (2.4)$$

$$\hat{c} = \underset{c}{\operatorname{argmax}} P(c) \cdot \prod_{i=1}^k P(f_i|c). \quad (2.5)$$

Isso significa que, por exemplo, para classificar corretamente uma imagem, será necessário executar três etapas. Primeiro, para cada classe, será calculado o produto da probabilidade de ocorrência das características (probabilidade *a posteriori*). Em seguida, haverá a multiplicação dessas probabilidades pela frequência relativa (no conjunto de treinamento) de sua respectiva classe (probabilidade *a priori*). Por fim, será selecionada a classe com maior probabilidade.

Além disso, como mencionado anteriormente, existem vários classificadores Naive Bayes. O utilizado nesta dissertação foi o Naive Bayes Gaussiano, que basicamente assume que a probabilidade *a priori*, ou mesmo função densidade de probabilidade, tem distribuição Gaussiana, também conhecida como distribuição Normal, e pode ser matematicamente descrita como:

$$P(f_i|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \cdot \exp\left(-\frac{(f_i - \mu_c)^2}{2\sigma_c^2}\right) \quad (2.6)$$

onde μ_c e σ_c representam a média e o desvio padrão de uma determinada classe e são calculados a partir das amostras no conjunto de treinamento.

2.2 k -Vizinhos Mais Próximos

k -Vizinhos Mais Próximos (*k-Nearest Neighbors*, kNN) é um método de aprendizado de máquina supervisionado, fundamentado em um algoritmo de aprendizado não-paramétrico e baseado em instância (DUDA; HART; STORK, 2001).

Ao não fazer suposições explícitas sobre a função de densidade de probabilidade dos dados, o kNN impede, ao mesmo tempo, o processo de aprender essa distribuição e os perigos potenciais de modificá-la. No entanto, vale a pena ressaltar que esse treinamento/aprendizado mínimo traz custos consideráveis em relação à demanda de memória e ao tempo de computação. A demanda de memória é um fator crítico, pois esse classificador precisa armazenar um número potencialmente alto de instâncias (amostras com seus atributos e rótulos). Além disso, como é necessário processar todo o conjunto de dados para prever a classe de uma determinada observação, o tempo de computação também torna-se um fator crítico para o algoritmo kNN.

O processo de inferência do kNN pode ser descrito, à grosso modo, como um processo de votação majoritária entre as k amostras de treinamento mais próximas da instância invisível, em que o voto de uma determinada amostra de treinamento é o rótulo/classe e quanto mais perto significa que o valor em uma determinada métrica de dissimilaridade é

menor, entre elas as distâncias euclidiana, Mahalanobis, Minkowski, Hamming e outras. A distância euclidiana, por exemplo, é uma das métricas de dissimilaridade mais populares e pode ser formulada como:

$$d(\mathbf{X}, \mathbf{X}^i) = \sqrt{(x_1 - x_1^i)^2 + (x_2 - x_2^i)^2 + \dots + (x_p - x_p^i)^2} \quad (2.7)$$

onde $\mathbf{X}$ e $\mathbf{X}^i$ são dois pontos/instâncias arbitrários em um espaço dimensional p e d é uma função de medida de dissimilaridade.

Formalmente, dado que $\mathbf{X}$ é uma observação invisível e $k \in N^*$, o algoritmo de classificação kNN pode ser dividido em três etapas. Na primeira etapa, o kNN calcula a distância entre $\mathbf{X}$ e cada amostra de treinamento (as k amostras de treinamento com valores mais baixos de distância pertencem a um conjunto $\mathbf{A}$). Na segunda etapa, para cada classe possível, a probabilidade condicional é estimada. De forma específica, a probabilidade de uma observação invisível $\mathbf{X}$ pertencer à classe c é dada pela taxa de amostras do subconjunto $\mathbf{A}$ (com k elementos) que possuem um rótulo igual a c . Por fim, a instância invisível é rotulada com a classe mais provável (valor mais alto).

O processo de estimativa descrito para a segunda etapa de classificação do kNN pode ser matematicamente descrito como:

$$P(c) = \frac{1}{K} \cdot \sum_{i \in \mathbf{A}} I(y_i, c) \quad (2.8)$$

onde c é uma determinada classe do problema, y_i é um determinado rótulo da amostra $\mathbf{A}$, $\mathbf{A}$ são as amostras de treinamento mais próximas k e I é uma função indicadora que retorna 1 (um), quando suas entradas são iguais, ou 0 (zero), caso contrário.

2.3 Máquina de Vetores de Suporte

Para entender o classificador Máquina de Vetores de Suporte (*Support Vector Machines*, SVM) é importante compreender os conceitos de classificador de hiperplano, classificador de margem máxima e classificador de margem suave.

2.3.1 Classificador de Hiperplano

Primeiramente, suponha que cada instância de um determinado problema possa ser representada como um ponto $\mathbf{X}$ em um espaço dimensional p , onde cada dimensão/eixo desse espaço está associado a um recurso desse problema. Este ponto pode ser formulado como:

$$\mathbf{X} = (x_1, x_2, \dots, x_p) \quad (2.9)$$

Em seguida, também suponha que exista um hiperplano dimensional $p - 1$ que separe perfeitamente um conjunto de instâncias de um determinado problema por seus rótulos. Sua definição matemática é dada pela equação:

$$\beta_0 + \beta_1\alpha_1 + \beta_2\alpha_2 + \dots + \beta_p\alpha_p = 0. \quad (2.10)$$

onde β são os coeficientes do hiperplano, $\alpha_1, \dots, \alpha_p \in P_O$ e P_O é dado um ponto cruzado pelo hiperplano.

Portanto, de acordo com a posição relativa de um determinado ponto $\mathbf{X}$ para um determinado hiperplano, é possível definir uma função de indicador que use esse hiperplano como limite. Esta regra de decisão é dada por:

$$y = \begin{cases} +1, & \text{if } \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_px_p > 0 \\ -1, & \text{if } \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_px_p < 0 \end{cases}. \quad (2.11)$$

onde os pontos do lado positivo do hiperplano estão associados a um rótulo positivo ($y = +1$) e o oposto também é válido.

Em seguida, o Classificador de Hiperplano usa a equação 2.11 como um processo de estimativa, onde os pontos são amostras invisíveis e sua posição relativa ao hiperplano define de qual classe eles pertencem.

2.3.2 Classificador de Margem Máxima

No entanto, se um conjunto de amostras for linearmente separável (perfeitamente separado usando um hiperplano), um número infinito desses hiperplanos poderá existir. Para selecionar os melhores hiperplanos dentre esses, foi proposto um método chamado Classificador de Margem Máxima.

O principal intuito por trás do Classificador de Margem Máxima é que o hiperplano de margem máxima, que consiste no hiperplano mais distante de todas as observações de treinamento, é a decisão ideal. O termo "mais distante" representa a soma com o maior valor da distância perpendicular de todas as amostras de treinamento a um determinado hiperplano. Em contrapartida, a menor dessas distâncias perpendiculares é conhecida como margem.

Portanto, encontrar esse melhor hiperplano de separação que pode ser formulado como um problema de maximização:

$$\underset{\beta_1, \dots, \beta_p}{\text{maximize}} M, \quad (2.12)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M, \forall i = 1, \dots, n. \quad (2.13)$$

Observe que M é o comprimento da margem, n é o número de amostras de treinamento, $X_1, \dots, X_n \in R_p$ e rótulos de classe $y_1, \dots, y_n \in \{-1, +1\}$.

O hiperplano resultante dessa otimização é conhecido como hiperplano de margem máxima. Para criar o Classificador de Margem Máxima, este hiperplano ideal deve ser aplicado à equação 2.11.

2.3.3 Classificador de Margem Suave

No entanto, uma das principais desvantagens do Classificador de Margem Máxima é sua sensibilidade a observações individuais. Para superar isso, o Classificador de Margem Suave, também conhecido como Classificador de Vetor de Suporte, adiciona uma variável de folga ϵ para cada observação no conjunto de treinamento. O objetivo dessa variável é permitir que algumas amostras sejam classificadas incorretamente se isso aumentar a margem do classificador.

Portanto, o processo de otimização para encontrar o melhor Classificador de Margem Suave para separar a maior parte das amostras do conjunto de treinamento pode ser modelado como o seguinte problema de otimização:

$$\underset{\beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M, \quad (2.14)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \quad (2.15)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C, \quad (2.16)$$

onde C é um hiperparâmetro maior que zero.

2.3.4 Classificador SVM

Uma das principais limitações do Classificador de Margem Suave é que seu limite de decisão é sempre linear. Para estender seus conceitos a problemas não-linearmente separáveis, o SVM é proposto. A idéia principal do classificador SVM é transformar o espaço dos atributos de entrada através de uma maneira específica, para que um limite de decisão linear (nesse novo espaço) possa separar corretamente as amostras. Como descrever exatamente como o Classificador de Vetores de Suporte é calculado está além do escopo

desta dissertação, verifica-se que a regra de decisão do SVM pode ser matematicamente descrita como:

$$f(X) = \beta_0 + \sum_{i=1}^n \alpha_i K(X^T, X'), \quad (2.17)$$

onde α é um coeficiente do SVM aprendido durante o treinamento e K é a transformação do espaço de destino. Por exemplo, um *kernel* linear (transformação) pode ser expresso por:

$$K(X, X') = \sum_{j=1}^p X_j^T X'_j, \quad (2.18)$$

e a transformação radial é definida como:

$$K(X, X') = \exp(-\gamma \sum_{j=1}^p (X_j^T X'_j)^2). \quad (2.19)$$

2.4 Floresta Aleatória

Proposta pela primeira vez por Ho (1995), Floresta Aleatória (*Random Florest* - RF) é um método de conjunto que combina as estimativas realizadas por um conjunto de Árvores de Decisão (*Decision Trees* - DTs) (BREIMAN, 2001). Em geral, DTs são um algoritmo de aprendizagem de máquina do tipo supervisionado não-paramétrico utilizado para tarefas de classificação e regressão. No contexto da classificação, seu objetivo é estimar o rótulo de uma determinada instância usando regras de decisão (limites) aprendidas com os dados de treinamento.

No entanto, encontrar a árvore de decisão ideal é um problema NP-completo (BREIMAN, 2001), o algoritmo prático de treinamento de DTs produz uma solução subótima e pode ser associado conceitualmente a uma busca gananciosa, na qual são tomadas decisões locais em cada nó da árvore. Outra desvantagem das DTs é a suscetibilidade de aprender também o ruído no conjunto de treinamento (*overfitting*).

Para reduzir a variação das predições das DTs, Floresta Aleatória usa uma abordagem de conjunto chamada *Random Bagging*. Nesse contexto, o *Random Bagging* é construído e treina vários estimadores (por exemplo, DTs) em diferentes subconjuntos aleatórios do conjunto de treinamento e, em seguida, combina seus resultados (por exemplo, estima o voto da maioria) para compor a predição final. Mais formalmente, acontece que cada DT pode ser vista como uma vizinhança ponderada (*weighted neighborhood*) (LIN;

JEON, 2006) e é expressa como:

$$\hat{y} = \sum_{i=1}^n W(X_i, X'_i) y_i, \quad (2.20)$$

onde X' é uma observação invisível, $\hat{y}$ é seu rótulo estimado, X_i é a i – ésima amostra de treinamento, y_i é seu respectivo rótulo e W é a função de peso.

Essa função de peso é alterada de acordo com o método de classificação usado, por exemplo, no contexto de Árvores de Decisão:

$$W(X_i, X'_i) = \begin{cases} \frac{1}{k} & , \text{ if } X_i \in K \\ 0 & , \text{ otherwise } \end{cases}, \quad (2.21)$$

onde K são pontos de k na mesma folha que X' . Observe que, a única diferença entre essa função de peso e a calculada para o método kNN é que K é um conjunto composto pelos pontos de k mais próximos da observação invisível. Além disso, é possível estender a equação 2.21 para compor a predição da Floresta Aleatória, que é dada por:

$$\hat{y} = \frac{1}{m} \sum_{j=1}^m \sum_{i=1}^n W_j(X_i, X'_i) y_i, \quad (2.22)$$

onde m é o número de estimadores (árvores) usados e W_j são suas respectivas funções de peso.

2.5 Perceptron Múltiplas Camadas

Perceptron Múltiplas Camadas (*Multilayer Perceptron*, MLP) é um método de aprendizagem de máquina do tipo supervisionado não-paramétrico. Além disso, esse classificador também é uma extensão do classificador Perceptron para conjuntos de dados não-linearmente separáveis (HAYKIN, 2008).

Introduzido por ROSEMBLATT (1958), o algoritmo Perceptron calcula uma soma ponderada das entradas e aplica o resultado a uma função de ativação. Abstratamente, cada peso indica a relevância de um determinado recurso para prever uma determinada classe. Mais formalmente, a previsão de Perceptron pode ser expressa como mostra a Equação 2.23.

$$\hat{y} = \varphi \left(\sum_{i=1}^p x_i w_i + b \right), \quad (2.23)$$

onde φ é uma função de ativação (por exemplo, logística/*softmax*), x é um vetor de entrada, p é o número de atributos de uma determinada entrada, w é o vetor de pesos,

que combinado com b , são parâmetros aprendíveis do Perceptron. Além disso, o processo de treinamento desse classificador é realizado de forma iterativa, fornecido por:

$$w_i(t+1) = w_i(t) + \eta \cdot (y - \hat{y}(t)) \cdot x_i \quad \forall i = 1, \dots, p, \quad (2.24)$$

em que t é uma determinada etapa do tempo de treinamento, y é o rótulo verdadeiro, $\hat{y}$ é o rótulo previsto e η é a taxa de aprendizado. No entanto, apesar de ter sido um algoritmo simples e poderoso, o Perceptron não é capaz de executar uma classificação não linear.

Para superar isso, o MLP foi proposto. Esse classificador consiste em um gráfico computacional composto por uma pilha de camadas de perceptron. Cada uma dessas camadas, exceto a última, possui suas funções de ativação logística, as quais são empregadas para criar um classificador linear, substituídas por outra função não-linear, como a função sigmóide, que é usada para executar uma transformação não-linear no espaço de atributos. Mais formalmente, a regra de decisão do MLP pode ser matematicamente formulada como:

$$\hat{y}_{l_k} = \begin{cases} \varphi_{0_k}(\sum_{i=1}^p x_i \cdot w_{0_{k_i}} + b_{0_k}) & \text{if } l = 0 \\ \varphi_{l_k}(\sum_{i=1}^D \hat{y}_{l-1_i} \cdot w_{l_{k_i}} + b_{l_k}) & \text{otherwise} \end{cases}, \quad (2.25)$$

onde x é o vetor de entrada de p dimensões, l é um determinado índice de camada, D é o número de neurônios na camada $l-1$ e φ é a função não-linear. Vale ressaltar que, em geral, os parâmetros aprendíveis do MLP (pesos e viés) são inicializados com valores aleatórios próximos de zero.

Além disso, para melhorar seu desempenho, MLP geralmente utiliza matematicamente um processo iterativo, expresso pela equação:

$$\hat{w}_{l_{k_i}}(t+1) = \begin{cases} w_{0_{k_i}}(t) + \eta \cdot \delta_{l_i}(t) \cdot x_k(t) & \text{if } l = 0 \\ w_{l_{k_i}}(t) + \eta \cdot \delta_{l+1_i}(t) \cdot \hat{y}_{l_k}(t) & \text{otherwise} \end{cases}, \quad (2.26)$$

onde η é a taxa de aprendizado e $\delta_{l_k}(t)$ é o gradiente local do k -ésimo neurônio da camada l -ésima. Esse gradiente local é dado por:

$$\delta_{l_k}(t) = \begin{cases} \varphi'_{l_k}[u_{l_k}(t)] \cdot \frac{1}{2M} \sum_{o=1}^M [y_{l_o}(t) - \hat{y}_{l_o}(t)]^2 & \text{if } l = z \\ \varphi'_{l_k}[u_{l_k}(t)] \cdot \sum_{j=1}^J w_{l_{k_j}}(t) \cdot \delta_{l+1_j}(t) & \text{otherwise} \end{cases}, \quad (2.27)$$

onde z é a última camada, M é o número de neurônios na última camada (z) e J é o número total de neurônios na camada l -ésima.

Observe que a derivada $\varphi'_i[u_{l_i}(t)]$ possui valores diferentes, dependendo da função de ativação escolhida, por exemplo:

$$\varphi'_{l_k}[u_{l_k}(t)] = \begin{cases} y_{l_k}(t)[1 - y_{l_k}(t)], & \text{if logistic/sigmoid} \\ \frac{1}{2}[1 - (y_{l_k}(t))^2], & \text{if hyperbolic tangent} \end{cases} \quad (2.28)$$

3 Rede Neural Convolucional

Neste capítulo, é apresentada uma visão geral sobre os blocos de construção essenciais das modernas arquiteturas de Rede Neural Convolucional (Convolutional Neural Network, CNN) com seu princípio de funcionamento e modelos.

Uma CNN consiste em um algoritmo de Aprendizagem Profunda desenvolvido por LeCun et al. (1990) que somente tornou-se popular quando Krizhevsky, Sutskever e Hinton (2012) obteve bons resultados no ImageNet 2012 *LargeScale Visual Recognition Challenge* (ILSVRC-2012). Esse desafio consiste em resolver uma tarefa de classificação com 1000 classes e mais de 1 milhão de amostras (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; DENG et al., 2009). Desde então, várias arquiteturas de CNNs foram propostas na tentativa de superar os resultados obtidos por Krizhevsky, Sutskever e Hinton (2012), dentre elas: VGG16 (SIMONYAN; ZISSERMAN, 2014b), VGG19 (SIMONYAN; ZISSERMAN, 2014b), MobileNet (HOWARD et al., 2017b), ResNet50 (HE et al., 2016) e Inception-ResNet-V2 (SZEGEDY et al., 2016).

Em uma Rede Neural Convolucional, uma convolução representa uma operação linear que implica na multiplicação de um conjunto de pesos com a entrada, bastante similar com uma rede neural tradicional (LECUN et al., 1990). Considerando que o método foi desenvolvido para entrada bidimensional, a multiplicação é feita entre uma matriz de dados de entrada e uma matriz bidimensional de pesos, a qual é chamada de filtro ou *kernel*, do inglês (LECUN et al., 1990).

CNN consegue manter as correlações entre os *pixels* vizinhos da imagem, uma vez que opera sobre campos receptivos locais, efetuando o processo de convolução. Logo, ela consegue diminuir a sensibilidade à translação, à rotação e à distorção das imagens (LECUN et al., 1998). Outros tipos de redes neurais não têm a capacidade de preservar esse tipo de relação, visto que consideram as imagens como uma entrada unidimensional.

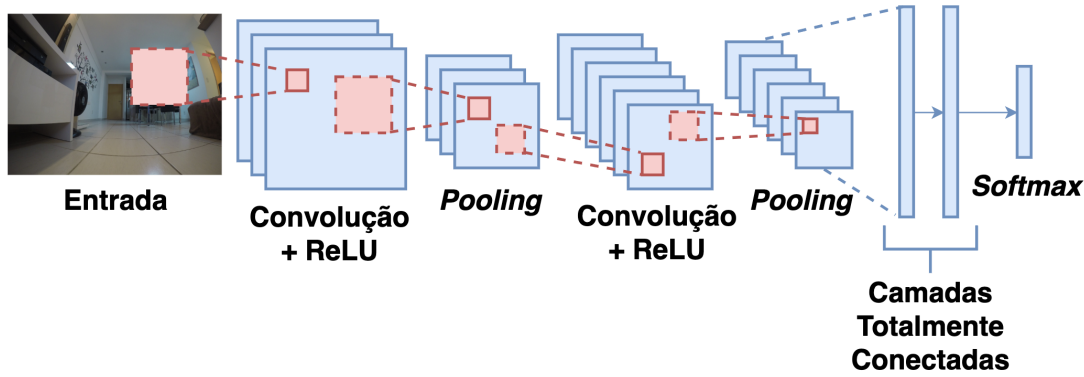
3.1 Estrutura de uma Rede Neural Convolucional

As Redes Neurais Convolucionais são redes neurais *feedforward*, na qual sua arquitetura foi elaborada para tornar a rede invariante à translação, à rotação e à distorção da imagem de entrada (LECUN et al., 1998). Elas são formadas por conexões locais, por pesos compartilhados, pela camada de *pooling* e outras diferentes camadas (LECUN; BENGIO; HINTON, 2015).

A organização da rede é dada por camadas com funções distintas. A camada convolutiva é responsável pela extração das características da imagem. A camada de *pooling*

efetua uma sub-amostragem da imagem, enquanto as camadas totalmente conectadas têm como tarefa interpretar as características extraídas. A Figura 2 apresenta o arranjo geral da estrutura de uma CNN.

Figura 2 – Modelo genérico de uma Rede Neural Convolucional.



Fonte: Autor.

3.1.1 Camada Convolutiva

Na camada de convolução, as unidades são dispostas em *feature maps*, na qual cada unidade está ligada a uma porção da camada precedente através de um conjunto de pesos denominados filtros ou *kernel* (LECUN; BENGIO; HINTON, 2015). Os neurônios da rede empregam um método chamado pesos compartilhados, que equivale a diversas conexões sendo manipuladas por um mesmo parâmetro (peso) (LECUN, 1989).

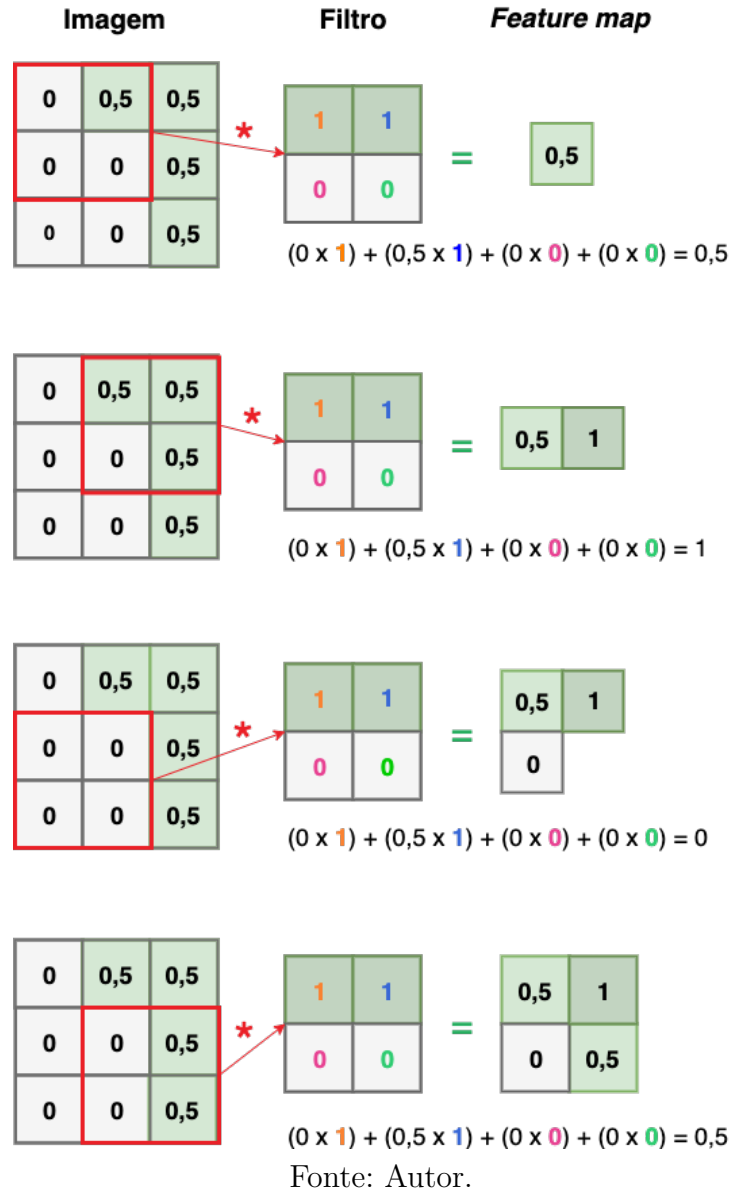
Esse método restringe consideravelmente a quantidade de parâmetros da rede. Além disso, esse modelo de organização das conexões configura o processo de convolução, visando extrair determinadas características das imagens, dentre elas as linhas e os contornos. Os filtros também podem consistir de duas ou três dimensões. A cada instante, eles ligam-se a uma certa parte de uma matriz, denominada de campo receptivo local. Nessa situação, os filtros podem ser interpretados como pequenos quadrados que deslizam sobre uma matriz, realizando a operação de convolução. Esses filtros têm um tamanho definido, como 5×5 ou $5 \times 5 \times 3$ (para entradas tridimensionais) e sua ação de deslocamento sobre a matriz de *pixels* da imagem é chamada de *stride*.

A Figura 3 apresenta a aplicação de um filtro (2×2) em uma imagem (3×3), resultando em um *feature map* (2×2). O cálculo do tamanho do *feature map* é realizado segundo a Equação 3.1:

$$T_m = \frac{M - k}{s} + 1 \quad (3.1)$$

onde T_m representa o tamanho da *feature map*, M é a dimensão da imagem, k é o tamanho do filtro (*kernel*) e s consiste no passo do deslocamento (*stride*). As variáveis T_m , M e k

Figura 3 – Operação de convolução em imagem.



correspondem ao tamanho em uma das dimensões (horizontal ou vertical), mas é padrão o uso de imagens e filtros quadrados (altura e largura iguais). Desse modo, as variáveis representam os tamanhos em ambas as dimensões. O Algoritmo 1 resume o processo executado na camada de convolução.

Algoritmo 1 Convolução

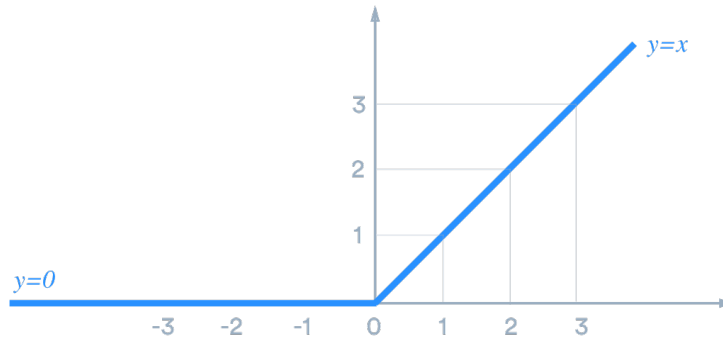
- 1: Atribuição de valores pequenos aos pesos w
 - 2: **para** $i = 1$ até $(M - k)/s + 1$ **faça**
 - 3: **para** $j = 1$ até $(N - k)/s + 1$ **faça**
 - 4: $u_{i,j} = \sum_{c=1}^{c=1} \sum_{d=1}^{d=1} x_{i \cdot j - 1 + c, j \cdot s - 1 + d} \cdot w_{c,d} + b$
 - 5: $y_{i,j} = f(u_{i,j})$
 - 6: **fim para**
 - 7: **fim para**
-

No Algoritmo 1, $x_{i,j}$ consiste no valor do *pixel* na posição (i, j) da imagem x . M corresponde à altura e N corresponde à largura da imagem. A variável $u_{i,j}$ representa o valor da posição (i, j) do *feature map* e $y_{i,j}$ é o resultado da aplicação da variável $u_{i,j}$ pela função de ativação. A variável $w_{c,d}$ simboliza o peso do filtro na posição (c, d) .

3.1.2 Não-linearidade ReLU

Para que qualquer tipo de rede neural seja poderosa, ela precisa conter não-linearidade. Por isso, em uma CNN, o resultado da operação de convolução é passado através da função *ReLU* (do inglês, *Rectified Linear Unit*) (??). Logo, os valores finais dos *feature maps* não são realmente os valores das somas, mas a função *ReLU* aplicada a eles.

Figura 4 – A função de ativação da Unidade Linear Retificada (*Rectified Linear Unit*, ReLU)).



Fonte: (<https://goo.gl/FLHUrZ>).

A Figura 4 apresenta uma representação gráfica da função ReLU. A consolidação dessa função foi motivada principalmente pelos experimentos de Nair e Hinton (2010), os quais indicam que o uso de ReLUs acelera significativamente o processo de convergência do treinamento, quando comparado com outras funções de não-linearidade, como tangente logística ou hiperbólica. A função ReLU pode ser matematicamente definida como na Equação 3.2.

$$g(z) = \max(0, z) \quad (3.2)$$

3.1.3 Camada de *Pooling*

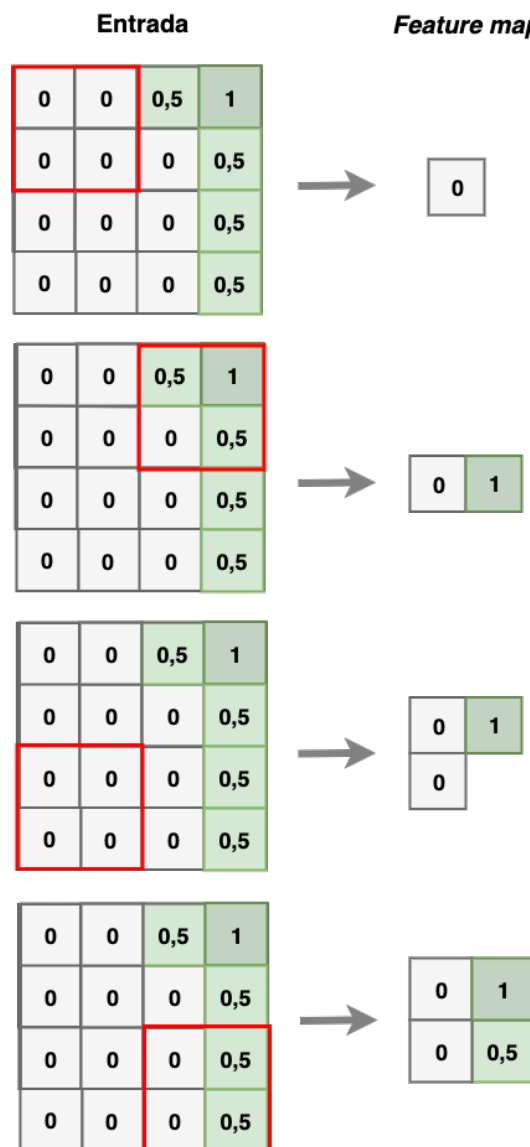
Como a camada convolutacional não reduz o tamanho da entrada, um passo extra pode ser necessário para diminuir a saída da convolução, preferencialmente sem adicionar novos parâmetros ao modelo. Por isso, é empregada a camada de *pooling*. Essa camada tem como objetivo alcançar um determinado nível de invariância ao deslocamento, diminuindo a resolução dos *feature maps* (GU et al., 2015).

A camada de *pooling* opera de modo similar a uma camada convolutiva, contudo, o *stride* normalmente tem o mesmo tamanho do filtro (ao contrário da camada anterior, onde

o *stride* era 1 (um)). Dessa forma, os campos receptivos tornam-se totalmente distintos e a redução dos *features maps* alcança em torno de 75%. Os tipos mais comuns de *pooling* são o *max pooling* e o *average pooling*. No *max pooling*, o maior valor do campo receptivo é selecionado, enquanto no *average pooling*, a média dos valores é calculado (BENGIO; GOODFELLOW; COURVILLE, 2015).

Como pode ser visto na Figura 5, a *camada de pooling* tem como entrada uma imagem em tons de verde e efetua a operação de *max pooling* sobre a imagem. A saída retornada é uma imagem com a metade da altura e metade da largura original. O Algoritmo 2 descreve o processo de *pooling*, aplicado após uma camada de convolução.

Figura 5 – Demonstração do processo de *max pooling*.



Fonte: Autor.

Algoritmo 2 *Pooling*

```

1: para  $i = 1$  até  $(M - k)/s + 1$  faça
2:   para  $j = 1$  até  $(N - k)/s + 1$  faça
3:     se max pooling então
4:        $y_{i,j} = \max(x_{(i-1) \cdot k + 1, (i-1) \cdot k + 1}, \dots, x_{i \cdot k, j \cdot k})$ 
5:     fim se
6:     se average pooling então
7:        $y_{i,j} = (\sum_{c=1}^k \sum_{d=1}^k x_{i \cdot s - 1 + c, j \cdot s - 1 + d}) / (k^2)$ 
8:     fim se
9:   fim para
10: fim para

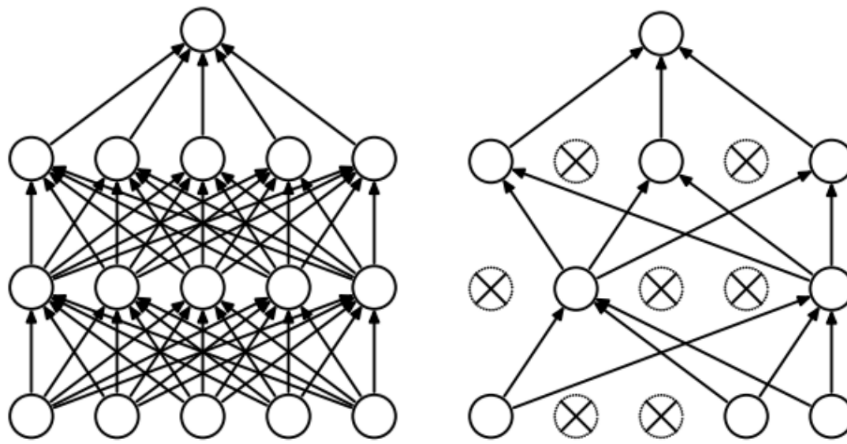
```

3.1.4 *Dropout*

É bastante comum que uma rede aprenda a classificar muito bem as imagens de seu conjunto de treinamento, todavia, em outras imagens, ela pode ter uma queda no seu desempenho. Para evitar que ocorra esse sobreajuste (em inglês, *overfitting*) em seu aprendizado, utiliza-se a técnica de *Dropout*.

O método de *Dropout* consiste em eliminar aleatoriamente algumas unidades da rede neural (SRIVASTAVA et al., 2014), como mostrado na Figura 6. De acordo com essa idéia, esses neurônios desativados (definidos como zero) não contribuem para o encaminhamento da rede e, conseqüentemente, não têm seus pesos atualizados durante a propagação reversa do gradiente. Uma probabilidade p é estabelecida e, assim, cada unidade tem essa probabilidade de seu sinal ser propagado na rede.

Figura 6 – Em (a), a ilustração representa o fluxo de informações convencionais durante uma passagem direta de uma rede e, em (b), esse fluxo é interrompido (definido como zero) em algum neurônio, consistindo no processo chamado de *Dropout*.



(a) Módulo simples de criação.

(b) Rede Neural Padrão.

Fonte: (SRIVASTAVA et al., 2014).

3.1.5 Camada Totalmente Conectada

A imagem de entrada é processada pelas camadas convolutivas e de *pooling*, que distinguem desde as características mais simples até às mais complexas, chegando nas camadas totalmente conectadas. Geralmente, a camada anterior é uma camada de *pooling*, na qual seus *feature maps* dispõem de mais de uma dimensão e, por isso, são redimensionados para apenas uma dimensão, a qual é representada em forma de vetor. Dessa forma, é possível conectar-se à parte final da rede. As camadas totalmente conectadas operam como uma rede neural multicamadas *feedforward* e são responsáveis pela interpretação das características extraídas pelas camadas iniciais (LECUN et al., 1998). O Algoritmo 3 descreve o processamento da camada totalmente conectada.

Algoritmo 3 Camada totalmente conectada

```

1: se camada anterior é de pooling ou de convolução então
2:   redimensionar( $x$ )
3: fim se
4: para todo neurônio  $j$  da camada totalmente conectada faça
5:   para toda entrada  $i$  da camada anterior faça
6:      $u_j = \sum_{i=1}^n x_i \cdot w_{i,j} + b_j$ 
7:      $y_j = f(u_j)$ 
8:   fim para
9: fim para

```

3.1.6 Softmax

A última camada é responsável por apresentar os resultados. O número de neurônios dispostos nessa camada é determinado pela quantidade de classes do problema. Cada neurônio exibe a saída em forma de probabilidade (0 a 1) e a resposta consiste no número do neurônio com saída de maior valor. Esse processo é realizado a partir da função *Softmax*, a qual é descrita pela Equação 3.3:

$$y_j = \frac{e^{y(j)}}{\sum_{i=1}^n e^{y(i)}} \quad (3.3)$$

onde a saída y do neurônio j passa a ser o valor dessa mesma saída dividido pelo somatório de todas as saídas dos neurônios dessa camada.

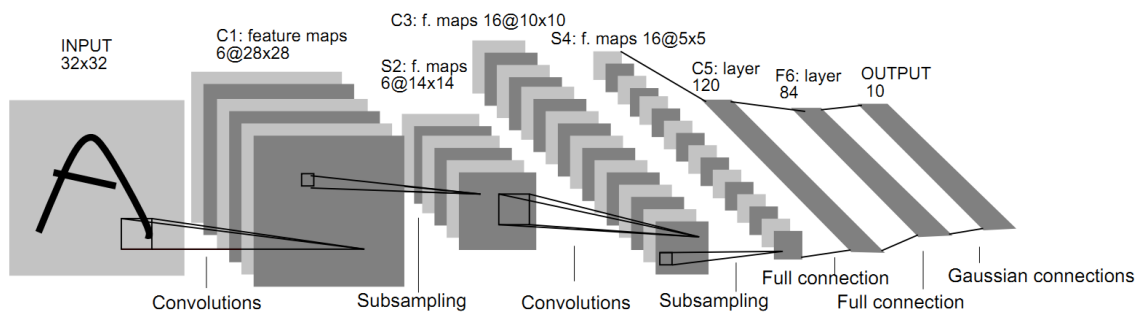
3.2 Lenet-5

Inspiradas pelas Redes Neurais Artificiais (HAYKIN, 2008), arquiteturas Neocognitron (FUKUSHIMA; MIYAKE, 1982), a estratégia de otimização *back-propagation* (BENGIO; GOODFELLOW; COURVILLE, 2015) e o conceito de pesos compartilhados

(também chamado de conectividade local (RUMELHART; HINTON; WILLIAMS, 1986), e ainda motivado pela necessidade de resolver problemas de reconhecimento de imagem do mundo real sem um estágio de pré-processamento computacionalmente caro, LeCun et al. (1998) desenvolveu uma rede convolucional pioneira de 7 níveis para classificar números escritos à mão em imagens de cheques (LECUN et al., 1990).

A arquitetura apresentada na Figura 7 foi nomeada como LeNet-5 e as suas principais contribuições para o campo de pesquisa em Aprendizagem Profunda são: a introdução de camadas convolucionais e de sub-amostragem.

Figura 7 – Arquitetura da Rede Neural Convolucional LeNet-5.



Fonte: (LECUN et al., 1998).

3.3 AlexNet

Apesar de ter sido desenvolvido no final dos anos 80, as CNNs só se tornaram populares em 2012, quando Krizhevsky, Sutskever e Hinton (2012) propôs um modelo de CNN chamado AlexNet. Esse modelo ganhou, por uma grande margem, o desafio *Large-Scale Visual Recognition Challenge* (ILSVRC) (RUSSAKOVSKY et al., 2015), para a tarefa de detecção de objetos, uma importante referência para as técnicas de visão computacional (DENG et al., 2009; KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Vale ressaltar que, a atenção dada a essa competição do ILSVRC deve-se principalmente ao tamanho e à complexidade do conjunto de dados. O *ImageNet*, nome dado a esse conjunto de dados, possui 1000 categorias e mais de 1,2 milhões de imagens (DENG et al., 2009).

Assim, desde a descoberta da AlexNet, todo o vencedor subsequente do ILSVRC usou modelos baseados na CNN. No entanto, apesar desse entusiasmo, o modelo AlexNet tem uma arquitetura semelhante ao seu antecessor, LeNet-5 (LECUN et al., 1998), sendo composto basicamente por camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas.

Além disso, entre as contribuições mais notáveis do artigo da AlexNet para o campo de pesquisa das CNNs estão: Unidade Linear Retificada (*Rectified Linear Unit*, ReLU) e

Dropout.

3.4 Arquiteturas VGG

As experiências realizadas no artigo do Oxford Visual Geometry Group (VGG) (SIMONYAN; ZISSERMAN, 2014a), incluindo o modelo vencedor das tarefas de detecção e localização do ILSVRC 2014 (RUSSAKOVSKY et al., 2015), demonstraram que, para todas as arquiteturas avaliadas (11, 13, 16 ou 19 camadas), existe uma correlação positiva entre o aumento da profundidade (número de camadas) dos modelos e sua precisão no *ImageNet Dataset*. Esse aumento, sem superajustar o modelo, somente foi possível através de uma estratégia de regularização denominada Convoluções Fatoradas.

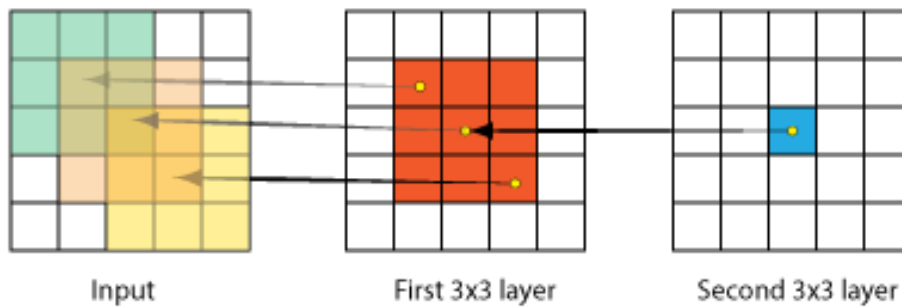
3.4.1 Convoluções Fatoradas

Inspirado no artigo Zeiler e Fergus (2014), que usava um tamanho de janela receptivo menor (*kernel*/filtro) que o AlexNet (por exemplo, na primeira camada de convolução (ZEILER; FERGUS, 2014), a rede usava *kernels* de tamanho 7×7 , enquanto a AlexNet usava *kernels* de tamanho 11×11), a rede VGG usou *kernels* muito pequenos (3×3) em todas as suas camadas de convolução.

Há três aspectos importantes que devem ser esclarecidos para justificar porque os pequenos *kernels* permitiram à rede VGG dominar a maioria dos modelos anteriores. Primeiro, em uma convolução, um determinado ponto no mapa de ativação é influenciado apenas por uma sub-região de entrada conectada, por exemplo, uma convolução de 3×3 é influenciada por uma janela de entrada 3×3 . Além disso, se uma segunda convolução de 3×3 for realizada, mas agora sobre o primeiro mapa de ativação, cada ponto do novo mapa de ativação será influenciado por uma janela de 3×3 do primeiro. No entanto, como cada ponto do primeiro mapa de ativação combina uma informação de 3×3 da entrada, uma combinação de 3×3 desses primeiros pontos de ativação é influenciada por uma informação de 5×5 da entrada. Portanto, se esse processo for repetido mais uma vez, cada ponto de saída da pilha de convolução 3×3 de três camadas será influenciado por uma sub-região de 7×7 da entrada. Em outras palavras, uma pilha de convolução 3×3 de três camadas tem o mesmo campo receptivo efetivo de uma única convolução 7×7 . A Figura 8 ilustra um exemplo desse processo de equivalência.

Segundo, uma pilha de convolução de 3×3 de três camadas possui menos parâmetros que uma única camada convolucional de 7×7 . Terceiro, como cada camada tem sua não-linearidade, uma pilha de camadas de convolucionais tem mais poder discriminativo do que apenas uma camada. Portanto, as redes VGG tornaram-se bastante populares não apenas por causa de seus bons resultados no ILSVRC 2014 (RUSSAKOVSKY et al., 2015), mas também por sua arquitetura uniforme.

Figura 8 – O campo receptivo efetivo de uma pilha de duas convoluções 3×3 .



Fonte: (SIMONYAN; ZISSERMAN, 2014a).

3.5 Arquiteturas Inception

Desde o LeNet5, os elementos básicos das arquiteturas neurais profundas são os mesmos (camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas), mas isso muda quando a equipe do *Google Brain* lança o *GoogleNet* (SZEGEDY et al., 2015). Inception, como também é chamado o *GoogleNet*, define resultados de ponta na tarefa de detecção ILSVRC 2014 (RUSSAKOVSKY et al., 2015), atingindo uma taxa de erro entre os 5 principais de 6,67%.

Além disso, entre as contribuições dos modelos *Inception* ao campo de pesquisa de Aprendizagem Profunda estão o módulo *Inception*, *Batch-Normalization* e Convoluções Assimétricas.

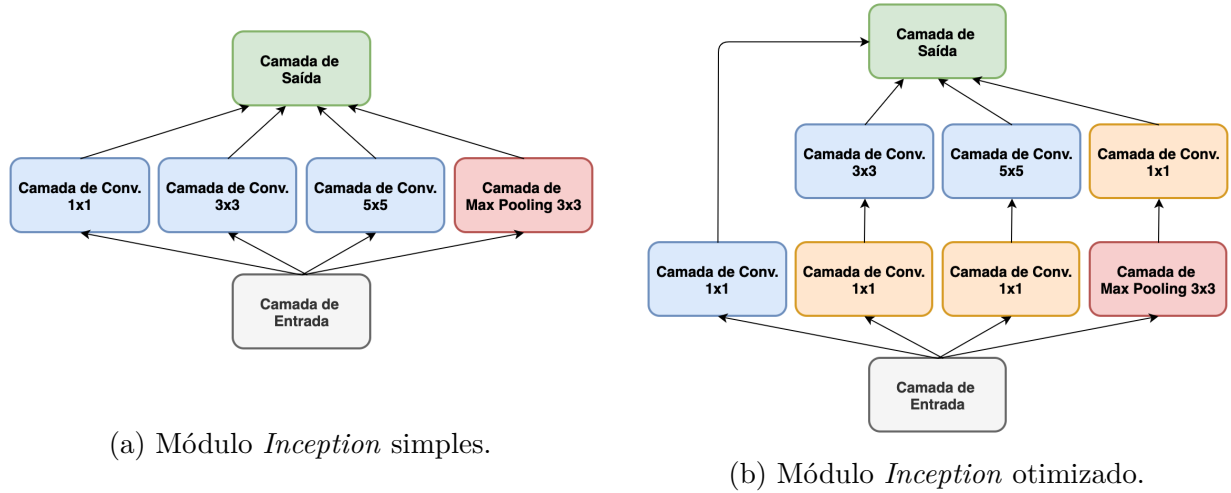
3.5.1 Módulo Inception

Projetar uma arquitetura CNN geralmente consiste em escolher quais operações usar (convolução 1×1 , convolução 3×3 , convolução 5×5 ou *pooling*) em cada camada. Então, por que não executar todas as operações de uma só vez e concatenar suas saídas, aproveitando a extração de recursos em vários níveis? Uma ilustração dessa arquitetura paralela é mostrada na Figura 9(a).

A principal desvantagem deste Módulo *Inception* simples (abordagem paralela descrita anteriormente) é que, depois de algumas camadas como essa, a quantidade de parâmetros adicionados à rede provavelmente se tornaria excessiva durante o treinamento.

No entanto, para abordar a ineficiência desse parâmetro, o *Google Brain* realizou uma convolução de 1×1 para compactar as entradas das camadas de convolução e a saída das camadas de *pooling*. Uma ilustração do Módulo *Inception* é apresentada na Figura 9(b). Essa adaptação simples preservou a maior parte da precisão da abordagem paralela

Figura 9 – Duas ilustrações de arquitetura inicial: (a) Módulo *Inception* simples e (b) Módulo *Inception* com redução de dimensionalidade.



Fonte: Autor.

e reduziu significativamente a quantidade de parâmetros livres.

3.5.2 Camada *batch-normalization*

Um dos problemas mais desafiadores na otimização de redes profundas é o Deslocamento Covariado Interno (*Internal Covariate Shift*) (Ioffe; Szegedy, 2015), que é, basicamente, uma alteração na distribuição de entradas de camadas intermediárias no momento do treinamento. Isso acontece porque as entradas de uma determinada camada são afetadas por todos os pesos das camadas anteriores, que geralmente mudam com a otimização. Além disso, como os pesos das camadas precisam se adaptar a uma nova distribuição de entrada a cada atualização de peso, um efeito negativo aparece e a convergência do modelo se torna mais lenta.

Para resolver o problema de *Internal Covariate Shift*, Ioffe e Szegedy (2015) propôs a normalização em lote *batch-normalization*, um novo mecanismo que incorpora ativações de normalização na própria arquitetura.

Com apenas dois parâmetros extras por ativação, o *batch-normalization* normaliza as saídas do produto interno entre pesos e entradas de uma determinada camada, com base em sua média e desvio padrão. Isso é equivalente a “limpar” os dados, fazendo uma distribuição centralizada em zero e normalizando seu intervalo ao longo do lote. Além disso, entre os benefícios do *batch-normalization* estão: a camada após um *batch-normalization* se concentra em como melhor associar insumos; o *batch-normalization* tem um poderoso efeito de regularização e reduz ou elimina a necessidade de *dropout*.

Considerando γ e β os parâmetros a serem aprendidos no treinamento, a saída é

calculada segundo a Equação 3.4.

$$y = \gamma \hat{x} + \beta, \quad (3.4)$$

onde $\hat{x}$ representa a entrada normalizada. Para realizar o cálculo de $\hat{x}$, a partir da Equação 3.7, em um *mini-batch* $M = \{x_1, x_2, \dots, x_b\}$, sendo b o seu tamanho, é necessário inicialmente calcular a média e a variância do *mini-batch*, através das Equações 3.5 e 3.6, respectivamente.

$$\mu_M \leftarrow \frac{1}{b} \sum_{i=1}^b x_i, \quad (3.5)$$

$$\sigma_M^2 \leftarrow \frac{1}{b} \sum_{i=1}^b (x_i - \mu_M)^2, \quad (3.6)$$

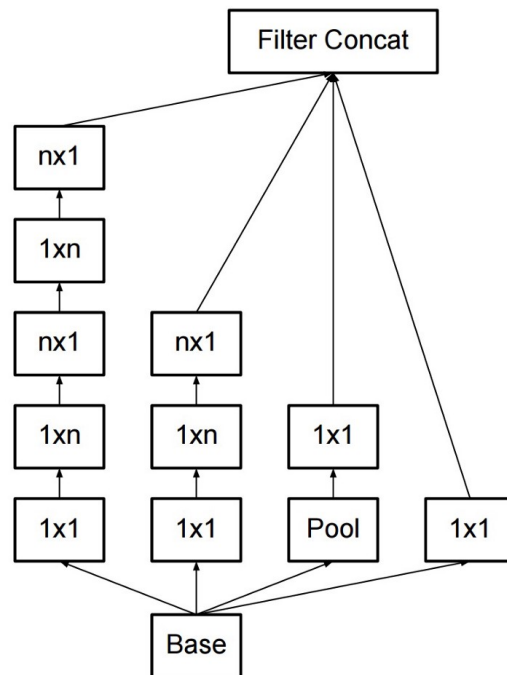
$$\hat{x}_i = \frac{x_i - \mu_M}{\sqrt{\sigma_M^2 + \lambda}}, \quad (3.7)$$

em que a constante λ tem como objetivo proporcionar estabilidade numérica.

3.5.3 Convoluções Assimétricas

Estendendo o conceito de Convoluções Fatoradas, aplicado anteriormente no artigo VGG (SIMONYAN; ZISSERMAN, 2014a), e também com base nas *Flattened CNNs* (JIN; DUNDAR; CULURCIELLO, 2014), a equipe do *Google Brain* usa Convoluções Assimétricas para propor uma nova versão dos Módulos *Inception*. Com o objetivo de melhorar a eficiência dos parâmetros do Módulo *Inception*, eles percebem que, ao invés de executar uma convolução $n \times n$, seria melhor aplicar uma convolução $n \times 1$ seguida por uma convolução $1 \times n$, assim, fatorando-a em duas convoluções assimétricas. Uma arquitetura genérica de um Módulo Inception com convoluções assimétricas é apresentada na Figura 10.

Figura 10 – Uma ilustração da arquitetura do Módulo Inception com convoluções assimétricas.



Fonte: (SZEGEDY et al., 2016).

3.6 Arquiteturas ResNet

Apesar das indicações do VGG, apontando para uma correlação positiva entre a profundidade e a precisão da rede, outras experiências, realizadas em um artigo da Rede Neural Residual (ResNet) (WU; ZHONG; LIU, 2017), não suportam essa visão, pelo menos para CNNs clássicas (redes compostas apenas por camadas de convoluções, camadas de *pooling* e camadas totalmente conectadas).

Os autores da ResNet argumentam que CNNs clássicas muito profundas são mais difíceis de otimizar, principalmente devido ao problema do gradiente de fuga (BENGIO; SIMARD; FRASCONI, 1994). Na prática, uma rede de 18 camadas apresenta erros de teste e treinamento mais altos do que uma rede de 34 camadas, mesmo que essas 16 camadas adicionais executem apenas um mapeamento de identidade.

Para superar esse problema e inspirados no conceito de "skip connections" das Redes Neurais Recorrentes, os autores da ResNet introduziram um novo bloco de construção de redes neurais profundas, o Bloco Residual.

3.6.1 Bloco Residual

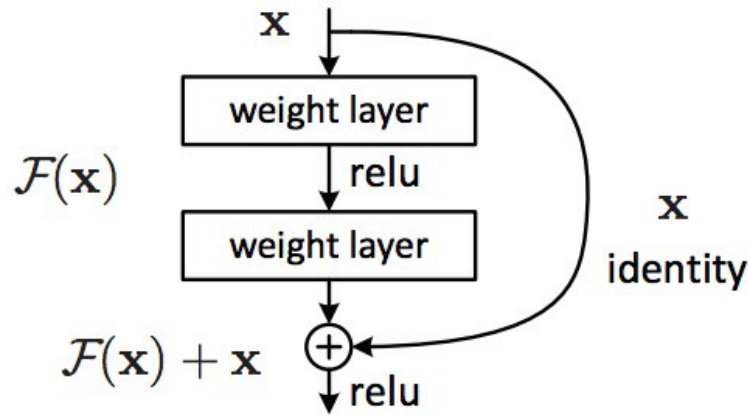
Em um bloco residual, a entrada flui para duas ramificações, uma conecta a entrada ao estágio de mesclagem e a outra executa uma pilha de camadas de convoluções (geralmente 2 ou 3) preservando as dimensões da entrada (por exemplo, se a entrada tiver C canais, então, C filtros são aplicados) e envia os resultados para o estágio de mesclagem. Finalmente, as saídas de ambos os ramos são somadas. Observe que a última camada de convolução da segunda ramificação não possui não-linearidades após as operações de convoluções. Essas funções de ativação ausentes são aplicadas à saída do estágio de mesclagem, compondo a saída do bloco residual. Uma ilustração de uma versão genérica desse bloco residual é apresentada na Figura 11

Com essa estratégia simples, a ResNet definiu resultados de última geração em tarefas de detecção, localização e segmentação de objetos no ILSVRC (RUSSAKOVSKY et al., 2015) e COCO 2015 (LIN et al., 2014), atingindo um erro de classificação entre os 5 melhores, com taxa de 3,57% (ILSVRC) e superando até o desempenho em nível humano no conjunto de dados *ImageNet*.

3.7 Arquiteturas Inception-ResNet

Em 2016, menos de um ano depois da ResNet, a equipe do *Google Brain* realizou experimentos de design de arquiteturas combinando as idéias de Conexões Residuais (WU; ZHONG; LIU, 2017) com o Módulo Inception (SZEGEDY et al., 2015). Entre esses experimentos está o modelo Inception-ResNet-V2 (LÄNGKVIST; KARLSSON; LOUTFI,

Figura 11 – Um exemplo da arquitetura de um Bloco Residual.



Fonte: (WU; ZHONG; LIU, 2017).

2014), que atingiu um erro de classificação top 1 de 16,5% e um top 5 de 3,1%, superando, assim, os resultados ResNet e Inception.

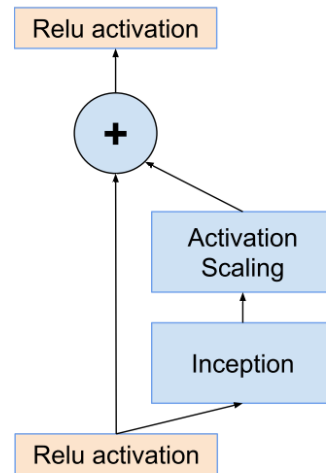
3.7.1 Módulo Inception com Conexões Residuais

A única diferença entre uma combinação direta de um Módulo *Inception* com Conexões Residuais é uma camada de escala na saída do Módulo *Inception*. Aparentemente, sem esse estágio de dimensionamento, a rede começa, após algumas milhares de iterações, a produzir apenas zeros. Os autores realizaram várias experiências para lidar com isso, mas apenas diminuindo, por um fator entre 0,1 e 0,3, a saída inicial antes de adicioná-las às entradas estabilizou o procedimento de otimização. A ilustração genérica do Módulo *Inception* com Conexões Residuais é apresentada na Figura 12.

3.8 Arquitetura Xception

Em 2016, François Chollet lançou o documento *Extreme Inception* (Xception) (CHOLLET, 2017), investigando a hipótese de que as correlações entre canais podem ser totalmente dissociadas das correlações espaciais. Para isso, o autor propõe a camada de Convolução Separável por Profundidade. Vale ressaltar que, com 36 camadas de Convolução Separáveis por Profundidade estruturadas em 14 módulos (a maioria delas com Conexões Residuais), a arquitetura Xception alcançou bons resultados no conjunto de dados JFT (Hinton; Vinyals; Dean, 2015), um banco de dados de 350 milhões de imagens com 17.000 rótulos possíveis, superando o desempenho das arquiteturas ResNet e ImageNet.

Figura 12 – Uma ilustração da arquitetura geral quando combinado Conexões Residuais com Módulo *Inception*.



Fonte: (LÄNGKVIST; KARLSSON; LOUTFI, 2014).

3.8.1 Convoluções Separáveis por Profundidade

A idéia por trás de uma camada de convolução 2D são recursos aprendidos diretamente de um espaço 3D (largura, altura e canais). Assim, uma convolução mapeia correlações entre canais e espaciais simultaneamente.

Além disso, em um Módulo *Inception*, uma convolução é fatorada em um conjunto de convoluções 1×1 , que são influenciadas especificamente por correlações entre canais, seguidas por convoluções regulares (por exemplo, convoluções 3×3 ou 5×5), que mapeiam as correlações espaciais e os canais cruzados.

No entanto, em Convolução Separável por Profundidade, primeiro é realizada uma convolução em canal, que mapeia apenas correlações espaciais; então, um conjunto de convoluções no sentido do ponto é calculado, mapeando as correlações entre canais e, em seguida, finalmente, todas as saídas de convoluções são concatenadas em profundidade. Portanto, a correlação espacial é totalmente dissociada das correlações entre canais.

Entre os benefícios da Convolução Separável por Profundidade sobre o Módulo *Inception* clássico estão uma redução no número de parâmetros livres e uma execução mais eficiente. Essas duas características são valiosas para aplicativos móveis ou mesmo embarcados.

3.9 Arquitetura MobileNet

Em 2017, os pesquisadores do Google projetaram e lançaram uma arquitetura *mobile-first* chamada MobileNet (HOWARD et al., 2017a). Essa rede de 28 camadas é

essencialmente (exceto a primeira camada) construída em convoluções separáveis por profundidade, que foram introduzidas por (HOWARD et al., 2017a) em 2016 e alcançaram, em sua configuração de linha de base, 99,1% da acurácia de detecção do InceptionV3 (SZEGEDY et al., 2015) (no banco de dados *ImageNet*) com 87,7% menos de parâmetros e 88,6 % menos de operações.

A abordagem MobileNet distingue-se do modelo Xception pela introdução de dois hiperparâmetros (multiplicador de largura, multiplicador de resolução), que permitem que o modelo corresponda, de forma eficiente, às restrições de *hardware* dos sistemas móveis e embarcados.

3.9.1 Multiplicadores de Largura

Para criar arquiteturas flexíveis e eficientes, os autores do MobileNets propõem um hiperparâmetro α , também chamado de "multiplicador de largura". No artigo MobileNets, esse parâmetro α é um fator de redução no número de convoluções em canais, em camadas de convolução separáveis por profundidade. Além disso, α afeta quase quadraticamente o número de parâmetros e o custo computacional do modelo. Observe que $\alpha \in (0, 1]$, em que 1 é a configuração da linha de base do MobileNet.

3.9.2 Multiplicadores de Resolução

O segundo hiperparâmetro proposto é o multiplicador de resolução ρ . Na prática, esse parâmetro ρ adapta a arquitetura da rede para processar diferentes resoluções de entrada (largura e altura). Por exemplo, se ρ for igual a 1, a entrada de rede será uma imagem 224×224 , se ρ for igual a 0,75, a entrada será uma imagem 168×168 e assim por diante. Observe que, como α , $\rho \in (0, 1]$ (sendo $\rho = 1$ a linha de base) e ρ também têm um efeito quadraticamente sobre o custo computacional.

Assim, o multiplicador de largura e os hiperparâmetros de resolução oferecem uma troca útil entre tamanho, precisão e latência, ao direcionar um sistema com limitações de *hardware*.

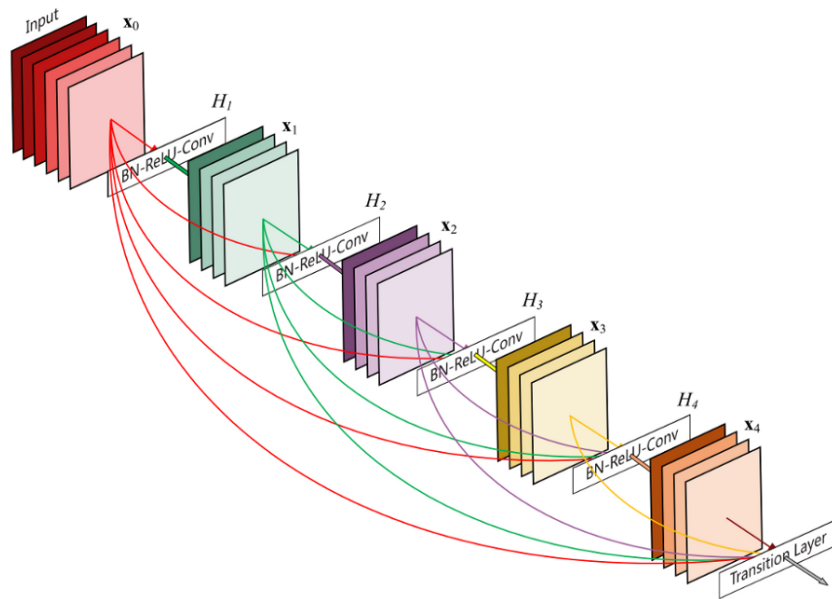
3.10 Arquiteturas DenseNet

A Rede Convolutacional Densa (DenseNet), proposta por Huang et al. (2017), estendeu o conceito de conexões mais curtas, anteriormente explorado por (SZEGEDY et al., 2015), compondo o que é chamado de Bloco Denso.

3.10.1 Dense Blocks

Existem duas diferenças principais entre as arquiteturas DenseNet e ResNet. Primeiro, em vez de executar conexões mais curtas entre as camadas adjacentes, o DenseNet conecta cada camada a todas as outras camadas com a mesma resolução de entrada (largura e altura). Esse conjunto de camadas interconectadas é denominado Bloco Denso. Uma ilustração de um bloco denso é apresentada na Figura 13. Segundo, toda camada DenseNet executa uma concatenação profunda das saídas anteriores, ao invés de somar.

Figura 13 – Ilustração da arquitetura de um Bloco Denso.



Fonte: (HUANG et al., 2017).

Assim, como resultado dessa estratégia, o DenseNet supera ligeiramente os resultados de acurácia do ResNet na tarefa de detecção do ImageNet, exigindo aproximadamente 55% menos parâmetros e 50% menos operações que o ResNet.

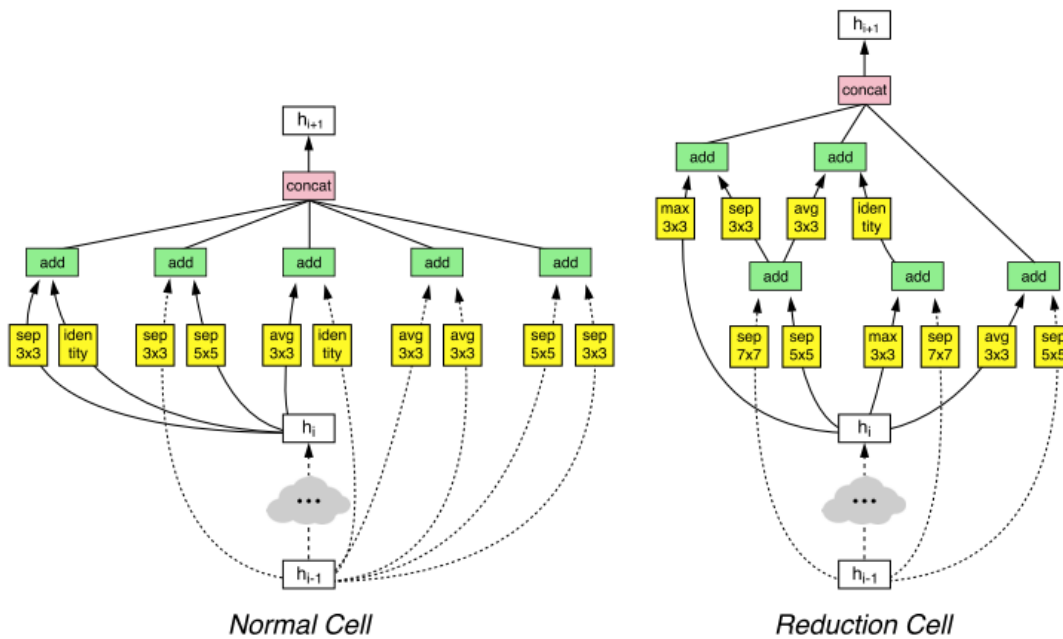
3.11 Arquiteturas NASNet

Com base no pressuposto de que projetar arquiteturas de classificação de imagem precisas e eficientes geralmente é um processo complexo de engenharia de arquitetura, o artigo *Neural Architecture Search* (NAS) (ZOPH; LE, 2016) explora a hipótese possível de aprender com eficiência uma arquitetura de modelo diretamente a partir de um conjunto de dados de interesse. Inspirada nos experimentos de NAS, a equipe do *Google Brain* propôs um novo paradigma de design de arquiteturas convolucionais, o espaço de busca da NASNet.

3.11.1 NASNet Search Space

Esse paradigma restringe o amplo espaço de pesquisa, combinando vários blocos de construção (por exemplo, identidade, convolução 1×3 e 3×1 , convolução 1×7 e 7×1 , *max pooling* 5×5 , convolução 1×1 , convolução 3×3 , convolução separáveis por profundidade 3×3 etc.), para formar uma rede inteira, em uma menor, procurando apenas dois módulos arquitetura (célula normal e célula de redução). Uma ilustração dos módulos de precisão (no ImageNet) encontrados no espaço de pesquisa da NASNet é apresentada na Figura 14. Esses módulos são alternados e repetidos várias vezes para compor a arquitetura final.

Figura 14 – Ilustração dos módulos da NASNet.



Fonte: (ZOPH; LE, 2016).

Além disso, para acelerar esse processo de pesquisa, a NASNet se baseia na suposição de que é possível transferir um módulo aprendido em um pequeno conjunto de dados, por exemplo, CIFAR-10, para um maior, como o ImageNet.

Portanto, a NASNet alcançou resultados de acurácia de detecção de última geração no conjunto de dados ImageNet, especificamente 82,7% no top 1 e 96,2% no top 5. Além disso, em sua versão *lite*, o NASNet corresponde à eficiência dos parâmetros do MobileNet, porém, é mais preciso e mais rápido. No entanto, vale ressaltar que, apesar de usar 500 GPUs Nvidia K40, a pesquisa de arquitetura mais rápida levou 4 dias.

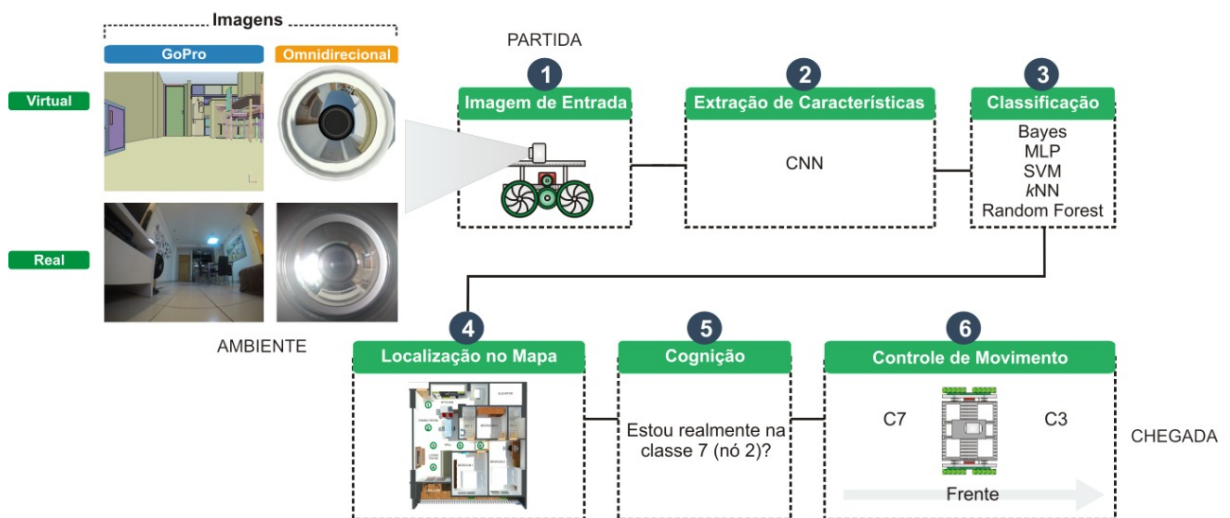
4 Metodologia

Este capítulo descreve a metodologia adotada para o desenvolvimento da abordagem proposta. No início do capítulo, são apresentadas as etapas que compõem este trabalho, as quais explicam, de modo sucinto, a captura da imagem, a extração de características e a classificação. Por último, são detalhadas as métricas de avaliação consideradas.

A Figura 15 representa o fluxo da metodologia, na qual o primeiro passo consiste na aquisição da imagem. A Seção 4.1 mostra o robô equipado com os dois tipos de câmera utilizados, bem como o mapa topológico do ambiente de navegação, e a Seção 4.2 explica o procedimento para a captura das imagens. Tendo adquirido a imagem digital, torna-se possível a aplicação dos extratores de características citados nos capítulos precedentes. A Seção 4.3 detalha como esses extratores de características foram empregados. Em seguida, o vetor de característica, o qual foi obtido na etapa de extração, é usado como entrada para os classificadores realizarem a tarefa de predição. A Seção 4.4 especifica a etapa de predição. Desse modo, o robô finaliza sua atividade de localização, o que permite, portanto, que conclua a sua tarefa de navegação.

O sistema foi analisado com imagens de alta resolução provenientes de uma câmera convencional e, em seguida, com imagens a partir de uma câmera omnidirecional. Os dois campos de visão obtidos foram modelados em simuladores, com o objetivo de efetuar um estudo prévio com as imagens sintéticas resultantes. Posteriormente, foram consideradas as imagens reais para a avaliação da tarefa de localização e navegação do robô.

Figura 15 – Infograma da metodologia empregada na abordagem proposta.



Fonte: Autor.

4.1 Mapa Topológico e Robô

Para este trabalho, foi utilizado um robô móvel constituído de uma base de metal, provido com dois motores, baterias, placas de circuito lógico e um sistema de engrenagens. A comunicação com o robô é realizada através de telemetria, possuindo um software próprio para a troca de dados com um microcontrolador.

A Figura 16(a) apresenta o robô equipado com uma câmera de alta resolução (GoPro[®]), enquanto a Figura 16(b) mostra o robô com uma câmera omnidirecional, a qual possui uma lente para captura de imagens em 360 graus. A Tabela 3 apresenta mais especificações do veículo e das câmeras utilizadas. A navegação do robô é executada em um ambiente do tipo interno, o qual consiste em um apartamento, como mostra a Figura 17(c). A escolha deste ambiente deu-se por apresentar aspectos que favorecem o seu reconhecimento, contribuindo para uma melhor navegação do robô.

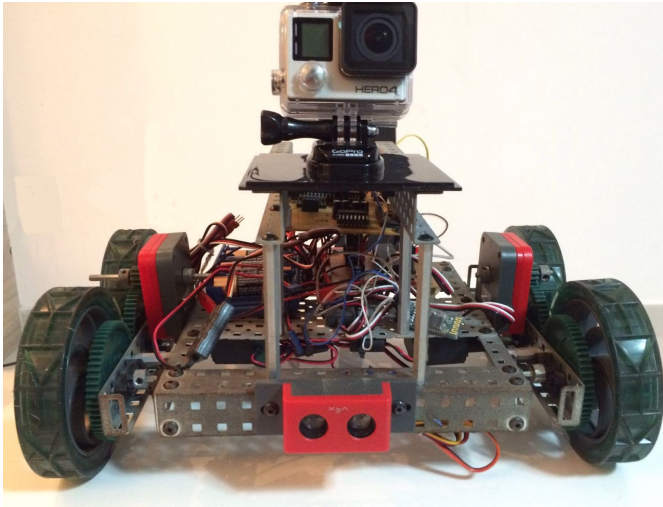
Tabela 3 – Especificações do robô móvel considerado.

Características do Robô Móvel	
Modelo	VEX EDR Kit
Motor	Servo EDR 393
Navegação	Interno
Fonte de Energia	6 baterias NiMH 1.2V e 2600mAh
Características da Câmera GoPro [®]	
Modelo	GoPro Hero3+
Conexão sem Fio	Wireless LAN
Resolução da Foto	12 MP
Características da Câmera Omnidirecional	
Modelo das Lentes	The Dot of Kogeto
Camera	iPhone 6
Resolução da Foto	8 MP

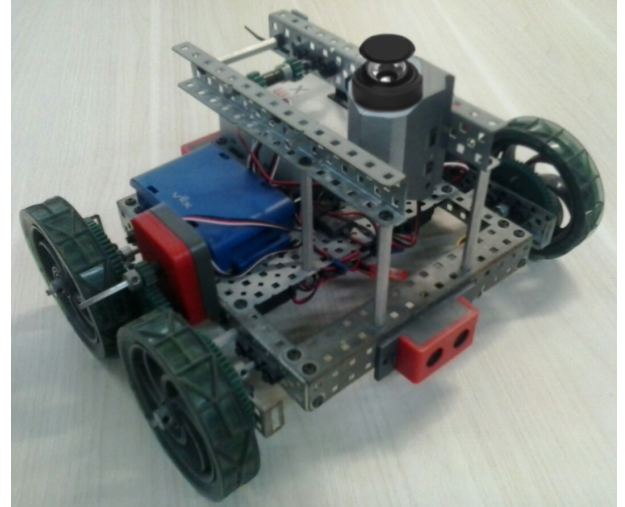
Fonte: Autor.

A Figura 17 contextualiza o mapa topológico do apartamento, mostrando os nós do local, os quais caracterizam pontos no apartamento, e estão disposto como apresenta a Figura 17(a), sendo numerados de 1 a 6. As imagens foram capturadas em posições estratégicas, situadas a partir desses nós. Essas posições referem-se à orientação dentro de cada nó e são rotuladas de c1 a c15, representando as classes para a localização do robô. Como exemplo, o robô pode encontrar-se na sala de estar do apartamento (nó 4) e posicionado de frente para a porta da varanda (classe 4) ou em direção à sala de jantar (classe 5). O sistema de visão computacional proposto nesse trabalho tem a capacidade de identificar essas diferentes posições. As arestas do mapa topológico são simbolizadas pelos caminhos praticáveis pelo robô em sua navegação, como demonstrado na Figura 17(b).

Figura 16 – Robô móvel usado na abordagem proposta.



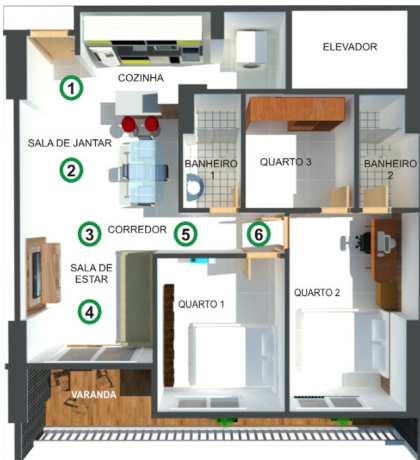
(a) Robô equipado com câmera GoPro®.



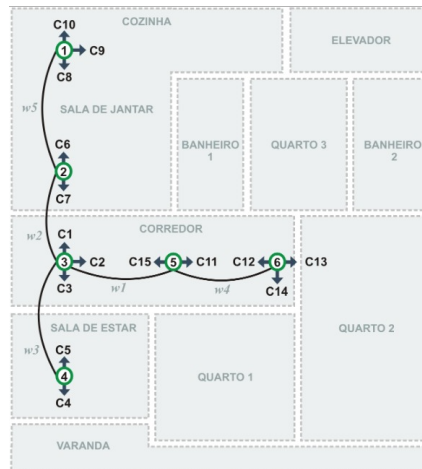
(b) Robô equipado com câmera omnidirecional.

Fonte: Autor.

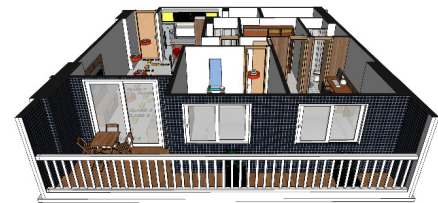
Figura 17 – Mapa topológico do ambiente interno (apartamento).



(a) Vista superior do apartamento.



(b) Mapa topológico com nós e arestas.



(c) Vista em perspectiva.

Fonte: Autor.

4.2 Captura de Imagens

O fornecimento das imagens sintéticas deu-se por meio de simuladores que possibilitam a reconstrução do ambiente com campo de visão convencional e omnidirecional. Para a modelagem do ambiente com imagens da câmera de alta resolução foi utilizado o simulador V-Rep (Rohmer; Singh; Freese, 2013). Foram adquiridas 40 imagens de cada classe no ambiente sintético. Também foram capturadas 40 imagens para cada classe no ambiente real. A aquisição de uma quantidade de imagens considerável para cada classe

foi feita para que fosse estabelecido um funcionamento correto do sistema de navegação, abrangendo, dessa forma, possíveis circunstâncias nas quais o robô esteja situado nas proximidades do nó, porém, não exatamente nele. Para a modelagem do ambiente com imagens da câmera omnidirecional foi empregado, em conjunto, os simuladores *Sketchup* (GASPAR, 2013) e *V-Ray* (LI-ZHEN, 2010). Foram capturadas 50 imagens por classe no ambiente sintético e a mesma quantidade de imagens também foi adquirida no ambiente real, com a mesma câmera.

Desse modo, foram avaliados quatro bancos de imagens, como detalha a Tabela 4, totalizando 750 imagens. A Figura 18 mostra exemplos de imagens do ambiente real e sintético para ambas as câmeras consideradas. Essas imagens são referentes à classe 5, a qual está situada no nó 4.

Tabela 4 – Especificações dos bancos de imagem.

Ambiente Real		
Tipo de Câmera	N. de imagens por classe	Resolução das Imagens
GoPro	40	4000 × 3000
Omnidirecional	50	4160 × 3120
Ambiente Virtual		
Tipo de Câmera	N. de imagens por classe	Resolução das Imagens
GoPro	40	640 × 480
Omnidirecional	50	800 × 600

Fonte: Autor.

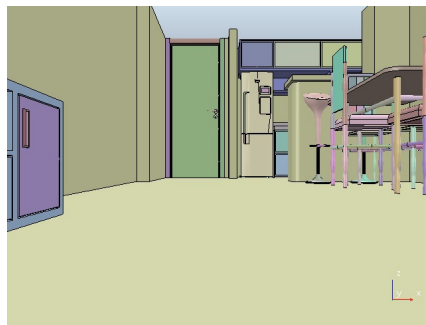
Figura 18 – Amostras de imagens da classe 5 (nó 4) para (a) ambiente real com câmera GoPro[®], (b) ambiente real com câmera omnidirecional, (c) ambiente virtual com câmera GoPro[®] e (d) ambiente virtual com câmera omnidirecional.



(a) GoPro[®] (real).



(b) Omnidirecional (real).



(c) GoPro[®] (virtual).



(d) Omnidirecional (virtual).

Fonte: Autor.

4.3 Extração de Características

Neste trabalho, a CNN utilizou o conceito de *Transfer Learning*, que está relacionado ao poder descritivo de uma CNN pré-treinada. Portanto, a camada totalmente conectada (mais especificamente, a última camada convolucional) é removida e o redimensionamento da entrada da camada removida em um vetor unidimensional é realizado (PAN; YANG, 2010; TAN et al., 2018). Após esse processo, o modelo pré-treinado não se comporta mais como um classificador e, portanto, é usado como um extrator de características. O *Transfer Learning* aplicado à CNN tem sido extensivamente empregado em diversos trabalhos (GRUBINGER; CHASPARIS; NATSCHLÄGER, 2017; MAZO et al., 2018; SUH et al., 2018; LIN et al., 2018; WANG et al., 2019).

As arquiteturas CNN consideradas foram VGG16, VGG19, MobileNet, InceptionV3, Xception, ResNet50 e InceptionResNetV2, NASNetMobile, DenseNet121, DenseNet169 e DenseNet201, as quais são pré-treinadas com o banco de dados *ImageNet* e usadas como extratores de características no sistema de visão computacional para navegação de robô móvel, compondo a abordagem desenvolvida neste trabalho. A Tabela 5 mostra o número de atributos retornado por cada arquitetura de CNN.

Tabela 5 – Número de atributos retornados por cada extrator de características.

Arquitetura CNN	Número de Atributos
VGG16	512
VGG19	512
MobileNet	1024
InceptionV3	2048
Xception	2048
ResNet50	2048
InceptionResNetV2	1536
NASNetMobile	1056
DenseNet121	1024
DenseNet169	1664
DenseNet201	1920

Fonte: Autor.

4.3.1 Rede Neural Convolutacional como Extrator de Características

Essa tópico detalha o processo de construção e inicialização das arquiteturas, seu treinamento (com os bancos de dados de referência) e suas transformações em extratores de características.

4.3.1.1 Construção e Inicialização de Arquiteturas

Nos últimos anos, especialmente após 2012, o desempenho na tarefa de detecção de objetos aumentou significativamente com o desenvolvimento de várias arquiteturas da CNN.

No entanto, como é inviável avaliar a hipótese explorada neste trabalho em todas as arquiteturas elaboradas pela comunidade científica, 11 arquiteturas respeitáveis (muitas delas vencedores de desafios complexos, como ILSVRC (RUSSAKOVSKY et al., 2015) e COCO (LIN et al., 2014)) são selecionados para representar a classe de método CNN.

Mais especificamente, o gráfico computacional que representa cada uma das arquiteturas VGG16, VGG19, MobileNet, Xception, InceptionV3, ResNet50, Inception-ResNet-V2, DenseNet169, DenseNet201, NASNetMobile e NASNetLarge é implementado e inicializado conforme especificado em seus respectivos artigos.

4.3.1.2 Treinamento das Arquiteturas

A segunda etapa desta seção consiste em treinar as 11 arquiteturas contempladas neste trabalho no banco de dados de referência, ou seja, o banco de dados do qual o "conhecimento" é transferido.

Com base na premissa de que as características/padrões aprendidos por uma CNN para discriminar um conjunto de classes são mais genéricos do que aqueles aprendidos a discriminar apenas um subconjunto dessas classes, o banco de dados usado nesta etapa

devem ter o maior número possível de classes. Além disso, como todas as onze arquiteturas foram elaboradas com o objetivo de solucionar o problema representado pelo banco de dados *ImageNet*, que consiste em 1,2 milhão de imagens não médicas e divididas em 1000 categorias, as metodologias de treinamento usadas por cada uma dessas arquiteturas são documentadas detalhadamente em seus respectivos artigos.

4.3.1.3 Convertendo CNNs em Extratores de Características

Na terceira e última etapa desta seção, as CNNs treinadas no conjunto de dados de referência são convertidas em extratores de recursos. Entretanto, para isso, é necessário entender as quatro transformações realizadas por essas redes neurais.

Primeiro, a imagem de entrada da CNN, ou seja, a imagem a ser processada, é submetida inicialmente a uma sequência de transformações não lineares, definidas de acordo com a arquitetura utilizada. Mais especificamente, esse processo converte a imagem de entrada em um conjunto de pequenas matrizes (em nenhuma das arquiteturas usadas neste trabalho, a resolução dessas matrizes é maior do que $11 \times 11 \times 1$). Posteriormente, cada uma dessas matrizes é redimensionada para um vetor unidimensional. Então, o conjunto desses vetores é concatenado, formando assim um único vetor. Este vetor, por sua vez, pode ser interpretado como os atributos extraídos pela CNN. O processo descrito é ilustrado na Figura 19, onde NF_{prim_cam} simboliza a quantidade de filtros aplicados nas primeiras camadas, $NF_{últ_cam}$ representa a quantidade de filtros aplicados na última camada e T_{img} é o tamanho da imagem original após as transformações de todas as camadas da CNN.

Por fim, esses vetores de atributos são submetidos a uma ou mais camadas totalmente conectadas. Em geral, esse processo se assemelha ao processamento realizado pelos classificadores Perceptron ou MLP. Deve-se notar que as funções de ativação usadas nessas camadas são ReLU ou Sigmoid, para as primeiras camadas, e Softmax, para a última (se a rede tiver apenas uma camada totalmente conectada, a função de ativação usada é a Softmax).

No entanto, conforme ilustrado na Figura 20, as últimas camadas totalmente conectadas (usadas para treinar a rede no banco de dados de referência) são removidas do modelo.

Com isso, quando uma imagem é submetida a essa arquitetura modificada, o resultado não pode mais ser interpretado como um conjunto de probabilidades de uma imagem de entrada pertencente a uma determinada classe do conjunto de dados de referência (*ImageNet*), mas como um vetor de informação, que seria usado por camadas totalmente conectadas para compor as probabilidades de classe.

Figura 19 – Ilustração das duas primeiras etapas de uma CNN. No primeiro, a imagem de entrada é submetida a uma sequência de camadas convolucionais e sub-amostragem (*pooling*). No segundo, os resultados da primeira etapa são redimensionados e concatenados em uma única lista, chamada de *deep features*.

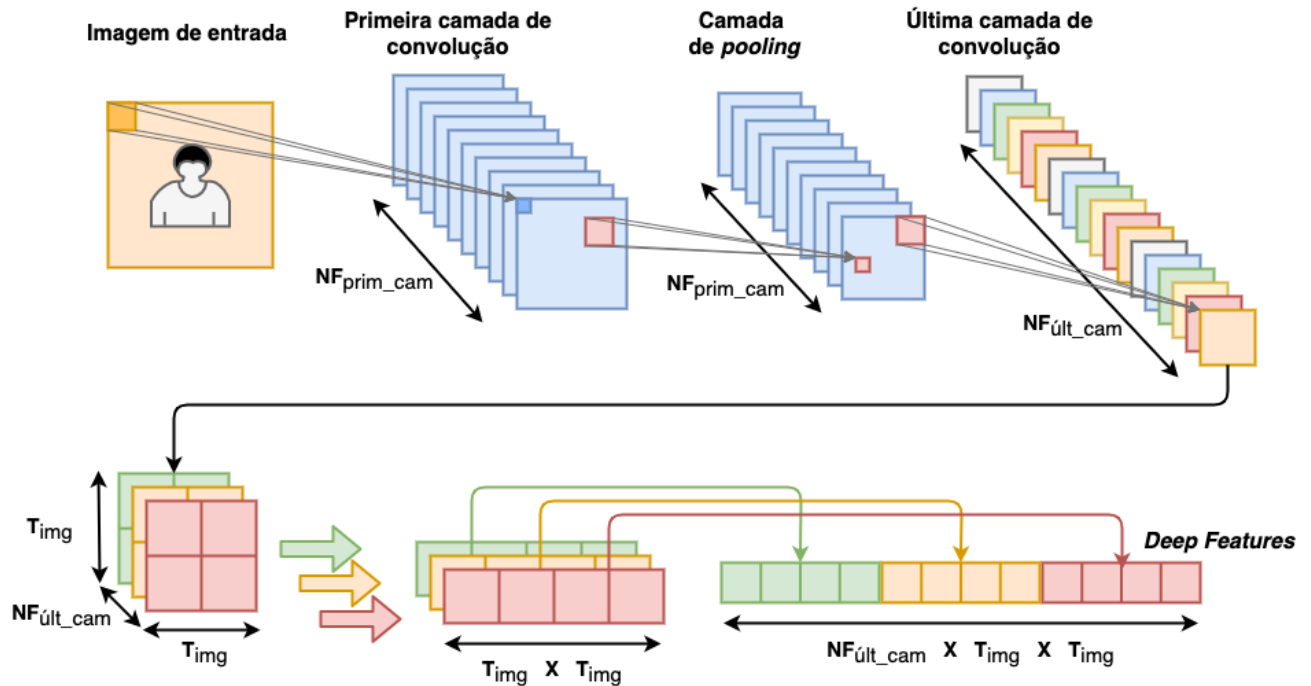
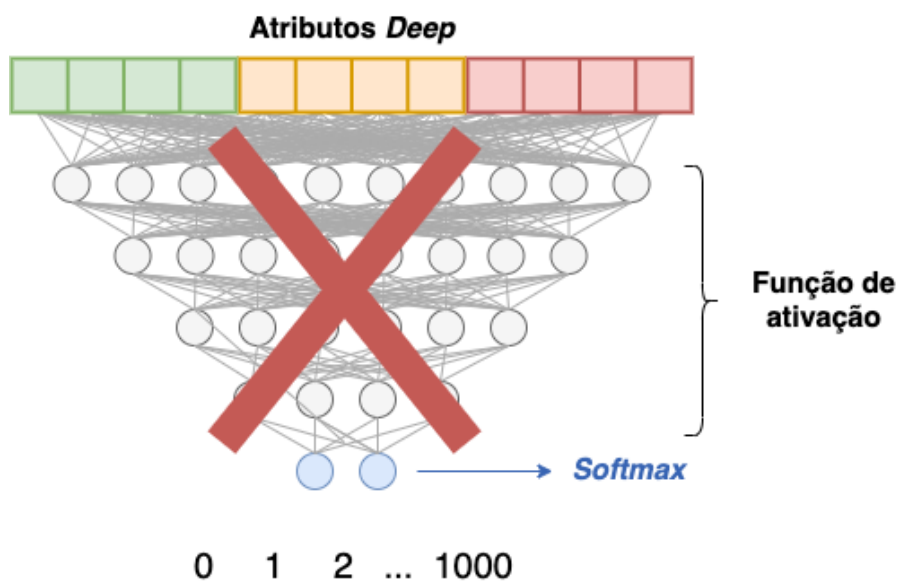


Figura 20 – Ilustração do processo usado para transformar uma CNN em um extrator de características.



Fonte: Autor.

4.4 Métodos de Classificação

Os classificadores utilizados nesse trabalho apresentam princípios diversos, permitindo a abrangência de diferentes aspectos. Naive Bayes é baseado em probabilidade e estatística, MLP consiste de uma rede neural artificial, kNN é uma técnica baseada em distância, *Random Forest* apresentada características de um classificador de *bagging* e SVM é fundamentada pela Teoria do Aprendizado.

No processo de classificação, a técnica Naive Bayes operou com a função de densidade de probabilidade gaussiana. O hiperparâmetro k do classificador kNN foi escolhido por meio de uma pesquisa em grade (*grid search*), testando os valores ímpares de 1 a 11. O MLP realizou seu treinamento usando o método Levenberg-Marquardt e neurônios variando de 2 a 1000 na camada oculta. O SVM usou kernels Linear, Polonomial e RBF, e hiperparâmetros C e γ com os valores 2^{-5} , 2^{-4} , 2^{-3} , ..., 2^{15} and 2^{-15} , 2^{-14} , ..., 2^3 , respectivamente. Os hiperparâmetros do MLP e da SVM foram determinados através de validação cruzada (*cross validation*) com 10-folds. A busca aleatória a partir do *Random Forest* foi usada para encontrar: o número de atributos para a melhor divisão (1,2,3, ... ou 10); a profundidade máxima da árvore a ser utilizada (6 ou nenhuma), o número mínimo de amostras necessárias para dividir um nó interno (1,2,3, ... ou 10), o número mínimo de amostras por folha a serem empregadas (1,2,3, ... ou 10), se amostras de autoinicialização devem ser usadas na construção de árvores ou não, e qual função deve ser usada para medir a qualidade de uma divisão (gini ou entropia). Além disso, o processo de treinamento do *Random Forest* utilizou 3000 estimadores.

4.5 Análise de Desempenho

A seguir, são detalhados os parâmetros utilizados para avaliação do resultados atingidos, verificando o desempenho da abordagem proposta neste trabalho.

4.5.1 Métricas de Avaliação

Para avaliar a performance dos classificadores na abordagem proposta, quatro métricas de avaliação foram consideradas, as quais são: Acurácia, F1-Score, *Recall* e *Precision* (SOKOLOVA; LAPALME, 2009). O cálculo dessas medidas é feito segundo a matriz de confusão. É importante destacar que a classe corresponde à posição e à direção do robô no ambiente, segundo o mapa topológico.

Para calcular cada métrica de avaliação, é necessário conhecer as seguintes definições:

- **Verdadeiro Positivo (VP):** equivale à quantidade de predições as quais afirmam que o robô foi corretamente localizado pelo método de classificação;

- **Verdadeiro Negativo (VN):** equivale à quantidade de predições as quais afirmam que outra localização foi corretamente prevista pelo método de classificação;
- **Falso Positivo (FP):** equivale à quantidade de predições as quais afirmam que outra localização foi incorretamente prevista como localização em avaliação pelo método de classificação;
- **Falso Negativo (FN):** equivale à quantidade de predições as quais afirmam que a localização em análise foi incorretamente prevista como outra localização pelo modelo de classificação.

A Tabela 6 apresenta a matriz de confusão que retrata as localizações do robô preditas correta ou incorretamente por um método de classificação. A classe 2 encontra-se destacada, em azul, na Tabela 6, visando um melhor entendimento das métricas consideradas. Com base na quantidade de classes, observa-se que a matriz é de ordem 15.

Tabela 6 – Matriz de confusão para um problema de classificação multiclasse.

Classe Real \ Classe Prevista	1	2	3	...	15
1	VP	FN	FN	FN	FN
2	FP	VN	FN	FN	FN
3	FP	FN	VP	FN	FN
⋮	FP	FN	FN	VN	FN
15	FP	FN	FN	FN	VN

Fonte: Autor.

A Acurácia representa a quantidade de acertos dividido pela quantidade de predições, não ponderando as desproporções entre a localização analisada e as demais. Essa medida é calculada segundo a Equação 4.1.

$$Acuracia = \frac{n^{\circ} \text{ total de acerto}}{n^{\circ} \text{ total de amostras no conjunto de dados}} \quad (4.1)$$

O *Recall* é a fração de verdadeiros positivos, ou seja, essa métrica representa a eficiência do sistema em indicar corretamente a localização para situações em que o robô realmente está no local previsto. Essa métrica é calculada a partir da Equação 4.2.

$$Recall = \frac{VP}{VP + FP} \quad (4.2)$$

O *Precision* consiste na parcela de verdadeiros positivos relativa a todas as predições positivas. Essa medida é calculada conforme a Equação 4.3.

$$Precision = \frac{VP}{VP + FN} \quad (4.3)$$

O F1-Score caracteriza a relação entre as métricas *Recall* e *Precision*, de forma mais específica, corresponde à média harmônica entre *Recall* e *Precision*. Essa métrica é calculada de acordo com a Equação 4.4.

$$F1Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.4)$$

4.5.2 Validação Cruzada

A validação cruzada (*cross validation*) é uma técnica, baseada em estatística, bastante empregada para análise e comparação de métodos de classificação. Ela consiste na divisão dos dados em dois grupos diferentes, um grupo para a fase de treinamento e outro grupo para a fase de teste dos algoritmos (STONE, 1974).

Há dois tipos mais comuns de validação cruzada: *hold out* e *k-fold*. O *textithold out* fraciona o conjunto de amostras em dois subconjuntos, em que um dos subconjuntos é dedicado para a etapa de treinamento e o outro para a etapa de validação. Neste trabalho, cada base de dados formada pelos atributos resultantes após a etapa de extração de características, teve seus padrões dividido em dois conjuntos, onde oitenta por cento das amostras foram destinadas para a fase de treinamento e o remanescente foi submetido para a fase de teste. Foram realizadas 10 iterações desse processo.

Quanto ao *k-fold*, ele efetua a partição das amostras em *k* subgrupos. São executadas *k* iterações para o treinamento e, em cada uma, um subconjunto é dedicado para a etapa de teste, enquanto os subconjuntos restantes formam o conjunto destinado para o treino, ou seja, são aplicados *k-folds* para a fase de treinamento e os outros *folds* são destinados para a fase de teste do algoritmo. Em nossa abordagem, para cada repetição do *hold out*, foi considerada a validação cruzada com 10 *folds* e, ainda, a busca em grade (*grid search*) (HAYKIN, 2008).

5 Resultados

Os resultados desta dissertação são apresentados em duas etapas. Na primeira etapa, todas as combinações de arquiteturas CNN com os classificadores considerados tiveram suas métricas de avaliação comparadas. A análise é realizada a partir dos quatro conjuntos de imagens provenientes do ambiente real e sintético através dos dois tipos de câmera: GoPro[®] e omnidirecional. Na segunda etapa, as melhores combinações têm seus tempos de processamento analisados.

Os experimentos foram realizados em um iMac com processador Core i5 2.5 GHz, 4GB RAM e sistema operacional Mac OS X Mojave 10.14.6. Tanto os extratores quanto os classificadores foram implementados em Python.

5.1 Análise de Desempenho

A seguir, são apresentados os valores de Acurácia, F1-Score, *Recall* e *Precision* retornados pelo processamento dos bancos de imagens em ambiente real e simulado, a partir da captura realizada com a câmera GoPro[®] e com a câmera omnidirecional.

5.1.1 Resultados no Ambiente Real

Primeiramente, foram processados os bancos compostos por imagens capturadas no ambiente real. As arquiteturas de CNN foram aplicadas nas imagens dando origem aos vetores de entrada dos classificadores.

As Tabelas 7, 8, 9 e 10 apresentam os valores médios e desvios padrões de Acurácia, *Recall*, *Precision* e F1-Score, respectivamente, retornados pelas combinações das arquiteturas de CNN com os classificadores, considerando as imagens do ambiente real, tanto com o campo de visão convencional quanto com o campo de visão omnidirecional. Os melhores valores estão destacados em verde.

Observando a Tabela 7, pode-se afirmar que os melhores valores de Acurácia foram atingidos pelas imagens da câmera GoPro[®], com as quais todas as arquiteturas de CNN conseguiram alcançar uma Acurácia de 100%, em combinação com, pelo menos, três classificadores. A única exceção é a arquitetura NASNetMobile, que não atingiu 100% de Acurácia em nenhuma combinação. Porém, o maior valor de Acurácia retornado por essa arquitetura foi de 98,833%, número considerado bastante satisfatório. Ainda observando a Tabela 7, quanto às imagens omnidirecionais, a combinação que retornou o maior valor de Acurácia foi VGG16 com *Random Forest*, com o valor de 98,667%.

Tabela 7 – **Acurácia** alcançada, no **ambiente real**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,667±0,667	100,000±0,000
VGG19	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,333±0,333	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,833±0,333	100,000±0,000	99,833±0,333	99,833±0,333
InceptionV3	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	96,833±1,333	100,000±0,000
Xception	99,667±0,408	100,000±0,000	99,833±0,333	99,833±0,333	100,000±0,000	98,333±1,394	100,000±0,000
ResNet50	98,833±1,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,500±0,667	100,000±0,000
InceptionResNetV2	100,000±0,000	100,000±0,000	99,500±0,408	99,833±0,333	100,000±0,000	70,667±4,028	100,000±0,000
NASNetMobile	99,667±0,667	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333
DenseNet121	99,833±0,333	99,833±0,333	99,833±0,333	100,000±0,000	100,000±0,000	95,000±1,394	100,000±0,000
DenseNet169	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,667±0,816	100,000±0,000
DenseNet201	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,167±1,871	100,000±0,000

Omni Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	95,867±1,293	98,133±1,485	98,400±0,800	98,667±0,843	98,000±1,193	91,200±1,424	98,533±0,980
VGG19	96,667±0,943	97,467±1,222	98,267±0,998	98,000±1,265	98,533±0,777	93,467±2,437	98,533±0,777
MobileNet	95,467±0,653	98,133±0,777	98,533±0,980	97,733±0,800	98,667±0,843	95,333±0,422	98,800±0,653
InceptionV3	92,800±1,950	98,000±0,943	98,133±0,884	97,600±1,236	97,600±1,236	90,933±1,373	97,733±1,373
Xception	95,067±0,680	97,467±1,147	98,400±0,998	96,933±0,800	97,333±0,943	96,400±0,533	98,133±0,777
ResNet50	89,867±2,286	97,067±1,373	98,000±0,422	97,600±0,680	98,267±0,680	83,333±1,978	98,267±0,327
InceptionResNetV2	92,400±1,497	96,800±0,499	96,933±0,680	97,067±0,680	97,467±0,499	91,867±1,904	97,733±0,680
NASNetMobile	92,933±0,904	96,533±0,980	96,133±1,424	97,067±1,373	97,467±1,147	94,667±1,333	97,467±1,147
DenseNet121	96,800±1,147	98,267±0,533	98,533±0,777	98,133±0,653	98,267±0,533	85,467±1,424	98,267±0,327
DenseNet169	94,933±1,960	98,133±0,499	98,533±1,147	97,733±0,904	98,000±0,596	91,467±1,360	98,267±0,904
DenseNet201	95,067±1,497	98,000±0,730	98,267±0,533	97,600±0,680	98,267±0,533	92,800±2,325	98,400±0,533

Fonte: Autor

Analisando a Tabela 8, desprende-se também que os maiores valores de *Recall* foram alcançados pelas imagens da câmera GoPro[®], com as quais todas as arquiteturas de CNN conseguiram alcançar um *Recall* de 100%, em combinação com, pelo menos, três classificadores. A única exceção também é a arquitetura NASNetMobile, que não atingiu 100% de *Recall* em nenhuma combinação. Contudo, o maior valor de *Recall* atingido por essa arquitetura foi de 98,833%, resultado considerado bastante satisfatório. Ainda analisando a Tabela 8, quanto às imagens omnidirecionais, o maior valor de *Recall* foi retornado pela associação da arquitetura MobileNet com SVM(RBF), a qual atingiu 98,800%.

De acordo com a Tabela 9, ainda é possível afirmar que os melhores valores de *Precision* foram alcançados pelas imagens da câmera GoPro[®], com as quais todas as arquiteturas de CNN alcançaram um *Precision* de 100%, em combinação com, pelo menos, três classificadores. A exceção ainda é a arquitetura NASNetMobile, que não atingiu 100% de *Precision* em nenhuma associação com classificador. Todavia, o maior valor de *Precision* atingido por essa arquitetura foi de 99,852%, que é considerado bastante convincente. Quanto às imagens omnidirecionais, ainda de acordo com a Tabela 9, a combinação MobileNet com SVM(RBF) alcançou o maior valor de *Precision*, com a marca de 98,909%.

Conforme a Tabela 10, as imagens resultantes da captura com a câmera GoPro[®] também proporcionaram os maiores valores de F1-Score, com as quais todas as arquiteturas de CNN retornaram F1-Score de 100%, em combinação com, pelo menos, três classificadores. A exceção também é a arquitetura NASNetMobile, que não atingiu 100% de F1-Score

Tabela 8 – **Recall** alcançado, no **ambiente real**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,667±0,667	100,000±0,000
VGG19	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,333±0,333	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,833±0,333	100,000±0,000	99,833±0,333	99,833±0,333
InceptionV3	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	96,833±1,333	100,000±0,000
Xception	99,667±0,408	100,000±0,000	99,833±0,333	99,833±0,333	100,000±0,000	98,333±1,394	100,000±0,000
ResNet50	98,833±1,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,500±0,667	100,000±0,000
InceptionResNetV2	100,000±0,000	100,000±0,000	99,500±0,408	99,833±0,333	100,000±0,000	70,667±4,028	100,000±0,000
NASNetMobile	99,667±0,667	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333	99,833±0,333
DenseNet121	99,833±0,333	99,833±0,333	99,833±0,333	100,000±0,000	100,000±0,000	95,000±1,394	100,000±0,000
DenseNet169	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,667±0,816	100,000±0,000
DenseNet201	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,167±1,871	100,000±0,000

Omni Real							
Classificador	Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	95,867±1,293	98,133±1,485	98,400±0,800	98,667±0,843	98,000±1,193	91,200±1,424	98,533±0,980
VGG19	96,667±0,943	97,467±1,222	98,267±0,998	98,000±1,265	98,533±0,777	93,467±2,437	98,533±0,777
MobileNet	95,467±0,653	98,133±0,777	98,533±0,980	97,733±0,800	98,667±0,843	95,333±0,422	98,800±0,653
InceptionV3	92,800±1,950	98,000±0,943	98,133±0,884	97,600±1,236	97,600±1,236	90,933±1,373	97,733±1,373
Xception	95,067±0,680	97,467±1,147	98,400±0,998	96,933±0,800	97,333±0,943	96,400±0,533	98,133±0,777
ResNet50	89,867±2,286	97,067±1,373	98,000±0,422	97,600±0,680	98,267±0,680	83,333±1,978	98,267±0,327
InceptionResNetV2	92,400±1,497	96,800±0,499	96,933±0,680	97,067±0,680	97,467±0,499	91,867±1,904	97,733±0,680
NASNetMobile	92,933±0,904	96,533±0,980	96,133±1,424	97,067±1,373	97,467±1,147	94,667±1,333	97,467±1,147
DenseNet121	96,800±1,147	98,267±0,533	98,533±0,777	98,133±0,653	98,267±0,533	85,467±1,424	98,267±0,327
DenseNet169	94,933±1,960	98,133±0,499	98,533±1,147	97,733±0,904	98,000±0,596	91,467±1,360	98,267±0,904
DenseNet201	95,067±1,497	98,000±0,730	98,267±0,533	97,600±0,680	98,267±0,533	92,800±2,325	98,400±0,533

Fonte: Autor

Tabela 9 – **Precision** alcançado, no **ambiente real**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,704±0,593	100,000±0,000
VGG19	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,407±0,296	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,852±0,296	100,000±0,000	99,852±0,296	99,852±0,296
InceptionV3	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,656±0,852	100,000±0,000
Xception	99,704±0,363	100,000±0,000	99,852±0,296	99,852±0,296	100,000±0,000	98,530±1,246	100,000±0,000
ResNet50	99,073±0,773	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,585±0,542	100,000±0,000
InceptionResNetV2	100,000±0,000	100,000±0,000	99,556±0,363	99,852±0,296	100,000±0,000	85,726±3,329	100,000±0,000
NASNetMobile	99,685±0,630	99,852±0,296	99,852±0,296	99,852±0,296	99,852±0,296	99,852±0,296	99,852±0,296
DenseNet121	99,852±0,296	99,852±0,296	99,852±0,296	100,000±0,000	100,000±0,000	97,197±0,450	100,000±0,000
DenseNet169	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	98,128±0,614	100,000±0,000
DenseNet201	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,821±1,264	100,000±0,000

Omni Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	96,296±1,237	98,327±1,331	98,566±0,718	98,808±0,752	98,244±1,048	93,058±1,035	98,663±0,896
VGG19	97,081±0,808	97,651±1,123	98,366±0,975	98,131±1,184	98,630±0,732	94,370±2,016	98,648±0,730
MobileNet	95,996±0,679	98,272±0,748	98,698±0,900	97,998±0,681	98,806±0,738	95,762±0,410	98,909±0,594
InceptionV3	93,705±1,821	98,198±0,864	98,304±0,765	97,866±1,009	97,950±0,947	92,585±1,219	98,072±1,077
Xception	95,637±0,721	97,679±1,040	98,509±0,931	97,230±0,596	97,500±0,930	96,743±0,467	98,230±0,784
ResNet50	91,446±1,791	97,354±1,130	98,114±0,460	97,815±0,579	98,432±0,629	87,494±1,585	98,378±0,341
InceptionResNetV2	93,295±1,221	97,112±0,363	97,183±0,534	97,391±0,481	97,761±0,407	93,975±0,893	97,978±0,628
NASNetMobile	93,696±0,865	96,735±0,968	96,365±1,399	97,294±1,256	97,619±1,121	95,265±1,264	97,619±1,121
DenseNet121	97,131±0,974	98,440±0,517	98,723±0,624	98,367±0,572	98,456±0,486	88,884±1,320	98,512±0,237
DenseNet169	95,529±1,704	98,331±0,477	98,695±1,019	98,072±0,682	98,210±0,581	92,771±1,302	98,420±0,875
DenseNet201	95,810±1,183	98,234±0,669	98,456±0,372	97,870±0,550	98,432±0,484	94,031±1,995	98,554±0,506

Fonte: Autor

em nenhuma combinação. Entretanto, o maior valor de F1-Score conseguido por essa arquitetura foi de 99,833%, número considerado bastante satisfatório. Com relação às imagens omnidirecionais, ainda segundo a Tabela 10, a associação da arquitetura MobileNet com SVM(RBF) também retornou o melhor valor da métrica, com 98,788% de F1-Score.

Tabela 10 – **F1-Score** alcançado, no **ambiente real**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,665±0,669	100,000±0,000
VGG19	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,331±0,335	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,833±0,335	100,000±0,000	99,833±0,335	99,833±0,335
InceptionV3	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	96,850±1,310	100,000±0,000
Xception	99,665±0,410	100,000±0,000	99,833±0,335	99,833±0,335	100,000±0,000	98,321±1,397	100,000±0,000
ResNet50	98,809±1,028	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	99,494±0,676	100,000±0,000
InceptionResNetV2	100,000±0,000	100,000±0,000	99,498±0,410	99,833±0,335	100,000±0,000	71,585±4,128	100,000±0,000
NASNetMobile	99,666±0,668	99,833±0,335	99,833±0,335	99,833±0,335	99,833±0,335	99,833±0,335	99,833±0,335
DenseNet121	99,833±0,335	99,833±0,335	99,833±0,335	100,000±0,000	100,000±0,000	94,781±1,613	100,000±0,000
DenseNet169	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,700±0,799	100,000±0,000
DenseNet201	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	97,134±1,932	100,000±0,000
Omni Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	95,820±1,313	98,128±1,487	98,393±0,805	98,660±0,849	97,981±1,210	91,106±1,506	98,527±0,985
VGG19	96,679±0,939	97,455±1,232	98,248±1,023	97,980±1,292	98,523±0,791	93,377±2,483	98,521±0,792
MobileNet	95,451±0,624	98,113±0,789	98,514±0,990	97,717±0,824	98,666±0,835	95,329±0,395	98,788±0,666
InceptionV3	92,632±2,029	97,967±0,971	98,101±0,920	97,583±1,244	97,562±1,278	90,710±1,385	97,695±1,414
Xception	95,074±0,673	97,472±1,136	98,397±1,000	96,941±0,786	97,335±0,935	96,398±0,546	98,131±0,777
ResNet50	89,652±2,368	97,065±1,370	97,972±0,431	97,590±0,686	98,245±0,696	83,013±2,185	98,247±0,340
InceptionResNetV2	92,287±1,471	96,816±0,467	96,932±0,695	97,034±0,691	97,452±0,516	92,075±1,772	97,733±0,696
NASNetMobile	92,891±0,879	96,481±0,990	96,112±1,439	97,055±1,381	97,444±1,170	94,640±1,320	97,444±1,170
DenseNet121	96,767±1,143	98,259±0,551	98,538±0,766	98,144±0,646	98,272±0,530	85,175±1,530	98,264±0,328
DenseNet169	94,916±1,974	98,117±0,511	98,518±1,166	97,723±0,910	97,983±0,613	91,306±1,427	98,245±0,921
DenseNet201	95,061±1,500	98,009±0,728	98,264±0,524	97,600±0,663	98,261±0,536	92,830±2,306	98,394±0,544

Fonte: Autor

5.1.2 Resultados no Ambiente Virtual

Em seguida, similarmente ao processo realizado com as imagens do ambiente real, foram processados os bancos de imagens sintéticas. As arquiteturas de CNN foram aplicadas nas imagens dando origem aos vetores de entrada dos classificadores.

As Tabelas 11, 12, 13 e 14 exibem os valores médios e desvios padrões de Acurácia, *Recall*, *Precision* e F1-Score, respectivamente, alcançados pelas combinações das arquiteturas de CNN com os classificadores, considerando as imagens do ambiente simulado, tanto com o campo de visão convencional quanto com o campo de visão omnidirecional. Os melhores valores estão destacados em verde.

Analisando as Tabelas 11, 12, 13 e 14, podemos notar que em ambos os banco de imagens sintéticas há combinações que atingiram 100% de Acurácia, *Recall*, *Precision* e F1-Score. Porém, com as imagens sintéticas omnidirecionais, a quantidade de associações que alcançaram o valor máximo nas métricas é maior, com destaque para a arquitetura MobileNet, que atingiu 100% em todas as métricas em conjunto com a maioria dos classificadores, com exceção do método Floresta Aleatória.

Dentre as combinações que retornaram 100% de Acurácia, *Recall*, *Precision* e F1-Score destacam-se as associações da arquitetura VGG16 com os classificadores kNN, SVM(Linear) e SVM(RBF) e da arquitetura MobileNet com kNN, pois, essas mesmas combinações, atingiram o valor máximo nas métricas em ambos os bancos de imagens sintéticas.

Tabela 11 – **Acurácia** alcançada, no **ambiente virtual**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Virtual							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	98,704±0,512	99,933±0,133	100,000±0,000	99,967±0,065	100,000±0,000	37,596±27,194	100,000±0,000
VGG19	98,904±0,292	99,801±0,067	99,934±0,081	99,801±0,067	99,967±0,067	87,088±3,537	99,967±0,067
MobileNet	99,470±0,217	99,933±0,133	100,000±0,000	99,933±0,133	99,933±0,133	99,933±0,133	99,967±0,067
InceptionV3	96,947±0,411	99,536±0,121	99,635±0,245	99,237±0,246	99,834±0,105	29,705±20,852	99,867±0,125
Xception	97,377±0,676	99,668±0,297	99,635±0,123	99,469±0,193	99,967±0,067	40,506±19,856	99,967±0,067
ResNet50	95,019±0,931	99,138±0,240	99,104±0,340	99,005±0,229	99,769±0,221	34,514±21,696	99,769±0,221
InceptionResNetV2	93,958±0,488	99,436±0,307	99,336±0,258	98,307±0,693	99,602±0,166	89,976±0,874	99,602±0,166
NASNetMobile	90,968±1,147	98,938±0,415	97,078±0,790	98,805±0,607	99,502±0,180	97,242±0,624	99,403±0,288
DenseNet121	97,572±0,663	99,601±0,343	99,867±0,125	99,835±0,148	99,835±0,148	43,689±16,080	99,801±0,123
DenseNet169	97,076±0,748	99,800±0,125	99,900±0,082	99,800±0,245	99,933±0,082	33,969±23,605	99,933±0,082
DenseNet201	97,676±0,428	99,967±0,067	99,801±0,125	99,801±0,163	99,967±0,067	39,006±25,099	99,967±0,067
Omni Virtual							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	99,867±0,267	100,000±0,000
VGG19	100,000±0,000	99,867±0,267	100,000±0,000	99,733±0,327	100,000±0,000	100,000±0,000	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	100,000±0,000	100,000±0,000
InceptionV3	99,600±0,533	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	98,800±1,067	99,867±0,267
Xception	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267
ResNet50	99,867±0,267	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	98,667±0,943	100,000±0,000
InceptionResNetV2	99,733±0,533	99,867±0,267	100,000±0,000	100,000±0,000	99,733±0,533	99,600±0,533	100,000±0,000
NASNetMobile	96,400±1,555	99,867±0,267	99,867±0,267	99,333±0,843	99,867±0,267	98,933±0,533	99,867±0,267
DenseNet121	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	99,600±0,533	100,000±0,000
DenseNet169	99,867±0,267	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	99,733±0,327	99,867±0,267
DenseNet201	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,067±0,998	99,867±0,267

Fonte: Autor

Tabela 12 – **Recall** alcançado, no **ambiente virtual**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Virtual							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	98,704±0,512	99,933±0,133	100,000±0,000	99,967±0,065	100,000±0,000	37,596±27,194	100,000±0,000
VGG19	98,904±0,292	99,801±0,067	99,934±0,081	99,801±0,067	99,967±0,067	87,088±3,537	99,967±0,067
MobileNet	99,470±0,217	99,933±0,133	100,000±0,000	99,933±0,133	99,933±0,133	99,933±0,133	99,967±0,067
InceptionV3	96,947±0,411	99,536±0,121	99,635±0,245	99,237±0,246	99,834±0,105	29,705±20,852	99,867±0,125
Xception	97,377±0,676	99,668±0,297	99,635±0,123	99,469±0,193	99,967±0,067	40,506±19,856	99,967±0,067
ResNet50	95,019±0,931	99,138±0,240	99,104±0,340	99,005±0,229	99,769±0,221	34,514±21,696	99,769±0,221
InceptionResNetV2	93,958±0,488	99,436±0,307	99,336±0,258	98,307±0,693	99,602±0,166	89,976±0,874	99,602±0,166
NASNetMobile	90,968±1,147	98,938±0,415	97,078±0,790	98,805±0,607	99,502±0,180	97,242±0,624	99,403±0,288
DenseNet121	97,572±0,663	99,601±0,343	99,867±0,125	99,835±0,148	99,835±0,148	43,689±16,080	99,801±0,123
DenseNet169	97,076±0,748	99,800±0,125	99,900±0,082	99,800±0,245	99,933±0,082	33,969±23,605	99,933±0,082
DenseNet201	97,676±0,428	99,967±0,067	99,801±0,125	99,801±0,163	99,967±0,067	39,006±25,099	99,967±0,067
Omni Virtual							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	99,867±0,267	100,000±0,000
VGG19	100,000±0,000	99,867±0,267	100,000±0,000	99,733±0,327	100,000±0,000	100,000±0,000	100,000±0,000
MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	100,000±0,000	100,000±0,000
InceptionV3	99,600±0,533	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	98,800±1,067	99,867±0,267
Xception	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267
ResNet50	99,867±0,267	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	98,667±0,943	100,000±0,000
InceptionResNetV2	99,733±0,533	99,867±0,267	100,000±0,000	100,000±0,000	99,733±0,533	99,600±0,533	100,000±0,000
NASNetMobile	96,400±1,555	99,867±0,267	99,867±0,267	99,333±0,843	99,867±0,267	98,933±0,533	99,867±0,267
DenseNet121	100,000±0,000	100,000±0,000	100,000±0,000	99,867±0,267	100,000±0,000	99,600±0,533	100,000±0,000
DenseNet169	99,867±0,267	99,733±0,327	99,867±0,267	99,867±0,267	99,867±0,267	99,733±0,327	99,867±0,267
DenseNet201	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,867±0,267	99,067±0,998	99,867±0,267

Fonte: Autor

Tabela 13 – **Precision** alcançado, no **ambiente virtual**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Virtual								
Arquitetura CNN	Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
VGG16		98,771±0,482	99,937±0,127	100,000±0,000	99,968±0,064	100,000±0,000	29,350±32,200	100,000±0,000
VGG19		98,954±0,252	99,807±0,062	99,936±0,079	99,805±0,065	99,967±0,065	88,397±3,410	99,967±0,065
MobileNet		99,488±0,205	99,937±0,127	100,000±0,000	99,937±0,127	99,937±0,127	99,937±0,127	99,967±0,065
InceptionV3		97,077±0,376	99,553±0,111	99,647±0,233	99,263±0,233	99,840±0,100	24,233±28,364	99,872±0,119
Xception		97,540±0,637	99,676±0,288	99,646±0,119	99,490±0,184	99,967±0,065	48,126±19,643	99,967±0,065
ResNet50		95,430±0,722	99,164±0,232	99,123±0,332	99,030±0,225	99,776±0,213	22,870±30,828	99,776±0,213
InceptionResNetV2		94,433±0,472	99,452±0,297	99,358±0,248	98,391±0,661	99,610±0,165	91,280±0,909	99,616±0,161
NASNetMobile		91,405±1,050	98,959±0,408	97,178±0,750	98,829±0,593	99,515±0,177	97,342±0,600	99,413±0,287
DenseNet121		97,746±0,610	99,608±0,340	99,870±0,122	99,839±0,144	99,839±0,144	42,043±21,453	99,806±0,120
DenseNet169		97,303±0,668	99,805±0,122	99,902±0,080	99,803±0,242	99,935±0,080	25,282±31,066	99,935±0,080
DenseNet201		97,841±0,331	99,967±0,065	99,805±0,123	99,807±0,157	99,967±0,065	32,295±29,475	99,967±0,065

Omni Virtual								
Arquitetura CNN	Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
VGG16		100,000±0,000	100,000±0,000	100,000±0,000	99,879±0,242	100,000±0,000	99,879±0,242	100,000±0,000
VGG19		100,000±0,000	99,879±0,242	100,000±0,000	99,758±0,297	100,000±0,000	100,000±0,000	100,000±0,000
MobileNet		100,000±0,000	100,000±0,000	100,000±0,000	99,879±0,242	100,000±0,000	100,000±0,000	100,000±0,000
InceptionV3		99,636±0,485	99,758±0,297	99,879±0,242	99,879±0,242	99,879±0,242	98,943±0,963	99,879±0,242
Xception		99,758±0,297	99,879±0,242	99,879±0,242	99,879±0,242	99,879±0,242	99,879±0,242	99,879±0,242
ResNet50		99,879±0,242	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	98,920±0,740	100,000±0,000
InceptionResNetV2		99,758±0,485	99,879±0,242	100,000±0,000	100,000±0,000	99,778±0,444	99,657±0,450	100,000±0,000
NASNetMobile		96,811±1,322	99,879±0,242	99,879±0,242	99,414±0,752	99,879±0,242	99,051±0,450	99,879±0,242
DenseNet121		100,000±0,000	100,000±0,000	100,000±0,000	99,879±0,242	100,000±0,000	99,657±0,450	100,000±0,000
DenseNet169		99,879±0,242	99,758±0,297	99,879±0,242	99,879±0,242	99,879±0,242	99,758±0,297	99,879±0,242
DenseNet201		99,879±0,242	99,879±0,242	99,879±0,242	99,879±0,242	99,879±0,242	99,172±0,872	99,879±0,242

Fonte: Autor

Tabela 14 – **F1-Score** alcançado, no **ambiente virtual**, pelas arquiteturas CNN e extratores clássicos em combinação com os classificadores a partir das imagens da câmera GoPro[®] e da câmera omnidirecional.

GoPro Virtual								
Arquitetura CNN	Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
	VGG16	98,708±0,510	99,934±0,133	100,000±0,000	99,967±0,065	100,000±0,000	29,708±30,938	100,000±0,000
	VGG19	98,905±0,289	99,801±0,067	99,934±0,081	99,801±0,067	99,967±0,067	86,219±4,017	99,967±0,067
	MobileNet	99,469±0,218	99,933±0,133	100,000±0,000	99,933±0,133	99,933±0,133	99,933±0,133	99,967±0,067
	InceptionV3	96,941±0,407	99,537±0,120	99,634±0,247	99,237±0,245	99,834±0,105	22,114±23,924	99,867±0,125
	Xception	97,357±0,673	99,668±0,297	99,635±0,123	99,469±0,192	99,967±0,067	37,594±21,067	99,967±0,067
	ResNet50	95,039±0,919	99,137±0,240	99,101±0,342	99,003±0,229	99,769±0,222	24,273±26,891	99,769±0,222
	InceptionResNetV2	93,943±0,493	99,436±0,308	99,336±0,258	98,307±0,693	99,602±0,167	89,981±0,975	99,602±0,167
	NASNetMobile	90,957±1,152	98,937±0,416	97,084±0,788	98,804±0,606	99,502±0,180	97,241±0,614	99,402±0,289
	DenseNet121	97,557±0,665	99,602±0,343	99,867±0,125	99,835±0,148	99,835±0,148	36,590±18,901	99,801±0,123
	DenseNet169	97,051±0,758	99,800±0,125	99,900±0,082	99,800±0,245	99,933±0,082	25,632±28,097	99,933±0,082
	DenseNet201	97,647±0,456	99,967±0,067	99,801±0,125	99,801±0,164	99,967±0,067	30,572±29,121	99,967±0,067
Omni Virtual								
Arquitetura CNN	Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
	VGG16	100,000±0,000	100,000±0,000	100,000±0,000	99,866±0,267	100,000±0,000	99,866±0,267	100,000±0,000
	VGG19	100,000±0,000	99,866±0,267	100,000±0,000	99,733±0,327	100,000±0,000	100,000±0,000	100,000±0,000
	MobileNet	100,000±0,000	100,000±0,000	100,000±0,000	99,866±0,267	100,000±0,000	100,000±0,000	100,000±0,000
	InceptionV3	99,599±0,535	99,733±0,327	99,866±0,267	99,866±0,267	99,866±0,267	98,800±1,071	99,866±0,267
	Xception	99,733±0,327	99,866±0,267	99,866±0,267	99,866±0,267	99,866±0,267	99,866±0,267	99,866±0,267
	ResNet50	99,866±0,267	100,000±0,000	100,000±0,000	100,000±0,000	100,000±0,000	98,645±0,963	100,000±0,000
	InceptionResNetV2	99,725±0,550	99,866±0,267	100,000±0,000	100,000±0,000	99,731±0,539	99,597±0,538	100,000±0,000
	NASNetMobile	96,343±1,603	99,866±0,267	99,866±0,267	99,322±0,859	99,866±0,267	98,929±0,538	99,866±0,267
	DenseNet121	100,000±0,000	100,000±0,000	100,000±0,000	99,866±0,267	100,000±0,000	99,597±0,538	100,000±0,000
	DenseNet169	99,866±0,267	99,733±0,327	99,866±0,267	99,866±0,267	99,866±0,267	99,733±0,327	99,866±0,267
	DenseNet201	99,866±0,267	99,866±0,267	99,866±0,267	99,866±0,267	99,866±0,267	99,055±1,007	99,866±0,267

Fonte: Autor

5.2 Análise do Tempo de Processamento

A seguir, são analisados tempos de treinamento e de teste para os bancos de dados que obtiveram os melhores resultados em cada ambiente nas métricas consideradas na seção anterior. Portanto, os bancos de dados avaliados nessa seção são: imagens com campo de visão convencional no ambiente real e imagens com campo de visão omnidirecional no ambiente simulado. São apresentados ainda os tempos de extração das arquiteturas de CNN.

5.2.1 Tempo de Treino

A Tabela 15 apresenta o tempo de treino de cada classificador para o processamento do conjunto de atributos retornado por cada arquitetura de CNN, quando aplicadas ao banco de imagens capturadas pela câmera GoPro[®] no ambiente real e ao banco de imagens capturadas pela câmera omnidirecional no ambiente simulado. Os melhores resultados estão grifados em verde.

Tabela 15 – Tempo de **treino**, em **segundos**, para o banco de imagens **convencionais no ambiente real** e para o banco de imagens **omnidirecionais no ambiente virtual**.

GoPro Real								
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)	
Arquitetura CNN								
VGG16	0,086±0,036	20,574±2,250	0,012±0,002	4,223±0,182	0,791±0,038	1,362±0,034	0,593±0,029	
VGG19	0,078±0,066	13,045±2,350	0,016±0,003	5,803±0,075	1,282±0,067	1,313±0,037	0,589±0,026	
MobileNet	0,119±0,082	6,891±1,917	0,043±0,006	26,929±0,184	1,301±0,018	1,141±0,034	2,522±0,056	
InceptionV3	0,181±0,077	26,464±2,606	0,105±0,019	9,315±0,131	2,218±0,041	4,301±0,049	3,394±0,052	
Xception	0,112±0,037	38,379±3,738	0,119±0,019	15,333±0,082	1,746±0,089	4,558±0,039	1,745±0,047	
ResNet50	0,137±0,076	44,045±6,198	0,088±0,024	3,856±0,041	1,171±0,038	4,057±0,083	1,277±0,015	
InceptionResNetV2	0,182±0,082	49,207±4,298	0,045±0,012	4,040±0,024	1,477±0,032	3,459±0,047	1,411±0,038	
NASNetMobile	0,144±0,053	22,646±0,817	0,033±0,007	13,231±0,088	1,473±0,039	1,088±0,025	1,049±0,048	
DenseNet121	0,055±0,008	17,631±2,763	0,049±0,007	9,060±0,087	0,994±0,036	2,366±0,039	1,102±0,039	
DenseNet169	0,105±0,018	13,899±1,179	0,051±0,006	7,259±0,154	2,234±0,096	3,747±0,069	1,612±0,068	
DenseNet201	0,091±0,026	26,143±1,449	0,046±0,011	13,466±0,248	3,286±0,032	4,364±0,106	2,291±0,079	
Omni Virtual								
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)	
Arquitetura CNN								
VGG16	0,105±0,031	18,563±2,104	0,033±0,007	6,623±0,193	1,884±0,027	1,913±0,037	0,539±0,029	
VGG19	0,134±0,072	18,145±2,515	0,026±0,001	5,041±0,069	1,923±0,043	1,914±0,040	0,569±0,025	
MobileNet	0,154±0,061	15,806±2,251	0,049±0,011	6,698±0,047	2,339±0,015	2,719±0,021	2,507±0,035	
InceptionV3	0,219±0,062	102,367±5,247	0,152±0,019	10,021±0,137	1,674±0,033	7,113±0,127	2,480±0,044	
Xception	0,142±0,068	84,137±11,666	0,135±0,024	6,642±0,082	1,491±0,015	2,790±0,017	1,622±0,090	
ResNet50	0,193±0,076	156,009±19,586	0,108±0,041	10,018±0,049	1,528±0,047	7,043±0,105	1,594±0,024	
InceptionResNetV2	0,180±0,150	53,504±5,267	0,083±0,016	9,492±0,182	1,932±0,034	1,329±0,035	2,047±0,042	
NASNetMobile	0,093±0,027	246,339±12,238	0,036±0,001	10,786±0,222	0,933±0,034	1,506±0,036	1,252±0,025	
DenseNet121	0,110±0,096	33,819±2,134	0,055±0,024	5,442±0,170	3,380±0,091	3,649±0,036	1,212±0,027	
DenseNet169	0,105±0,071	31,986±3,819	0,084±0,016	5,424±0,134	3,964±0,043	5,758±0,110	4,048±0,073	
DenseNet201	0,115±0,006	42,903±4,391	0,079±0,023	15,116±0,072	3,630±0,077	6,714±0,078	2,595±0,030	

Fonte: Autor

Quanto ao banco de imagens com a câmera GoPro[®] no ambiente real, analisando a Tabela 15, destaca-se o classificador kNN por retornar os menores tempos de treino, sendo suas melhores marcas com as arquiteturas VGG16, VGG19 e NASNetMobile, com 0,012 s, 0,016 s e 0,033 s, respectivamente.

Quanto ao banco de imagens com a câmera omnidirecional no ambiente simulado, ainda analisando a Tabela 15, também sobressai o classificador kNN por apresentar os melhores tempos de treino, sendo seus menores tempos com as arquiteturas VGG19, VGG16 e NASNetMobile, com 0,026 s, 0,033 s e 0,036 s, respectivamente.

5.2.2 Tempo de Teste

A Tabela 16 mostra o tempo de teste de cada classificador para o processamento do conjunto de atributos retornado por cada arquitetura de CNN, quando aplicadas ao banco de imagens capturadas pela câmera GoPro[®] no ambiente real e ao banco de imagens capturadas pela câmera omnidirecional no ambiente simulado. Os melhores tempos estão destacados em verde.

Tabela 16 – Tempo de teste, em **milisegundos**, para o banco de imagens **convencionais no ambiente real** e para o banco de imagens **omnidirecionais no ambiente virtual**.

GoPro Real							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	0,003±0,001	0,007±0,000	0,018±0,000	0,156±0,001	0,010±0,000	0,012±0,000	0,008±0,000
VGG19	0,002±0,000	0,005±0,001	0,018±0,000	0,159±0,002	0,012±0,000	0,012±0,000	0,008±0,000
MobileNet	0,003±0,001	0,003±0,001	0,036±0,000	0,159±0,001	0,018±0,000	0,017±0,000	0,025±0,001
InceptionV3	0,004±0,000	0,008±0,001	0,074±0,004	0,164±0,001	0,038±0,002	0,053±0,005	0,040±0,001
Xception	0,002±0,001	0,012±0,002	0,075±0,003	0,160±0,001	0,029±0,001	0,057±0,005	0,029±0,001
ResNet50	0,003±0,001	0,007±0,001	0,053±0,003	0,163±0,001	0,034±0,001	0,059±0,008	0,035±0,000
InceptionResNetV2	0,003±0,001	0,011±0,002	0,054±0,001	0,159±0,002	0,031±0,000	0,046±0,001	0,023±0,001
NASNetMobile	0,003±0,002	0,006±0,001	0,037±0,000	0,156±0,001	0,020±0,000	0,018±0,000	0,018±0,001
DenseNet121	0,002±0,001	0,006±0,001	0,036±0,000	0,163±0,001	0,015±0,000	0,026±0,000	0,016±0,000
DenseNet169	0,002±0,001	0,005±0,001	0,060±0,002	0,161±0,002	0,035±0,002	0,055±0,005	0,025±0,001
DenseNet201	0,002±0,000	0,007±0,000	0,066±0,003	0,159±0,001	0,044±0,002	0,054±0,006	0,034±0,001
Omni Virtual							
Classificador	Naive Bayes	MLP	kNN	RF	SVM(Linear)	SVM(Polin.)	SVM(RBF)
Arquitetura CNN							
VGG16	0,002±0,000	0,005±0,001	0,016±0,000	0,131±0,001	0,015±0,000	0,015±0,000	0,008±0,000
VGG19	0,002±0,001	0,004±0,000	0,016±0,000	0,130±0,001	0,014±0,000	0,015±0,000	0,007±0,000
MobileNet	0,004±0,001	0,004±0,000	0,044±0,001	0,129±0,001	0,034±0,001	0,036±0,000	0,031±0,001
InceptionV3	0,003±0,001	0,015±0,001	0,076±0,003	0,135±0,002	0,038±0,001	0,088±0,002	0,036±0,002
Xception	0,002±0,000	0,013±0,002	0,062±0,002	0,130±0,001	0,040±0,001	0,071±0,004	0,031±0,001
ResNet50	0,004±0,001	0,011±0,002	0,049±0,001	0,136±0,001	0,037±0,002	0,081±0,011	0,032±0,002
InceptionResNetV2	0,002±0,001	0,012±0,001	0,070±0,002	0,135±0,001	0,035±0,002	0,023±0,001	0,028±0,001
NASNetMobile	0,002±0,001	0,011±0,003	0,038±0,001	0,130±0,002	0,018±0,000	0,030±0,001	0,020±0,000
DenseNet121	0,002±0,001	0,006±0,001	0,037±0,001	0,131±0,002	0,030±0,001	0,035±0,001	0,020±0,001
DenseNet169	0,002±0,001	0,006±0,000	0,077±0,003	0,128±0,002	0,055±0,005	0,056±0,003	0,053±0,005
DenseNet201	0,002±0,000	0,007±0,001	0,068±0,001	0,134±0,002	0,061±0,006	0,076±0,004	0,036±0,001

Fonte: Autor

Quanto ao banco de imagens com a câmera GoPro[®] no ambiente real, observando a Tabela 15, o classificador Naive Bayes destaca-se por possuir os melhores tempos de teste em combinação com as arquiteturas VGG19, Xception, DenseNet121, DenseNet169 e DenseNet201, com a marca de 0,002 s.

Em relação ao banco de imagens com a câmera omnidirecional no ambiente simulado, ainda observando a Tabela 16, o classificador Naive Bayes ainda sobressai, quando em conjunto com as arquiteturas VGG16, VGG19, Xception, InceptionResNetV2, NAS-

NetMobile, DenseNet121, DenseNet169 e DenseNet201, todos retornando um tempo de teste de 0,002 s.

Com relação ao tempo de extração, a Tabela 17 apresenta o tempo para cada arquitetura de CNN. Pode-se concluir por essa Tabela que os melhores tempos de extração são retornados pelas arquiteturas MobileNet, VGG16 e VGG19, as quais cumprem sua tarefa em 94.01s, 164.42s e 188.11s, respectivamente.

Tabela 17 – Tempo de extração para cada arquitetura CNN.

Arquitetura CNN	Tempo de Extração (ms)
VGG16	164.42±0.50
VGG19	188.11±0.14
MobileNet	94.01±0.55
InceptionV3	274.61±0.30
Xception	264.12±0.42
ResNet50	188.30±0.64
InceptionResNetV2	632.33±0.32
NASNetMobile	298.43±0.08
DenseNet121	507.37±0.02
DenseNet169	666.03±0.11
DenseNet201	832.14±0.20

Fonte: Autor

5.3 Análise Geral dos Resultados

Considerando as análises realizadas na Seção 5.1, é possível afirmar que as imagens obtidas com a câmera de alta resolução GoPro[®] no ambiente real proporcionam um melhor desempenho para o sistema de localização proposto, pois apresenta uma maior quantidade de combinações que atingiram o valor de 100% em todas as métrica de avaliação utilizadas. Apesar das imagens omnidirecionais sintéticas também terem apresentado combinações que retornaram 100% em todas as métrica, quantitativamente, o banco de imagens convencionais reais sobressai.

Na análise do tempo de processamento, realizada na Seção 5.2, o classificador kNN destaca-se na fase de treinamento, por cumprir em 0,002 s essa etapa. Em contrapartida, na fase de classificação, Naive Bayes é destacado por ser o mais rápido, realizando essa etapa em 0,002 s. Tanto o kNN, na fase de treinamento, quanto o Naive Bayes, na fase de teste, sobressaem para ambos os bancos de dados analisados na Seção 5.2.

6 Conclusão

Neste artigo, é proposta uma nova abordagem localização e navegação de robôs móveis em mapas topológicos a partir de atributos gerados por Redes Neurais Convolucionais (CNN) em imagens com campo de visão convencional e omnidirecional. A área para a navegação do robô é do tipo interna e as imagens são obtidas utilizando uma câmera de alta resolução e uma câmera com lente omnidirecional, considerando o ambiente real e simulado.

Para analisar o sistema de visão computacional proposto, foram realizadas combinações das arquiteturas de CNN com os classificadores selecionadas, afim de obter a melhor configuração, definida segundo os resultados de parâmetros como Acurácia, *Recall*, *Precision* e F1-Score, além de uma análise de tempo de processamento, fundamental em aplicações para sistemas embarcados.

Em busca de uma intercessão entre os melhores resultados na análise de desempenho (Seção 5.1) com os melhores valores na análise do tempo de processamento (Seção 5.2), as combinações das arquiteturas VGG16 e VGG19 com o classificador kNN mostraram-se adequadas para a fase de treinamento, obtendo os menores tempos nessa etapa e ainda atingindo 100% em todas as métricas com os bancos de dados destacados na Seção 5.2. Na fase de teste, Naive Bayes mostrou-se satisfatório, retornando os melhores tempos e um resultado de 100% em todas as métricas quando em associação com as arquiteturas VGG16 e VGG19, considerando os bancos de dados enfatizados na Seção 5.2.

Como, em sistemas embarcados, parâmetros como tempo de teste, de extração e quantidade de armazenamento são os mais relevantes, a combinação da arquitetura VGG16 com o classificador Naive Bayes configura-se como bastante adequada para a implantação no sistema de navegação, atingindo o valor de 100% em todas as métricas, possuindo o menor tempo de classificação, com 0,002s, enquanto a arquitetura VGG16 apresenta ainda a menor quantidade de atributos a serem processados, retornando 512 atributos, como mostra a Tabela 5. Apesar de não apresentar o menor tempo de extração, mas o segundo melhor tempo nessa tarefa, essa combinação ainda se mantém como mais adequada, por destacar-se nos demais quesitos, principalmente, no que se refere à necessidade de armazenamento.

Dessa forma, os resultados afirmam a importância da abordagem proposta para a realização das tarefas de localização e, conseqüentemente, de navegação de robôs móveis, compreendendo sistemas de visão computacional tanto do ponto de vista real quando do ponto de vista simulado (virtual).

6.1 Trabalhos Futuros

O trabalho apresentado nesta dissertação pode ser estendido em várias direções. Como trabalhos futuros, sugerem-se:

- Realizar um estudo com outros tipos de ambientes, tanto internos quanto externos;
- Analisar a influência da resolução das imagens;
- Realizar testes de validação estatística, como Friedman, Wilcoxon e outros;
- Empregar outras arquiteturas de CNN como ResNeXt-50 (XIE et al., 2017) e CapsNet (SABOUR; FROSST; HINTON, 2017);
- Utilizar outros métodos de Aprendizagem de Máquina, como Floresta de Caminhos Ótimos (*Optimum-Path Forest*, OPF) (NUNES et al., 2014).

Referências

ALARIFI, A. et al. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, v. 16, n. 5, p. 707, 2016. Citado 2 vezes nas páginas 19 e 20.

BENGIO, Y.; GOODFELLOW, I. J.; COURVILLE, A. Deep learning. *Nature*, Citeseer, v. 521, n. 7553, p. 436–444, 2015. Citado 2 vezes nas páginas 39 e 41.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, Institute of Electrical and Electronics Engineers (IEEE), v. 5, n. 2, p. 157–166, mar 1994. Disponível em: <<https://doi.org/10.1109/72.279181>>. Citado na página 48.

BESSA, J. A. et al. Global location of mobile robots using artificial neural networks in omnidirectional images. *IEEE Latin America Transactions*, v. 13, n. 10, p. 3405–3414, 2015. Citado na página 21.

BONIN-FONT, F.; ORTIZ, A.; OLIVER, G. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, v. 53, n. 3, p. 263–296, 2008. Citado na página 21.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. Citado na página 30.

CALDERONI, L. et al. Indoor localization in a hospital environment using random forest classifiers. *Expert Syst. Appl.*, v. 42, n. 1, p. 125–134, 2015. Citado na página 20.

CAMPOS, R. S.; LOVISOLO, L.; CAMPOS, M. L. R. de. Wi-fi multi-floor indoor positioning considering architectural aspects and controlled computational complexity. *Expert Syst. Appl.*, v. 41, n. 14, p. 6211–6223, 2014. Citado na página 20.

CHENG, H.; CHEN, H.; LIU, Y. Topological indoor localization and navigation for autonomous mobile robot. *IEEE Trans. Automation Science and Engineering*, v. 12, n. 2, p. 729–738, 2015. Citado na página 19.

CHIN, W. H. et al. Multi-channel bayesian adaptive resonance associate memory for on-line topological map building. *Applied Soft Computing*, v. 38, p. 269 – 280, 2016. ISSN 1568-4946. Citado na página 18.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. Disponível em: <<https://doi.org/10.1109/cvpr.2017.195>>. Citado 2 vezes nas páginas 22 e 49.

CHOWDHURY, T. J. S. et al. Advances on localization techniques for wireless sensor networks: A survey. *Computer Networks*, v. 110, p. 284–305, 2016. Citado na página 20.

Ciabattoni, L. et al. Real-time fall detection system by using mobile robots in smart homes. In: *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. [S.l.: s.n.], 2017. p. 15–16. ISSN 2166-6822. Citado na página 17.

- COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado na página 22.
- DEAK, G.; CURRAN, K.; CONDELL, J. A survey of active and passive indoor localisation systems. *Computer Communications*, v. 35, n. 16, p. 1939–1954, 2012. Citado 2 vezes nas páginas 19 e 20.
- DEBSKI, A. et al. Open-source localization device for indoor mobile robots. *Procedia Computer Science*, v. 76, p. 139–146, 2015. Citado na página 20.
- DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. [S.l.], 2009. p. 248–255. Citado 2 vezes nas páginas 35 e 42.
- DIOP, M. et al. A computer vision-aided motion sensing algorithm for mobile robot's indoor navigation. In: *IEEE 14th International Workshop on Advanced Motion Control, AMC 2016, Auckland, New Zealand, April 22-24, 2016*. [S.l.: s.n.], 2016. p. 400–405. Citado na página 17.
- DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2. ed. New York: Wiley, 2001. Citado na página 26.
- FUENTES-PACHECO, J.; ASCENCIO, J. R.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. *Artif. Intell. Rev.*, v. 43, n. 1, p. 55–81, 2015. Citado na página 21.
- FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: *Competition and cooperation in neural nets*. [S.l.]: Springer, 1982. p. 267–285. Citado na página 41.
- GÁLVEZ-LÓPEZ, D.; TARDÓS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robotics*, v. 28, n. 5, p. 1188–1197, 2012. Citado na página 21.
- GARCIA-FIDALGO, E.; ORTIZ, A. Vision-based topological mapping and localization by means of local invariant features and map refinement. *Robotica*, v. 33, n. 7, p. 1446–1470, 2015. Citado na página 21.
- GARCIA-FIDALGO, E.; ORTIZ, A. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, v. 64, p. 1–20, 2015. Citado na página 19.
- GASPAR, J. *Sketchup Pro 2013 Step by Step*. [S.l.]: GetPro Books, 2013. ISBN 8561453184, 9788561453183. Citado na página 58.
- GRUBINGER, T.; CHASPARIS, G. C.; NATSCHLÄGER, T. Generalized online transfer learning for climate control in residential buildings. *Energy and Buildings*, v. 139, p. 63 – 71, 2017. ISSN 0378-7788. Citado na página 59.
- GU, J. et al. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015. Citado na página 38.
- HARTMANN, B. *Human Worker activity recognition in industrial environments*. Karlsruhe, Germany: KIT Scientific Publishing, 2011. Citado na página 20.

- HAYKIN, S. *Neural Networks and Learning Machines*. McMaster University, Canada: Prentice Hall, 2008. Citado 4 vezes nas páginas 22, 31, 41 e 65.
- HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado 2 vezes nas páginas 22 e 35.
- Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, mar. 2015. Citado na página 49.
- HO, T. K. Random decision forests. In: IEEE. *Document analysis and recognition, 1995., proceedings of the third international conference on*. [S.l.], 1995. v. 1, p. 278–282. Citado na página 30.
- HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. Citado 3 vezes nas páginas 22, 50 e 51.
- HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. Citado na página 35.
- HUANG, G. et al. Densely connected convolutional networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. Disponível em: <<https://doi.org/10.1109/cvpr.2017.243>>. Citado 3 vezes nas páginas 22, 51 e 52.
- Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv e-prints*, fev. 2015. Citado na página 45.
- Jiang, R. et al. Geometric map-assisted localization for mobile robots based on uniform-gaussian distribution. *IEEE Robotics and Automation Letters*, v. 2, n. 2, p. 789–795, April 2017. ISSN 2377-3766. Citado na página 18.
- JIMENEZ, A. et al. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In: *IEEE International Symposium on Intelligent Signal Processing, 2009. WISP 2009*. [S.l.: s.n.], 2009. p. 37–42. Citado na página 20.
- JIN, J.; DUNDAR, A.; CULURCIELLO, E. Flattened convolutional neural networks for feedforward acceleration. *CoRR*, abs/1412.5474, 2014. Disponível em: <<http://arxiv.org/abs/1412.5474>>. Citado na página 47.
- JOHNS, E.; YANG, G. Global localization in a dense continuous topological map. In: *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. [S.l.: s.n.], 2011. p. 1032–1037. Citado na página 21.
- KIM, H.-S.; CHOI, J.-S. Advanced indoor localization using ultrasonic sensor and digital compass. In: *International Conference on Control, Automation and Systems, 2008. ICCAS 2008*. [S.l.: s.n.], 2008. p. 1114–1120. Citado na página 20.
- KOTHARI, N. et al. Robust indoor localization on a commercial smart phone. In: *Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies (ANT 2012), the 9th International Conference on Mobile Web Information Systems (MobiWIS-2012), Niagara Falls, Ontario, Canada, August 27-29, 2012*. [S.l.: s.n.], 2012. p. 1114–1120. Citado na página 20.

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 2 vezes nas páginas 35 e 42.

KURDI, M. M.; DADYKIN, A. K.; ELZEIN, I. Navigation of mobile robot with cooperation of quadcopter. In: *Ninth International Conference on Advanced Computational Intelligence, ICACI 2017, Doha, Qatar, February 4-6, 2017*. [S.l.: s.n.], 2017. p. 30–36. Citado na página 17.

KWIECIEN, A. et al. Reliability of bluetooth smart technology for indoor localization system. In: *Computer Networks - 22nd International Conference, CN 2015, Brunów, Poland, June 16-19, 2015. Proceedings*. [S.l.: s.n.], 2015. p. 444–454. Citado na página 20.

LÄNGKVIST, M.; KARLSSON, L.; LOUTFI, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, Elsevier BV, v. 42, p. 11–24, jun 2014. Disponível em: <<https://doi.org/10.1016/j.patrec.2014.01.008>>. Citado 2 vezes nas páginas 49 e 50.

LECUN, Y. *Generalization and network design strategies*. [S.l.]: Elsevier, 1989. Citado na página 36.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 5 2015. ISSN 0028-0836. Citado 2 vezes nas páginas 35 e 36.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1990. p. 396–404. Citado 2 vezes nas páginas 35 e 42.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado 3 vezes nas páginas 35, 41 e 42.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineers (IEEE), v. 86, n. 11, p. 2278–2324, 1998. Citado na página 42.

LI, H. Low-cost 3d bluetooth indoor positioning with least square. *Wireless Personal Communications*, v. 78, n. 2, p. 1331–1344, 2014. Citado na página 20.

LI-ZHEN, X. *Vray render 3ds max scene rendering techniques*. [S.l.]: Coal Technology, 2010. v. 9. 98 p. Citado na página 58.

LIAW, A.; WIENER, M. et al. Classification and regression by randomforest. *R news*, v. 2, n. 3, p. 18–22, 2002. Citado na página 22.

LIN, Q. et al. Investigation into the topology optimization for conductive heat transfer based on deep learning approach. *International Communications in Heat and Mass Transfer*, v. 97, p. 103 – 109, 2018. ISSN 0735-1933. Citado na página 59.

LIN, T. et al. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. Citado 2 vezes nas páginas 48 e 60.

- LIN, Y.; JEON, Y. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, Taylor & Francis, v. 101, n. 474, p. 578–590, 2006. Citado na página 31.
- LIU, M.; SIEGWART, R. DP-FACT: towards topological mapping and scene recognition with color for omnidirectional camera. In: *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*. [S.l.: s.n.], 2012. p. 3503–3508. Citado na página 21.
- LOWRY, S. et al. Visual place recognition: A survey. *IEEE Transactions on Robotics*, v. 32, n. 1, p. 1–19, Feb 2016. Citado na página 18.
- MAINETTI, L.; PATRONO, L.; SERGI, I. A survey on indoor positioning systems. In: *22nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2014, Split, Croatia, September 17-19, 2014*. [S.l.: s.n.], 2014. p. 111–120. Citado na página 20.
- MAUTZ, R.; TILCH, S. Survey of optical indoor positioning systems. In: *2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011, Guimaraes, Portugal, September 21-23, 2011*. [S.l.: s.n.], 2011. p. 1–7. Citado 2 vezes nas páginas 19 e 20.
- MAZO, C. et al. Transfer learning for classification of cardiovascular tissues in histological images. *Computer Methods and Programs in Biomedicine*, v. 165, p. 69 – 76, 2018. ISSN 0169-2607. Citado na página 59.
- MIAH, M. S.; GUEAIEB, W. A fuzzy logic approach for indoor mobile robot navigation using UKF and customized RFID communication. In: *Autonomous and Intelligent Systems - Second International Conference, AIS 2011, Burnaby, BC, Canada, June 22-24, 2011. Proceedings*. [S.l.: s.n.], 2011. p. 21–30. Citado na página 20.
- NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: FÜRKNRANZ, J.; JOACHIMS, T. (Ed.). *ICML*. Omnipress, 2010. p. 807–814. Disponível em: <<http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10>>. Citado na página 38.
- NOWICKI, M.; SKRZYPCZYNSKI, P. Indoor navigation with a smartphone fusing inertial and wifi data via factor graph optimization. In: *Mobile Computing, Applications, and Services - 7th International Conference, MobiCASE 2015, Berlin, Germany, November 12-13, 2015, Revised Selected Papers*. [S.l.: s.n.], 2015. p. 280–298. Citado na página 20.
- NUNES, T. M. et al. EEG signal classification for epilepsy diagnosis via optimum path forest - A systematic assessment. *Neurocomputing*, v. 136, p. 103–123, 2014. Citado na página 78.
- PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2010. Citado na página 59.
- RANGANATHAN, A.; DELLAERT, F. Online probabilistic topological mapping. *I. J. Robotics Res.*, v. 30, n. 6, p. 755–771, 2011. Citado na página 21.
- Ravindran, R.; Mills, J. P.; Krishnan, M. Autonomous multi-robot platoon monitoring. In: *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. [S.l.: s.n.], 2018. p. 328–331. ISSN 1558-3899. Citado na página 17.

Rohmer, E.; Singh, S. P. N.; Freese, M. V-rep: A versatile and scalable robot simulation framework. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2013. p. 1321–1326. Citado na página 57.

RUBIO, F. et al. Comparison between bayesian network classifiers and svms for semantic localization. *Expert Syst. Appl.*, v. 64, p. 434–443, 2016. Citado na página 20.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Springer Nature, v. 323, n. 6088, p. 533–536, oct 1986. Disponível em: <<https://doi.org/10.1038/323533a0>>. Citado na página 42.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado 5 vezes nas páginas 42, 43, 44, 48 e 60.

SABOUR, S.; FROSST, N.; HINTON, G. E. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017. Citado na página 78.

SAEDAN, M.; LIM, C. W.; ANG, M. H. Appearance-based slam with map loop closing using an omnidirectional camera. In: *2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. [S.l.: s.n.], 2007. Citado na página 21.

SCHINDLER, G.; METZGER, C.; STARNER, T. A wearable interface for topological mapping and localization in indoor environments. In: *Location- and Context-Awareness, Second International Workshop, LoCA 2006, Dublin, Ireland, May 10-11, 2006, Proceedings*. [S.l.: s.n.], 2006. p. 64–73. Citado na página 20.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. Disponível em: <<http://arxiv.org/abs/1409.1556>>. Citado 4 vezes nas páginas 22, 43, 44 e 47.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 35.

SIOUTIS, M.; TAN, Y. User indoor location system with passive infrared motion sensors and space subdivision. In: *Distributed, Ambient, and Pervasive Interactions - Second International Conference, DAPI 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings*. [S.l.: s.n.], 2014. p. 486–497. Citado na página 20.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, v. 45, n. 4, p. 427–437, 2009. Citado na página 63.

SON, J.; KIM, S.; SOHN, K. A multi-vision sensor-based fast localization system with image matching for challenging outdoor environments. *Expert Syst. Appl.*, v. 42, n. 22, p. 8830–8839, 2015. Citado na página 20.

SONG, Z.; JIANG, G.; HUANG, C. A survey on indoor positioning technologies. In: ZHOU, Q. (Ed.). *Theoretical and Mathematical Foundations of Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 198–206. ISBN 978-3-642-24999-0. Citado na página 17.

- SONG, Z.; JIANG, G.; HUANG, C. A survey on indoor positioning technologies. In: *Theoretical and Mathematical Foundations of Computer Science*. [S.l.]: Springer Berlin Heidelberg, 2011. p. 198–206. Citado na página 20.
- SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 40.
- STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, JSTOR, p. 111–147, 1974. Citado na página 65.
- SUH, H. K. et al. Transfer learning for the classification of sugar beet and volunteer potato under field conditions. *Biosystems Engineering*, v. 174, p. 50 – 65, 2018. ISSN 1537-5110. Citado na página 59.
- SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *AAAI*. [S.l.: s.n.], 2017. p. 4278–4284. Citado na página 22.
- SZEGEDY, C. et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. Disponível em: <<https://doi.org/10.1109/cvpr.2015.7298594>>. Citado 3 vezes nas páginas 44, 48 e 51.
- SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. Disponível em: <<https://doi.org/10.1109/cvpr.2016.308>>. Citado 3 vezes nas páginas 22, 35 e 47.
- TAN, C. et al. A survey on deep transfer learning. In: *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III*. [S.l.: s.n.], 2018. p. 270–279. Citado na página 59.
- THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition (Fourth Edition)*. USA: Academic Press, 2008. Citado 2 vezes nas páginas 22 e 25.
- VAPNIK, V. N. *Statistical Learning Theory*. Nova Jersey, EUA: John Wiley & Sons, 1998. Citado na página 22.
- VARSHAVSKY, A. et al. GSM indoor localization. *Pervasive and Mobile Computing*, v. 3, n. 6, p. 698–720, 2007. Citado na página 20.
- WANG, Q. et al. Transferred deep learning based waveform recognition for cognitive passive radar. *Signal Processing*, v. 155, p. 259 – 267, 2019. ISSN 0165-1684. Citado na página 59.
- WANG, Z. L. et al. Reviews on planar region detection for visual navigation of mobile robot under unknown environment. In: LEE, G. (Ed.). *Advances in Automation and Robotics, Vol. 2*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 593–601. ISBN 978-3-642-25646-2. Citado na página 18.

- Widyantara, H.; Rivai, M.; Purwanto, D. Gas source localization using an olfactory mobile robot equipped with wind direction sensor. In: *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*. [S.l.: s.n.], 2018. p. 66–70. Citado na página 17.
- WU, S.; ZHONG, S.; LIU, Y. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, Springer Nature, v. 77, n. 9, p. 10437–10453, feb 2017. Disponível em: <<https://doi.org/10.1007/s11042-017-4440-4>>. Citado 2 vezes nas páginas 48 e 49.
- XIE, S. et al. Aggregated residual transformations for deep neural networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. [S.l.: s.n.], 2017. p. 5987–5995. Citado na página 78.
- YAYAN, U.; YUCEL, H.; YAZICI, A. A low cost ultrasonic based positioning system for the indoor navigation of mobile robots. *Journal of Intelligent and Robotic Systems*, v. 78, n. 3-4, p. 541–552, 2015. Citado na página 20.
- ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014. p. 818–833. Disponível em: <https://doi.org/10.1007/978-3-319-10590-1_53>. Citado na página 43.
- ZOPH, B.; LE, Q. V. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. Citado 2 vezes nas páginas 52 e 53.
- ZOPH, B. et al. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017. Citado na página 22.
- Łukowska, A. et al. Soil sampling mobile platform for agriculture 4.0. In: *2019 20th International Carpathian Control Conference (ICCC)*. [S.l.: s.n.], 2019. p. 1–4. Citado na página 17.