PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

RODNEY SALES NOGUEIRA JUNIOR

MODELO BASEADO EM DEEP LEARNING PARA DETECÇÃO DE PORTAS E ESCADAS PARA AUXILIAR DEFICIENTES VISUAIS

Porto Alegre
2021

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MODELO BASEADO EM DEEP LEARNING PARA DETECÇÃO DE PORTAS E ESCADAS PARA AUXILIAR DEFICIENTES VISUAIS

RODNEY SALES NOGUEIRA JUNIOR

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Profª. Isabel Harb Manssour

Porto Alegre
2021

Ficha Catalográfica

N778m Nogueira Junior, Rodney Sales

Modelo baseado em deep learning para detecção de portas e escadas para auxiliar deficientes visuais / Rodney Sales Nogueira Junior. – 2018.

87 p.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Isabel Harb Manssour.

1. Visão Computacional. 2. Redes Neurais Convolucionais. 3. Ambientes Internos. 4. Deficientes Visuais. I. Manssour, Isabel Harb. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

RODNEY SALES NOGUEIRA JUNIOR

MODELO BASEADO EM DEEP LEARNING PARA DETECÇÃO DE PORTAS E ESCADAS PARA AUXILIAR DEFICIENTES VISUAIS

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado(a) em 29 de Agosto de 2018.

BANCA EXAMINADORA:

Profa. Dra. Soraia Raupp Musse (PPGCC/PUCRS)

Prof. Dr. Claudio Rosito Jung (UFRGS)

Profª. Isabel Harb Manssour (PPGCC/PUCRS - Orientadora)

DEDICATÓRIA

Dedico este trabalho a meus pais.

“The art of simplicity is a puzzle of complexity.”
(Douglas Horton)

AGRADECIMENTOS

Agradeço primeiramente a meus pais, por todo o apoio, e carinho. Também aos amigos e colegas, em especial aos colegas Carlos Alberto, e Erich, pela parceria nos tempos difíceis, e noites viradas fazendo trabalhos. Ao colega Marcelo, pela paciência, e pelas mil ajudas nas dificuldades do desenvolvimento desse trabalho, e também ao amigo Gyrão pelos gráficos e ajudas na revisão do texto. Agradeço também ao Francisco, e a *South System*, pela confiança no meu trabalho. E também a Isabel pela oportunidade, pelas lições, e pela paciência de orientar todo o trabalho.

MODELO BASEADO EM DEEP LEARNING PARA DETECÇÃO DE PORTAS E ESCADAS PARA AUXILIAR DEFICIENTES VISUAIS

RESUMO

Devido ao grande numero de pessoas com deficiência visual no mundo, e com o avanço tecnológico, os interesses em pesquisa e desenvolvimento de diferentes técnicas de apoio a mobilidade dessas pessoas aumentou. Nesse contexto a detecção de portas e escadas é um tópico de pesquisa importante, pois provê informações que podem auxiliar a mobilidade dessas pessoas. Este trabalho apresenta um modelo para auxílio a navegação em ambientes internos para deficientes visuais. Através de um estudo aprofundado da literatura, foram encontrados trabalhos que utilizam de técnicas de visão computacional para identificar corredores, obstáculos, portas e escadas. Entretanto, poucos utilizam os recentes avanços em visão computacional e redes neurais convolucionais para esse objetivo. Desta forma, o presente modelo engloba um experimento sobre as redes neurais convolucionais para o reconhecimento e detecção de portas e escadas. Usando o método YOLO, apresentamos um modelo que não só detecta diferentes tipos de portas, como também também é capaz de diferenciar escadas ascendentes e descendentes, com taxas de FPS próximas a 30, e mAP acima de 90%.

Palavras-Chave: Visão Computacional, Redes Neurais Convolucionais, Ambientes Internos, Deficientes Visuais.

A DEEP LEARNING BASED MODEL FOR THE DETECTION OF DOORS AND STAIRS TO AID THE VISUALLY IMPAIRED

ABSTRACT

Due to a large number of visually impaired persons in the world, and with the advance of technology, the research interest in the development of different approaches to support the mobility of these persons has increased. In this context, the detection of doors and stairs is an important research topic because it provides useful information that can aid in the mobility of these persons. In this work, we present a model to aid the visually impaired navigation in indoor environments. We found approaches that use computer vision techniques to identify corridors, obstacles, stairs, and doors through a literature review. However, few of them use recent techniques in computer vision and convolutional neural networks in their solutions. Thus, the presented model includes an experiment on convolutional neural networks to recognize and detect doors and stairs. Using the YOLO method, we present a model that detects not only different kinds of doors but also is capable of differentiating ascending and descending stairs, with FPS rates close to 30 and mAP above 90%.

Keywords: Computer Vision, Convolutional Neural Networks, Indoor Environments, Visually Impaired.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

SIFT – Scale-invariant feature transform

SURF – Speeded up robust features

CNN – Convolutional Neural Networks

RNN – Recurrent Neural Networks

DNN – Deep Neural Networks

FC – Fully Connected

RELU – Rectified Linear Unit

SGD – Stocastic Gradient Descent

MSE – Mean Squared Error

RPN – Region Proposal Network

R-CNN – Region-based Convolutional Neural Network

YOLO – You Look Only Once

SSD – Single Shot MultiBox Detector

AP – Average Precision

MAP – Mean Average Precision

IOU – Intersection over Union

VP – Verdadeiro Positivo

VN – Verdadeiro Negativo

FP – Falso Positivo

FN – Falso Negativo

DSD – Doors and Stairs Dataset

PR – Precision Recall

SUMÁRIO

1. INTRODUÇÃO

Conforme dados da OMS (Organização Mundial de Saúde), 39 milhões de pessoas em uma escala global são cegas, e 246 milhões tem problemas de visão. Navegação e assistência a mobilidade é essencial para a comunidade de deficientes visuais, uma vez que fornece um meio de alcançar a independência e equidade social [47]. Neste contexto, tecnologias de navegação podem trazer benefícios para uma grande quantidade de pessoas, auxiliando-as a se locomover em diversos ambientes e a encontrar determinados locais, como por exemplo, uma loja em um shopping, ou a biblioteca de uma universidade.

A tarefa de encontrar determinados locais, como, por exemplo, uma sala de aula em uma universidade, ou seja um ambiente interno e desconhecido, é complexa, pois há a necessidade de algum tipo de orientação como placas, mapas, ou auxilio de outras pessoas. Para as pessoas com deficiência visual é ainda mais complexo, tendo em vista que elas necessitam quase que exclusivamente da assistência de outra pessoa.

Mekhalfi et al. [42] citam que os requisitos para uma pessoa deficiente visual se deslocar em qualquer ambiente são basicamente: navegação, permitindo ao usuário encontrar os locais sem assistência de outras pessoas, e reconhecimento, provendo informações sobre o ambiente ao seu redor, como obstáculos e pessoas.

Para a navegação, a maioria dos trabalhos que provêm assistências são baseados no GPS [42]. Entretanto, tais soluções encontram dificuldades para trabalhar em ambientes internos, e, além disso, não fornecem informações suficientes sobre o ambiente ao redor de seus usuários. Outros trabalhos utilizam técnicas que envolvem outros tipos de sensores, como o de Sanchez et al. [66] que utiliza rádio-frequência (RFID-*Radio-Frequency IDentification*).

Tapu et al. [67] afirma que soluções baseadas em visão computacional constituem uma alternativa promissora para esses problemas. Neste contexto, técnicas de visão computacional em áreas como reconhecimento facial e reconhecimento de objetos estão se mostrando promissoras. Esses métodos recebem melhorias significativas em desempenho, acurácia e confiabilidade ao longo das pesquisas, auxiliando assim na resolução de diversos problemas [26]. Com os avanços na área de aprendizagem de máquina, com técnicas de *deep learning*, na qual redes neurais convolucionais tem batido recordes nas tarefas de reconhecimento de imagens [53], essas também vêm sendo utilizadas no contexto de ambientes internos, como no trabalho de Kumar e Meher [32].

Alguns trabalhos desenvolvidos com a finalidade de navegação em ambientes internos continuam dependendo de auxilio de mapas, como o trabalho proposto por Serrão et al. [57] que utiliza sistemas de informação geográficos para a localização no ambiente. Além desse, outros trabalhos detectam obstáculos no ambiente, mas não provêm informações de navegação, como no caso dos que apenas detectam portas [45, 58] ou obstáculos [67, 13]

no ambiente. Também há diversos trabalhos que são capazes de prover informações pertinentes em imagens com técnicas de *deep learning*, tais como a proposta de Mao et al. [41], que consegue descrever em detalhes cada imagem, ou o trabalho de Ren et al. [51] que consegue identificar diversos elementos de uma imagem.

Considerando estes trabalhos, surge, então, a seguinte pergunta de pesquisa: É possível desenvolver um modelo baseado em *deep learning* que auxilie deficientes visuais na localização de escadas e portas em ambientes internos, usando técnicas de visão computacional?

Sendo assim, o presente estudo apresenta como proposta de desenvolvimento um modelo para auxiliar pessoas com deficiência visual a navegar em ambientes internos baseado em técnicas de visão computacional, que visa informar o usuário sobre a localização de portas e escadas. Dessa forma esse documento está organizado da seguinte maneira: o Capitulo 2 expõe os fundamentos para o entendimento da área de visão computacional, reconhecimento de imagens, e *deep learning*; o Capitulo 3 apresenta uma revisão sistemática sobre a área de detecção de escadas, portas, e objetos; a proposta do projeto, metodologia para o desenvolvimento do projeto, e os experimentos realizados são apresentados no Capitulo 4; os resultados da pesquisa e do projeto são apresentados no Capitulo 5; e por fim as conclusões do trabalho são apresentadas no Capitulo 6.

2. FUNDAMENTAÇÃO TEÓRICA

Szeliski [65] afirma que visão computacional procura descrever o mundo que se vê nas imagens e reconstruir suas propriedades, tais como, formas, iluminação e distribuições de cores. Alguns exemplos de aplicações de visão computacional são: *Optical Character Recognition* (OCR), ou seja, leitura e reconhecimento de caracteres; reconhecimento facial; reconhecimento de objetos; análise de imagens médicas [65]; ou até auxílio a detecção de obstáculos e navegação para deficientes visuais [57, 45]. Nestes exemplos de aplicações, são procuradas informações relevantes em imagens, como características faciais ou informações dos objetos, como bordas, retângulos, entre outros. A partir disso, são aplicados métodos de processamento dessas informações para obter resultados relevantes, tal como qual objeto uma determinada imagem possui, ou onde está um determinado objeto.

Um exemplo de uma tarefa visão computacional é o reconhecimento de dígitos escritos a mão. Para isto, LeCun et al. [34] propuseram um conjunto de dados com imagens em preto e branco de dígitos escritos a mão, chamado de MNIST. Neste caso, o problema proposto é que dada uma imagem preto e branco de um número de 0 a 9, um algoritmo deve dizer qual é o respectivo número. Ou seja, o algoritmo deve interpretar, a partir de um conjunto de informações extraídas da imagem, conforme exposto na Figura 2.1, qual dígito esse conjunto de informações está representando. A tarefa de dada uma imagem x encontrar a qual classe y ela pertence, é chamada de classificação de imagem, e é usualmente desempenhada por algoritmos de aprendizagem de máquina.

Aprendizagem de máquina é uma área de pesquisa que consiste em, dada uma determinada tarefa, um programa de computador ser capaz de aprender por experiência como realizar essa tarefa. Mitchell [44] define que parte do aprendizado consiste em adquirir conhecimento a partir de exemplos específicos apresentados em uma etapa de treinamento. Um exemplo, portanto, é reconhecer dígitos escritos a mão a partir de um conjunto de imagens de dígitos escritos a mão.

O MNIST é um conjunto de dados público em que vários algoritmos são testados e os resultados são até demonstrados publicamente[1]. Ele contém 60,000 imagens de treinamento e 10,000 imagens de teste. Até o presente momento diversas técnicas foram empregadas para resolver esse problema, muitas foram submetidas e estão disponíveis publicamente. Algumas das técnicas iniciais utilizadas por Yan LeCun [34] foram baseadas em aprendizagem de máquina: classificadores lineares, com técnicas de *deskewing*, classificadores não-lineares, redes neurais e, até mesmo, redes neurais convolucionais.

Técnicas mais refinadas de visão computacional foram empregadas neste trabalho para resolver problemas da natureza de classificação de imagens, com o reconhecimento e a detecção de objetos de ambientes internos, como sofás, cadeiras, portas, ou escadas.

[1] http://yann.lecun.com/exdb/mnist/

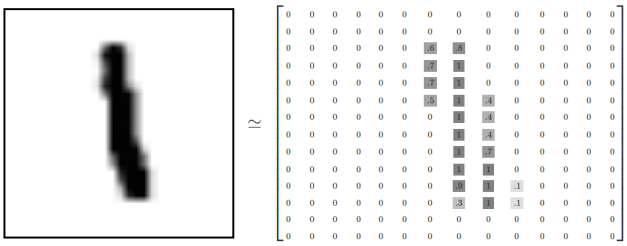


Figura 2.1 – Representação do MNIST.

Dessa forma este capítulo está organizado da seguinte maneira: a Seção 2.1 apresenta a diferença entre os problemas de detecção e reconhecimento de objetos; a Seção 2.2 aborda os métodos do estado da arte para detecção e reconhecimento de objetos; e a Seção 2.3 descreve as formas de avaliação destes métodos.

2.1 Reconhecimento e Detecção de Objetos

Para a identificação de objetos em imagens, existem três problemas distintos: reconhecimento de objetos, que consiste em classificar (ou simplesmente dizer) quais objetos estão presentes na imagem; localização de objetos, que consiste em encontrar as coordenadas de **um** objeto, possibilitando a construção de um *bounding box* sobre o mesmo; e a detecção de objetos, que é semelhante a localização, porém pode encontrar **múltiplos objetos**, e **múltiplas classes** de objetos em uma mesma imagem [18].

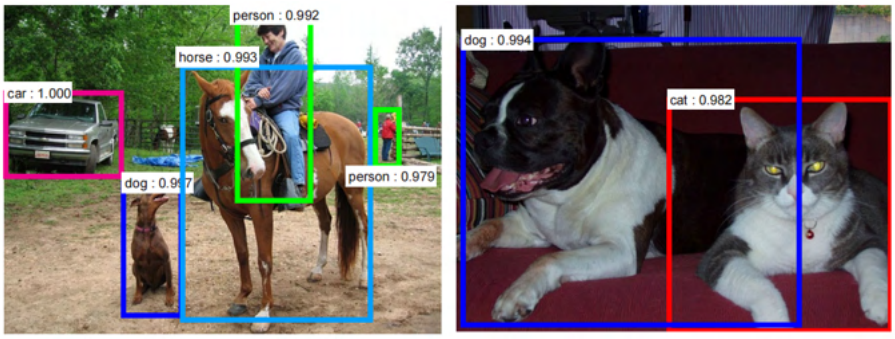
Outro fator importante que ajuda no entendimento de imagens e no reconhecimento de objetos, é a definição de um contexto. Dessa forma, o reconhecimento de cenas pode trazer uma contribuição importante [76]. O reconhecimento de cenas é responsável por classificar qual cena a imagem pertence, como por exemplo: corredores; elevadores; veículos (trens, aviões); entre outros.

A Figura 2.2 mostra um exemplo de reconhecimento de objetos, detecção de objetos, e reconhecimento de cenas. Percebe-se que através do reconhecimento de objetos e cenas dá para saber quais são os possíveis rótulos de uma determinada imagem. Já a detecção de objetos é capaz de rotular, e encontrar a região de cada objeto na imagem.

Reconhecer objetos distintos em uma imagem pode ser considerada uma tarefa complexa, tendo em vista que muitas vezes um mesmo tipo objeto possui cores e tamanhos diferentes. Segundo Vidal e Ullman [69], para fazer o reconhecimento de imagens devem ser realizados dois estágios: primeiro, o objeto a ser classificado deve ser representado por características que são extraídas da imagem. Depois, um classificador é aplicado nas características para que haja uma decisão de qual classe a imagem pertence.

Os primeiros algoritmos utilizados para a classificação de imagens incluem o Perceptron e os algoritmos de Winnow [38]. Posteriormente, outros algoritmos também foram

(a) Reconhecimento de Objetos



(b) Detecção de Objetos

Predictions:

- **Type of environment:** indoor
- **Scene categories:** entrance_hall (0.525), lobby (0.174)
- **Scene attributes:** enclosed area, no horizon, man-made, indoor lighting, glossy, socializing, cloth, congregating, touring

(c) Reconhecimento de Cenas

Figura 2.2 – Diferenças entre, reconhecimento de objetos, detecção de objeto,s e reconhecimento de cenas.

utilizados, tal como o SVM (*Support Vector Machine*) [69]. Além destes, outros tipos de redes neurais também foram bastante utilizados, como perceptrons de multi camadas, ou redes neurais convolucionais [33].

Entretanto, com o passar do tempo, novas representações de características foram apresentadas, tais como métodos que lidam com espaços de interesse nas imagens. O agrupamento desses espaços são chamados de descritores de características. Vidal e Ullman [69] afirmam que um bom descritor deve ter a habilidade de lidar com intensidade, rotação, escala e outras variações. Como exemplo de um conhecido descritor de características, pode-se citar o SIFT (*Scale-invariant feature transform*), que converte cada espaço na imagem em um vetor de 128 dimensões, em que cada imagem é representada como uma coleção desses vetores.

Uma variação do SIFT é o SURF (*Speeded-Up Robust Features*), cujo algoritmo consiste em: selecionar pontos de interesse em locais distintos da imagem, como cantos, ou aglomeração de pixels semelhantes; na sequência, a região vizinha de cada ponto de interesse é representada por um vetor de características; no final os vetores de características são comparados entre diferentes imagens, e para isso é utilizada uma medida de distância entre vetores, como distância Euclidiana ou de Mahalanobis, possibilitando, assim, classificar quais espaços nas imagens tem mais pontos em comum [4].

Além disso há outras representações, como o BoW (*Bag of Words*), representado na Figura 2.3. O BoW é um algoritmo que simplesmente computa a distribuição (histograma) de palavras visuais encontradas na imagem. As palavras visuais podem ser extraídas a partir dos espaços chave da imagem. A computação dos histogramas resultantes é utilizada para treinar um algoritmo de aprendizado de máquina que ficará responsável pela classificação da imagem.

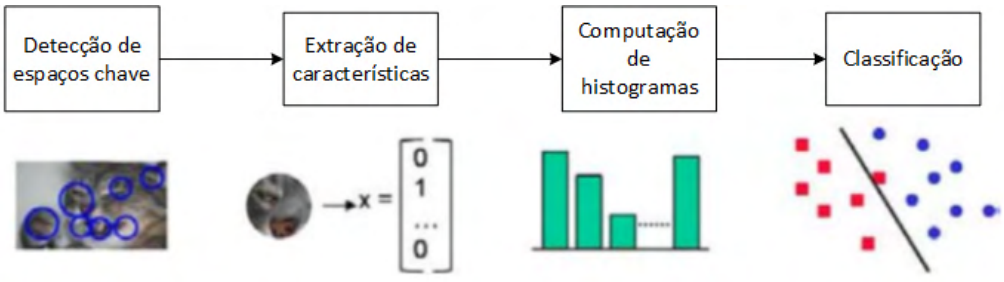


Figura 2.3 – Um processamento de pipeline típico de um reconhecimento de imagens utilizando técnicas de *bag of words* [65].

O treinamento e avaliação de um classificador de imagens pode ser feito através de diferentes estratégias. Uma estratégia comum consiste em separar o conjunto de imagens em duas partes: imagens para treinamento e imagens para teste. Além disso, para os casos de Redes Neurais, em alguns casos também são utilizados conjuntos de imagens para validação. Neste caso, por exemplo, o treinamento pode ser feito com um conjunto de

imagens para treinamento, e no fim de cada época de treinamento, o algoritmo é testado em um conjunto de imagens para validação. Esse tipo de técnica permite que avaliar o quão bem está sendo feito o treinamento, e também definir um critério de parada no treinamento.

No caso, o algoritmo analisa as imagens de treinamento para aprender suas características, e depois verifica se consegue classificar corretamente as imagens de validação. A precisão do algoritmo no conjunto de validação deve então ser utilizada como parâmetro para saber o quanto ele aprendeu, caso seja um valor não desejável, o algoritmo deve ser refinado, e o processo de treinamento refeito.

Somente depois desta etapa o algoritmo é testado com o conjunto de imagens de teste. É importante para a integridade do método, que o algoritmo não utilize as imagens de teste até o final do período de treinamento. Esse processo de encontrar um resultado adequado varia de acordo com os métodos, ou algoritmos utilizados tanto nos métodos de extração de características, quanto nos classificadores [44].

Ou seja, para refinar os métodos escolhidos para um determinado conjunto de imagens, são testadas várias formas de extração de características, combinadas com técnicas de otimização de hiper-parâmetros dos classificadores. A otimização de hiper-parâmetros nos classificadores inclui desde buscas em grade, ou aleatórias, até métodos mais refinados, como otimizações baseadas em gradiente descendente. Algoritmos não-lineares como redes neurais também utilizam métodos baseados em gradiente descendente [10].

Um exemplo de aplicação de reconhecimento de objetos é o desafio proposto por Russakovsky et al. [54], nomeado de *Imagenet Large Scale Visual Recognition Challenge*, que consiste no reconhecimento de 1000 objetos distintos em um conjunto de dados com mais de 10 milhões de imagens. Os primeiros vencedores do desafio no ano de 2010 utilizaram, para a extração de características, os descritores HoG (*Histogram of Oriented Gradients*) [12] e LBP (*Local Binary Patterns*) [1] combinados com classificadores SVM de larga escala, uma vez que eram capazes de classificar e treinar com a imensa quantidade de imagens do conjunto. As técnicas usadas nos anos subsequentes incluem o uso de redes neurais convolucionais, que são capazes de aprender as características a partir dos dados de treinamento.

Outro projeto que disponibiliza vários conjuntos de imagens é o Pascal VOC[2], que contém diversas competições no contexto de visão computacional, que incluem classificação, detecção e segmentação de imagens. Para tarefas de classificação as anotações consistem em apenas identificar a qual classe cada imagem pertence. Para tarefas de segmentação ou detecção são anotadas as posições e classes de cada objeto em uma imagem. A Figura 2.4 mostra como é feita uma anotação do conjunto de detecção de objetos do Pascal VOC. Os formatos dessas anotações podem variar por tipo de conjunto de dados, como a forma de representação dos objetos, os tipos de arquivos ou diretórios utilizados, a quantidade de informações extras, como variáveis que podem representar se o objeto está

[2]http://host.robots.ox.ac.uk/pascal/VOC/

com oclusão, entre outras. No caso do conjunto de anotações PASCAL VOC, ele é dado por uma série de arquivos em **xml** contendo as coordenadas das posições x, y, e x_1, y_1, os pontos iniciais e finais do retângulo dos objetos, e nas versões mais recentes foi adicionado a informação de **dificuldade**, que é usado para representar objetos com oclusão, ou distantes.

Figura 2.4 – Exemplo de uma anotação do Pascal VOC.

A detecção de objetos ainda é mais complexa que o reconhecimento, uma vez que inclui o problema de localização junto ao de reconhecimento [46]. Shantaiya et al. [55] conduziram um estudo sobre o estado da arte das técnicas de detecção de objetos, categorizando-as em quatro tipos: baseadas em característica; baseadas em *template*; baseadas em classificadores e baseadas em deslocamento.

Uma visão geral do estudo é apresentada na Tabela 2.1. Como forma de complementar este estudo, incluímos a linha 9 na tabela, que corresponde a um novo tipo de técnicas de detecção de objetos: abordagens baseadas em redes neurais profundas.

Para as técnicas baseadas em características, a padronização das características das imagens é importante. Uma ou mais características dos objetos de interesse são modeladas e extraídas através de algoritmos de processamento de imagens. Características variam de tamanho, formato ou cor dos objetos. Exemplos desse tipo de abordagem são: descritores HoG, modelos baseados em camadas ocultas de Markov (*Hidden Markov Models*), entre outros [55].

Para as abordagens baseadas em *template*, se há algum *template* capaz de descrever um objeto específico, o problema é resolvido através do processo de encontrar características entre o *template* e a imagem. Já as abordagens baseadas em deslocamento utilizam as informações do movimento para a detecções em tempo real.

Por fim, as abordagens baseadas em classificadores utilizam técnicas de aprendizagem de máquina, treinando as imagens com um conjunto específico de características e considerando uma classificação binária (objeto e fundo). Então, um algoritmo baseado

em abordagens como *sliding window*, utiliza esse classificador eliminando as detecções desnecessárias através de um algoritmo de NMS (*Non-Maxima Supression*).

Modelar um conjunto de características de alto nível para a detecção de objetos é uma tarefa trabalhosa. Ao longo dos anos, muitas abordagens foram propostas, tais como a abordagem de Viola e Jones [70], SIFT [40] e SURF [4].

Dessa forma, as técnicas de detecção baseadas em redes neurais profundas são similares às de detecção baseadas em classificadores, porém diferem pelo fato de redes neurais aprenderem características conforme a quantidade de camadas empregadas, apesar do alto custo computacional. Este alto custo dos classificadores baseados em redes neurais profundas é devido a quantidade de camadas da rede, que define a quantidade de operações matemáticas a serem realizadas.

Para esse caso de detecção existem abordagens que combinam redes neurais convolucionais como classificadores, com métodos como de *sliding window*. Novas abordagens que trabalham sobre o problema do alto custo computacional, utilizando arquiteturas específicas de detecção, chamadas de *Region proposal Networks* (RPN) [16], abordagens que usam as redes convolucionais como um extrator de características [16], e métodos que tratam a detecção como um problema de regressão [50].

Abordagem	Métodos Utilizados
Baseado em formas (Baseado em Características)	PCA-HOG, Modelos híbridos de camadas ocultas de Markov (HMM), Filtros gaussianos, Modelos de contornos ativos de Geodesic Bootstraping, Baseados em formas de contornos ativos
Baseado em cores (Baseado em características)	HMM, algoritmo de GHOSP Histograma de cores (HC) e Histogramas de gradientes orientados (HOG), Filtros de Kalman (SIFT), e outros
Modelo fixo	Content-adaptive progressive occlusion analysis (CAPOA), Local best match authentication (LBMA), filtros de Kalman
Baseado em modelos deformáveis	Prototype-based template, Adaptive tracking, Modelo hibrido
Baseado em classificadores	Redes neurais multi layer perceptron, Baseados em filtros de Bayes, Baseado em características de Harr, baseados em algoritmos de busca PSO, SVM's e sliding windows
Diferenciação de Frames (Baseados em movimento)	HMM, filtros Bayesianos Algoritmo de Subtração de Fundos, e de Eliminação de fundos
Optical-flow (Baseado em movimento)	SIFT / HoG, Estimativas de Optical Flow, Optical-Flow com base em classificação por SVM
Gaussian Mixture (Baseado em movimento)	Mistura de Gaussianas, Filtroo de Kalman
Baseados em redes neurais profundas	Redes neurais convolucionais, Redes neurais de de Region proposal, redes neurais convolucionais como um extrator de características, e métodos que utilizam regressão

Tabela 2.1 – Análise de abordagens de detecção de objetos [55].

2.2 Redes Neurais Convolucionais

Conforme demonstrado na seção anterior, um dos problemas principais de classificação de imagens é encontrar um conjunto de características de alto nível que se adaptem ao problema. A dificuldade é que o conteúdo das imagens pode estar associado a diferentes contextos, tais como objetos, faces, textos, entre outros, e, assim, possuir diferentes tipos de propriedades, como formas, cores e bordas. Recentemente, com os avanços em redes neurais e o surgimento, com as redes neurais convolucionais, de redes com muitas camadas de profundidade, esse problema é parcialmente resolvido, pois as redes neurais

convolucionais aprendem as características mais relevantes das imagens a partir dos dados de treinamento [53].

Existem diversos tipos de camadas de redes neurais e conjuntos de operações. As redes neurais com diversas camadas convolucionais são chamadas de redes neurais convolucionais profundas. Existem outros tipos, como as redes neurais recorrentes, redes neurais completamente conectadas, entre outras [18, 53]. Uma rede neural convolucional, como mostra a Figura 2.5, é estruturada com vários estágios. Nas camadas iniciais são feitas operações convolucionais (aplicação de um conjunto filtros, ou *kernels* [65]), e em alguns casos de *pooling* (redução do tamanho das matrizes resultantes da convolução [18]). No final tem as camadas de classificação, com funções de ativação, as comumente usadas são: softmax, e sigmoid [18, 53].

Nas camadas iniciais, são aplicados diferentes filtros, sendo que cada operação de convolução combina um espaço de *pixels* em outro de menor dimensionalidade, conforme mostra a Figura 2.6[53, 6, 68]. Assim, cada camada subsequente combina os resultados da camada anterior, como mostra a Figura 2.5, em que diversos filtros são aplicados e os resultados são combinados em cada estágio.

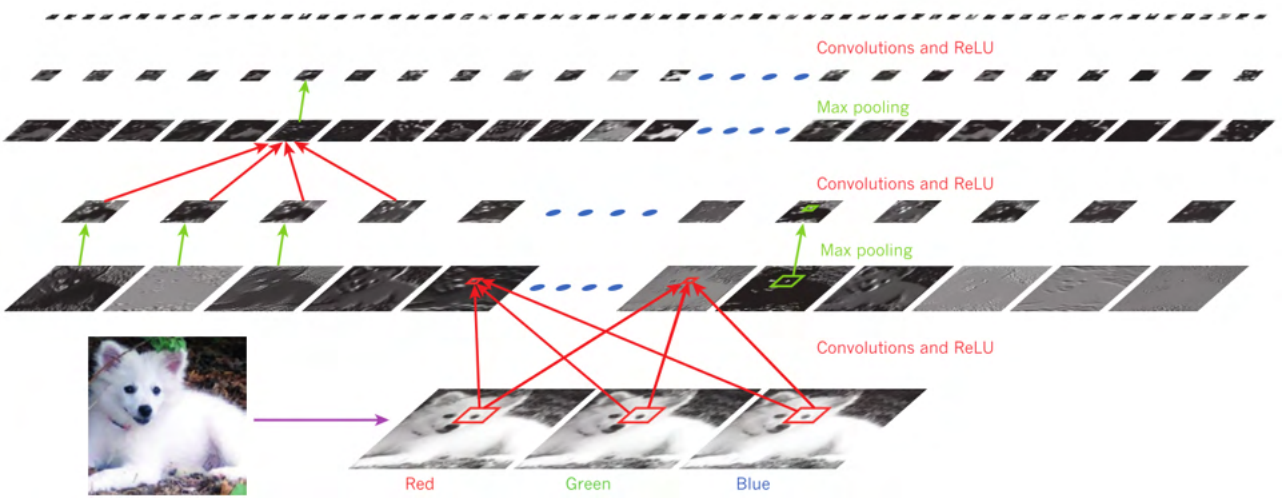


Figura 2.5 – Uma rede convolucional profunda [53].

As matrizes resultantes da camada convolucional passam por um cálculo de não linearidade, como, por exemplo, o **ReLU** (*Rectified Linear Units*), que é uma função de ativação que aplica a função $f(x) = max(0, x)$ em cada elemento do resultado da convolução [53, 6, 68]. Já a camada de *pooling* é responsável por reduzir o tamanho do tensor da camada anterior, concatenando-a em um espaço menor através de operações como $max(x)$ (*max pooling*), entre outras, conforme ilustra a Figura 2.7.

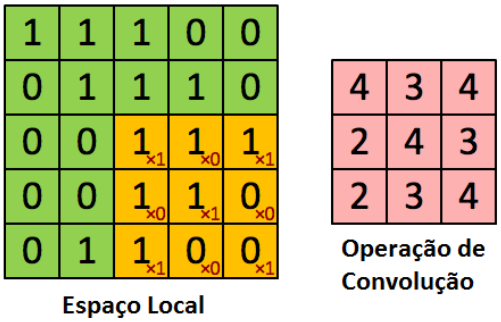


Figura 2.6 – Operação de uma camada de convolução[6].

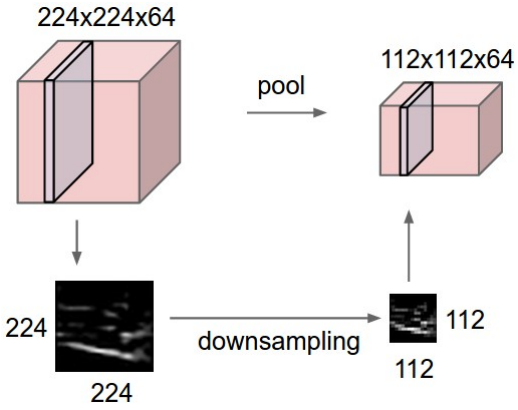


Figura 2.7 – Operação de uma camada de *pooling*[68].

Por fim, as camadas finais da rede consistem em camadas completamente conectadas (*fully connected*), comumente nomeadas de camadas do topo (*top layers*), são as mais influenciadas diretamente pelos dados de saída. A camada de saída fica logo após, e consiste em uma camada com ativação correspondente que deverá gerar os valores probabilísticos em casos de tarefas de classificação, ou demais valores para camadas de regressão [18, 53].

Em classificação as funções de ativação variam de acordo com o problema, ou a quantidade de classes. Em caso de classificação de multi classes, a função de ativação mais recomendada é a *softmax*, que consiste em reduzir os vetores de características vindos da camada posterior a um vetor de probabilidades [18].

Nas fases de treinamento, a rede aprende a extrair cada caraterística com base nos dados utilizados. Por exemplo, uma rede de classificação de portas pode aprender como distinguir formas retangulares e linhas, além de texturas de madeira ou vidro. Na fase de classificação a entrada irá ser processada em cada uma dessas camadas, portanto esse processo terá sua performance dependente do tamanho da rede neural [53].

O treinamento é feito com um algoritmo comum de *backpropagation*. Ele pode ser combinado com uma função de otimização, como um gradiente descendente estocástico (SGD), que é responsável por minimizar a taxa de erro do resultado da rede neural em

relação aos dados de treinamento. O valor da taxa de erro, é dado por uma função de custo, tal como erro quadrado médio (MSE) [18].

Além disto, é possível aprimorar os pesos de um modelo de rede já treinado, utilizando esses pesos para outras tarefas. Esse método de treinamento é chamado de *fine tunning* e também é feito utilizando o *backpropagation*. Também é possível modificar as camadas de saída para outra tarefa de classificação sem modificar os pesos atuais do modelo, essa tarefa é chamada de *transfer learning* [18].

Porém, conforme aumenta a profundidade das redes surge um problema chamado de *vanishing gradients*, que ocorre devido a perda do sinal gerado na propagação desse sinal da camada final até as camadas iniciais da rede. As redes residuais (ResNet) [21], resolvem o problema de treinamento de redes mais profundas através de um bloco chamado de módulo residual. O módulo residual, conforme ilustra a Figura 2.8, dita que ao invés de tentar aprender um mapeamento de x até $H(x)$, é melhor aprender a diferença entre os dois. Então, para calcular o $H(x)$, só adiciona-se o restante à entrada. Se o restante é $F(x) = H(x) - x$, ao invés de aprender $H(x)$ diretamente, a rede tentará aprender $F(x) + x$.

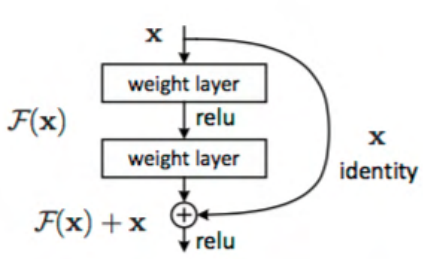


Figura 2.8 – Módulo Residual [21].

Outro exemplo de uma rede neural convolucional é a Inception (ou GoogLeNet), desenvolvida por Szegedy et al. [64]. que é uma rede profunda que possui 22 camadas. A arquitetura da rede proposta está representada na Figura 2.9, na qual cada retângulo corresponde a uma camada de convolução ou *pooling*, e os octógonos são as camadas de entrada e saída da rede.

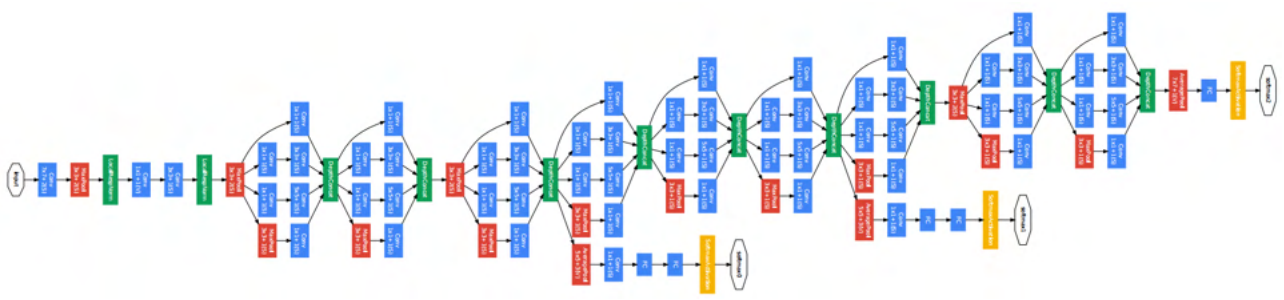


Figura 2.9 – Arquitetura da GoogLeNet [64].

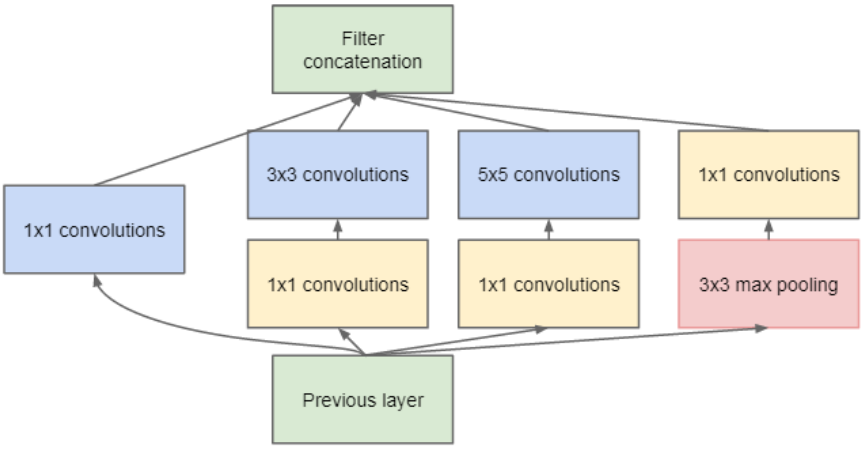


Figura 2.10 – Modulo Inception [64].

A ResNet resolve o problema de profundidade, já a Inception propõe que é possível também aumentar a largura da rede. Para isso a maior contribuição da Inception foi o "módulo Inception" conforme ilustra a Figura 2.10. As redes neurais profundas são computacionalmente custosas, com a ideia de diminuir este custo, os autores da Inception limitaram o número de canais de entrada adicionando convoluções menores (1x1), antes das convoluções maiores (3x3, e 5x5).

A Inception teve melhorias ao longo do tempo. As versões 2 e 3 apresentadas por Szegedy et al. [63] apresentaram: melhorias nos módulos de convoluções Inception com um canal maior de convolução (5x5) remodelado, diminuindo o custo computacional; mudanças na função de otimização para RMSProp; melhores normalizações em lote; e novas regularizações na função de custo para prevenções de *overfitting*.

A InceptionResNet [62] inspirou-se no módulo residual para combiná-lo com o módulo Inception. Isso é feito através da adição da conexão residual na operação de saída do módulo Inception. Para isso foram feitas as seguintes alterações no modelo Inception: são adicionadas após as camadas de convoluções originais, novas de de tamanho 1x1 para adequar-se aos tamanhos da ResNet; as operações de *pooling* do módulo Inception são substituidos pelas conexões residuais; e foi alterado os valores das ativações residuais.

Similar a Inception também existe a Xception [8], porém com pequenas diferenças: a ordem das convoluções muda para primeiro as convoluções maiores e depois as convoluções menores; a ausência de não-linearidade, pois no módulo Inception original há não-linearidade após a primeira operação e na modificação não há operações não-lineares (ReLU).

Além disso, também existe a Mobilenet [24] que utiliza o conceito de *depthwise separable convolutions*, que reduz drasticamente o custo computacional, possibilitando assim uma arquitetura que seja capaz de ser executada em dispositivos móveis, tais como *smartphones*.

Para cada tarefa distinta uma arquitetura diferente de rede neural profunda é utilizada, com um conjunto distinto de camadas, sendo que a profundidade da rede é dada pela quantidade de camadas que esta possui. No contexto de detecção de objetos utilizando redes profundas, os métodos iniciais utilizavam técnicas de *selective search*, ou *sliding window*.

Por exemplo, na proposta de Girshick et al. [17], nomeada de R-CNN, o método possui três passos: buscar na imagem por possíveis objetos utilizando *selective search*, gerando uma série de propostas de regiões (*region proposals*); executar uma rede neural convolucional em cada uma dessas regiões; executar a saída dessa rede neural em um classificador, como um SVM, para classificar a região, e depois em um algoritmo de regressão linear esticar a região do *bounding box* caso o objeto realmente exista.

Esse tipo de método é lento e de alto custo computacional, pois classificar várias regiões da imagem com uma rede neural convolucional de muitas camadas irá gerar muitas operações computacionalmente pesadas. Então, os autores atualizaram o método extraindo as características da imagem antes de gerar as regiões de interesse, dessa forma a CNN executará apenas uma vez sobre a imagem inteira. Além disso, substituíram o classificador SVM por uma camada de classificação com a função de *softmax*[16].

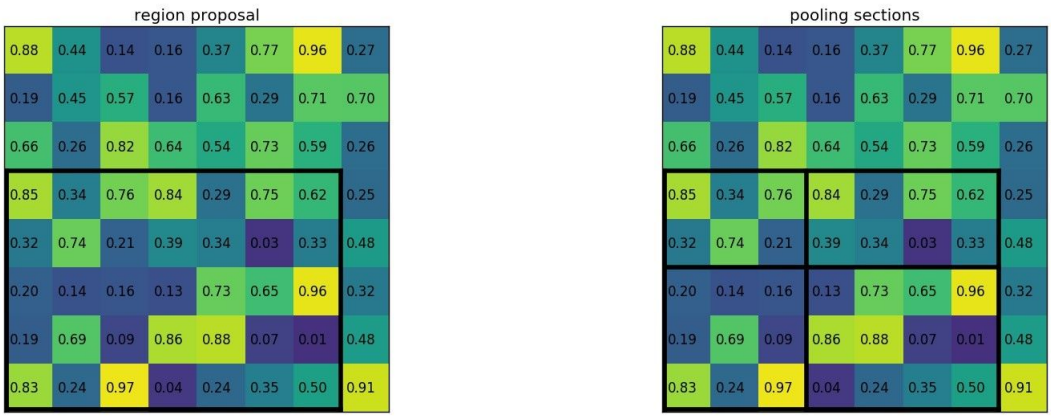
O método também incluiu novas camadas chamadas de RoI (*Region of Interest* ou região de interesse) *pooling*, que tem como entrada: um mapa de características vindo de uma rede convolucional; e uma matriz de $Nx5$ representando a lista das regiões de interesse, em que as primeiras colunas representam o índice da imagem, e as últimas colunas as coordenadas da região.

A partir disso, para cada RoI ele pegará a seção correspondente do mapa de características e redimensionará para um tamanho predefinido, pegando apenas a seção de maior interesse. O resultado é que com uma lista de retângulos de diferentes tamanhos, é possível pegar uma lista de mapas de características de tamanho fixo menor, com apenas os valores maiores no mapa de características [16].

A Figura 2.11 mostra um exemplo, em que no mapa de característica e na região proposta, são geradas as seções de *pooling*, de tamanhos distintos. Então a operação irá resultar apenas em quatro valores, a partir da operação *max* em cada região[3].

Por fim, os autores melhoraram o método, trocando o algoritmo baseado em busca seletiva por uma rede neural capaz de propor as regiões de interesse, nomeada de RPN (*region proposal network*). O método chamado de Faster R-CNN, substitui a última camada de uma CNN, com um método de *sliding window*, sobre os mapas de características, e os mapeia em uma dimensão menor. Para cada localização da janela, ele irá gerar múltiplas regiões possíveis baseado em um hiper-parâmetro. Cada região possível consiste em

[3]https://deepsense.ai/region-of-interest-pooling-explained/



(a) Mapa de características, e região proposta

(b) Seções de pooling

(c) Saída da operação de RoI Pooling

Figura 2.11 – Exemplo de uma Seção de Pooling em um mapa de Regiões de Interesse.

um mapa de probabilidades dos objetos, e as quatro coordenadas do objeto. Em outras palavras o método é centrado em uma própria camada da rede [51].

Esses métodos, embora capazes de detectar objetos, ainda não são capazes de fazer a detecção em tempo real. A detecção em tempo real[4], feito através do *feed* da câmera, tem como preocupação principal não apenas detectar de forma precisa, mas também com uma velocidade média de processamento alta, possibilitando várias detecções em cada *frame*, e em vários *frames* por segundo.

Dessa forma, as abordagens recentes, como a YOLO (*You Look Only Once [50]*), ou a SSD (*Single Shot MultiBox Detector*) [39], que são métodos para detecção de objetos em tempo real, utilizam pipelines diferentes, com redes neurais convolucionais de menor profundidade. A Figura 2.13 compara a abordagem de geração de regiões candidatas da YOLO [50], com a Overfeat [56], que utilizava métodos convencionais de *sliding window* com uma rede convolucional profunda.

A inovação proposta pelo método YOLO é transformar o problema de detecção em um problema de regressão. Assim, é proposto um pipeline direto, capaz de produzir as coordenadas das regiões dos objetos de interesse e as probabilidades dos objetos com apenas uma CNN. A Figura 2.12 ilustra esse pipeline, onde as saídas são distribuídas em

[4]https://tinyurl.com/yolossd

forma de uma grade. As células da grade contêm um conjunto de valores das regiões de interesse e as probabilidades dos objetos. Por fim, um filtro com NMS é aplicado deixando apenas as regiões com maior probabilidade dos objetos de interesse.

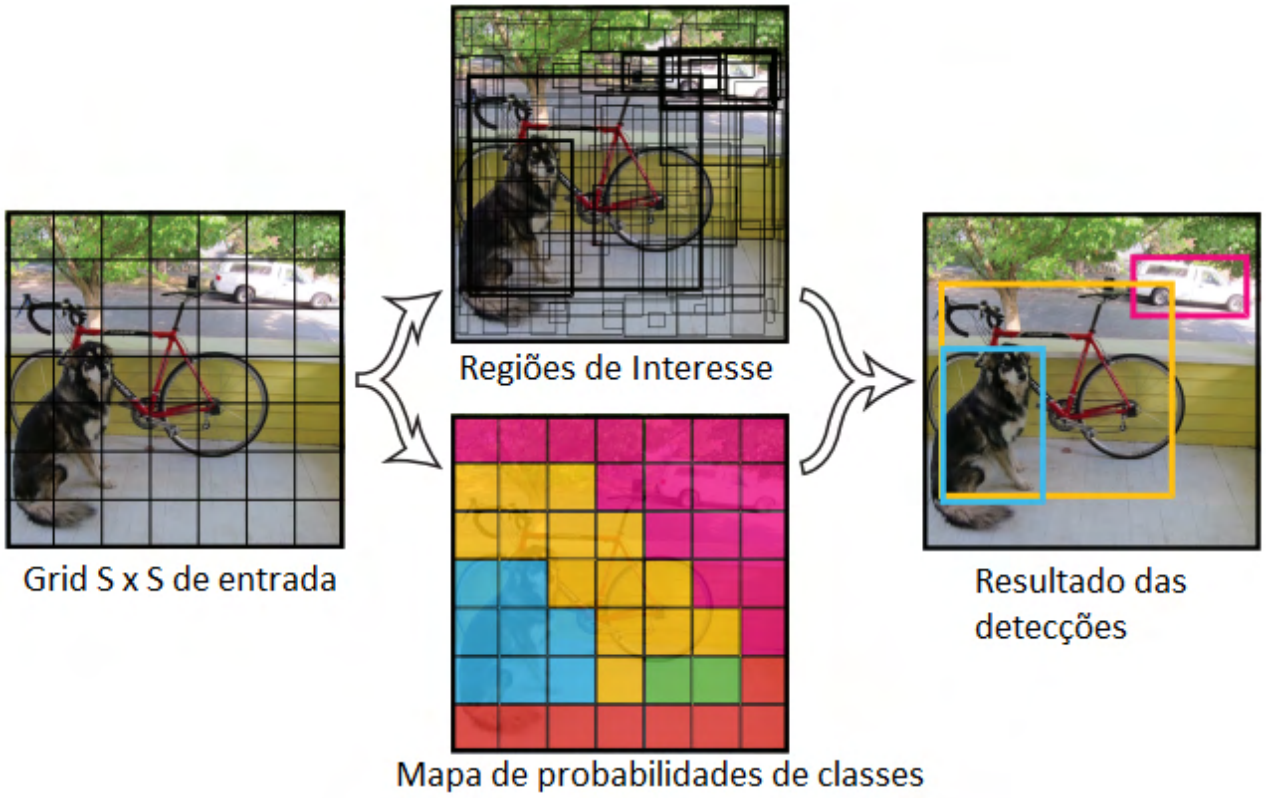


Figura 2.12 – Pipeline de detecção do método YOLO [50].

2.3 Métricas de Avaliação

Para a avaliação de qualquer um dos métodos abordados neste capítulo, são utilizadas diversas métricas. As mais comuns para a avaliação de um classificador são a acurácia, que calcula quantas vezes o classificador acertou, e a taxa de erro, que conta basicamente as vezes que o classificador não acertou. A acurácia é medida com x/t, sendo x o número de acertos, e t o número total de dados sendo avaliados. Para muitas classes, como no Imagenet [54], também são computadas as taxas de erro em top-N, ou seja, caso a classe correspondente está presente nos N (5, 3) itens de maior probabilidade na predição.

Outras métricas comuns são *precision* e *recall*, e a curva *F1 score*. *Precision* é similar a acurácia, medindo a porcentagem das predições corretas, ou positivas, como mostrado na Equação 2.1, onde VP equivale aos verdadeiros positivos, FP aos falso positivos e FN aos falsos negativos.

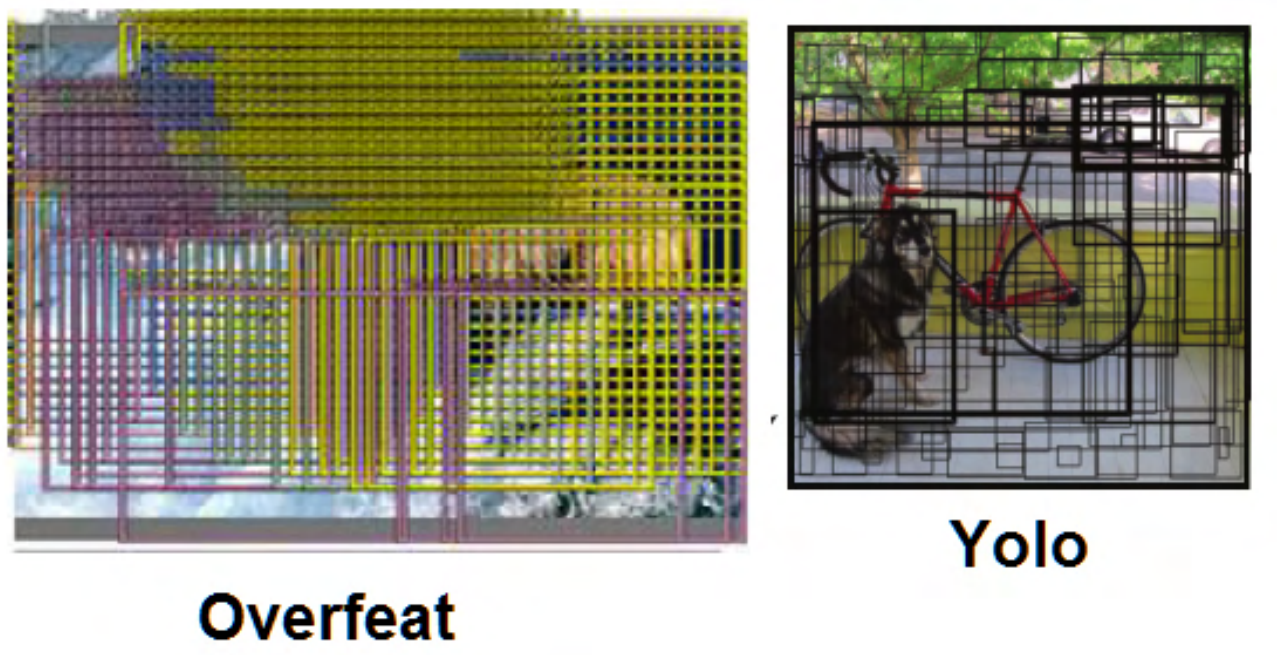


Figura 2.13 – Yolo comparado a Overfeat em relação a geração de geração de regiões candidatas antes da aplicação do NMS [50, 56].

$$Precision = \frac{VP}{VP + FP} \tag{2.1}$$

Recall, dado pela Equação 2.2, mede o quão bem são encontrados os positivos.

$$Recall = \frac{VP}{VP + FN} \tag{2.2}$$

E o *F1 score* dado pela Equação 2.3 considera tanto a *precision*, quanto a *recall*.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

Existem também métricas mais completas, e complexas, para cada problema. Para a detecção de objetos a métrica mais comum é o mAP (*mean average precision*). Para calcular o mAP em detecção de objetos, pode-se calcular a intersecção sobre a união, ou *Intersection over Union* (IoU), que auxilia medir o quão bem foi feita cada detecção.

IoU, conforme mostra a Figura 2.14, mede o quanto duas regiões se sobrepõem. Isso é importante para saber o quão bem o detector de objetos foi capaz de prever a localização do objeto em relação a posição de anotação. Existe uma padronização de que define o limiar para o IoU. O *Pascal VOC* [14] define um limiar de 0.5, ou seja, caso o IoU seja maior que 0.5 ele será considerado um VP, caso o contrário um FP.

Tendo os possíveis FP e VP, podemos, então, calcular os valores de *precision* e *recall* das detecções. A partir disso podemos calcular a precisão média (*average precision*),

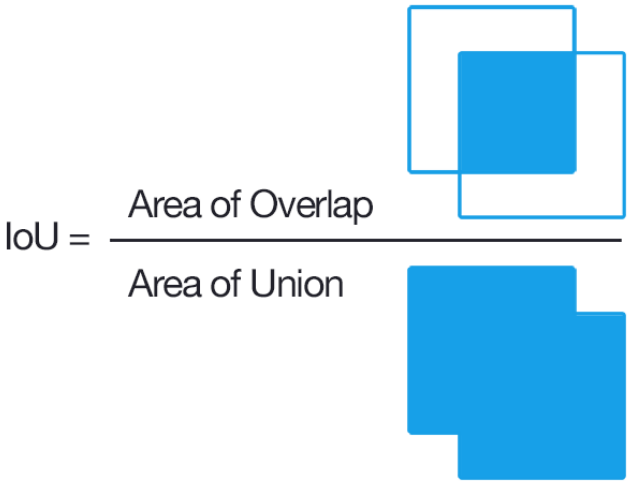


Figura 2.14 – *Intersection over Union*.

ou AP. Porém, inicialmente precisamos calcular o valor de *precision* interpolado $P_{interp^{(r)}}$, conforme mostra a Equação 2.4, na qual $p\tilde{r}$ é *precision* calculado no *recall* $\tilde{r}$. O valor de AP é dado pela equação 2.5.

$$P_{interp^{(r)}} = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{2.4}$$

$$AP = \frac{1}{11} \sum_{rE(0.0,\ldots,1.0)} P_{interp^{(r)}} \tag{2.5}$$

Para a equação de AP são escolhidos 11 limiares de confiança diferentes, tal que o *recall* desses valores varie de 0 a 1: $0.1, 0.2, ..., 0.9, 1.0$, então o AP é definido como a média dos valores de *precision* com estes 11 níveis de *recall*. Já o mAP é a média de todos os valores médios de *precision* sobre **todas** as classes já medidas, ou seja, é como uma média (*mean*) do AP calculado para todas as classes [25, 14, 37].

3. TRABALHOS RELACIONADOS

Utilizamos uma revisão sistemática como método de pesquisa de trabalhos relacionados, voltado para à detecção de escadas, portas e objetos. Dessa forma, esta Seção está organizada da seguinte maneira: a Seção 3.1 expõe a metodologia utilizada no processo de pesquisa; e a Seção 3.2 mostra os resultados e conclusões da pesquisa realizada.

3.1 Método de Pesquisa

As perguntas de pesquisa foram feitas considerando o objetivo de identificar e avaliar os trabalhos disponíveis no contexto de detecção de portas, escadas, e objetos, que podem ser utilizados para auxiliar deficientes visuais na navegação em ambientes internos.

Portanto, as seguintes perguntas de pesquisa foram formuladas:

1. RQ1: Quais são os métodos disponíveis de detecção, que incluem objetos de ambientes internos (portas, e escadas), e podem auxiliar os deficientes visuais?
2. RQ2: Quais são os melhores métodos de detecção de objetos disponíveis na literatura?
3. RQ3: Para quais objetos ainda há espaço para melhorias nas técnicas de detecção disponíveis?

As questões foram formuladas considerando que os métodos atuais de detecção de objetos utilizam conjuntos de dados específicos, que nem sempre contêm os objetos de interesse para o auxílio na navegação de ambientes internos. Além disso, a comparação entre os métodos de detecção de objetos pode mostrar quais deles poderiam ser utilizados no contexto de auxílio a navegação de deficientes visuais. E por fim, quais espaços podem ser preenchidos nessa área.

O processo de pesquisa foi separado nas seguintes etapas:

1. Identificar as potenciais fontes de consulta;
2. Filtrar os dados relevantes;
3. Selecionar os trabalhos;
4. Revisar os trabalhos;
5. Coletar e analisar dados.

Iniciamos o processo de pesquisa com uma busca por palavra chave, de *object detection*, no Google que levou a um repositório[1] que contém vários artigos na área de *deep learning* e visão computacional, incluindo os 15 trabalhos mais relevantes na área de detecção de objetos. Depois disso foi feita uma busca com o uso das palavras chave, *door detection* e *stairs detection*, na base de dados da IEEE[2].

Após a primeira busca, constatou-se que a maioria dos trabalhos relevantes da área de detecção de objetos, utilizam um conjunto de dados de desafio da área para a avaliação de seus resultados, como o Microsoft COCO [37], ou o Pascal VOC [14]. Os demais trabalhos, relacionados a portas, e escadas avaliaram de forma diferente seus resultados. Dessa foi feito um refinamento na *string* de busca para incluir os conjuntos de dados desejados, o publico alvo desejado, e o ambiente desejado, para que a seleção de trabalhos filtre os trabalhos mais relevantes de acordo com os critérios de inclusão e exclusão.

3.1.1 Critérios de Inclusão e Exclusão

Considerando o foco e as perguntas de pesquisa, foram definidos cinco temas de pesquisa para os critérios de inclusão e exclusão, conforme a Tabela 3.1: detecção de objetos, escadas e portas, incluindo conjuntos de dados usados para a avaliação (PASCAL VOC, Microsoft COCO, ou SUN Database); e por fim como a pesquisa tem foco em auxiliar deficientes visuais, foi incluso: *visually impaired* e *indoor environments*.

Exemplos de tópicos excluídos pelo item de detecção de objetos são tarefas relacionadas, como: detecção de objetos em imagens de alta resolução, que acabam utilizando outras técnicas de avaliação e detecção; detecção de objetos em contexto, que também utilizam outras técnicas de avaliação, e detecção, técnicas de detecção que auxiliam deficientes visuais que apenas usam sensores, e outros. Além disso para as buscas que encontraram o SUN Database, foram removidos os trabalhos que o utilizam apenas para o reconhecimento de cenas.

Temas	Tópicos Relevantes	Tópicos Não Relevantes
Detecção de Objetos	Técnicas de detecção de objetos, com CNN, ou outras	Detecção de objetos em contexto, ou segmentação, ou métodos antigos não baseados em CNN
Conjuntos de dados de avaliação	**Avaliação com métricas de válidas, como mAP, ou acurácia, e utilizando os conjuntos de dados selecionados (PASCAL VOC, e outros)**	**Conjuntos de dados diferentes, e métricas diferentes**
Detecção de escadas	Computer vision, Related work, or comparisons, Deep Learning	Reconhecimento de cenas, métodos não baseados em visão computacional
Detecção de portas	Computer vision, Deep Learning	Não baseadas em visão computacional, e outras
Deficientes Visuais	Tecnologias de visão computacional, voltadas para deficientes	Tecnologias voltadas para outras tarefas
Ambientes Internos	Tecnologias e métodos aplicados em ambientes internos	Métodos aplicados em outros ambientes, como smart cars, por exemplo

Tabela 3.1 – Temas do Critério de Exclusão e Inclusão.

[1] https://github.com/kjw0612/awesome-deep-vision
[2] http://ieeexplore.ieee.org

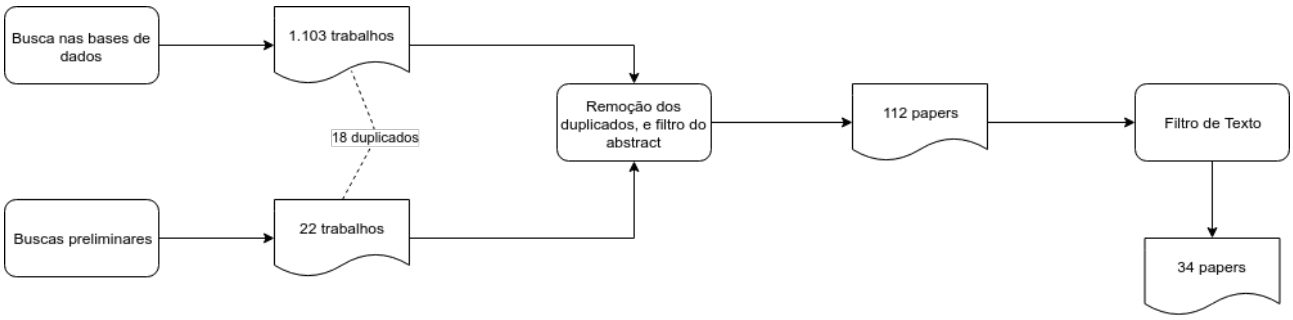


Figura 3.1 – Processo de seleção.

Para os conjuntos de avaliação, os resultados experimentais relevantes em casos de detecção de objetos utilizam ou o Pascal VOC, ou o Microsoft COCO. Além disso, a comparação deve ser feita de forma similar, com a métrica de mAP. Tópicos não relevantes incluem diferentes formas de avaliação, ou conjuntos de dados.

Por fim os itens de detecção de escadas, ou portas, são utilizados para encontrar trabalhos que especificamente realizem esse tipo de tarefa. Isso ajuda a responder as perguntas 2 e 3. Sendo que a maioria das abordagens de detecção no estado da arte encontrados na primeira avaliação não incluem a detecção de portas e escadas, esses itens são relevantes para isso. Os tópicos relevantes para esse estudo, são ligados a técnicas que façam detecção de escadas e portas em ambientes internos, e podem ser uteis para auxiliar a navegação nesses ambientes.

3.1.2 Seleção dos Estudos

A partir dos critérios de inclusão e exclusão, a *string* de pesquisa foi então formulada. O fator principal para a formulação é agrupar as palavras chave principais, com os conjuntos de dados de avaliação, e as palavras chaves do contexto de deficientes visuais. Incluindo então, por exemplo: *object detection*, e *Pascal*, e *deficientes visuais*.

A *string* final de pesquisa foi: **((object detection OR doors detection OR stairs detection) AND ((sun or pascal voc OR coco) OR ((visually impaired) AND indoor environments)))**. A ideia é que os trabalhos podem utilizar as palavras chaves *visually impaired* e *indoor environments*, e também podem fazer a avaliação com algum dos conjuntos de dados selecionados.

Para o processo de seleção cinco bases de dados foram escolhidas. A busca foi feita sobre os meta dados, resultando inicialmente em 1.103 trabalhos, como mostra a Figura 3.1. O critério para o filtro pelo *abstract* foi que estivessem presentes pelo menos os termos: *object detection*, *doors detection*, ou *stairs detection*, e *images* ou *computer vision*.

Depois do filtro, a quantidade de trabalhos resultante foi **112**. A Tabela 3.2 mostra uma visão geral detalhada dos resultados encontrados em cada base de dados. Além disso,

Conjuntos de Dados	URL	Resultados
Elsevier	http://sciencedirect.com	472
IEEExplore	http://ieeexplore.ieee.org/	227
ACM	http://dl.acm.org	13
Scopus	https://www.scopus.com/	391
Total	-	1.103

Tabela 3.2 – Resultados dos Conjuntos de Dados.

18 dos trabalhos principais encontrados na fase inicial foram encontrados nas bases com essa *string* de busca.

O último filtro foi o de texto completo, buscando através de uma leitura vertical detalhes específicos dos métodos no texto, e filtrando trabalhos que citam as frases em seus *abstracts*, mas não contemplam nenhum dos problemas desejados em suas abordagens. O resultado final depois desse filtro foi de **34** trabalhos.

Os estudos dessa revisão são todos trabalhos publicados em diversas fontes, como revistas e anais de eventos. A qualidade de muitos dos trabalhos pode ser mensurada tanto pelo evento, ou revista no qual o trabalho foi publicado, como pelo método de avaliação dos resultados, ou seja, trabalhos com uma má exposição da avaliação dos resultados podem ter uma qualidade duvidosa.

Um exemplos de trabalho com uma boa avaliação de resultados é o de Redmon et al. [50], que além de avaliar em um conjunto de dados conhecido, também mostra comparações dos resultados com outros trabalhos da área, conforme apresenta a Figura 3.2.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Yolo Results Table

Method	*mAP*	**FPS**	**# Boxes**
Faster R-CNN [2](VGG16)	73.2	7	300
Faster R-CNN [2](ZF)	62.1	17	300
YOLO [5]	63.4	45	98
Fast YOLO [5]	52.7	155	98
SSD300	72.1	58	7308
SSD500	**75.1**	23	20097

SSD Results Table

Figura 3.2 – Comparação entre Yolo [50] e SSD [39].

Outros trabalhos de menor qualidade expõem os resultados de forma qualitativa, mas alguns provêm uma boa conclusão do estudo com base em experiências de usuário, e experiências de trabalhos passados, ou integrados em um projeto maior [57, 45, 58].

3.2 Análise e Discussão dos Resultados

Todos os resultados da revisão sistemática são apresentados nesta seção. A Tabela 3.4 mostra uma visão geral de todos os trabalhos relacionados restantes depois do último filtro. Muitos destes trabalhos focam na detecção de objetos, e apenas alguns combinam detecção de objetos e detecção de escadas, ou detecção de portas, e muito menos todos os tipos de detecção.

No contexto de detecção de objetos o conjunto de dados mais utilizado é o *Pascal VOC*, e para o contexto de escadas e portas as avaliações foram feitas em conjuntos de dados próprios de cada trabalho.

O restante desta seção está dividida em: detecção de objetos, que contém uma análise detalhada dos métodos de detecção de objetos encontrados; detecção de portas e escadas, que apresentam os métodos e uma comparação detalhada de cada abordagem; e respostas de pesquisa, que contém as conclusões e respostas da pesquisa de trabalhos relacionados.

3.2.1 Detecção de Objetos

A revisão sobre os métodos de detecção de objetos foca em três tópicos: os métodos utilizados para a detecção, em forma de pipeline; os classificadores, ou a arquitetura das redes neurais utilizadas para a detecção; e uma revisão dos resultados de cada trabalho, e comparação de todos.

Algumas das abordagens de detecção de objetos utilizam pipelines únicos, como o trabalho de Redmon et al. [50]. Outras utilizam abordagens similares como as propostas R-CNN [17, 16, 52]. A Tabela 3.5 mostra uma visão geral dos pipelines de detecção, e suas conexões com as redes neurais convolucionais, como, por exemplo, a abordagem de Girshick et al. [17] que utiliza busca seletiva com uma rede neural convolucional.

Considerando esta visão geral, é possível ver que as primeiras abordagens baseadas em redes neurais convolucionais [17, 79, 48] eram baseadas em busca seletiva ou *sliding window*. Ao longo do tempo, abordagens mais robustas, com técnicas de *regional proposal* que começaram com [20, 22], foram melhoradas por [17, 16]. Por fim, surgiram métodos baseados em um pipeline único e métodos de regressão, como o de Redmon et al. [50] e o de Liu et al. [39].

Outras abordagens também surgiram, como a the Yan et al. [74] que usa segmentação e *conditional random fields*, e a abordagem de Zhu et al. [78] que também usa segmentação, mas suas redes neurais também são baseadas em R-CNNs. Porém, as R-

Abordagem	Detecções			Conjuntos de Avaliação			
	Escadas	Portas	Objetos	Pascal	Ms Coco	Sun	Outro
Wang and Tian[71]	x						x
Harms et al.[19]	x						x
Serrão et al.[57]	x	x					x
Shalaby et al.[58]		x	x				x
Moreno et al.[45]		x					x
Asad et al.[2]		x					x
Skulimowski et al.[61]		x	x				x
Quintana et al.[15]		x	x				x
Chen et al.[7]		x	x				x
Mekhalfi et al.[42]	x	x	x				x
Mekhalfi et al.[43]	x	x	x				x
Kumar and Meher[32]	x	x	x				x
Jose et al.[27]		x					x
Tapu et al.[67]			x				x
Redmon et al. [50]				x			
Liu et al. [39]				x	x		
Girshick et al.[17]				x			
Girshick et al.[16]				x			
Girshick et al.[51]				x			
Najibi et al.[46]				x			
Zou et al.[79]				x			
He et al.[22]			x	x			
Lenc et al.[36]			x	x			
Dai et al.[11]			x	x	x		
Kim et al.[28]			x	x			
Wang et al.[72]			x	x			
Zeng et al.[75]			x	x	x		
Lee et al.[35]			x	x	x		
Shen et al.[59]			x	x			
Papanderou et al.[49]			x	x			
Zhu et al.[78]			x	x			
Yan et al.[74]			x	x			
Bell et al.[5]			x	x	x		
Cinbis et al.[9]			x	x			

Tabela 3.4 – Visão Geral dos Trabalhos Relacionados.

CNNs são difíceis de treinar, tem um custo computacional alto, e não funcionam em tempo real, mesmo com as versões melhoradas da R-CNN (Fast/Faster R-CNN).

Alguns trabalhos propõem uma arquitetura nova de redes neurais profundas para detecção de objetos, outros usam ou estendem arquiteturas já conhecidas. A Tabela 3.6 mostra um sumário das arquiteturas de redes neurais. O sumário lista as camadas de base e as camadas de saída, usualmente há uma série de camadas convolucionais, juntamente

#	Método	Pipeline de Detecção
1	Redmon et al. [50]	Detecção única: uma única rede neural, e um algoritmo único de detecção.
2	Liu et al. [39]	Detecção única: uma única rede neural, e um algoritmo único, similar a YOLO, de detecção
3	Sermanet et al. [56]	Sliding window em conjunto com uma CNN
4	Szegedy et al. [64]	Similar ao R-CNN, porém com a arquitetura da GoogLeNet
5	Girshick et al. [17]	Três módulos: region propostals (selective search), CNN, e um SVM
6	Girshick et al. [16]	Baseado em uma DNN: Rede Convolutional, region proposals (RoI Layers), e uma camada de classificação
7	Ren et al. [51]	Baseado em uma DNN: Rede Convolutional, region proposals (RoI Layers), e o Fast R-CNN Detector
8	Najibi et al. [46]	Baseado em uma DNN: com um regressor em grade fixa, de multi escala, para prever a localização
9	Zou et al [79]	Padrões neurais densos integrados a um Framework de Regionlets
10	Hoffman et al [23]	Detecção em Larga escala através da adapção de um algoritmo, que adapta classificadores para detectores.
11	He et al. [22]	Baseado em uma DNN: Camadas de Pooling personalizadas usadas para detecção de objetos (similar to RoI Pooling), que inspiraram as camadas RoI Pooling
12	Lenc e Vedaldi [36]	Melhorias no método RCNN, mudando a arquitetura da CNN, para nela centralizar toda a detecção
13	Dai et al. [11]	Baseado em uma DNN: Utilizando a R-CNN, porém mudando as últimas camadas para comunicar-se com a R-FCN
14	Kim et al. [28]	Baseado na R-CNN com algumas melhorias
15	Wang et al. [72]	Baseado em Selective search com uma Rede Neural Convolucional
16	Zeng et al. [75]	Baseado na Fast R-CNN, utilizando uma camada bi-directional para auxiliar na extração de características e classificação
17	Ouyang et al. [48]	RCNN com Selective search
18	Lee et al. [35]	Baseado na R-CNN, modificando a arquitetura da rede. E mudando o método de selective search, por um método chamado de randomly sheared strategy
19	Papandreou et al. [49]	Estratégia baseada em Sliding window com uma rede neural convolucional
20	Shen et al. [59]	Baseado na R-CNN, modificando a arquitetura da rede. E mudando o método de selective search, por um método chamado de randomly sheared strategy
21	Zhu et al. [78]	R-CNN e Selective search based
22	Yan et al. [74]	Utiliza CRF para combinar detecção e segmentação. CRF utiliza as características extraídas com uma CNN
23	Bell et al. [5]	Baseado em uma DNN: Uma rede neural convolucional combinada com uma rede neural única, e milhares de camadas RoI, terminando com duas camadas completamente conectadas
24	Cinbis et al. [9]	Selective search com uma CNN e características de FV

Tabela 3.5 – Visão geral dos pipelines de detecção de objetos.

com camadas completamente conectadas. Também, nem todos os trabalhos utilizam um número fixo de camadas em sua arquitetura, significando que o método é testado com um número variável de camadas. Alguns dos métodos propõe camadas e arquiteturas de redes completamente novas, como [22, 17, 16], com as camadas chamadas de RoI *pooling*, ou as camadas chamadas de *Inception* [64]. Além disso outros trabalhos como [39, 50], utilizam arquiteturas comuns como a VGG 16 [60].

As contribuições principais vistas nas camadas e arquiteturas são as camadas de RoI pooling, que foram propostas pela SPP-CNN [22] e pela Fast/Faster R-CNN [16, 51].

#	Método	Utiliza outra rede	Camadas de Base	Camadas de Saída
1	Redmon et al. [50]	Não	24 Convolucionais	2 FC
2	Liu et al. [39]	VGG 16	5 Convolucionais	3 Convolucionais
3	Sermanet et al. [56]	Não	5 Convolucionais	3 fully connected
4	Szegedy et al. [64]	Não	22 inception	Sofmatx
5	Girshick et al. [17]	Sim	5 Convolucionais, com 2 FC	Multiple SVM's
6	Girshick et al. [16]	Sim	Rede convolucional com camadas de RoI pooling	Duas camadas de saída personalizadas
7	Ren et al. [51]	R-CNN, e a Fast R-CNN	5, ou 13 convolucionais	Multiplas camadas combinadas
8	Najibi et al. [46]	Várias arquiteturas de CNN	Convolucionais e RoI	FC e um Regressor
9	Zou et al. [79]	Sim	Várias convolucionais	3 FC
10	Hoffman et al. [23]	Não	1-5 convolucionais	2 FC ou mais
11	He et al. [22]	Sim	5 convolucionais	2 FC
12	Lenc e Vedaldi [36]	R-CNN	5 convolucionais, combinadas com 2 FC	softmax, ou SVM
13	Dai et al. [11]	Não	convolucional e RoI	1 FC
14	Kim et al. [28]	Inception/GoogLeNet e Faster R-CNN	9 convolucional, e 8 inception	1 convolucional
15	Wang et al. [72]	Não	6 convolucionais	3 FC
16	Zeng et al. [75]	BN-Net	multiplas inception, e RoI	multiplas inception
17	Lee et al. [35]	Não	5 convolucionais	4 FC
18	Ouyang et al. [48]	Clarifari, baseada na R-CNN	5 convolucionais	2 FC
19	Lee et al. [35]	Sim	5 convolucionais	4 FC
20	Papandreou et al. [49]	Não	Multiplas convoluções epitomicas	2 FC
21	Zhu et al. [78]	R-CNN	5 convolucionais, com 2 FC	Multiplos SVM
22	Yan et al. [74]	R-CNN	5 convolucionais, com 2 FC	Multiplos SVM
23	Bell et al. [5]	Não	1 convolucional, e 2 IRNNs, com várias RoI	2 FC
24	Cinbis et al. [9]	Não, mas foi baseada em [31]	5 convolucionais	3 FC

Tabela 3.6 – Visão geral das arquiteturas de *Deep Learning.*

Outras diferenças principais entre cada arquitetura são os tamanhos das camadas. Por exemplo, o método YOLO [50] utiliza camadas convolucionais de $1x1x1024$ nas últimas camadas, e a GoogLeNet [64] $7x7x1024$, o que muda drasticamente os custos computacionais. Muitas arquiteturas de redes também são reutilizadas em outros trabalhos, como a R-CNN [17] que é usada por diversas outras redes, como a DeepID Net [48].

Outro tópico importante é o treinamento de diversas arquiteturas utilizadas no contexto de detecção, é que, por exemplo, as R-CNN combinam conjuntos de dados de reconhecimento de objetos, e depois conjuntos de detecção de objetos. O treinamento garante que as redes convolucionais aprendam a extrair características relevantes de diversos objetos, e também gerar regiões de interesse, como as R-CNNs. Isso é possível e uma boa prática, pois as redes dividem a tarefa, e os conjuntos de reconhecimento de objetos, como o *Imagenet* [54], possuem milhões de dados, enquanto conjuntos de detecção, como o Pascal VOC [14], possuem milhares.

Para a revisão dos resultados boa parte dos trabalhos utilizam um conjunto de dados similar, para a avaliação dos seus resultados, sendo este o Pascal VOC [14]. A métrica de detecção de objetos neste contexto é o mAP. A Tabela 3.7 mostra os conjuntos de dados utilizados nas avaliações dos trabalhos encontrados, e os melhores resultados de mAP alcançados.

Conjuntos de Dados	# Trabalhos Avaliando	Melhor Resultado (mAP)
Pascal VOC 2007	20	83.8 [28]
Pascal VOC 2010	7	68.8 [16]
Pascal VOC 2012	12	78.8 [51]
MSCoco	6	64.3 [5]

Tabela 3.7 – Melhores Resultados.

Conforme mencionado, o problema na avaliação dos resultados é que as abordagens utilizam diferentes métodos de treinamentos, e diferentes conjuntos de dados. Isso significa que os resultados podem variar de acordo com os dados vistos inicialmente na fase de treinamento. Uma comparação mais detalhada é apresentada na Tabela 3.8, mostrando os resultados de cada trabalho com as suas avaliações, as métricas de mAP e velocidade utilizadas.

A Tabela 3.8 mostra alguns resultados interessantes. O método YOLO [50] utilizando a rede Fast YOLO obtém um dos tempos de detecção mais rápido. Porém, também existem abordagens competitivas, como a SSD 300 que tem um resultado de precisão de 72%. A razão principal disso é o tamanho das camadas da rede utilizadas, e o tamanho de entrada visto na SSD [39]. A SSD testa duas abordagens variando do tamanho de entrada de 300x300, para 500x500. Com tamanhos de entrada maiores, a rede aumenta a precisão, porém diminui a velocidade de detecção.

#	Método	Conjunto de Dados	mAP	Velocidade
1	Redmon et al. [50] (YOLO)	Pascal VOC 2007, 2012	63.4, 57.9	45 FPS
1	Redmon et al. [50] (Fast YOLO)	Pascal VOC 2007	52.7	155 FPS
2	Liu et al. [39] (SSD 300)	Pascal VOC 2007, 2012, MS Coco	72.1, 70.3, 38	58 FPS
2	Liu et al. [39] (SSD 500)	Pascal VOC 2007, 2012, MS Coco	75.1, 73.1, 43.7	23 FPS
3	Girshick et al.[17]	Pascal VOC 2007, 2010	58.5, 53.7	
4	Girshick et al. [16]	Pascal VOC 2007, 2010, 2012	70.0, 68.8, 68.4	0.5 FPS
5	Ren et al.[51]	Pascal VOC 2007, 2012	78.8, 78.8	18 FPS
6	Najibi et al. [46]	Pascal VOC 2007, 2012	57.2, 66.4	
7	Zou et al. [79]	Pascal 2007, 2010	46.1, 44.1	
8	He et al.[22]	Pascal VOC 2007,	60.9	
9	Lenc et al.[36]	Pascal VOC 2007	59.68	
10	Dai et al.[11]	Pascal VOC 2007, 2012, MSCoco	76.6, 82, 53.2	0.17 secs
11	Kim et al.[28]	Pascal VOC 2007, 2012	83.8, 82.5	31.3 FPS
12	Wang et al. [72]	Pascal VOC 2007, 2010	41.7, 39.7	
13	Zeng et al. [75]	Pascal VOC 2007, MS Coco	77.2, 68	
14	Lee et al. [35]	Pascal VOC 2012, MS Coco	77.8, 45.8	0.23
15	Shen et al. [59]	Pascal 2007, 2010, 2012	60.1, 56.4, 56.3	
16	Papandreou et al. [49]	Pascal 2007	56.4	
17	Zhu et al. [78]	Pascal 2010, 2012	58.5, 58.7	
18	Yan et al. [74]	Pascal 2007	61.4	
20	Bell et al. [5]	Pascal 2007, 2012, MS Coco	79.2, 76.4, 64.3	
21	Cinibis et al. [9]	Pascal 2007, 2010	47.3, 55.2	

Tabela 3.8 – Visão Geral dos Resultados de detecção de objetos.

3.2.2 Detecção de Escadas e Portas

A Tabela 3.9 mostra uma visão geral dos pipelines de detecção utilizados na detecção de portas e escadas.

#	Abordagem	Pipeline de Detecção
1	Wang and Tian.[71]	Detecção de linhas, Extração de características de profundidade, classificação SVM
2	Harms Et al.[19]	Detecção da orientação da superfície, detecção de bordas em 3d, rastreamento de linhas, predição do modelo de escada
3	Serrão Et Al.[57]	Detecção de bordas, e retângulos, e estimativas baseadas em limiares
4	Shalaby Et Al.[58]	Detecção de bordas, cantos e retângulos, predição de linhas, busca de polígonos agrupados, e predição geométrica via limiares
5	Asad and Ikram.[2]	Extração e seleção de características, e um módulo de decisões e predições pré estabelecidas
6	Skulimowski Et Al.[61]	Estimativa de orientação da superfície, detecção de linhas, e retângulos, seleção de linhas, e estimativa do modelo de porta, e detecção da maçaneta da porta
7	Quintana Et Al.[15]	
8	Chen Et Al.[7]	CNN
9	Kumar and Meher.[32]	CNN, e RNN
10	Mekhalfi Et Al.[42]	SIFT, BoW, e PCA
11	Mekhalfi Et Al.[43]	SIFT, BoW, e PCA

Tabela 3.9 – Pipelines de detecção de portas e escadas.

O método apresentado por Wang e Tian [71] utiliza uma abordagem com RGBD, combinando pontos de características, e a dimensão de profundidade, para depois classificar as imagens em escadas utilizando SVM. Harms et al. [19] propõe um método inovador para a detecção de escadas ascendentes utilizando uma câmera stereo, e técnicas de visão computacional, com detecção de bordas em 3D e rastreamento de linhas.

Serrão et al. [57] utiliza um pipeline com detecção de retângulos, e bordas, baseados em transformadas de Hough, e detecção de cantos, para depois buscar as escadas procurando por um certo padrão nas detecção, ao qual linhas conectadas devem formar o degrau de uma escada.

Para as detecções de portas, alguns autores [57, 58, 45, 2] utilizam um pipeline similar, com detecção de bordas e retângulos, baseados em transformadas de Hough e detecção de cantos. Então, os retângulos detectados são filtrados de acordo com um limiar, baseando-se nas propriedades das portas.

Já Skulimowski et al. [61] propõe um método de detecção de portas baseado em uma abordagem que utiliza cenas 3D, também com imagens de RGBD. Primeiro, uma equação estima a orientação e a localização do piso, então as bordas da imagem são extraídas e uma transformação de Hough probabilística é feita para buscar segmentos de linha, e pares de linas paralelas que estão perpendiculares ao chão são identificadas. A detecção também combina um método para a detecção das maçanetas da porta, melhorando a identificação da porta.

Quintana et al. [15] propõe uma técnica única, que integra ambas a geometria do ambiente, e a imagem, vindas de uma câmera e um scanner 3D a laser. Isso torna o método robusto a oclusões, variações de cores e condições de luzes.

Os únicos métodos encontrados que se baseiam em CNNs são os trabalhos de Chen et al. [7], e o de Kumar e Meher [32]. Chen et al. [7] propõe um método de detecção de portas e reconhecimento da orientação da porta (esquerda, direita). Eles treinam uma rede neural convolucional em um conjunto de dados próprio. Já Kumar e Meher [32] propõem um método de descrição completa dos objetos de um ambiente, incluindo também portas e escadas. Seu método é baseado na combinação de uma CNN com uma RNN.

A Tabela 3.10 mostra os resultados das avaliações realizadas para os trabalhos relacionados de detecção de portas e escadas. Os conjuntos de dados e os formatos de avaliação variam de acordo com os tipos de trabalho. Neste contexto, é difícil comparar qual trabalho pode ou não ser melhor. É importante ressaltar também que vários dos trabalhos utilizam imagens com profundidade (RGBD), principalmente no contexto da detecção de escadas.

Comparando com os resultados da Tabela 3.8 pode-se perceber a padronização no contexto de detecção de objetos, em que todos utilizam um conjunto de dados padrão, sendo ele o *Pascal VOC* 2007, ou 2010, ou o *Microsoft COCO*. Além disso alguns trabalhos, como o de Serrão et al. [57], avaliam de forma qualitativa, apenas comentando e mostrando os resultados em algumas imagens, dificultando ainda mais a comparação.

#	Abordagem	Método de Avaliação	Conjunto de dados de Avaliação	Resultados
1	Wang and Tian.[71]	Accuracy	Próprio	97.2%
2	Harms Et al.[19]	Porcentagem do Erro de altura (cm), Porcentagem do erro de Profundidade (cm), Porcentagem do erro de Largura(cm)	Próprio	0.8, 0.85, 10.7
3	Serrão Et Al.[57]	Qualitativamente	Próprio	Apenas Imagens
4	Shalaby Et Al.[58]	Sensividade	Próprio, Simples, e Complexo	100%, 64%
5	Asad Et Al.[2]	Acurácia	Três Conjuntos Próprios	84.1%, 90.4%, 95.5%
6	Skulimowski Et Al.[61]	Sensibilidade, com o coeficiente de similaridade de Jaccard	Próprio	63%
7	Quintana Et Al.[15]	Precision, and Recall	Próprio	0,986, 0,983
8	Chen Et Al.[7]	Taxa de Erro	Próprio	2.82%
9	Kumar and Meher.[32]	Acurácia Média	Próprio	Todas as classes: 94.27% Portas: 96.3% Escadas: 93.33%
10	Mekhalfi Et Al.[42]	Baseado em distância Euclidiana	Próprio	Todas as Classes: 89%
11	Mekhalfi Et Al.[43]	Baseado em distância Euclidiana	Próprio	Todas as Classes: 89%

Tabela 3.10 – Resultados de Detecção de Portas e Escadas.

3.2.3 Respostas de Pesquisa

A revisão sistemática da literatura foi proposta para buscar os trabalhos relevantes referentes a esse contexto. Desta revisão foram selecionados **34** trabalhos e agrupados pela sua finalidade de pesquisa, incluindo detecção de objetos do ambiente, detecção de escadas, e detecção de portas. A partir destes trabalhos, foi feita uma análise dos métodos utilizados e resultados obtidos.

Após a revisão da literatura e a análise dos trabalhos encontrados, foi possível responder as perguntas de pesquisa formuladas:

1. RQ1: Quais são os métodos disponíveis de detecção, que incluem objetos de ambientes internos (portas e escadas), e podem auxiliar os deficientes visuais?
2. RQ2: Quais são os melhores métodos de detecção de objetos disponíveis na literatura?
3. RQ3: Para quais objetos ainda há espaço para melhorias nas técnicas de detecção disponíveis?

Para responder a RQ1, consideramos os trabalhos de Mekhalfi et al. [42, 43] e Kumar e Meher [32] que propõem um método com a capacidade de reconhecer objetos, escadas e portas nas imagens. Porém, Mekhalfi et al. [42] não dispõem de uma avaliação com as métricas mais utilizadas pelo estado da arte, e Kumar e Meher [32] não propõem métodos para a detecção de objetos, nem escadas, nem portas, apenas métodos de reconhecimento desses objetos.

Além disso apenas o trabalho de Mekhalfi et al. [42, 43] se dispõe a auxiliar deficientes visuais com navegação em ambientes a partir de técnicas de detecção, e inclui objetos, escadas, e portas.

Para os métodos de detecção de portas, Skulimowski et al. e Quintana et al. [61, 15] utilizam técnicas que geram bons resultados, porém dependem de imagens RGBD. Apenas os métodos de Kumar e Meher e Chen et al. [32, 7] utilizam como base CNNs e abrangem portas, e nenhum dos trabalhos encontrados utilizam CNNs e abrangem escadas.

Considerando a RQ2, achamos que os melhores métodos de detecção disponíveis na literatura e que possibilitam uma comparação, são a YOLO [50] e a SSD [39], pois ambas têm uma precisão aceitável com possibilidade de executar em tempo real. Nesse contexto há espaço para treinar essas técnicas para a detecção de portas e escadas, e incluir melhorias nas arquiteturas das redes e no formato de treinamento, utilizando um conjunto de imagens de ambientes. Esses conjuntos podem ser voltados para cenas de ambientes, como o *Places2* [77], para extração das características das imagens de ambientes internos, ou conjuntos como o *Open Images* [30], que possuem tanto objetos que fazem parte da cena, como escadas, portas, placas e objetos comuns, como cadeiras, mesas, entre outros.

Considerando a RQ3, identificamos que as técnicas de detecção de objetos se mostram capazes de resolver o problema de detecção em tempo real, com bons resultados de mAP e em imagens RGB. Porém, elas não contemplam objetos específicos de ambientes, como portas, e escadas. Dado o uso de redes neurais convolucionais, essas técnicas são capazes de aprender a partir de dados de treinamento. Dessa forma há um espaço para o treinamento e avaliação das técnicas baseadas em redes neurais convolucionais, como YOLO [50] ou SSD [39], em conjuntos de dados que contemplem objetos, portas e escadas. E também, conforme a Tabela 3.10, percebe-se que há um espaço para construção de um conjunto de dados público que seja considere diversas imagens de portas, e escadas em ambientes internos.

4. MODELO DE APOIO A NAVEGAÇÃO EM AMBIENTES INTERNOS PARA DEFICIENTES VISUAIS

Conforme mencionado anteriormente, o objetivo deste trabalho consiste na proposição de um modelo que auxilie pessoas com deficiência visual na navegação em ambientes internos. Para que isso seja possível, a premissa é que através do uso de visão computacional se consiga passar alguma informação do ambiente ao usuário, como, por exemplo, avisar que à sua direita está uma porta, ou uma escada.

De acordo com as respostas das perguntas de pesquisa formuladas na revisão sistemática (Seção 3.1), encontramos um espaço no que se diz respeito à utilização de técnicas do estado da arte de detecção de objetos empregadas para a detecção de escadas e portas em ambientes internos. Dessa forma os objetivos específicos para este trabalho são:

1. Desenvolvimento e validação de algoritmos para detecção de portas e escadas;
2. Geração de informações a partir das detecções realizadas;
3. Elaboração de uma prova de conceito para validação dos algoritmos desenvolvidos.

Sendo assim, o modelo proposto, conforme ilustra a Figura 4.1, utiliza as coordenadas de detecções, e os objetos classificados para poder auxiliar o usuário com informações relevantes.

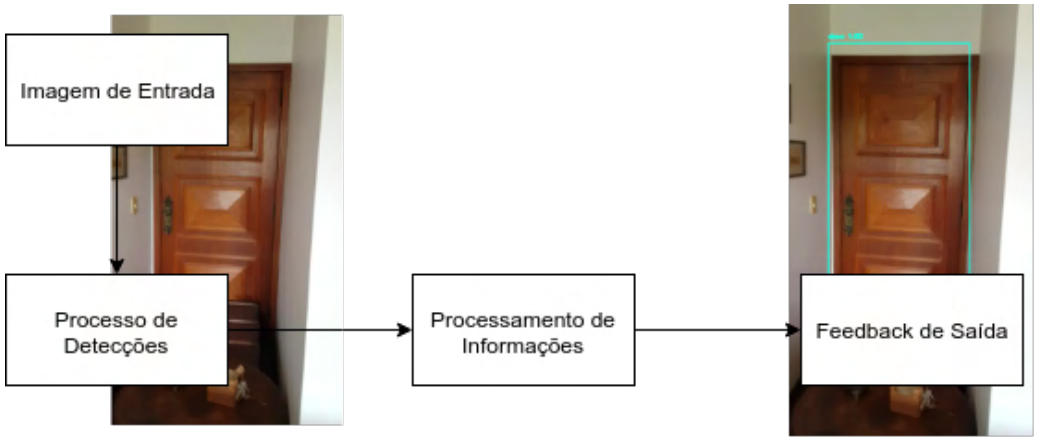


Figura 4.1 – Visão Geral do Modelo.

A metodologia deste trabalho baseou-se na realização de um experimento na área de visão computacional, mais especificamente em classificação de imagens e detecção de objetos. Para isso foram coletadas imagens contendo os objetos desejados (portas e escadas) no ambiente, e, então, foram realizados os experimentos com métodos de *deep learning*, tanto para a classificação de imagens, como para a detecção de objetos.

Dessa forma este capitulo esta organizado da seguinte maneira: a Seção 4.1 apresenta a metodologia utilizada para a realização deste trabalho; a Seção 4.2 mostra como foi realizada a construção do conjunto de dados, e suas informações; a Seção 4.3 mostra como foram realizados os experimentos para a construção deste trabalho.

4.1 Metodologia

A metodologia utilizada no desenvolvimento deste trabalho consistiu em avaliar algumas técnicas de classificação de imagens e detecção de objetos descritas na literatura, e utilizar estas técnicas considerando o contexto de portas e escadas em imagens. Neste trabalho, estamos considerando apenas portas e escadas em ambientes internos, tais como prédios comerciais e residenciais.

Usamos como base o trabalho de Mekhalfi et al [42], que propõe um método baseado em técnicas de BoVW, PCA e classificadores, e também o trabalho de Kumar e Meher [32], que propõe um método baseado em *deep learning* para classificar portas, escadas e diversos outros objetos. Entretanto, para o contexto de apoio a navegação é importante que o modelo forneça informações sobre os tipos de escadas e portas, como por exemplo: portas de elevadores ou se uma porta está aberta; ou se o usuário deve subir ou descer a escada.

Na literatura encontrada, alguns trabalhos [71, 19] propõe a detecção de escadas, sendo que Harms et al [19] informa se é uma escada ascendente, porém eles não utilizam nenhuma técnica de *deep learning*. Chen et al. [7] utilizam técnicas de *deep learning* e conseguem dizer a localização das portas no corredor, utilizando classificação de imagens.

Dessa forma optamos por realizar um experimento, nos métodos de classificação de imagens, e detecção de objetos, para esse contexto. Inicialmente por questões de hardware, e adequações na construção de nosso conjunto de dados conduzimos o experimento em técnicas de classificação, para depois realizar o experimento sobre técnicas de detecção de objetos. Nosso experimento utilizou um conjunto de dados próprio com imagens de vários tipos de portas, e escadas ascendentes e descendentes. Para que seja possível por exemplo, informar o usuário que há uma porta de elevador, ou uma escada ascendente, em que ele possa subir.

Para isso, a nossa metodologia para a realização deste trabalho, foi dividida em etapas, de acordo com a Figura 4.2. As etapas iniciais começam pela análise dos trabalhos relacionados e o experimento inicial, ao qual levou à definição do modelo, e da metodologia de pesquisa utilizada para a construção do modelo.

A metodologia de construção do modelo foi definida como um experimento: então foi necessário a construção do conjunto de dados para o contexto deste trabalho; testes com

algoritmos de *deep learning* para classificação das imagens do conjunto de dados; ajustes e melhorias no conjunto de dados, incluindo as anotações das regiões dos objetos; e por fim testes com métodos de detecção de objetos. Além disso para as etapas de melhorias do conjunto de dados, o processo foi feito de forma iterativa com as avaliações das técnicas tanto de detecção quanto de classificação.

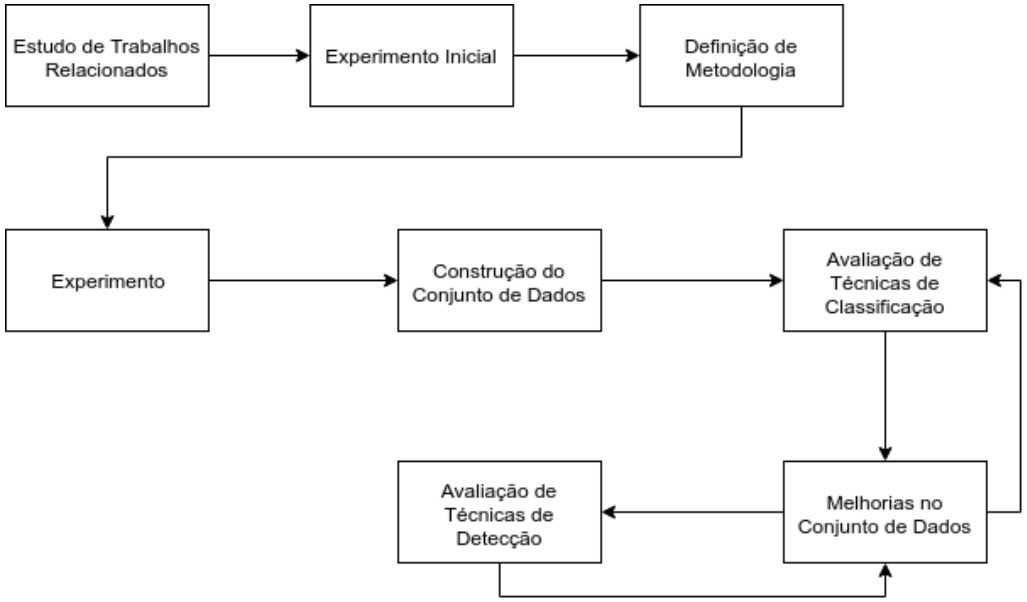


Figura 4.2 – Metodologia de Pesquisa.

Para a avaliação preliminar deste conjunto de dados, e do modelo proposto, foram selecionadas técnicas classificação de imagens, já consolidadas. As técnicas inicialmente selecionadas foram: HoG e SVM; BoVW e SVM; e uma rede neural convolucional, a InceptionV3 [63] (a versão mais recente da GoogLenet [64]). Depois da avaliação preliminar optou-se por continuar com as técnicas baseadas em redes neurais convolucionais e seguir na construção do conjunto de dados.

Na etapa da avaliação de técnicas de classificação, optamos pelo uso do método de treinamento de *transfer learning*, que conforme mencionado na Seção 2.2 consiste em treinar apenas as camadas de classificação de uma rede neural pré treinada. A construção do conjunto de dados foi feita de modo incremental, ou seja novas coletas foram realizadas, e testadas a partir de treinamentos nos modelos selecionados, e avaliadas em um mesmo conjunto de testes.

Para avaliar o modelo, primeiramente avaliamos se é possível classificar entre: ambientes internos, ou seja, imagens sem portas, ou escadas; imagens com portas; e imagens com escadas. Inicialmente foram considerados apenas uma categoria de porta, e uma de escada. Depois observamos se era possível diferenciar entre os tipos de escada e portas. Os tipos de escada escolhidos foram ascendente e descendente, ou seja escadas vista de baixo, ou de cima. Para portas escolhemos diversos tipos entre eles abertas, fechadas, semi abertas, duplas e de elevador.

Os resultados da classificação foram avaliados, a partir disso foi possível escolher as classes mais adequadas para fazer a anotação das imagens (com a classe e posição dos objetos) para a etapa de detecção. Além disso, foi possível filtrar do conjunto de imagens, os *outliers*, e as imagens que geram bons resultados no treinamento. Isso é importante, pois os métodos de detecção também são baseados em redes neurais convolucionais, e enfrentam problemas similares na etapa de treinamento, necessitando de uma boa qualidade nos dados. Por fim foi realizado o experimento para um modelo de detecção de portas e escadas, que possa auxiliar na navegação em ambientes internos.

Para a detecção de objetos também utilizamos métodos de *deep learning*. Conforme a avaliação dos trabalhos relacionados vista na Seção 3.2.3, é desejável que seja feita a detecção em tempo real, com um bom *trade-off* entre velocidade e precisão (mAP). Assim, os métodos escolhidos foram treinados e avaliados com o nosso conjunto de dados.

A avaliação consistiu em analisar a precisão (mAP) e a velocidade. O método de treinamento utilizado foi o de *fine tunning*, que conforme mencionado na Seção 2.2, consiste em utilizar pesos de modelos pré-treinados em um conjunto de dados maior, para que o modelo possa realizar a nossa tarefa de detecção de portas, e escadas.

4.2 Conjunto de Dados

Para a construção do nosso modelo decidimos utilizar um conjunto de dados próprio, o qual chamamos de DSD (*doors and stairs dataset*)[1]. O conjunto é composto de imagens que possuem as seguintes classes: portas, portas de elevador, portas duplas, portas abertas, e portas semi abertas; escadas, tanto ascendentes, como descendentes; e ambientes internos.

A Figura 4.3 mostra a hierarquia de classes dos objetos anotados. A classe de ambientes internos é considerada como um **negativo**, ou seja, serve para auxiliar os algoritmos a separarem no ambiente o que é ou não relevante, deste modo ela não está listada na hierarquia.

A hierarquia de classes foi definida considerando que é importante para os usuários terem informações mais detalhadas sobre o ambiente, pois assim poderiam saber como interagir com uma determinada porta. Por exemplo: no caso de uma porta de elevador, ele terá de procurar o painel do elevador; se uma porta estiver fechada, ele terá que procurar pela maçaneta para abri-la; e se uma porta estiver aberta, ele poderá entrar no ambiente. Já para as escadas é importante que o usuário saiba se é uma escada para subir ou para descer um andar.

[1] https://tinyurl.com/gvc-dsd

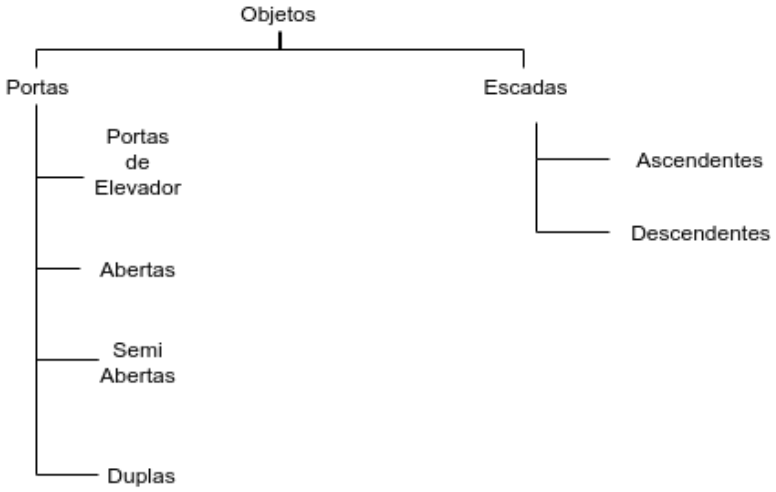


Figura 4.3 – Hierarquia de Classes do DSD.

Para o processo de construção do conjunto de dados, começamos com uma etapa de coleta e separação de imagens por categorias. A coleta das imagens foi feita através de uma busca em diversas fontes de imagens, tais como: sites de busca de imagens, como *Google Images* e Flickr; e outros conjuntos de dados de imagens, como SUN Dataset [73], MIT Places [77] e Imagenet [54]. Posteriormente na coleta de imagens da internet foi encontrado o conjunto de dados MCI Indoor [3], que contém imagens de ambientes internos, de melhor qualidade, coletadas em campo, ao qual também foram adicionadas ao DSD.

Dessa forma, em busca de mais imagens de melhor qualidade, foi feita uma coleta de imagens “em campo”, com fotos e vídeos capturados de ambientes variados, incluindo prédios da universidade e condomínios residenciais. Neste caso, nos deslocamos em diferentes ambientes e geramos vídeos de 1 a 3 minutos. Os *frames* foram extraídos dos vídeos para serem colocados no conjunto de dados, sendo que a quantidade de *frames* obtidos por segundo foi 5, para evitar que houvesse muita similaridade em cada uma das imagens geradas.

Depois da coleta, foi realizada uma etapa de limpeza das imagens de forma manual. Assim, foram removidas imagens borradas que acabam sendo geradas pela movimentação da câmera, e imagens com *frames* muito similares.

As imagens dos sites de busca foram coletadas uma a uma, através de uma consulta pelas palavras chave desejadas, como *stairs*, *doors*, *elevators*, entre outras. Além disso, foram selecionadas apenas as imagens públicas do *Flickr*.

Dessa forma o conjunto DSD foi dividido em dois subconjuntos, como mostram as Tabelas 4.1, e 4.2. O subconjunto A contém as imagens coletadas da internet, de menor qualidade, e o subconjunto B as imagens coletadas em campo, e do MCI Indoor [3]. Essa divisão foi feita para garantir um conjunto de imagens com maior controle de qualidade dos dados. Essa divisão foi necessária tendo em vista que o conjunto completo, e o subconjunto A, estavam provendo dificuldades no treinamento dos métodos de detecção de objetos.

Subconjunto	Coleta	Total de Imagens
A	Internet	8.501
B	Em Campo	3.516
Total	-	12.017

Tabela 4.1 – Subconjuntos do DSD.

Por fim, todas as imagens foram categorizadas e as posições dos objetos foram anotadas no formato Pascal VOC [14], utilizando a ferramenta *labelImg*[2]. A Tabela 4.2 mostra a quantidade final de anotações feitas para cada classe, bem como a quantidade de anotações achadas para cada fonte (internet e em campo). As anotações de ambientes internos não foram feitas com região, pois os algoritmos de pré processamento consideram todo o contexto que não é anotado como **negativo**.

A Figura 4.4 mostra como foram feitas as anotações utilizando a ferramenta *labelImg*. Neste caso, as coordenadas (*bounding boxes*) dos objetos são marcadas considerando cada uma das classes propostas. O formato Pascal VOC considera as coordenadas como os pontos *x* e *y*, iniciais e finais dos objetos, nomeados como: **xmin**, **ymin**; e **xmax**, **ymax**.

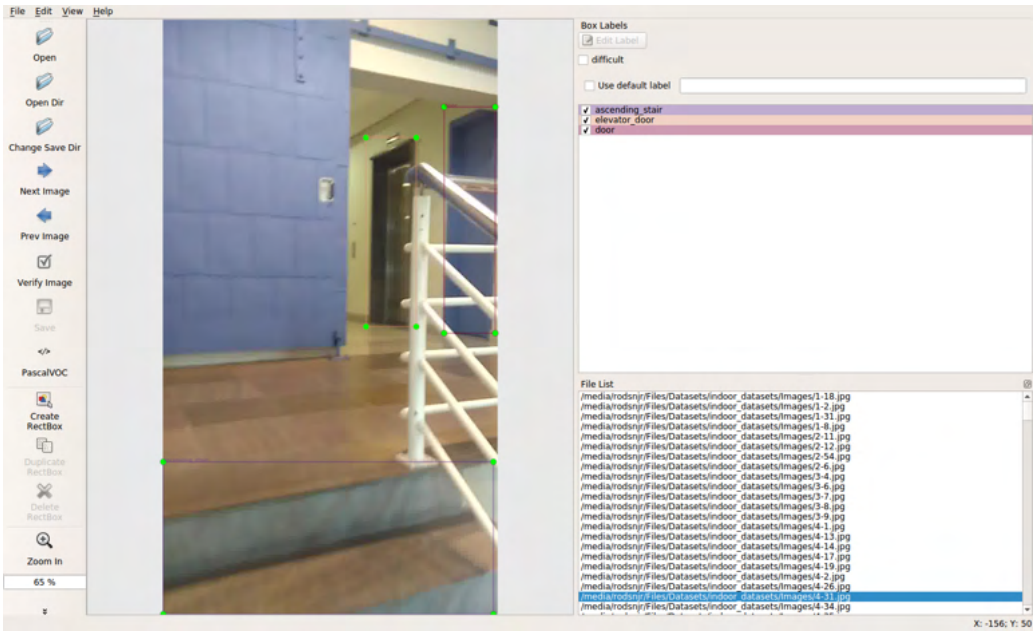
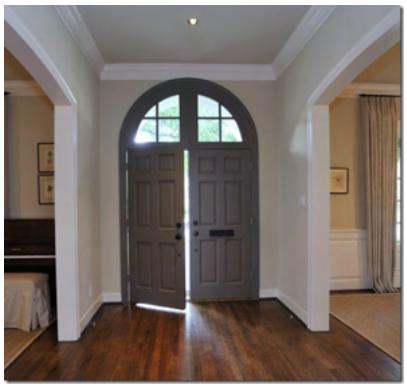


Figura 4.4 – Anotações usando *labelImg*.

As Figura 4.5, e 4.6 mostram algumas das imagens coletadas para o DSD, sendo das classes, de portas, portas de elevadores, ambientes internos e de ambas as classes de escadas (ascendentes e descendentes).

[2]https://github.com/tzutalin/labelImg

(a) Porta

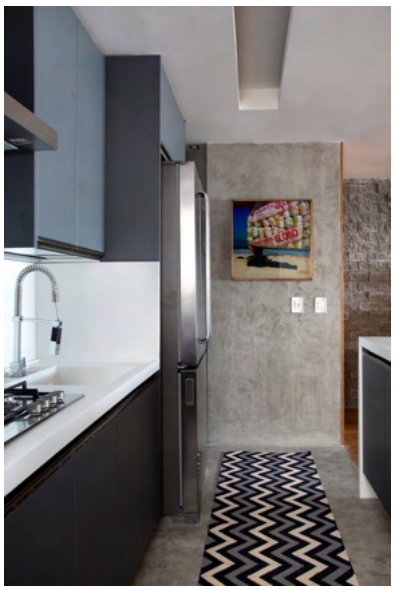


(b) Porta Dupla

(c) Porta de Elevador

(d) Porta Semi Aberta

(e) Porta Aberta



(f) Ambiente Interno

Figura 4.5 – Classes de Portas do DSD, e Ambientes Internos.

(a) Escadas Ascendentes (b) Escadas Descendentes

Figura 4.6 – Classes de Escadas do DSD.

4.3 Experimentos

Para a construção e avaliação do modelo proposto foram realizados experimentos no conjunto de dados DSD, tanto para a classificação das diversas classes propostas, quanto para a detecção dos objetos anotados.

A classificação é utilizada para avaliar quais métodos são capazes de identificar as classes propostas, apresentando os melhores resultados considerando estas classes. Como há diferentes tipos de classes nas imagens, que podem ser definidas como cenas

Classe	Anotação	Sub Conjunto		
		A	B	Total
Escada ascendente	Ascending Stair	1.924	852	2.776
Escada descendente	Descending Stair	381	935	1.316
Porta	Door	3091	1486	4577
Porta de Elevador	Elevator Door	646	1034	1680
Porta Dupla	Double Door	460	0	460
Porta Semi Aberta	Half Opened Door	328	0	328
Porta Aberta	Opened Door	702	0	702
Ambientes Internos	Indoor Environment	3.274	0	3.274

Tabela 4.2 – Visão geral das anotações.

(escadas ascendentes, escadas descendentes, e de ambientes internos), e classes que podem ser definidas como objetos (portas, portas de elevadores), esse processo pode se encaixar tanto como uma tarefa de reconhecimento de cenas, ou de objetos. Dessa forma neste texto utilizamos como definição a classificação de imagens, que seria a tarefa que ambos reconhecimento de cenas, ou objetos, realizam.

Após a avaliação dos resultados da classificação, utilizamos os modelos, e os resultados das validações com diferentes divisões de classe como base para o treinamento dos métodos de detecção. As subseções a seguir descrevem os processos de classificação das imagens e de detecção dos objetos.

4.3.1 Classificação de Imagens

Conforme descrito na fundamentação teórica (2.2), existem diversas arquiteturas de redes neurais convolucionais, como a InceptionV3 [63]. Assim o objetivo do experimento de classificação de imagens é identificar quais dos modelos disponíveis na literatura conseguem realizar a tarefa melhor, e quais são os melhores resultados com diferentes separações de classes. Para o treinamento nesta etapa de classificação foram selecionados os seguintes modelos, previamente treinados no ImageNet [54].

1. VGG16, e VGG19 [60];
2. Resnet 50 [21];
3. InceptionV3 [63];
4. Xception [8];
5. MobileNet [24];
6. InceptionResNetV2 [62].

Lista 4.1 – Modelos treinados para a classificação de imagens.

Então as anotações das classes do conjunto foram dividas conforme a Lista 4.2. A divisão inclui diferentes classes de portas em ambientes internos, e apenas as classes principais da hierarquia. Das portas, as mais fáceis de separar são portas normais, de portas de elevador.

A camada de entrada foi configurada para receber um tamanho de entrada de $224x224x3$, baseado no tamanho da arquitetura da InceptionV3 [63]. Para a escolha das camadas ocultas foi feita uma análise empírica com diferentes tamanhos, funções de ativação. A configuração final foi de 1 a 3 camadas completamente conectadas, de 128 a 2048

1. portas, escadas, ambientes internos;
2. portas, portas de elevador, escadas ascendentes, escadas descendentes, ambientes internos;
3. portas, portas duplas, portas semi abertas, portas abertas, portas de elevador, ambientes internos, escadas ascendentes, escadas descendentes.

Lista 4.2 – Divisões de Classes.

unidades ocultas, com um *dropout* de 0.5 para evitar *overfitting*, e uma camada de saída com a função de ativação *softmax*.

O processo de treinamento foi dividido em quatro passos: primeiro, extraímos as características das imagens com cada uma das redes convolucionais; depois, foram treinadas as camadas de classificação; na sequência os classificadores foram otimizados, buscando os melhores resultados no conjunto de dados de avaliação; e por fim os classificadores foram avaliados no conjunto de dados de testes.

Para a etapa de extração de características são salvos arquivos com apenas as características para cada uma das redes utilizadas, para agilizar os demais processos. Então esses arquivos são posteriormente utilizados nas etapas de treinamento, avaliação e testes, evitando o custo e tempo de extrair as características para cada lote de entrada, nas iterações do treinamento. Esse processo de treinamento é eficiente, e permite testar diversos tipos de entradas de anotações para cada imagem. Sendo assim, nas etapas de treinamento e avaliação são dadas as configurações de classes da Lista 4.2.

4.3.2 Detecção de Portas e Escadas

Conforme descrito na fundamentação teórica (2.2) também existem diversos métodos para a detecção de objetos. A partir da análise dos trabalhos relacionados constatou-se que os métodos YOLO e SSD são os recomendáveis por executarem detecções em tempo real, pois possuem um bom *trade-off* entre precisão, e velocidade de processamento. Além disso, a implementação Fast YOLO, que utiliza uma rede convolucional menor, tem um custo de processamento baixo o suficiente para executar em dispositivos móveis. Sendo assim para o treinamento da detecção das portas e escadas os métodos escolhidos foram baseados na YOLO [50]

O experimento consistiu em treinar alguns dos modelos de redes neurais convolucionais com o método YOLO, para a detecção dos objetos desejados. Os objetos selecionados na etapa de classificação de imagens foram: escadas ascendentes, escadas descendentes, portas e portas de elevador.

A Lista 4.3 mostra os modelos de redes neurais convolucionais utilizados no treinamento, adaptados ao método de detecção de objetos YOLO. O treinamento inicialmente foi feito nas imagens coletadas da internet, e devido alguns problemas, posteriormente foi feito apenas nas imagens coletadas em campo.

1. Inception [63];
2. Tiny YOLO [50].
3. Full YOLO [50].
4. Mobilenet [50].

Lista 4.3 – Modelos treinados para a detecção de objetos.

A Full YOLO contém camadas convolucionais e conexões residuais, com um total de 50, 594, 061 parâmetros, e o modelo final ocupa um espaço de 592.9MB em disco. A Tiny YOLO é um modelo mais simples, com uma rede convolucional de apenas 9 camadas convolucionais, resultando em 15, 785, 885 parâmetros, e ocupando um tamanho de 185MB em disco. O método de treinamento para a detecção consistiu nas seguintes etapas: dividir o conjunto de dados em treinamento, validação e testes; organizar os algoritmos de treinamento; treinar cada um dos modelos desejados; realizar aprimoramentos no conjunto de dados conforme os resultados do treinamento; e por fim avaliar os modelos no conjunto de testes.

Por questões de otimização de tempo, os modelos Mobilenet e Tiny YOLO foram selecionados para primeira rodada de treinamentos. Depois do treinamento, e geração de resultados no conjunto de testes com esses modelos, fizemos otimizações no conjunto de dados, até os resultados de mAP, chegarem a um número próximo ou acima de 50%. Além disso, as otimizações no conjunto de dados feitas durante esta etapa de treinamento incluíram: retirar imagens de má qualidade; ajustar anotações com erros, com localização do objeto errada, ou erros de digitação nas classes dos objetos. Apesar das otimizações os treinamentos no subconjunto A e no conjunto completo do DSD não geraram bons resultados. Dessa forma optamos por utilizar apenas o subconjunto B (com imagens coletadas em campo).

5. RESULTADOS

Como mencionado no Capítulo 4, dois experimentos foram realizados sobre o conjunto de dados DSD, um com métodos de classificação de imagens, e outro com métodos de detecção de objetos. Dessa forma os resultados apresentados neste capítulo estão organizados da seguinte maneira: a Seção 5.1 expõe os resultados referentes à classificação de imagens; e a Seção 5.2 mostra os resultados referentes aos métodos de detecção de objetos utilizados.

5.1 Classificação de Imagens

Conforme descrito na metodologia apresentada na Seção 4.1, foram realizados experimentos com classificação de imagens e com detecção de objetos. Para a classificação de imagens, o conjunto de dados foi dividido em diferentes classes conforme a Tabela 5.1, para testar a classificação de diferentes contextos.

O treinamento foi feito sobre 100 épocas, com um *batch size* de 16, utilizando a função de otimização adam [29]. Além disso, foram adicionados dois procedimentos feitos a cada época: um procedimento de *early-stopping*, com um critério de parada de 10 épocas, ou seja, caso o classificador não melhore por 10 épocas, o treinamento é terminado; e um procedimento de *checkpoint*, que caso no final da época atual haja melhorias no modelo, salva um arquivo com o mesmo.

Para treinar os modelos de redes neurais convolucionais, utilizamos as implementações que estão disponíveis no Keras [1]. Estas implementações permitem utilizar apenas as camadas convolucionais como extratores de características, facilitando o *transfer-learning*. Além disso, utilizamos uma implementação do VGG16 treinada no Places2 [77] [2], para comparar a diferença entre utilizar pesos capazes de reconhecer diferentes tipos de cenas.

A avaliação do modelo foi feita sobre o subconjunto A (com imagens coletadas da internet, como descrito na Seção 4.2). Conforme ilustra a Tabela 5.1, os subconjuntos para o treinamento foram divididos em: conjunto de treinamento, com 80% do total das imagens do conjunto; validação, com 10% do total das imagens do conjunto; e testes, com 10% do total das imagens do conjunto.

De acordo com o objetivo deste trabalho, dividimos o subconjunto A em 3 configurações de classes. Como mostra a Lista 5.1, em que ajustamos de acordo com a hierarquia mostrada na Seção 4.2, as diferentes classes de portas, e escadas em três configurações distintas.

[1] https://keras.io/applications/

[2] https://github.com/GKalliatakis/Keras-VGG16-places365

Classe	Treinamento	Validação	Testes
Escadas Ascendentes	766	76	104
Escadas Descendentes	191	30	22
Portas	1888	196	230
Portas Duplas	291	33	35
Portas Abertas	473	49	57
Portas Semi Abertas	232	26	22
Portas de Elevador	420	37	41
Ambientes Internos	2617	318	339

Tabela 5.1 – Divisão do Conjunto de Dados para Treinamento.

Conjunto 1: portas, escadas, ambientes internos;

Conjunto 2: portas, portas de elevador, escadas ascendentes, escadas descendentes, ambientes internos;

Conjunto 3: portas, portas duplas, portas semi abertas, portas abertas, portas de elevador, aescadas ascendentes, escadas descendentes, ambientes internos.

Lista 5.1 – Divisões de Classes.

A Tabela 5.3 mostra a quantidade de classes que ficaram para o Conjunto 1 da Lista 5.1. E a Tabela 5.2 mostra a quantidade de classes que ficaram para o Conjunto 2 da Lista 5.1.

Classe	Treinamento	Validação	Testes
Escadas Ascendentes	766	76	104
Escadas Descendentes	191	30	22
Portas	2884	300	344
Portas de Elevador	420	37	41
Ambientes Internos	2617	318	339

Tabela 5.2 – Divisão do Conjunto 2.

Classe	Treinamento	Validação	Testes
Escadas Ascendentes	957	106	126
Portas	3304	337	385
Ambientes Internos	2617	318	339

Tabela 5.3 – Divisão do Conjunto 1.

A Tabela 5.4 apresenta o resultado da acurácia de cada um dos modelos avaliados para cada uma das separações previstas, conforme a Lista 5.1. A Figura 5.2 exemplifica alguns dos resultados de reconhecimento obtidos a partir de alguns testes no modelo MobileNet utilizando o Conjunto 3.

Modelo	Conjunto 1	Conjunto 2	Conjunto 3
InceptionV3	0.836	0.789	0.706
IncepvionResNetV2	0.851	0.805	**0.718**
MobileNet	**0.868**	**0.812**	0.714
Resnet50	0.687	0.623	0.544
VGG16	0.816	0.777	0.694
VGG19	0.829	0.776	0.690
VGG16 Places	0.804	0.778	0.692
Xception	0.833	0.806	0.712

Tabela 5.4 – Resultados de Acurácia.

Analisando a Tabela 5.4, podemos perceber que embora algumas das classes do DSD (escadas e ambientes internos) possam pertencer a um contexto de reconhecimento de cenas, não houve ganho significativo ao utilizar pesos do Places2 [77]. Além disso, os resultados da separação de todas as classes não foram tão satisfatórios, podemos perceber que a diferença entre os resultados do Conjunto 1 e 2 não estão tão distantes, porém há uma maior discrepância com o Conjunto 3. Então, concluímos que as melhores classes para aplicar no experimento com detecção são: portas e portas de elevadores, que são mais distintas e fáceis de separar, além de escadas ascendentes e descendentes.

A Tabela 5.5 mostra a relação das camadas do topo usadas para cada um dos modelos utilizados. Devido a quantidade de dados, todas as camadas tem um nível de *dropout* entre elas. Além disso, o tamanho de saída da camada final varia de acordo com a divisão de classes de cada um dos três conjuntos, sendo 3 para o Conjunto 1, 5 para o Conjunto 2, e 8 para o Conjunto 3.

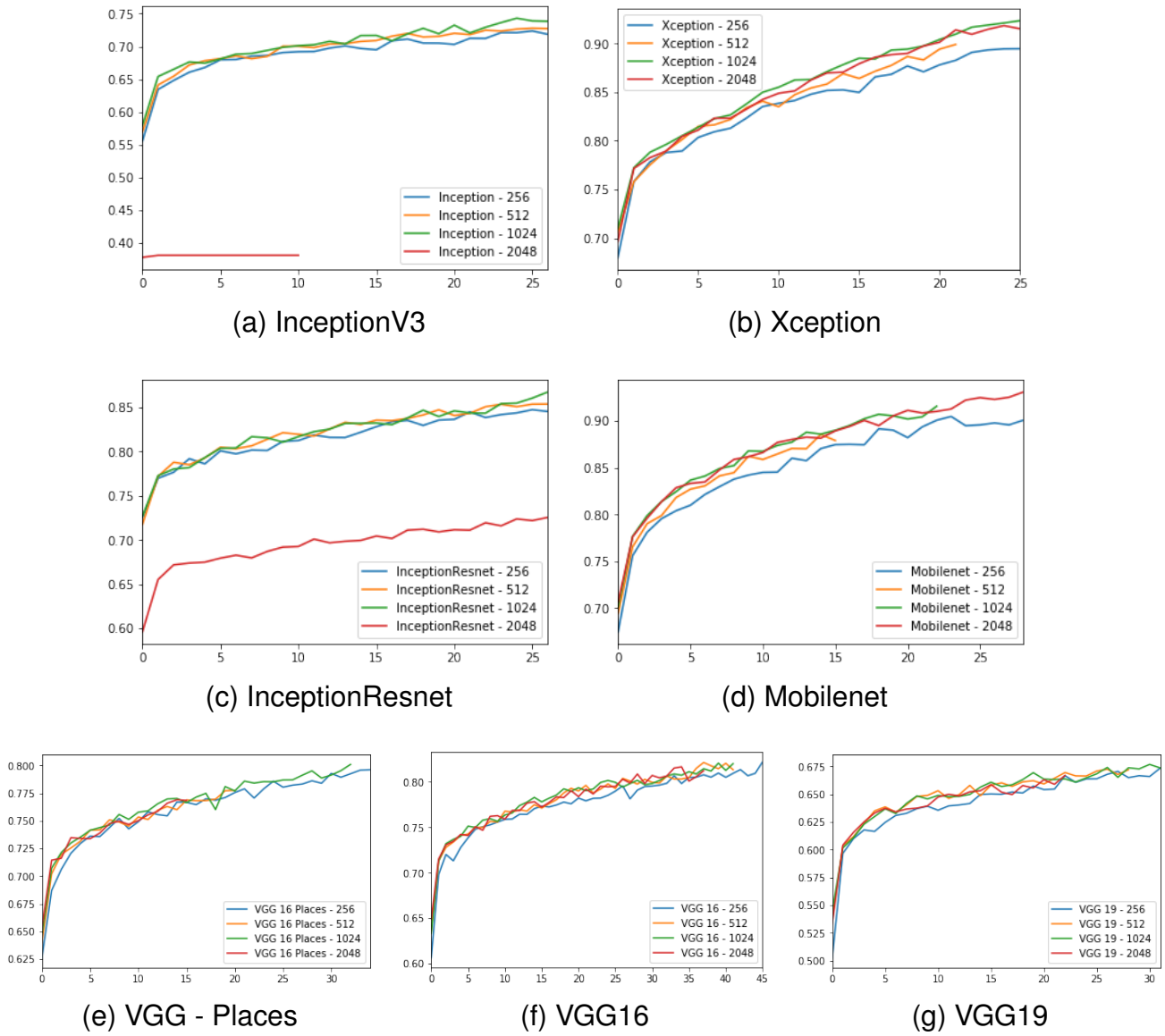
As redes com menor número de parâmetros utilizaram camadas maiores, porém de acordo com os resultados de acurácia no treinamento, expostos na Figura 5.1 alguns modelos não variaram muito alterando a quantidade de unidades de ativação. A MobileNet com o aumento da quantidade de unidades de ativação obteve uma mudança significativa no treinamento, já a InceptionV3, e a InceptionResnet com 2048 unidades tiveram os resultados piorados. E a VGG16 Places foi a única que diminuindo a quantidade de unidades de ativação teve melhorias significativas.

Visualizando os resultados qualitativos na Figura 5.2 conseguimos verificar que há uma certa dificuldade para distinguir a categoria de cada uma das portas. Na última imagem, por exemplo, verificamos que uma porta dupla, tendo apenas uma maçaneta gera uma certa dificuldade para o classificador prever se é uma porta, ou uma porta dupla.

Escadas também são difíceis de separar em ascendentes e descendentes. Para um reconhecimento correto da escada é necessário ter informações do ambiente em volta da mesma, o que pode dificultar a detecção. Essa análise é importante para fazer a ano-

Modelo	Camadas	Unidades de Ativação
InceptionV3	3	256
InceptionResNet	3	2048
MobileNet	3	2048
ResNet	1	128
VGG16	2	256
VGG19	2	512
VGG16 Places	1	128
Xception	3	256

Tabela 5.5 – Camadas do Topo dos Modelos.



(a) InceptionV3 (b) Xception

(c) InceptionResnet (d) Mobilenet

(e) VGG - Places (f) VGG16 (g) VGG19

Figura 5.1 – Resultados de Acurácia de Treinamento por Época.

tação das regiões das escadas, tentando não pegar apenas o espaço da escada, mas também objetos próximos à ela, como, por exemplo, o corrimão.

Figura 5.2 – Exemplos de Classificação de Imagens com Mobilenet.

Depois da avaliação de acurácia, medimos as redes treinadas no Conjunto 3, em uma aplicação em tempo real, executando em um *laptop* Dell Inspiron 7000, com uma NVidia Geforce 940MX, com 4GB de memória. A Tabela 5.6 apresenta os resultados da medição da taxa de *frames* neste experimento.

Modelo	FPS (*Frames Per Second*)	Tempo de Classificação em segundos
MobileNet	18.66	0.0320
InceptionV3	11.56	0.0821
IncepvionResNetV2	6.45	0.1960
Resnet50	9.88	0.0863
VGG16	8.71	0.0866
VGG16 Places	8.71	0.0866
VGG19	6.53	0.1188
Xception	11.82	0.0799

Tabela 5.6 – Tempo de Classificação.

Analisando a Tabela 5.6 verificamos que a InceptionV3, Xception, e a MobileNet alcançaram resultados satisfatórios de classificação e taxa de *frames*, mostrando a viabilidade do uso em aplicações em tempo real. Os resultados da MobileNet são satisfatórios, mesmo tendo em vista que ela contém o menor número de parâmetros. Isso pode ser devido ao fato que há uma quantidade menor de dados de treinamento do que usualmente utilizado em conjuntos de dados desse âmbito, provendo uma maior dificuldade no treinamento dos demais métodos. Mas, no contexto deste estudo ela provê um ótimo resultado, e também uma boa portabilidade para ser usada em aplicações em tempo real, como em dispositivos móveis por exemplo.

5.2 Detecção de Portas e Escadas

Conforme mencionado na Seção 4.1 para avaliar o modelo corretamente com a detecção de portas e escadas no ambiente, uma avaliação no conjunto DSD é essencial. Nossa avaliação foi feita com o método de detecção YOLO, com diversos modelos de redes neurais convolucionais. Para isso, estabelecemos um período de 96 horas (4 dias) para o treinamento dos métodos, e realizamos o treinamento no mesmo hardware e ambiente de software.

Para o treinamento utilizamos uma implementação da Darknet-19 [50] feita em Keras [3]. A implementação já oferece o método de pré processamento das imagens de treinamento sugerida pelo Pascal VOC [14].

Todas as redes fornecidas pela implementação já são pré-treinadas no Microsoft COCO [37], que contém melhores pesos para a tarefa de detecção de objetos. Para a estratégia de pré processamento das imagens são utilizados: *random crops* em 90% da área de imagem, *flips* horizontais sobre a imagem; e *blurrings*, com *gaussian blur*, *median blur* e average blur.

Conforme mencionado na Seção 4.3.2, a etapa de treinamento não chegou em um ponto de convergência quando foi utilizado o subconjunto de imagens A. Ou seja, os valores de saída da função de custo por época estabilizaram entre 3.500, e 3.300. Mesmo fazendo algumas otimizações no subconjunto de imagens A durante o processo de treinamento (ver Seção 4.1), não conseguimos chegar a um ponto de convergência. Sendo assim, não foram feitas avaliações com este subconjunto de imagens. Mas, com uma investigação neste subconjunto de imagens constatou-se que: há impureza nas imagens; muitas das imagens tem resoluções completamente diferentes; muitas das imagens apresentam oclusão de portas e escadas; e também pode haver problemas pela distribuição de classes não ter ficado uniforme.

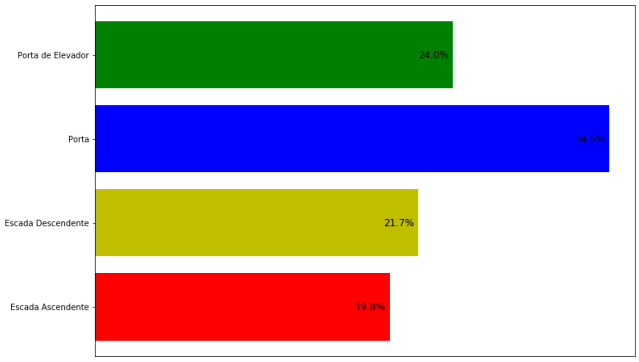


Figura 5.3 – Distribuição por Classes do Conjunto DSD Completo.

[3] https://github.com/experiencor/keras-yolo2

Analisando a Figura 5.3, pode-se perceber que não há uma distribuição uniforme da quantidade de imagens por classe, pois mais da metade dos objetos são portas. As imagens do subconjunto B possuem uma distribuição de classes mais uniforme, conforme mostra a Figura 5.4, pois não foram incluídas diversas classes de portas.

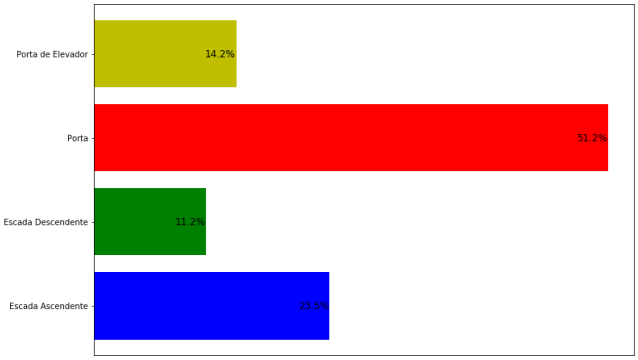


Figura 5.4 – Distribuição por Classe das Imagens do Subconjunto B.

Além disso, a Figura 5.5, mostra as diferentes resoluções por conjunto de classes, na qual **Outros** correspondem a resoluções diferentes com um número de representatividade menor que 3%. Podemos perceber a quantidade de imagens com distribuição de tamanhos completamente diferentes, por exemplo, chegando a 73.27% de imagens de resoluções diferentes para as portas.

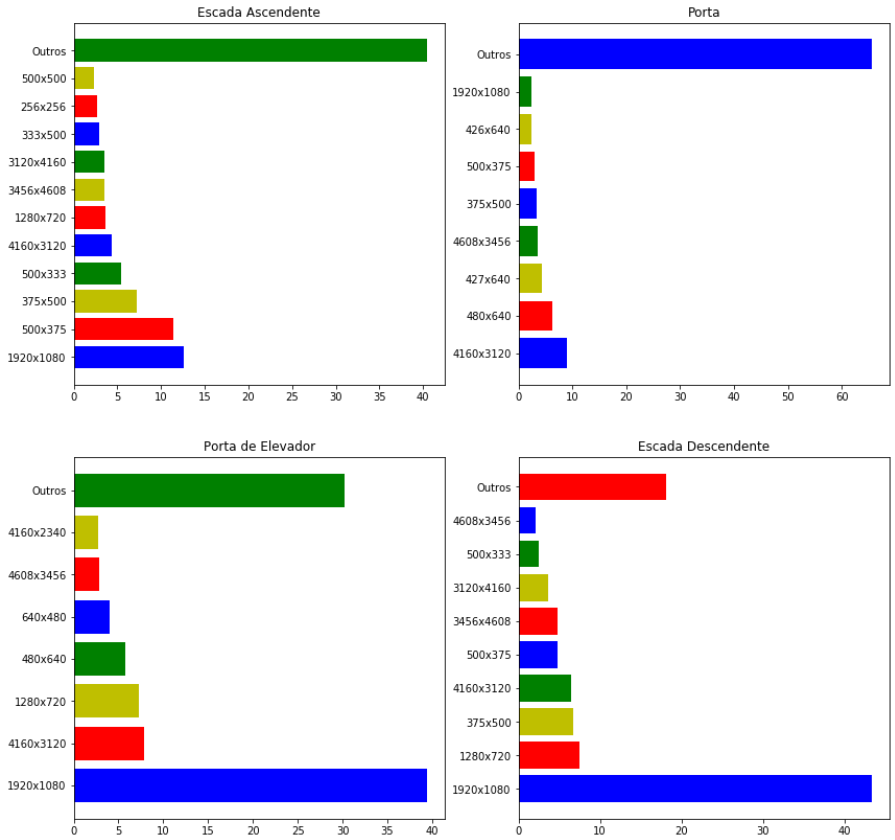


Figura 5.5 – Distribuição das resoluções por classe.

A partir desses problemas, optamos por utilizar o conjunto de imagens coletadas em campo, o subconjunto B, para treinamento e avaliação dos resultados. A divisão do conjunto de dados foi de 70% das imagens para treinamento, 15% para validação, e 15% para testes.

O total de imagens utilizadas foi de 3, 516, com 2, 426 utilizadas para treinamento, 544 para validação, e 546 para testes. A Tabela 5.7 mostra em detalhes como ficou a divisão do conjunto de dados para cada classe, com a divisão de treinamento, validação, e testes.

A distribuição final desse conjunto ficou mais uniforme, mesmo havendo uma quantidade mais significativa de portas em todos os conjuntos. Porém, os modelos chegaram a um estado ótimo de convergência, com valores de *loss* abaixo de 0.600.

Classes	Treinamento	Validação	Testes
Escadas Ascendentes	596	128	128
Escadas Descendentes	655	140	140
Portas	1040	223	223
Portas de Elevador	596	155	155

Tabela 5.7 – Distribuição dos dados do DSD.

Os resultados apresentados na Tabela 5.8 mostram que o método Full YOLO provê o melhor resultado do mAP total. Porém, o modelo apresentou o melhor AP apenas para as classes de escadas, e o segundo melhor AP para as classes de portas. Entretanto, dado a capacidade do modelo, esperávamos que ele teria o melhor resultado de AP para todas as classes.

O segundo melhor mAP foi da Inception V3, com apenas uma pequena diferença para a Full YOLO. Entretanto, ele obteve o melhor resultado de AP para as classes de portas, e obteve resultados competitivos para as classes de escadas.

Ambos os métodos, Full YOLO e Inception V3 com YOLO, apresentaram resultados promissores e similares, e ambos são capazes de ser embarcados em smartphones, considerando o tempo de predição apresentado na Tabela 5.9. Os modelos mais simples, Tiny YOLO e MobileNet com YOLO, produziram os piores resultados de mAP, porém ambos produziram os melhores tempos de predição.

Método	Escadas Ascendentes (AP)	Escadas Descendentes (AP)	Portas (AP)	Portas de Elevador (AP)	Total (mAP)
Full YOLO	**92.67**	**91.58**	89.85	93.78	**91.97**
Tiny YOLO	73.65	69.34	74.82	90.24	77.01
Inception v3	88.72	86.40	**92.69**	**94.40**	90.55
MobileNet	70.49	76.97	59.35	45.35	63.04

Tabela 5.8 – Resultados dos teste de detecção.

A Tabela 5.9 confirma nossa intuição que o Tiny YOLO alcança os menores tempos de predição. Porém, o Full YOLO também apresenta resultados impressionantes, com 39 de

FPS, permitindo que o modelo consiga executar em tempo real. Considerando a velocidade, a Inception V3 também é rápida, porém a Full YOLO é 1.5 vezes mais rápida que ela, a MobileNet apresenta um ganho de 1.2 vezes sobre a Full YOLO, e a Tiny YOLO apresenta um ganho de 1.25 vezes sobre a MobileNet.

Método	Tempo de Detecção (em Segundos)	FPS
Full YOLO	0.0255 ± 0.0022	39
Tiny YOLO	$\mathbf{0.0169} \pm 0.0017$	**59**
Inception v3	0.0367 ± 0.0064	27
MobileNet	0.0209 ± 0.0071	47

Tabela 5.9 – Tempo de Detecção.

Além disso, tendo em vista que os modelos podem ser embarcados em diversos tipos de hardware, com dispositivos com menor capacidade de processamento é possível utilizar o método Tiny YOLO, por exemplo. Assim, é possível garantir que o método execute em tempo real, sacrificando um pouco de precisão.

Para ilustrar os resultados qualitativos obtidos com os experimentos realizados nesse estudo, a Figura 5.6 apresenta 4 imagens selecionadas aleatoriamente distribuídas em 4 colunas, sendo que cada coluna corresponde a um dos métodos de detecção. As imagens foram selecionadas do nosso conjunto de testes, e compara a predição (em amarelo, ou laranja) em constaste com a anotação (em azul). Abaixo de cada imagem é informada a confiança (valor percentual dado pela predição) e o limiar aplicado foi de 0.3.

Analisando as predições na Figura 5.6, percebemos que os resultados são similares e as maiores diferenças entre as posições das caixas de detecção. A MobileNet não conseguiu detectar uma das amostras, e em relação à confiança, percebemos que a Tiny YOLO, e a MobileNet apresentam a maior discrepância.

A Figura 5.7 ilustra as performances de todos os modelos em contraste com as curvas de *precision*, e *recall* (PR). O limiar de confiança varia de 0 a 1, com incrementos de 0.01. Assumimos que as predições são corretas com o IoU sendo maior que 0.5 (o mesmo limiar do Pascal VOC [14]).

A performance geral de cada modelo, pode ser sintetizada pelo mAP e o AP de cada classe. Percebe-se que a Tiny YOLO e a MobileNet apresentam pontos de dispersão no PR (Figuras 5.7c, 5.7d), especialmente quando comparadas com a Full YOLO, e a Inception V3 (Figuras 5.7a, 5.7b), que mostram pontos de distribuição densos. Isso é dado pela concentração de maiores pontos de confiança nas predições corretas dados pelo Full YOLO e a Inception V3.

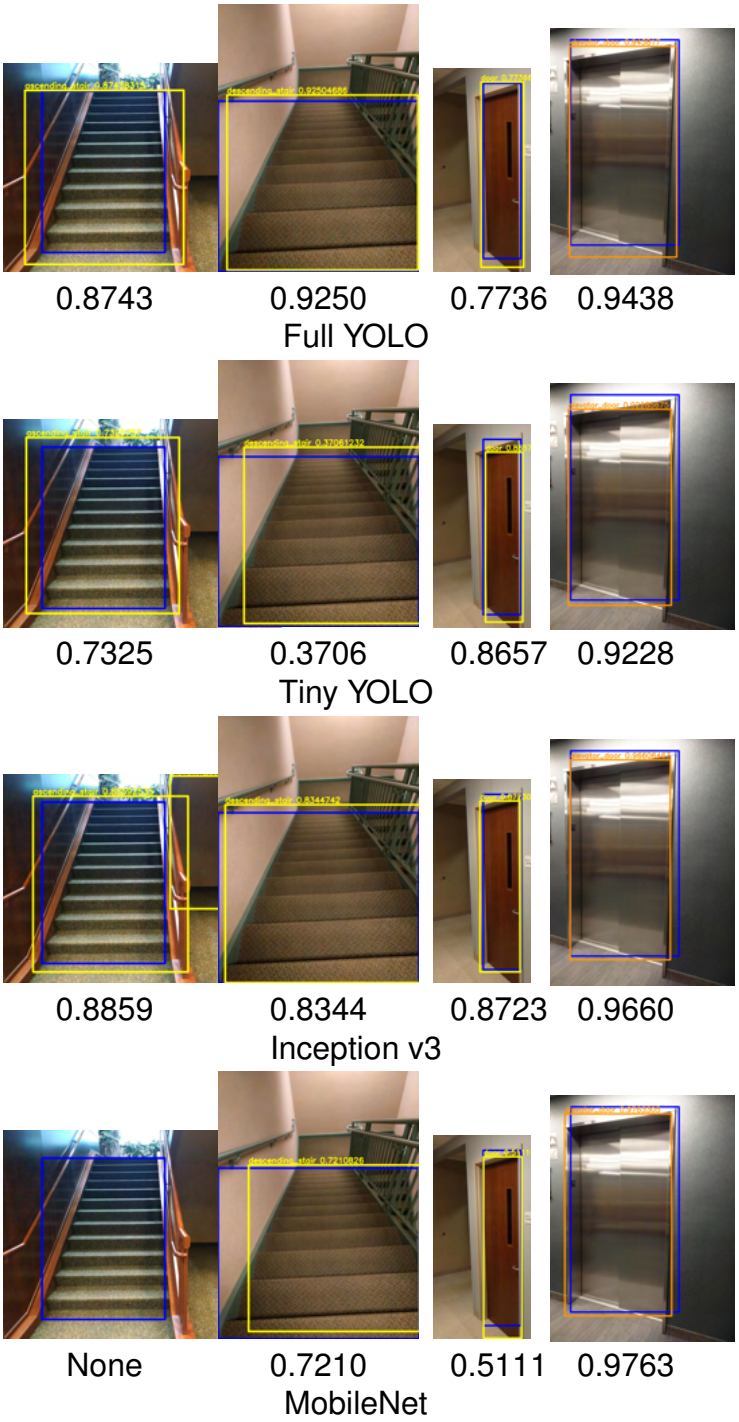
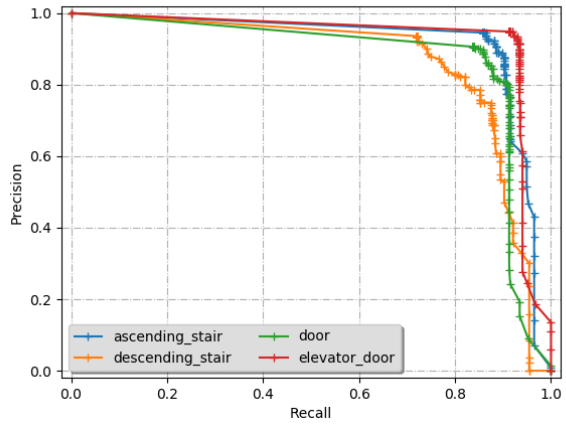
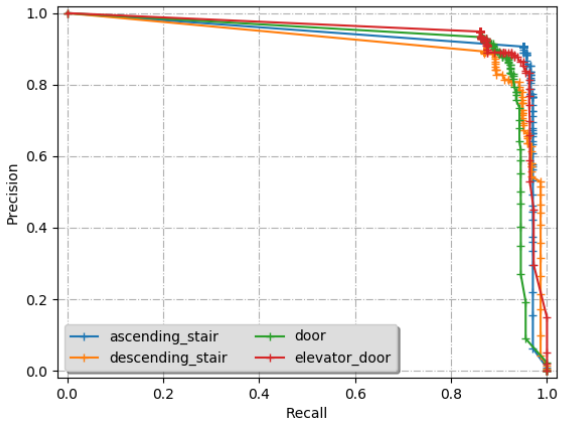


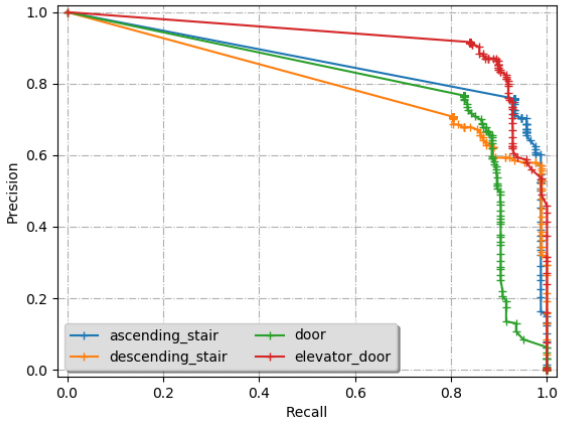
Figura 5.6 – Avaliação qualitativa dos Modelos.



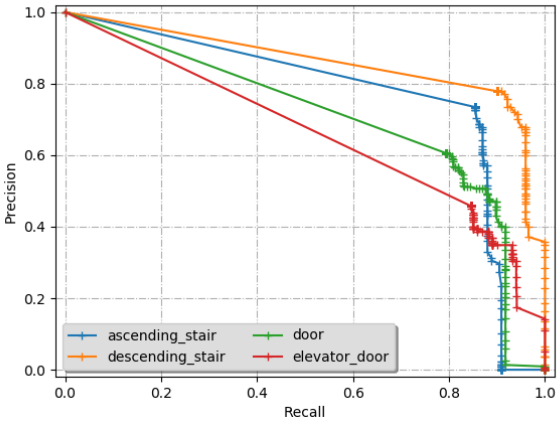
(a) Full YOLO



(b) Inception v3



(c) Tiny YOLO



(d) MobileNet

Figura 5.7 – Curvas de Precision e Recall sobre os Métodos.

Por fim, para avaliar nosso experimento em um cenário real, testamos ambos a Full YOLO, e a Tiny YOLO em uma breve navegação sobre um dos prédios da PUC. Disponibilizamos o resultado no Youtube [4]. Avaliando o vídeo é possível perceber que, mesmo em uma primeira avaliação, e no treinamento apenas com o conjunto de imagens do ambiente, os modelo são capazes de detectar corretamente os objetos. Porém, há um pequeno número de falso positivos sobre os *frames*, principalmente nas escadas, e detecções de escadas no chão. O número e falso positivos para o método Tiny YOLO é maior, conforme esperado, tendo em vista os resultados de mAP, e as avaliações qualitativas dos modelos.

[4] https://youtu.be/3ZEzURb6_vM

6. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho partiu da seguinte pergunta de pesquisa: É possível desenvolver um modelo baseado em *deep learning* que auxilie deficientes visuais na localização de escadas e portas em ambientes internos, usando técnicas de visão computacional? Para responder a esta questão, desenvolvemos um modelo baseado em visão computacional, que utiliza técnicas recentes de *deep learning* para classificação e detecção de objetos em imagens. Para isso foi criado um conjunto de dados contendo imagens de portas e escadas, que foi avaliado com técnicas de classificação e detecção.

Considerando os objetivos específicos e os resultados obtidos, podemos afirmar que, através de uma prova de conceito, é possível detectar os objetos propostos (porta e escada), trazendo informações úteis para seus usuários. Sendo assim, as contribuições deste trabalho são: um conjunto de dados [1] com 12.501 imagens, contendo portas, escadas e ambientes internos; uma avaliação deste conjunto com diferentes modelos de redes neurais convolucionais para classificação de imagens; uma avaliação do conjunto com o método YOLO para a detecção de objetos, com diferentes modelos de redes neurais convolucionais.

A construção do conjunto de dados foi feita de forma iterativa, através da realização de experimentos com estes dados usando técnicas de classificação de imagens e detecção objetos. Concluímos que a construção de um conjunto de dados desta maneira deve ser bem avaliada, pois não é uma tarefa simples. São necessárias milhares de imagens com anotações de qualidade, e bem padronizadas. Além disso, é importante garantir uma distribuição de classes uniforme, principalmente quando se tem uma quantidade menor de imagens, para auxiliar na avaliação do conjunto.

Os resultados das técnicas de classificação (Seção 5.1) apresentados neste trabalho, provaram que o modelo pode ser treinado a partir de pesos pré gerados em treinamentos com conjuntos tanto de reconhecimento de objetos, quanto de reconhecimento de cenas. Além disso, verificamos que com o uso da técnica de *transfer learning* é possível obter resultados de acurácia acima de 80%, e concluímos que a divisão de classes mais confiável para esse contexto é: portas; portas de elevador; escadas ascendentes; escadas descendentes; e ambientes internos.

[1] https://tinyurl.com/gvc-dsd

A partir dos resultados obtidos nos experimentos de classificação, anotamos o conjunto de dados com as regiões de cada objeto, e realizamos um experimento sobre técnicas de detecção de objetos com o método YOLO.

Apesar dos problemas com o treinamento no subconjunto A (Seção 4.2), os resultados (Seção 5.2) foram satisfatórios, com valores de mAP acima de 90%, e com tempo de processamento com taxas de FPS acima de 30. Além disso, avaliamos os resultados obtidos em um vídeo, com uma taxa de *frames* acima dos 30%, conforme o esperado. Analisando as detecções neste vídeo, percebemos que o modelo já detecta corretamente os objetos, embora haja ainda alguns falso positivos.

Como trabalhos futuros pretendemos conduzir uma nova coleta e anotação do conjunto de dados, para trazer imagens em demais ambientes internos aprimorando o conjunto existente, e limpar os outliers. Também pretendemos avaliar esse novo conjunto em novos métodos de detecção, tais como a SSD [39] e a Faster RCNN [16]. Além disso, considerando o contexto e a motivação para o desenvolvimento do trabalho, pretendemos explorar algoritmos de coerência temporal, para amenizar a taxa de erros entre os *frames* de um vídeo. E por fim pretendemos embarcar a solução encontrada em dispositivos móveis, para posteriormente fazermos testes com usuários.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ahonen, T.; Hadid, A.; Pietikäinen, M. “Face description with local binary patterns: Application to face recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28–12, Out 2006, pp. 2037–2041.

[2] Asad, M.; Ikram, W. “Smartphone based guidance system for visually impaired person”. In: 3rd International Conference on Image Processing Theory, Tools and Applications, 2012, pp. 442–447.

[3] Bashiri, F. S.; LaRose, E.; Peissig, P.; Tafti, A. P. “MCIndoor20000: A fully-labeled image dataset to advance indoor objects detection”, *Data in Brief*, vol. 17, Abr 2018, pp. 71–75.

[4] Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L. V. “Speeded-up robust features (surf)”, *Computer Vision and Image Understanding*, vol. 110–3, Jun 2008, pp. 346–359.

[5] Bell, S.; Zitnick, C. L.; Bala, K.; Girshick, R. B. “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks”, *CoRR*, vol. abs/1512.04143, Dez 2015, pp. 1–11.

[6] Britz, D. “Understading convolutional neural networks for nlp”. Capturado em: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp, Acesso em: Nov 2016.

[7] Chen, W.; Qu, T.; Zhou, Y.; Weng, K.; Wang, G.; Fu, G. “Door recognition and deep learning algorithm for visual based robot navigation”. In: IEEE International Conference on Robotics and Biomimetics, 2014, pp. 1793–1798.

[8] Chollet, F. “Xception: Deep learning with depthwise separable convolutions”, *CoRR*, vol. abs/1610.02357, Out 2016, pp. 1–8.

[9] Cinbis, R. G.; Verbeek, J. J.; Schmid, C. “Weakly supervised object localization with multi-fold multiple instance learning”, *CoRR*, vol. abs/1503.00949, Mar 2015, pp. 1–15.

[10] Claesen, M.; Moor, B. D. “Hyperparameter search in machine learning”, *CoRR*, vol. abs/1502.02127, Fev 2015, pp. 0–5.

[11] Dai, J.; Li, Y.; He, K.; Sun, J. “R-FCN: object detection via region-based fully convolutional networks”, *CoRR*, vol. abs/1605.06409, Mai 2016, pp. 1–11.

[12] Dalal, N.; Triggs, B. “Histograms of oriented gradients for human detection”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 886–893.

[13] Elhariri, E.; El-Bendary, N.; Hassanien, A. E.; Snasel, V. “An Assistive Object Recognition System for Enhancing Seniors Quality of Life”, *Procedia Computer Science*, vol. 65, Out 2015, pp. 691–700.

[14] Everingham, M.; Gool, L. V.; Williams, C. K. I.; Winn, J.; Zisserman, A. “The pascal visual object classes (voc) challenge”. Capturado em: http://host.robots.ox.ac.uk/pascal/VOC/, Acesso em: Nov 2016.

[15] Galera, B. Q.; Prieto, S. A.; Adan, A.; Bosché, F. “Door detection in 3d coloured point clouds of indoor environments”, *Automation in Construction*, vol. 85, Jan 2018, pp. 146 – 166.

[16] Girshick, R. B. “Fast R-CNN”. In: IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[17] Girshick, R. B.; Donahue, J.; Darrell, T.; Malik, J. “Rich feature hierarchies for accurate object detection and semantic segmentation”, *CoRR*, vol. abs/1311.2524, Nov 2013, pp. 1–21.

[18] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning”. MIT Press, 2016, 787p.

[19] Harms, H.; Rehder, E.; Schwarze, T.; Lauer, M. “Detection of ascending stairs using stereo vision”. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 2496–2502.

[20] He, K.; Zhang, X.; Ren, S.; Sun, J. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: 13th European Conference on Computer Vision, 2014, pp. 346–361.

[21] He, K.; Zhang, X.; Ren, S.; Sun, J. “Deep residual learning for image recognition”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[22] He, K.; Zhang, X.; Ren, S.; Sun, J. “Spatial pyramid pooling in deep convolutional networks for visual recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37–9, Set 2015, pp. 1904–1916.

[23] Hoffman, J.; Guadarrama, S.; Tzeng, E.; Donahue, J.; Girshick, R. B.; Darrell, T.; Saenko, K. “LSDA: large scale detection through adaptation”, *CoRR*, vol. abs/1407.5035, Jul 2014, pp. 1–9.

[24] Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, *CoRR*, vol. abs/1704.04861, Abr 2017, pp. 1–9.

[25] Hui, J. “Map (mean average precision) for object detection”. Capturado em: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, Acesso em: Abr 2018.

[26] Jafri, R.; Ali, S. A.; Arabnia, H. R.; Fatima, S. “Computer vision-based object recognition for the visually impaired in an indoors environment: a survey”, *Visual Computer*, vol. 30–11, Nov 2014, pp. 1197–1222.

[27] Jose, J.; Farrajota, M.; Rodrigues, J. M.; Buf, J. H. D. “The SmartVision local navigation aid for blind and visually impaired persons”, *International Journal of Digital Content Technology and its Applications*, vol. 5, Mai 2011, pp. 362–375.

[28] Kim, K.; Cheon, Y.; Hong, S.; Roh, B.; Park, M. “PVANET: deep but lightweight neural networks for real-time object detection”, *CoRR*, vol. abs/1608.08021, Ago 2016, pp. 1–7.

[29] Kingma, D. P.; Ba, J. “Adam: A method for stochastic optimization”, *CoRR*, vol. abs/1412.6980, Dez 2014, pp. 1–15.

[30] Krasin, I.; Duerig, T.; Alldrin, N.; Ferrari, V.; Abu-El-Haija, S.; Kuznetsova, A.; Rom, H.; Uijlings, J.; Popov, S.; Kamali, S.; Malloci, M.; Pont-Tuset, J.; Veit, A.; Belongie, S.; Gomes, V.; Gupta, A.; Sun, C.; Chechik, G.; Cai, D.; Feng, Z.; Narayanan, D.; Murphy, K. “Openimages: A public dataset for large-scale multi-label and multi-class image classification.” Capturado em: https://storage.googleapis.com/openimages/web/index.html, Acesso em: Abr 2018.

[31] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In: 26th Annual Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.

[32] Kumar, R.; Meher, S. “A novel method for visually impaired using object recognition”. In: International Conference on Communications and Signal Processing (ICCSP), 2015, pp. 0772–0776.

[33] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86–11, Dez 1998, pp. 2278–2324.

[34] LeCun, Y.; Cortes, C. “MNIST handwritten digit database”. Capturado em: http://yann.lecun.com/exdb/mnist/, Acesso em: Maio 2017.

[35] Lee, H.; Kwon, H.; Bency, A. J.; Nothwang, W. D. “Fast object localization using a CNN feature map based multi-scale search”, *CoRR*, vol. abs/1604.03517, Abr 2016, pp. 1–16.

[36] Lenc, K.; Vedaldi, A. “R-CNN minus R”, *CoRR*, vol. abs/1506.06981, Jun 2015, pp. 1–9.

[37] Lin, T.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. “Microsoft COCO: common objects in context”, *CoRR*, vol. abs/1405.0312, Mai 2014, pp. 1–15.

[38] Littlestone, N. “Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm”, *Machine Learning*, vol. 2–4, Abr 1988, pp. 285–318.

[39] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Cheng-Yang; Berg, A. C. “Ssd: Single shot multibox detector”. In: 14th European Conference on Computer Vision, 2016, pp. 21–37.

[40] Lowe, D. G. “Distinctive image features from scale-invariant keypoints”, *International Journal of Computer Vision*, vol. 60–2, Nov 2004, pp. 91–110.

[41] Mao, J.; Xu, W.; Yang, Y.; Wang, J.; Yuille, A. L. “Deep captioning with multimodal recurrent neural networks (m-rnn)”, *CoRR*, vol. abs/1412.6632, Dez 2014, pp. 1–17.

[42] Mekhalfi, M. L.; Melgani, F.; Bazi, Y.; Alajlan, N. “Toward an assisted indoor scene perception for blind people with image multilabeling strategies”, *Expert Systems with Applications*, vol. 42–6, Abr 2015, pp. 2907–2918.

[43] Mekhalfi, M. L.; Melgani, F.; Zeggada, A.; De Natale, F. G. B.; Salem, M. A. M.; Khamis, A. “Recovering the sight to blind people in indoor environments with smart technologies”, *Expert Systems with Applications*, vol. 46, Mar 2016, pp. 129–138.

[44] Mitchell, T. M. “Machine Learning”. McGraw-Hill, 1997, 432p.

[45] Moreno, M.; Shahrabadi, S.; José, J.; Du Buf, J. M. H.; Rodrigues, J. M. F. “Realtime local navigation for the blind: Detection of lateral doors and sound interface”, *Procedia Computer Science*, vol. 14, Dez 2012, pp. 74–82.

[46] Najibi, M.; Rastegari, M.; Davis, L. S. “G-CNN: an iterative grid based object detector”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, pp. 2369–2377.

[47] Organization, W. H. “Blindness and visual impairment: Global facts”. Capturado em: http://www.iapb.org/vision-2020/global-facts, Acesso em: Nov 2016.

[48] Ouyang, W.; Luo, P.; Zeng, X.; Qiu, S.; Tian, Y.; Li, H.; Yang, S.; Wang, Z.; Xiong, Y.; Qian, C.; Zhu, Z.; Wang, R.; Loy, C. C.; Wang, X.; Tang, X. “Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection”, *CoRR*, vol. abs/1409.3505, Set 2014, pp. 1–13.

[49] Papandreou, G.; Kokkinos, I.; Savalle, P. “Untangling local and global deformations in deep convolutional networks for image classification and sliding window detection”, *CoRR*, vol. abs/1412.0296, Nov 2014, pp. 1–13.

[50] Redmon, J.; Divvala, S.; Girshick, R. B.; Farhadi, A. “You only look once: Unified, real-time object detection”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[51] Ren, S.; He, K.; Girshick, R. B.; Sun, J. “Faster r-cnn: Towards real-time object detection with region proposal networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39–6, Jun 2017, pp. 1137–1149.

[52] Ren, S.; He, K.; Girshick, R. B.; Zhang, X.; Sun, J. “Object detection networks on convolutional feature maps”, *CoRR*, vol. abs/1504.06066, Abr 2015, pp. 1–8.

[53] Rusk, N. “Deep learning”, *Nature Methods*, vol. 13–1, Mai 2015, pp. 35–35.

[54] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L. “Imagenet large scale visual recognition challenge”, *International Journal of Computer Vision*, vol. 115–3, Abr 2015, pp. 211–252.

[55] Sanjivani, S.; Keshri, V.; Kamal, M. “A Survey on Approaches of Object Detection”, *International Journal of Computer Applications*, vol. 6518, Mar 2013, pp. 975–8887.

[56] Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. “Overfeat: Integrated recognition, localization and detection using convolutional networks”, *CoRR*, vol. abs/1312.6229, Dez 2013, pp. 1–16.

[57] Serrão, M.; Rodrigues, J. M. F.; Rodrigues, J. I.; Du Buf, J. M. H. “Indoor localization and navigation for blind persons using visual landmarks and a gis”, *Procedia Computer Science*, vol. 14, Dez 2012, pp. 65–73.

[58] Shalby, M.; Salem, M. A.-M. M.; Khamis, A.; Melgani, F. “Geometric model for vision-based door detection”. In: 9th IEEE International Conference on Computer Engineering and Systems, 2015, pp. 41–46.

[59] Shen, Z.; Xue, X. “Do more dropouts in pool5 feature maps for better object detection”, *CoRR*, vol. abs/1409.6911, Set 2014, pp. 1–9.

[60] Simonyan, K.; Zisserman, A. “Very deep convolutional networks for large-scale image recognition”. In: 3rd International Conference on Learning Representations, 2015, pp. 1–14.

[61] Skulimowski, P.; Owczarek, M.; Strumillo, P. “Door detection in images of 3d scenes in an electronic travel aid for the blind”. In: 10th International Symposium on Image and Signal Processing and Analysis, 2017, pp. 189–194.

[62] Szegedy, C.; Ioffe, S.; Vanhoucke, V. “Inception-v4, inception-resnet and the impact of residual connections on learning”, *CoRR*, vol. abs/1602.07261, Fev 2016, pp. 1–12.

[63] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. “Rethinking the inception architecture for computer vision”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[64] Szegedy, C.; Wei, L.; Yangqing, J.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. “Going deeper with convolutions”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[65] Szeliski, R. “Computer Vision: Algorithms and Applications”. Springer-Verlag New York, Inc., 2010, 812p.

[66] Sánchez, J.; Aguayo, F. A.; Hassler, T. M. “Independent outdoor mobility for the blind”. In: Virtual Rehabilitation, 2007, pp. 114–120.

[67] Tapu, R.; Mocanu, B.; Zaharia, T. “A computer vision system that ensure the autonomous navigation of blind people”. In: E-Health and Bioengineering Conference, 2013, pp. 1–4.

[68] University, S. “Convolutional neural networks (cnns / convnets)”. Capturado em: http://cs231n.github.io/convolutional-networks/, Acesso em: Nov 2016.

[69] Vidal-Naquet, M.; Ullman, S. “Object recognition with informative features and linear classification”. In: IEEE International Conference on Computer Vision, 2003, pp. 281–288.

[70] Viola, P.; Jones, M. “Rapid object detection using a boosted cascade of simple features”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, pp. I–511.

[71] Wang, S.; Tian, Y. “Detecting stairs and pedestrian crosswalks for the blind by rgbd camera”. In: IEEE International Conference on Bioinformatics and Biomedicine Workshops, 2012, pp. 732–739.

[72] Wang, X.; Yang, M.; Zhu, S.; Lin, Y. “Regionlets for generic object detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37–10, Out 2015, pp. 2071–2084.

[73] Xiao, J.; Ehinger, K. A.; Hays, J.; Torralba, A.; Aude, O. “Sun database: Exploring a large collection of scene categories”, *International Journal of Computer Vision*, vol. 119–1, Ago 2016, pp. 3–22.

[74] Yan, J.; Yu, Y.; Zhu, X.; Lei, Z.; Li, S. Z. “Object detection by labeling superpixels”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 5107–5116.

[75] Zeng, X.; Ouyang, W.; Yan, J.; Li, H.; Xiao, T.; Wang, K.; Liu, Y.; Zhou, Y.; Yang, B.; Wang, Z.; Zhou, H.; Wang, X. “Crafting gbd-net for object detection”, *CoRR*, vol. abs/1610.02579, Out 2016, pp. 1–16.

[76] Zhou, B.; Lapedriza, À.; Xiao, J.; Torralba, A.; Oliva, A. “Learning deep features for scene recognition using places database”. In: 27th International Conference on Neural Information Processing Systems, 2014, pp. 487–495.

[77] Zhou, B.; Àgata Lapedriza; Khosla, A.; Oliva, A.; Torralba, A. “Places: A 10 million image database for scene recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40–6, Jun 2018, pp. 1452–1464.

[78] Zhu, Y.; Urtasun, R.; Salakhutdinov, R.; Fidler, S. “segdeepm: Exploiting segmentation and context in deep neural networks for object detection”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 4703–4711.

[79] Zou, W. Y.; Wang, X.; Sun, M.; Lin, Y. “Generic object detection with dense neural patterns and regionlets”. In: British Machine Vision Conference, 2014, pp. 1–11.