UNIVERSIDADE FEDERAL DE SANTA CATARINA

REDES NEURAIS EM DISPOSITIVOS RASPBERRY PI PARA DETECÇÃO DE PESSOAS

Caio Cargnin Cardoso

Florianópolis - SC

2018 / 2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

REDES NEURAIS EM DISPOSITIVOS RASPBERRY PI PARA DETECÇÃO DE PESSOAS

Caio Cargnin Cardoso

Trabalho de Conclusão de Curso submetido ao Programa de graduação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Sistemas de Informação

Florianópolis - SC

2018 / 2

CAIO CARGNIN CARDOSO

REDES NEURAIS EM DISPOSITIVOS RASPBERRY PI PARA DETECÇÃO DE PESSOAS

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Sistemas de Informação”, e aprovado em sua forma final pelo Curso de Bacharelado em Sistemas de Informação.

_____

Prof. Renato Cislaghi, Dr.
Coordenador

_____

Prof. Elder Rizzon Santos, Dr.
Orientador

Banca Examinadora:

_____

Prof.ª Jerusa Marchi, Dr.ª

_____

Thiago Angelo Gelaim, Me.

RESUMO

Os avanços observados na área da Visão Computacional nos últimos anos, propiciados pela larga utilização de Redes Neurais Convolucionais, fazem-se notar em tarefas de classificação de imagens, localização de objetos e detecção de objetos. Técnicas consideradas o estado-da-arte nestes tipos de problema baseiam-se na utilização de arquiteturas de redes neurais compostas por diversas camadas de neurônios, e consequentemente, por uma grande quantidade de parâmetros. O treinamento e a inferência realizada com redes como estas demandam grande capacidade computacional, não só em termos de processamento, mas também em consumo de memória. As exigências provenientes de arquiteturas consideradas o estado-da-arte inviabilizam a utilização de tecnologias como estas em ambientes caracterizados pela restrição de recursos computacionais, como sistemas embarcados ou dispositivos utilizados no contexto da Internet das Coisas (IoT), onde a sua aplicação em soluções envolvendo segurança residencial, nas quais tempos de resposta eficientes são requisitos essenciais, pode ser comprometida. Enquanto modelos de classificação ou detecção de objetos baseados em redes neurais podem ser utilizados neste contexto, delegando-se a computação para computadores remotos, o tempo de resposta envolvendo requisições de rede para servidores alocados remotamente pode ser inaceitável para este tipo de aplicação. A popularidade da plataforma Raspberry Pi no ambiente de IoT, aliada ao poder computacional relativamente alto, para dispositivos utilizados neste contexto, fazem com que o desafio de se utilizar Redes Neurais

Convolucionais para classificação e detecção de objetos, executando-se a inferência diretamente a partir destes dispositivos torne-se uma tarefa viável. Este trabalho propõe a busca por uma solução baseada em modelos de rede neural, dentre um conjunto de arquiteturas caracterizadas por uma demanda computacional menor, capazes de realizar detecção de pessoas em imagens com baixo tempo de execução e consumo de memória, no contexto de uma aplicação em segurança residencial, diretamente a partir de um Raspberry Pi, apresentando eficiência aceitável. Mais especificamente, são realizados experimentos com os modelos de detecção de objetos TinyYolo, MobileSSD e Pelee, comparando-se tempo de execução e eficiência na tarefa de detecção. Além disso, experimenta-se uma solução composta por uma etapa inicial com um modelo de classificação, seguida de uma etapa posterior utilizando um modelo de detecção de objetos, visando a melhoria do desempenho tanto em tempo de execução quanto na acurácia na tarefa de detecção de pessoas.

Palavras-chave: inteligência artificial, aprendizagem de máquina, redes neurais, visão computacional, classificação de imagens, detecção de objetos, iot, raspberry pi

ABSTRACT

The recent advances in Computer Vision in last years, mainly due massive utilization of Convolutional Neural Networks, can be observed in tasks such as image classification, object localization and object detection. State of the art techniques in these kind of problems are based on neural net architectures with many layers of neurons, and by consequence, with a large number of parameters. Training and inference with these networks demand huge computational capacity, not only in processing power, but in memory consumption too. Demands presented by architectures considered the state of the art in the field make the utilization of technologies like these not viable in environments denoted by scarce computational resources, such as embedded systems or devices used in the context of the Internet of Things (IoT), where applications in residential security systems, where the demands for efficient response times are critical, can be compromised. Although neural network-based image classification and object detection models can be used in this context, delegating the computation to remote computers, the response time of network requests can be unacceptable for such kind of application. The popularity of the Raspberry Pi platform in IoT environments, with the relatively high processing power compared to devices used in such context, implies that the challenge of using Convolutional Neural Networks for image classification and object detection, applying the inference directly from these devices, turn to be a viable task. The present work tries to find a neural net-based solution, within a set of low-demanding processing power architectures, but still able to

detect people in images, with low response times and memory consumption, in a residential security system context, with acceptable efficiency. More specifically, an experiment is realized with object detection models, including TinyYolo, MobileSSD and Pelee models, comparing response times and detection quality. Additionally, another experiment with a hybrid solution, composed by a classification step, followed by a detection step, tries to achieve higher response time and detection accuracy.

Keywords: artificial intelligence, machine learning, neural networks, computer vision, image classification, object detection iot, raspberry pi

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

IoT ……Internet of Things

SVM ……...Support Vector Machines

GPU ……Graphical Processing Unit

MAP ……....Mean Average Precision

IoU ……...Intersection over Union

OCR ……...Optical Character Recognition

HOG ……...Histogram of Gradients

ILSVRC……ImageNet Large-Scale Visual Recognition Challenge

SIFT ……Scale Invariant Feature Transform

ROC ……...Receiving Operating Characteristic

DET ……....Detection Error Trade-off

ReLU ……Rectified Linear Unit

MLP ……..Multi-layer Perceptron

CNN ……Convolutional Neural Network

SSD ……..Single Stage Detector

YOLO ……....You Only Look Once

FPS ……..Frames per Second

1. INTRODUÇÃO

Detectar pessoas em imagens trata-se de um problema amplamente explorado na literatura de Visão Computacional ([1], [2] e [3]). Problemas relacionados com o reconhecimento de pedestres, no contexto de veículos autônomos [4], ou de intrusos, no contexto de sistemas de segurança [5], podem ser resolvidos com algoritmos de aprendizagem de máquina, utilizando-se algoritmos como SVMs [6] e Redes Neurais [7]. Dentre as possibilidades de abordagem, encontram-se modelos de classificação de imagens, que possibilitam a identificação de pessoas na imagem, e modelos de detecção de objetos, que aplicados à pessoas, permitem que identifique-se a quantidade e a posição relativa das pessoas detectadas na imagem. A detecção de objetos é um dos problemas fundamentais na área da Visão Computacional, e proporcional à sua relevância, é a sua complexidade quando comparada à problemas similares [8] . Enquanto a classificação de imagens consiste em inferir a classe à qual pertence uma imagem, na detecção de objetos, podem existir diversos objetos em uma imagem, os quais também devem ser classificados e localizados [8]. Comparada à localização de objetos, que consiste em encontrar as coordenadas correspondentes ao único objeto que deve ser localizado em uma imagem, na detecção de objetos é necessário encontrar as coordenadas correspondentes a todos os possíveis objetos presentes na imagem. Recentemente, modelos de classificação e detecção

têm sido continuamente desenvolvidos e melhorados a partir de abordagens conexionistas, e o desempenho dos modelos de Redes Neurais utilizados para estes fins apresentam-se como possibilidades concretas na resolução deste tipo de problema [9] .

Em paralelo a estes avanços, a popularização da Internet das Coisas (IoT) proporciona o desenvolvimento de aplicações baseadas na ampla utilização de dados de sensores , que por vezes encontram limitações no ambiente onde serão executadas. Enquanto o fato de estarem conectados em rede proporciona que dispositivos utilizados neste contexto possam delegar as demandas computacionais para servidores remotos, em determinados tipos de problema, o próprio dispositivo deve ser responsável por realizar a computação necessária para solucionar determinadas tarefas [10]. Assim, este trabalho pretende encontrar uma solução para problemas envolvendo a detecção de pessoas em imagens, diretamente à partir de dispositivos computacionais tipicamente utilizados em ambientes de IoT, mais especificamente, a plataforma de hardware Raspberry Pi [11].

1.1. Problema

Diante dos avanços constatados em diversos problemas no âmbito da Visão Computacional, proporcionados principalmente em razão da utilização de Redes Neurais Convolucionais, a crescente demanda por recursos computacionais envolvida na utilização deste tipo de modelo de Aprendizagem

de Máquina, por vezes inviabiliza a sua utilização em ambientes onde haja alguma restrição dos recursos computacionais disponíveis.

Os modelos de classificação de imagens e detecção de objetos que apresentam os melhores melhores resultados são compostos por inúmeras camadas convolucionais e um número elevado de parâmetros, fazendo com que tanto o treinamento como a inferência realizadas com estes modelos dependam de hardware com grande poder computacional, como é o caso das unidades de processamento gráfico, popularmente conhecidas como placas de vídeo (GPUs) [12].

No contexto da Internet das Coisas, a limitação dos recursos computacionais disponíveis exige que as soluções projetadas precisem delegar o processamento mais oneroso para servidores remotos, ou utilizem alternativas que exijam menos do hardware existente. Para determinadas aplicações, onde o tempo envolvido com a latência de rede ao se requisitar um servidor remotamente pode ser um fator impeditivo, exige-se soluções capazes de serem utilizadas no próprio hardware disponível localmente [11].

Para que seja possível realizar a tarefa de detecção de pessoas com redes neurais em ambientes computacionais restritos, torna-se necessária a utilização de modelos que exijam menos do hardware envolvido, sem que com isso perca-se substancialmente a eficiência das soluções baseadas em classificação ou detecção [13]. Este trabalho de conclusão de curso tem a motivação em demonstrar as limitações e as possibilidades envolvidas em se utilizar Redes Neurais no contexto de uma solução em segurança residencial baseada na detecção de pessoas em imagens, utilizando modelos de

classificação de imagens e detecção de objetos, executados diretamente em dispositivos Raspberry Pi, realizando uma análise do desempenho dos modelos selecionados, e propondo um modelo que atenda os requisitos descritos.

1.2. Objetivos

1.2.1. OBJETIVO GERAL

O presente trabalho de conclusão de curso tem como finalidade propor uma solução de baseada em aprendizagem de máquina, capaz de detectar pessoas utilizando a plataforma de hardware Raspberry Pi, no contexto da aplicação em um sistema de segurança residencial.

1.2.2. OBJETIVOS ESPECÍFICOS

1. Analisar a fundamentação teórica e o estado da arte no problema de classificação de imagens e detecção de objetos em ambientes computacionais restritos;
2. Analisar modelos de aprendizagem de máquina capazes de realizar classificação e detecção de objetos em ambientes computacionais restritos;

3. Definir/estabelecer os critérios de qualidade/avaliação dos modelos considerando-se o contexto de segurança residencial
4. Analisar o desempenho, ao realizar a inferência, entre os modelos de aprendizagem selecionados segundo os critérios estabelecidos;
5. Propôr uma solução para detectar pessoas em imagens com desempenho superior aos modelos de detecção pré-treinados selecionados, em termos de acurácia e tempo de processamento;

1.3. Escopo do Trabalho

O escopo do presente trabalho consiste em analisar a possibilidade da utilização de modelos de detecção de objetos baseados em redes neurais para a detecção de pessoas em imagens, no caso de uso de um sistema de segurança residencial, em ambientes computacionais com recursos limitados. Também está incluso no escopo, o treinamento de um modelo de classificação no desenvolvimento de uma solução híbrida envolvendo etapas de classificação e detecção.

Entre os itens que não fazem parte do escopo deste trabalho, pode-se citar:

- Análise de métodos de classificação de imagens e detecção de objetos que não sejam baseados em aprendizagem de máquina;

- Utilização e análise de modelos de redes neurais que não possam ser executados em ambientes computacionais restritos, como no caso da plataforma Raspberry Pi;
- Projeto e implementação de um artefato de *software* capaz de ser utilizado em sistemas de segurança residencial;
- Estudo e comparação de métodos alternativos de treinamento de redes neurais;
- Treinamento de modelos de redes neurais para detecção de objetos;

1.4. Método

O trabalho proposto utilizou-se de pesquisas realizadas a partir dos portais de periódicos ACM Portal, IEEE Xplorer e pela plataforma Arxiv. Os modelos e técnicas utilizados foram baseados no conteúdo da especialização em Deep Learning, oferecida pela plataforma Coursera [14], além dos vídeos gravados das aulas ministradas na disciplina CS231n (Deep Learning for Computer Vision) da Universidade americana de Stanford, disponíveis na plataforma Youtube [15].

A partir da análise do estado-da-arte da família de modelos que atendem as restrições apresentadas no trabalho, estabeleceu-se as possibilidades de modelos e técnicas a serem utilizadas.

Para a análise do desempenho dos modelos selecionados, foram planejados dois experimentos, visando obter respostas quanto ao desempenho

dos modelos selecionados, no cenário proposto pelo trabalho. No primeiro experimento realiza-se uma análise comparativa de modelos pré-treinados de detecção de objetos, utilizando diferentes valores para os hiperparâmetros de cada um destes modelos, comparando-os quanto aos critérios estabelecidos. No segundo experimento, os modelos que apresentam o melhor desempenho no experimento anterior, com suas respectivas configuração de hiperparâmetros, são utilizados em uma solução híbrida, envolvendo etapas distintas, de classificação e detecção, na tentativa de se obter melhores resultados. Treinou-se um modelo de classificação baseado na arquitetura MobileNet [16], utilizado na primeira etapa da solução, que combinado aos modelos selecionados a partir do experimento anterior, formam o conjunto de soluções que são comparadas neste experimento.

Para a realização do experimento, construiu-se um conjunto de dados a partir de imagens de câmeras de segurança capturadas pelo autor do trabalho, junto a imagens obtidas do conjunto de dados de imagens de pessoas INRIA [90]. As imagens de ambos os conjunto foram aleatoriamente distribuídas entre 3 subconjuntos distintos: conjunto de treinamento, conjunto de testes e conjunto de validação. O conjunto de treinamento foi exclusivamente utilizado para o treinamento do modelo de classificação utilizado no segundo experimento. O conjunto de validação foi utilizado para selecionar os melhores valores de hiperparâmetros, tanto no primeiro como no segundo experimento. Os resultados finais dos experimentos foram obtidos avaliando-se os modelos com o conjunto de testes, sem que este tenha sido utilizado durante seleção de

hiperparâmetros ou treinamento, evitando assim que os modelos apresentem qualquer viés em relação a estes dados [17].

Os critérios de avaliação de ambos os experimentos foram definidos levando-se em consideração os requisitos de um suposto sistema de segurança residencial, em que a desempenho na detecção de pessoas na imagem, sem avaliar a qualidade das *bounding boxes* produzidas, pode ser traduzida utilizando uma métrica comumente utilizada para avaliar modelos de classificação, a acurácia. Optou-se por não utilizar métricas relacionadas à sistemas de detecção de objetos, como *mean average precision* (MAP) [18] em conjunto com *intersection over union* (IoU) [18], pela dificuldade de coletar este tipo de métrica em tempo hábil de realização deste trabalho, devido às exigências em produzir-se um conjunto de dados com anotações referentes às *bounding boxes* das pessoas presentes nas imagens. Além da capacidade de detectar pessoas, em um sistema de segurança residencial exige-se que o tempo de processamento da solução desenvolvida não inviabilize sua utilização. Neste sentido, no trabalho foi utilizada a métrica de quantidade de *frames* (quadros) por segundo (FPS), que traduz diretamente a capacidade da solução de processar imagens em um intervalo fixo de tempo [20].

1.5. Estrutura do Trabalho

Este trabalho de conclusão de trabalho está organizado em 5 capítulos. No capítulo atual, o capítulo 1, introduz-se o tema, os objetivos são apresentados e explica-se qual é a proposta.

No capítulo 2, é realizada a fundamentação teórica, as definições necessárias para o tema proposto, como detecção de objetos, classificação de imagens, aprendizagem de máquina, redes neurais, transferência de aprendizagem, e tópicos específicos relacionados à abordagem utilizada.

O capítulo 3 apresenta os trabalhos relacionados à área da Visão Computacional, aos problemas de classificação de imagens e detecção de objetos, e aos modelos de rede neural que atendem os requisitos do problema.

O desenvolvimento do trabalho, a metodologia utilizada e os resultados dos experimentos realizados são descritos no capítulo 4.

No capítulo 5, são apresentadas as conclusões obtidas a partir do experimentos realizados e as possíveis direções a serem seguidas a partir dos resultados em trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Visão Computacional

Os problemas abordados na área da Visão Computacional envolvem descrever o mundo observado a partir de imagens, reconstruindo suas propriedades como a distribuição de formas, cores e iluminação. Enquanto humanos e animais realizam estas tarefas de maneira natural, sem que seja necessário maior esforço, os algoritmos utilizados em Visão Computacional são extremamente sujeitos a erros. É comum não atentar-se para a complexidade dos problemas abordados na área, dado que historicamente na pesquisa em Inteligência Artificial deu-se mais importância para os aspectos cognitivos da inteligência do que para aspectos envolvendo a percepção [21].

Atualmente, diversas aplicações envolvendo Visão Computacional, tanto na pesquisa, quanto em aplicações comerciais, demonstram que os avanços recentes na área abrem uma vasta gama de possibilidades. Pode-se destacar aplicações como o Reconhecimento Óptico de Caracteres (OCR), aplicado ao reconhecimento de códigos postais em correspondências [22], reconstrução de objetos em 3D a partir de fotografias [23], Reconhecimento Facial [24], entre outros.

No contexto do presente trabalho, existe especial interesse em problemas envolvendo o reconhecimento e detecção de objetos em imagens, onde aplicações que incluem sistemas de segurança, veículos autônomos e

monitoramento de tráfego, compartilham a necessidade de detectar a presença de pessoas na imagens analisadas.

2.1.1. Detecção de objetos

Entre os diferentes problemas na área da Visão Computacional, analisar uma imagem e reconhecer os diferentes objetos presentes na cena está entre as tarefas mais complicadas [21]. A tarefa de detecção é uma extensão natural à tarefa de classificação de imagens, apresentando a complexidade adicional de localizar as instâncias dos objetos na imagem. O objetivo na detecção de objetos é identificar a posição na imagem onde estão os diferentes objetos a serem detectados. Além da classe do objeto, deseja-se saber as coordenadas referentes a estes objetos [21]. Resolver o problema da detecção de objetos é um objetivo precursor para a resolução do problema da compreensão semântica do ambiente [25].

As dificuldades envolvidas na tarefa tem relação com as possíveis variações que uma determinada classe de objetos pode apresentar, como diferentes ângulos, poses, condições de iluminação, oclusão, tamanho, e variações naturais que diferentes instâncias de um mesmo objeto podem apresentar. O problema conhecido como *Semantic Gap* [26] aborda a diferença entre a informação possível de ser extraída de uma informação visual, baseada somente em diferentes valores de intensidade de *pixels*, e a sua interpretação semântica por alguém em uma situação específica. Trata-se de um dos desafios fundamentais na detecção de objetos.

Abordagens precursoras na detecção de objetos utilizavam mecanismos baseados no alinhamento entre modelos 2D/3D de objetos em relação à imagem, através de atributos simples, como contornos [27], pontos-chaves (*key-points*) [28], ou *templates* [29].

A utilização de algoritmos de aprendizagem de máquina neste contexto foi a primeira revolução acontecida na área [25], onde passa-se a tratar o problema como um tarefa de aprendizagem supervisionada. Pode-se destacar a utilização de algoritmos baseados em *boosting* [30], e abordagens baseadas em *support vector machines* (SVMs) [1]. Esta primeira geração de algoritmos de aprendizagem baseia-se na engenharia de atributos visuais, focando na extração de atributos que descrevessem as imagens de maneira relevante, como *Haar wavelets* [31], *edgelets* [32], *shapelets* [33], *histograms of oriented gradient* (HOG) [1] e *bags-of-visual-words* [34]. Um dos modelos de maior destaque antes da utilização de redes neurais neste contexto trata-se do *Deformable Part Model* [35] e suas variações [36].

Entre os métodos utilizados neste contexto, o método da *janela deslizante (sliding window)* baseia-se na utilização de uma janela de dimensões n x m, que percorre a imagem executando um classificador múltiplas vezes, nas diferentes posições da imagem. Como o objeto a ser identificado pode estar em versões com diferentes tamanhos, tamanhos variados da janela precisam ser utilizados.

Uma limitação encontrada no método citado, está na sobreposição de janelas, dado que diferentes janelas em posições adjacentes podem acabar detectando o mesmo objeto múltiplas vezes, fazendo com que diversas *bounding-boxes*

tenham uma intersecção que não reflete a existência de diferentes objetos. Para contornar este problema, pode-se utilizar uma estratégia conhecida como *non-maximum suppression*, onde janelas de detecção que tenham encontrado exemplos positivos dos objetos a serem detectados com intersecção acima de um determinado limiar, são suprimidas em favor das janelas que também estejam presentes na intersecção mas possuam um nível de confiança maior na detecção realizada [8].

Algoritmos baseados no método das janelas deslizantes são descritos na literatura, entre os mais conhecidos estão [20] aplicado na detecção de faces, e [1], na detecção de pedestres. Este último, utiliza a combinação de uma modelo de aprendizagem baseado em SVM linear em conjunto com extração de atributos HOG [1], onde um conjunto de histogramas de gradientes orientados sobrepostos é utilizado para descrever as características da imagem. Neste caso, cada histograma acumula a frequência de gradientes em orientações específicas, nas diversas regiões da imagem [8].

Apesar de redes neurais serem utilizadas para abordar o problema desde [37], na detecção de faces e [38] na detecção de mãos, foi após o trabalho seminal em [39], no contexto da classificação de imagens, seguido de [40], já na tarefa da detecção de objetos, ambos realizados no contexto da competição *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) [41], que impulsionaram a ampla utilização de Redes Neurais compostas por múltiplas camadas para a solução de problemas envolvendo imagens.

Abordagens baseadas em Redes Neurais semelhantes às utilizadas no método das janelas deslizantes são utilizadas em [42], onde aplica-se um

classificador em cada possível posição da imagem, e em [43], onde elegem-se regiões de interesse (*region proposals*), e em cada uma delas realiza-se a tarefa de classificação e localização isoladamente. Os métodos mais recentes baseiam-se na utilização de Redes Neurais Convolucionais Profundas, de maneira *end-to-end*, dispensando a utilização de métodos de extração de atributos como HOG [1] ou SIFT [28]. Como base para os modelos de rede neural utilizados para detecção de objetos, são utilizados modelos de classificação, que funcionam de maneira análoga aos componentes de extração de atributos, de modo que modelos que apresentam melhor desempenho em tarefas de classificação, também fazem com que o desempenho na tarefa de detecção apresente melhores resultados.

Para a avaliação de resultados em detecção de objetos, costuma-se utilizar métricas como *mean average precision* (MAP), *receiver operating characteristic* (ROC) ou *detection error trade-off* (DET) [44], comumente utilizadas na detecção de faces. A avaliação baseada em AP parte da existência de caixas retangulares (*bounding boxes*) consideradas como a seção da imagem onde estão presentes os objetos a serem detectados, comparando-as com as *bounding boxes* produzidas pelo modelo, de modo que a intersecção entre as seções retangulares, medida através de *intersection over union* (IOU), ultrapasse o limiar de 0.5. O desempenho no conjunto de dados é avaliado considerando-se a média entre os valores de AP das diferentes classes.

2.1.2. CLASSIFICAÇÃO DE IMAGENS

Na classificação de imagens, o objetivo é inferir a classe de uma imagem, a partir dos seus atributos. Comparada à detecção de objetos, trata-se de uma tarefa mais simples, pois neste caso, considera-se apenas a presença da classe que deseja-se identificar na imagem. Diversos problemas na área podem ser resolvidos a partir de classificação. Como um problema de aprendizagem supervisionada, as etapas envolvidas envolvem a definição, ou construção de um conjunto de dados rotulados, dos quais se extraem atributos, que podem ser os próprios pixels da imagem, ou atributos derivados, como é o caso de atributos HOG e atributos SIFT, para que então se treine um classificador [8].

Assim como na detecção de objetos, até recentemente, era comum a utilização de classificadores baseados em *Support Vector Machines* (SVMs) neste contexto, combinado com atributos HOG e SIFT [8]. Até que, em 2012, no desafio de classificação de imagens ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), demonstrou-se a eficiência em se utilizar redes neurais convolucionais, fazendo com que, desde então, modelos considerados o estado-da-arte na área são modificações baseadas na arquitetura de redes neurais convolucionais [17].

Ao se avaliar a tarefa de classificação de imagens, além da acurácia, costuma-se ainda utilizar precisão, *recall*, ou *f1-score*. Curvas de Precisão-*Recall* também são formas de avaliar classificadores [8]. A utilização

da acurácia é recomendável para casos em que o número de exemplos presentes no conjunto de dados não apresenta diferenças consideráveis [45].

2.2. Aprendizagem de Máquina

Aprendizagem de máquina, ou *machine learning*, é uma área da Inteligência Artificial dedicada em fornecer aos computadores a habilidade de aprender sem que precisem ser explicitamente programados [46]. Um algoritmo de aprendizado de máquina é um algoritmo capaz de aprender a partir dos dados [17].

Segundo [47], pode-se dizer que um programa de computador aprende a partir da experiência E, relacionada a uma classe de tarefas T e uma medida de performance P, se a performance nas tarefas em T, avaliada através de P, aumentar com a experiência E.

Quando um algoritmo de *machine learning* aprende a partir de exemplos rotulados, é chamado de aprendizado supervisionado, uma vez que são fornecidos os valores de saída (classe, ou rótulos) desejados para os exemplos utilizados. Uma outra forma de aprendizado é o aprendizado não-supervisionado, que não envolve o uso de exemplos rotulados, em que, ao invés de aprender o mapeamento das entradas e saídas, o objetivo pode ser modelar uma distribuição de probabilidade dos dados ou descobrir agrupamentos e outras estruturas subjacentes aos dados [48]. Pode-se citar ainda outros tipos de aprendizado: o aprendizado por reforço, em que um

agente autônomo, que recebe informações do ambiente e interage com ele, pode aprender a escolher as melhores ações para atingir seus objetivos [47]; e o aprendizado semi-supervisionado, em que alguns exemplos possuem rótulos fornecidos enquanto outros podem não possuir [17].

2.3. Redes Neurais

Redes neurais são uma forma de aprendizado de máquina com origem inspirada no funcionamento do neurônio biológico [47], uma abordagem conhecida como conexionista [49], desenvolvida a partir do modelo de um neurônio artificial proposto em [50].

Através do algoritmo da retropropagação, é possível treinar redes com múltiplas camadas utilizando-se de métodos de otimização baseados em gradientes [49], fazendo com a abordagem conexionista seja uma opção viável de aprendizagem. Somando-se isto ao fato de redes com pelo menos uma camada oculta serem capazes de aproximar qualquer função [51], tornam-se claras as possibilidades oferecidas pelas Redes Neurais no contexto da Aprendizagem de Máquina. Um conceito central do conexionismo é a ideia das representações distribuídas [52], em que uma entrada no sistema deve ser representada através de diversos atributos, e cada atributo deve estar envolvido na representação de diversas destas entradas [17].

2.3.1. DEEP LEARNING

Entre os obstáculos historicamente enfrentados por soluções baseadas em Redes Neurais, está o treinamento de redes com múltiplas camadas de neurônios. As limitações envolvidas no treinamento de modelos como estes passam a ser superadas a partir do momento em que [53] demonstra ser possível treinar eficientemente redes com múltiplas camadas, utilizando uma estratégia chamada *greedy layerwise pretraining*, para treinar as suas *Deep Belief Networks*. Mas foi com [39] que ficam demonstradas as possibilidades de utilizar-se Deep Learning em problemas reais, quando uma Rede Neural Convolucional vence o desafio de classificação de imagens *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), derrubando as taxas de erro por uma ampla margem [17].

O termo *deep*, do inglês “profundo”, está relacionado ao fato das redes neurais treinadas sob esta perspectiva apresentam diversas camadas de neurônios, apresentando portanto profundidade. A vantagem de se utilizar redes “profundas” deve-se a sua maior capacidade de representação, onde um conceito complexo pode ser representado nas camadas mais profundas como uma composição de representações mais simples, provenientes das camadas anteriores. Ao utilizá-las em imagens, estas representações mais simples podem identificar conceitos como linhas ou contornos, que em camadas intermediárias podem compôr representações mais complexas, como formas ou partes de objetos, até que nas camadas mais profundas representem objetos completos, ou conceitos ainda mais abstratos [17].

Outra justificativa para a profundidade de uma rede é a redução exponencial no número de unidades de ativação (neurônios) necessárias para se obter a capacidade de representação em uma rede profunda, quando comparada com uma rede sem profundidade [17].

Apesar de diversas inovações terem sido introduzidas desde a publicação do algoritmo da retropropagação, como novos métodos de otimização [54], [55] novas arquiteturas [56], [57] e novas funções de ativação [58], grande parte do atual sucesso desta abordagem pode ser explicado pelos avanços em termos de hardware, através da utilização de placas gráficas (GPUs) para o treinamento destas redes neurais, além de conjuntos de dados abertos cada vez maiores [57]. Outro aspecto importante para o desenvolvimento das redes neurais modernas é o surgimento de diversas bibliotecas especializadas, que tornam o processo de treinar e executar uma rede neural extremamente simples, podendo-se citar as seguintes: Theano [59], Torch [60], Caffe [61], MXNet [62], e TensorFlow [63].

2.3.2. FUNÇÕES DE ATIVAÇÃO

No modelo de neurônio artificial de [50], as entradas do neurônio são multiplicadas por um conjunto de pesos, fazendo com que o neurônio seja "ativado" caso o produto vetorial do vetor de pesos e o vetor de entrada ultrapasse um determinado limiar [47]. O conceito da função de ativação, que no modelo original é a função-degrau, ou função de *Heaviside* [64], nas redes neurais atuais é alguma forma de função diferenciável, responsável por

introduzir uma não-linearidade no cálculo dos neurônios de uma rede, sendo fundamental para a capacidade de representação deste tipo de modelo. Entre as função de ativação mais convencionais, pode-se citar as seguintes:

Função de ativação Sigmoidal - a função sigmóide, ou função logística, uma das primeiras funções de ativação utilizadas na literatura de redes neurais [17], pode ser descrita de acordo com as seguintes equações:

$$z = wx + b \qquad \textit{(Equação 1)}$$

$$a = \mathrm{g(z)} \qquad \textit{(Equação 2)}$$

$$g(z) = \frac{1}{1+e^{-z}} \qquad \textit{(Equação 3)}$$

Onde o valor z, também chamado de pré-ativação, corresponde à combinação linear entre o vetor de entradas x e o vetor de pesos w correspondente, somado a um termo *bias* b. A ativação a é calculada pela função g, que neste tipo de unidade de ativação corresponde à função logística, ou função sigmóide, recebendo a pré-ativação z como entrada, resultando em um valor escalar. A forma funcional da função sigmóide apresenta algumas desvantagens, como o fato de sua saída não ser centrada em zero, causando dificuldade no treinamento de redes neurais [17], e os gradientes da função, quando esta resulta em valores positivos ou negativos extremos, podem ser extremamente baixos, tendendo a zero, também

causando dificuldades durante a retropropagação. Apesar dos problemas mencionados, este tipo de função de ativação é ainda a mais adequada para se utilizar na última camada de uma modelo de classificação binária [17].

Função de ativação ReLU - projetada para contornar os problemas relacionados com as outras funções de ativação [39], a função ReLU apresenta a seguinte formulação:

$$g(z) = max(0, z) \qquad \textit{(Equação 4)}$$

Onde z é a pré-ativação, assim como na função de ativação sigmóide, conforme descrito na Equação 1. Apesar deste tipo de função de ativação também apresentar seus próprios problemas, como o problemas das *dead relus* [39], atualmente são a escolha padrão de funções de ativação em arquiteturas de redes modernas [17].

2.3.3. ARQUITETURAS DE REDE

Como mencionado anteriormente, redes com múltiplas camadas são capazes de aproximar qualquer função [51]. O tipo e o número de unidades em uma camada, e a maneira com que estas camadas conectam-se umas às outras, determinam a arquitetura de uma rede.

De maneira geral, redes neurais multicamadas são organizadas de maneira que os neurônios de uma determinada camada recebem como entrada as saídas dos neurônios da camada anterior. O vetor de entradas x de uma rede neural é considerado como a camada de entrada rede. As camadas intermediárias são conhecidas como camadas ocultas, e a última camada da rede como camada de saída, e o seu valor de ativação representa a saída da função representada pela rede neural.

Diferentes arquiteturas apresentam distintos padrões de conexão, que podem ser mais indicadas para determinados tipos de problemas que outros. Entre as topologias mais convencionais destacam-se as seguintes:

Redes Neurais Feedforward - também conhecidas Perceptron Multicamadas (do inglês Multi-Layer Perceptron - MLP), neste tipo de arquitetura cada camada recebe como entrada os valores de ativação provenientes da camada anterior, de modo que uma rede com L camadas é composta por 1 camada L de saída e L-1 camadas ocultas, sendo que a camada de entrada não é considerada nesta contagem [17]. Apesar de poderem apresentar conexões entre camadas não adjacentes, conhecidas como *skip-connections* [65], não apresentam conexões recorrentes.

Na Figura 1 é mostrada uma rede neural com as conexões entre as unidades de cada camada, onde cada componente do vetor x (camada 0 ou camada de entrada) conecta-se a cada unidade j da camada $l = 1$, responsável por calcular o valor de pré-ativação $z_j[l]$, que serve como entrada para a função de ativação $g[l]$, resultando no valor de ativação $a_j[l]$. Cada

unidade da camada l recebe como entrada os valores de ativação resultantes das unidades da camada $l-1$. A última camada, a camada L, é responsável pelo valor escalar de saída $\hat{y}$ da função representada pela rede neural.

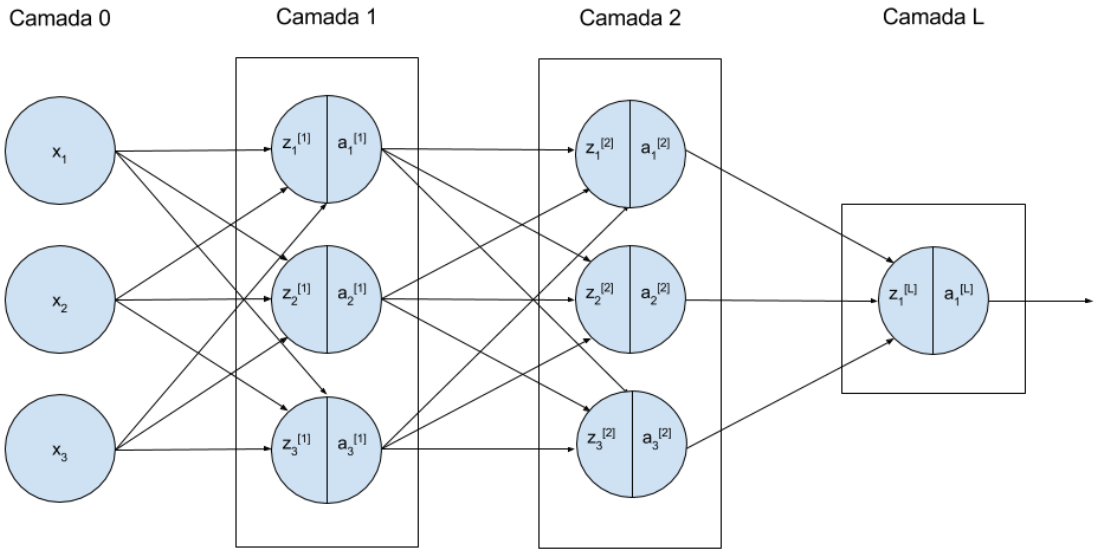

Figura 1: Uma rede neural feedforward simples, mostrando as conexões entre as unidades de cada camada

As camadas são encadeadas de maneira que o vetor de ativação $a^{[l]}$, ou a saída, de uma camada l pode ser definida em função da ativação da camada anterior $l-1$:

$$z^{[l]} = W^{[l]\top} a^{[l-1]} + b^{[l]} \qquad \textit{(Equação 5)}$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \qquad \textit{(Equação 6)}$$

sendo $a^{[0]}$ o vetor de entrada x, $z^{[l]}$ é o vetor de pré-ativação, $g^{[l]}$ é a função de ativação, $W^{[l]}$ é a matriz de pesos e $b^{[l]}$ é o vetor bias , correspondentes à camada l. A camada de saída, a camada L, é responsável por calcular a

saída $\hat{y}$ da rede neural, sendo $\hat{y}$ um valor escalar ou vetor, no caso de uma cada *softmax* [17].

$$\hat{y} = a^{[L]} = g^{[L]}(z^{[L]}) \qquad \textit{(Equação 7)}$$

sendo $g^{[L]}$ uma função de ativação possivelmente diferente das funções de ativação utilizadas nas camadas ocultas. Em problemas de classificação binária, é comum g ser a função logística. Já em problemas que envolvem múltiplas classes, pode-se usar a função softmax:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \qquad \textit{(Equação 8)}$$

sendo z um vetor de pré-ativações, que ao passar pela função softmax resulta em um vetor de saída normalizado, de modo que soma dos seus componentes resulte em 1.

Na Figura 2, uma rede neural é representada de maneira mais compacta, apenas com as conexões entre as camadas, ao invés do todas as conexões entre as unidades. A saída da última camada L, uma camada *softmax*, é um vetor com C componentes, onde C é o número de classes em um problema de classificação multi-classe.

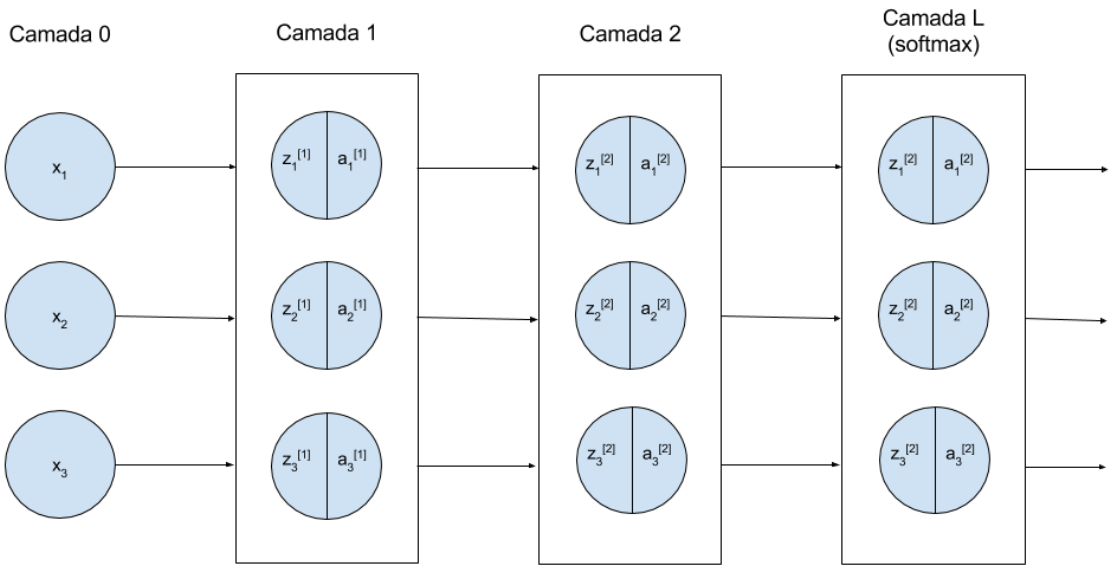

Figura 2: Uma rede neural feedforward sim, mostrando as conexões entre as camadas

Redes Neurais Convolucionais - utilizadas frequentemente em tarefas de Visão Computacional, este tipo de arquitetura de rede neural realiza operações de convolução ao invés de utilizar apenas multiplicações entre as matrizes de pesos e os vetores de ativação da camada anterior [17]. Uma camada convolucional recebe um tensor de entrada, que pode ser uma imagem, com determinada largura, altura e número de canais, sobre o qual operam um conjunto de filtros, ou *kernels*, de dimensões $f \, x \, f$, e mesmo número de canais, que percorrem este tensor de entrada, gerando um mapa de características (*feature map*) para cada filtro. O número de canais $n_c^{[l]}$ do tensor de saída é definido pelo número de filtros $K^{[l]}$ da camada l. A altura e a largura de cada *feature map* dependem dos valores de *padding* e *stride* da camada, que especificam a maneira que estes filtros deslizam. Especificamente, a altura

$n_h^{[l]}$ e a largura $n_w^{[l]}$ do tensor de saída da camada l podem ser calculados de acordo com:

$$n_h^{[l]} = \lfloor \frac{n_h^{[l-1]} - f + 2p}{s} \rfloor + 1 \qquad \textit{(Equação 9)}$$

$$n_w^{[l]} = \lfloor \frac{n_w^{[l-1]} - f + 2p}{s} \rfloor + 1 \qquad \textit{(Equação 10)}$$

onde f é o tamanho dos filtros da camada l, p é o valor de *padding* da camada, e s é o valor de *stride* da camada.

Tecnicamente, a operação descrita como convolução comumente implementada em redes neurais, é uma correlação-cruzada, que ao contrário da convolução, não realiza o espelhamento dos filtros, horizontal e verticalmente, antes de deslizar pela imagem. A convolução de uma entrada bidimensional I com um *kernel K*, pode ser definida como:

$$S(i, j) = (I * K)(i,j) = \sum_m \sum_n I(i - m, j - n)\, K(m, n) \qquad \textit{(Equação 11)}$$

enquanto uma correlação-cruzada tem a seguinte formulação:

$$S(i, j) = (I * K)(i,j) = \sum_m \sum_n I(i + m, j + n)\, K(m, n) \qquad \textit{(Equação 12)}$$

onde o I é o tensor de entrada, K é o kernel, S é o *feature map* resultante, i e j são os índices que indicam a posição do *kernel*, m e n correspondem ao tamanho horizontal e vertical do filtro, respectivamente [17].

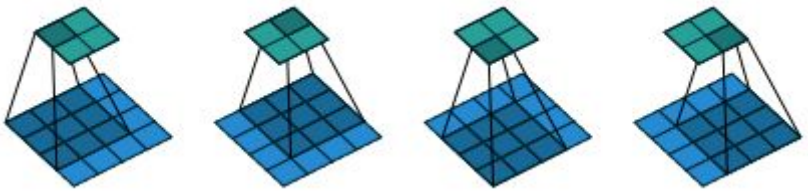

Figura 3: Filtro convolucional 3x3 percorrendo uma imagem 4x4, com *stride* 1 e *padding* 0, produzindo um *feature map* 2x2 [66].

Os filtros de uma camada convolucional deslizam pela imagem primeiramente na horizontal, da esquerda para a direita, em seguida, na vertical, de cima para baixo. O *stride* define quantas posições o filtro do tensor se movimenta a cada passo. Um valor de *stride* mais alto faz com que as dimensões dos *features maps* resultantes sejam reduzidas. Na prática, um único valor é definido tanto para o deslocamento horizontal quanto para o vertical.

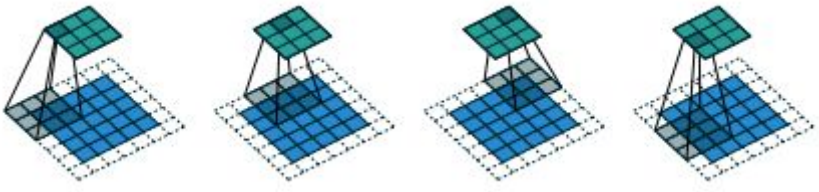

Figura 4: Filtro convolucional 3x3 percorrendo uma imagem 5x5, com *stride* 2 e *padding* 1, produzindo um *feature map* 3x3 [66].

Para evitar que a saída de uma camada convolucional tenha suas dimensões reduzidas, pode-se usar um valor de *padding* que adicione linhas e colunas preenchidas por zeros nas bordas do tensor de entrada. Dessa maneira é possível utilizar mais camadas convolucionais em uma rede neural, sem que as dimensões reduzam-se demais. Um valor de padding que mantenha tamanho do tensor de saída igual ao de entrada pode ser chamado de *'same'*, enquanto não utilizar padding também é chamado de *'valid'* [66]. Na Figura 5 é representada a conexão entre camadas convolucionais, onde cada uma recebe um tensor de entrada, com largura w, altura h e um número de canais c que determina sua profundidade. Através de operações de convolução, resulta em um tensor de saída, com largura, altura e profundidade determinados pelo tamanho f dos filtros convolucionais utilizados na camada, o valor de *padding* p e o valor de *stride* s

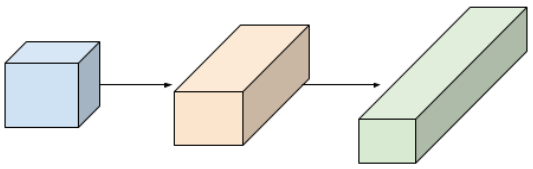

Figura 5: Conexão entre camadas convolucionais, que recebem um tensor de entrada e emitem outro tensor como saída

As camadas convolucionais geralmente são seguidas por função de ativação não-linear. Além das camadas responsáveis pelas convoluções, é

comum intercalá-las com camadas de *pooling* [17]. O *pooling* também é realizado através de filtros que percorrem o tensor de entrada, mas diferentemente dos filtros convolucionais, não possuem pesos, tendo apenas duas dimensões, operando em cada canal isoladamente. A cada passo de uma operação de *pooling*, é produzido um valor correspondente a alguma estatística relacionada com os valores da entrada correspondentes à posição em que o filtro está posicionado. O tipo de *pooling* define qual é a estatística coletada. No caso de um *max-pooling*, o maior valor entre os elementos selecionados corresponde à saída daquele passo do *pooling*. No caso do *average-pooling*, a média entre os valores selecionados é o resultado da operação [17]. As diferentes camadas em uma rede convolucional podem ser consideradas individualmente, ou como diferentes etapas de uma só uma camada. Na figura 6, é mostrada uma camada convolucional onde a primeira etapa formada por uma mais convoluções, seguidas de uma função de ativação, intercaladas por etapas de pooling.

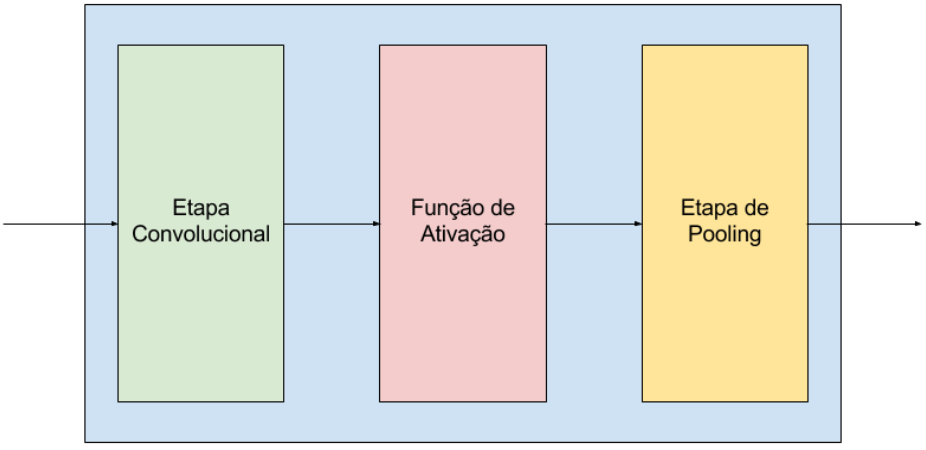

Figura 6: As diferentes etapas de uma camada convolucional

Além da arquitetura convencional de redes convolucionais, arquiteturas mais recentes sugerem variações na topologia dessas redes, envolvendo *skip-connections*, conectando camadas não adjacentes [65], que facilitam o aprendizado da função-identidade, permitindo o treinamento de redes ainda mais profundas; ou a utilização de módulos mais complexos, como os módulos *Inception*, que realizam convoluções com diferentes tamanhos de filtro em paralelo, concatenando a saída de cada um destes conjuntos de filtros em único tensor de saída [56], utilizando ainda *Depthwise Separable Convolutions*, reduzindo a quantidade de computações realizadas nas camadas iniciais do modelo [67].

A motivação para as rede convolucionais é baseada em três propriedades principais: a conectividade esparsa, resultante do tamanho de *kernel* menor que o tamanho das entradas, reduzindo a quantidade total de parâmetros e a eficiência computacional; o compartilhamento de parâmetros, também responsável por reduzir o número de parâmetros; e a equivariância translacional, responsável por possibilitar que este tipo de rede seja capaz de identificar um padrão, mesmo que ocorra em posições distintas em uma imagem [17].

2.3.4. FUNÇÃO-CUSTO

Para que um algoritmo de Aprendizagem de Máquina baseado em Redes Neurais seja capaz de encontrar a melhor configuração de parâmetros

(pesos) para a ligação entre as camadas da redes, utiliza-se um algoritmo de otimização. Apesar de poder ser interpretado como um problema de otimização não-linear [68], o treinamento de Redes Neurais difere de um simples problema de otimização pela necessidade de que a função aprendida seja capaz de generalizar para exemplos não considerados durante o treinamento. Neste contexto, deve-se utilizar um função de custo, como uma forma de direcionar a aprendizagem, de forma que os parâmetros da função, aprendidos durante o treinamento, sejam aqueles que minimizem a função de custo utilizada. No caso de problemas de regressão, onde deseja-se aproximar uma função contínua, é comum utilizar uma função de custo baseada no erro quadrático mínimo, enquanto em problemas de classificação costuma-se utilizar a função *cross-entropy* [68].

Cross-entropy - Considerando-se problemas de classificação binária, onde a saída de uma Rede Neural é caracterizada por uma função de ativação sigmoidal, utiliza-se como uma forma de medir a diferença entre as saídas emitidas pelo modelo de aprendizagem e o reais valores dos rótulos dos exemplos utilizados para treinamento, uma função de erro baseada na entropia-cruzada entre duas distribuições de probabilidade [17]. Desta forma, tem-se que:

$$J = -\sum_{n=1}^{N} \{y_n log\ \hat{y}_n + (1 - y_n)\ log\ (1 - \hat{y}_n)\} \qquad \textit{(Equação 13)}$$

Onde N corresponde ao número total de exemplos de treinamento, y_n corresponde ao rótulo do exemplo n, e $\hat{y}_n$ ao rótulo inferido pelo modelo para o exemplo n.

2.3.5. OTIMIZAÇÃO

Para encontrar os valores dos parâmetros que minimizem o valor da função de custo utilizada, é necessário utilizar um algoritmo de otimização. No treinamento de Redes Neurais, é comum utilizar-se algoritmos baseados no Método Gradiente Estocástico [69] , de forma que os parâmetros da rede sejam minimizados de maneira iterativa e aproximada, diferente de métodos de otimização de segunda ordem, como no caso do Método de Newton [70].

Rmsprop - Surgido a partir de uma modificação realizada no algoritmo de otimização AdaGrad [71], que por sua vez trata-se de um algoritmo de otimização iterativo adaptativo. O método RMSprop [54] comporta-se melhor em cenários onde a função a ser otimizada é uma função não-convexa, como é natural em cenários envolvendo redes neurais. Enquanto o AdaGrad adapta-se diminuindo a taxa de aprendizagem levando em consideração todo o histórico dos gradientes, fazendo com que a taxa frequentemente diminua antes do algoritmo entrar em um regime de convergência, o RMSprop utiliza uma média móvel com decaimento exponencial para evitar que os valores dos gradientes

em etapas muito distantes do treinamento interfiram de maneira prejudicial nos passos atuais [17].

2.3.6 RETROPROPAGAÇÃO

Ao se utilizar uma rede neural que recebe um exemplo x que produz uma saída $\hat{y}$, a informação passa através da rede de maneira que o exemplo x é a informação inicial propagada ao longo das camadas da rede até que produza $\hat{y}$. Esta etapa recebe o nome de *forward propagation* [17]. Durante o treinamento, esta etapa pode continuar até que se produza o valor de custo J. A etapa conhecida como *Backpropagation* [49], permite que a informação possa ser transmitida a partir do custo, passando novamente pelas camadas ocultas, permitindo que se compute os valores dos gradientes para cada unidade da rede [17].

O algoritmo da retropropagação faz com que os gradientes de cada camada da rede possam ser calculados em relação custo, permitindo que possa-se ajustar os valores dos pesos de cada unidade da rede de forma a diminuir o custo. Desta forma, ao aplicar-se sucessivamente o algoritmo da retropropagação, em pequenos conjuntos de exemplos de treinamento, tem-se a cada passo, uma aproximação dos valores dos gradientes a serem utilizados no ajuste dos pesos [17].

2.3.7. REGULARIZAÇÃO

Durante o treinamento de Redes Neurais, é comum encontrar situações em que os parâmetros da rede são ajustados de forma a minimizarem o custo total quando observam-se os exemplos de treinamento, mas falham quanto à generalização proporcionada pelo modelo. A função de custo quando computada em exemplos que não foram utilizados durante o treinamento, não repete o fenômeno observado quando calculada levando-se em conta somente os exemplos de treinamento. Neste caso, considera-se que a rede está em regime de *overfitting*. Regularização, neste contexto, é o nome dado para qualquer modificação realizada em um algoritmo de aprendizagem com o intuito de reduzir o erro de generalização, mas não o erro de treinamento [17]. Entre as formas mais comuns de regularização encontram-se técnicas como *early stopping,* regularização-l2, *dropout* e *data augmentation*.

Dropout - A técnica conhecida como *Dropout* [72], consiste em descartar aleatoriamente unidades da rede durante uma etapa do treinamento, fazendo com que a rede seja obrigada a calcular os gradientes na etapa posterior desconsiderando um subconjunto aleatório de unidades a cada etapa. A quantidade de unidades a serem descartadas é definida de acordo com a taxa de *dropout* utilizada, que define um percentual de unidades a serem descartadas a cada etapa do algoritmo. Pode-se interpretar este efeito como uma forma de *ensemble*, dado que diferentes versões da mesma rede são utilizadas para ajustar os parâmetros da rede durante o treinamento.

Data Augmentation - Uma maneira simples de combater-se o *overfitting* é treinar o algoritmo de aprendizagem com mais dados. Em situações em que a quantidade de dados é limitada, e a coleta de novos exemplos é inviável, é comum utilizar transformações em cima dos dados existentes, de maneira que o conjunto de dados seja aumentado de forma artificial. Em problemas de classificação de imagens, onde a posição, tamanho e orientação dos elementos encontrados nas imagens pode variar, a utilização desta técnica pode melhorar desempenho de generalização da rede. Transformações como rotações, translações verticais e horizontais, ampliações e reduções no tamanho original da imagem são possibilidades de acordo com as especificidades do domínio do problema [17].

2.3.8. BATCH NORMALIZATION

A técnica de *Batch Normalization* [73] consiste em uma forma de ajustar os valores dos pesos de uma camada da rede, de maneira que os valores de ativações de uma camada possam ser normalizados, introduzindo-se parâmetros relativos à média e ao desvio-padrão das ativações de uma camada, possibilitando que as ativações de uma camada sigam uma distribuição gaussiana, facilitando a atuação do algoritmo da retropropagação. Não se trata portanto, de um algoritmo de otimização, nem de uma técnica de regularização, apesar de possuir efeitos de regularização quando utilizado [17],

mas sim de uma técnica de reparametrização que facilita o treinamento de uma rede neural.

2.3.9. TRANSFERÊNCIA DE APRENDIZAGEM

Quando uma rede neural é treinada utilizando-se um conjunto de treinamento qualquer, e posteriormente esse treinamento prévio é aproveitado para melhor a generalização da rede em outro contexto, diz-se que realizou-se transferência de aprendizagem [17]. Esta técnica costuma ser utilizada em contextos onde as características entre os diferentes conjuntos de treinamento possuem naturezas semelhantes.

O processo chamado *fine-tuning* é uma forma de transferência de aprendizagem onde aproveitam-se os pesos de uma rede pré-treinada em um conjunto de treinamento qualquer, como os valores iniciais dos parâmetros da rede. Desta forma, pode-se ainda congelar os pesos das camadas iniciais da rede, complementando-se o treinamento em outro conjunto de dados com características semelhantes. Neste cenário, recomenda-se utilizar taxas de aprendizagem baixas, para evitar que os gradientes calculados no início desta nova etapa de treinamento prejudiquem os valores de parâmetros já treinados anteriormente [74].

3. TRABALHOS RELACIONADOS

Considerando-se o contexto da utilização de Redes Neurais Convolucionais Profundas na tarefa de Detecção de Objetos, o trabalho de levantamento no nível de *survey* apresentado em [25] descreve os modelos utilizados atualmente, além de relacionar um breve histórico de métodos alternativos utilizados anteriormente às redes neurais convolucionais apresentarem-se como abordagem dominante. Além dos modelos utilizados e o estado-da-arte, o trabalho ainda relaciona as diferentes arquiteturas de rede, como *single stage detectors*, *double stage detectors* e modelos baseados em partes, detalhando também as técnicas mais comumente aplicadas para o treinamento destes modelos. São apresentadas novas abordagens na área como Redes baseadas em Grafos e Redes Adversárias, assim como também são apresentados os desafios relacionados ao problema, como a variância de escala, variância rotacional, adaptação ao domínio, oclusão, e detecção de pequenos objetos.

Outro trabalho em nível de *survey* que merece ser destacado é o apresentado em [75], onde também descreve-se o problema, a arquitetura dos principais modelos baseados em redes neurais, assim como as diferentes arquiteturas dos componentes responsáveis pela extração dos atributos, a base convolucional dos modelos. Detalhes referentes ao treinamento destas redes, incluindo aspectos relacionados ao treinamento distribuído, também são abordados no trabalho.

Entre os diferentes modelos utilizados para detecção de objetos, podem-se citar abordagens não-baseadas em aprendizagem, como é caso de modelos baseados em *edge detection* [27], baseados em *key-points* [28], e modelos baseados em *templates* [29]. Além destes, [76] e [77] merecem destaque por abordarem o problema utilizando detectores de face baseados em *appearance*, e introduzirem diversas inovações adotadas em diversos trabalhos posteriores.

Ao relacionar-se diferentes trabalhos envolvendo aprendizagem de máquina, mas que não consideram a utilização de redes neurais, cabe destacar os trabalhos pioneiros de [30] e [31], envolvendo a utilização da técnica de *boosting*, e [1], em que utiliza-se Support Vector Machines (SVM) para abordar o problema.

Entre os primeiros trabalhos a destacarem-se na utilização de Redes Neurais Convolucionais para resolver os problemas de detecção e localização estão [37] e [38], e posteriormente [78], [79], [80] e [81]. Este último, apresenta direta relação com o presente trabalho, por ser utilizado na detecção de pedestres.

Abordagens modernas na detecção de objetos, baseadas em Redes Neurais Convolucionais, já consideradas *Deep Learning*, incluem trabalhos como [42], que apresenta uma abordagem conexionista para método da janela deslizante, [43], que elege regiões de interesse (*region proposals*) onde as janelas deslizantes podem ser aplicadas, no modelo conhecido como R-CNN. Em trabalhos subsequentes, [82] e [83] propõem, respectivamente, os modelos conhecidos como Fast R-CNN, e Faster R-CNN, onde são realizadas

modificações na arquitetura original, tendo em vista a melhoria no tempo de processamento do modelo.

Outros trabalhos em que o tempo de processamento é encarado como prioridade, e portanto, apresentam maior aderência com o trabalho aqui desenvolvido, incluem a família de modelos conhecidos como *Single Step Object Detectors*, onde os modelos realizam inferência em um só passo, tanto das classes dos objetos a serem detectados, quanto das suas coordenadas. Entre os modelos que fazem parte desta categoria de modelos, podem ser citados o modelo YOLO [18], os modelos SSD [83] e o modelo agora considerado o estado-da-arte na tarefa de detecção, RetinaNet [85]. A combinação destes modelos com bases convolucionais em que o tempo de processamento também é o foco do seu desenvolvimento, resultam nos modelos TinyYOLO [18], que utiliza a implementação de referência Darknet [18], e o modelo MobileSSD, que combina a base convolucional da rede MobileNet [16] com os modelos apresentados em [83].

Considerando-se os trabalhos que apresentam os principais conjuntos de dados utilizados em trabalhos cujo o foco é a detecção de objetos, podem citados o conjunto PascalVOC [44] e MS-COCO [19]. Conjuntos de dados onde os objetos a serem detectados são compostos exclusivamente por pessoas incluem o conjunto *INRIA Person Dataset* [90] e o *HDA Person Dataset* [86].

4. DESENVOLVIMENTO

Visando a aplicação de detecção de pessoas utilizando Redes Neurais na plataforma computacional Raspberry Pi, no contexto de um sistema de segurança residencial, o presente trabalho realizou dois experimentos, objetivando compreender as limitações e possibilidades envolvendo a utilização de modelos baseados em redes neurais convolucionais especificamente desenvolvidos para serem utilizados em ambientes computacionais restritos.

Para a tarefa de detecção de pessoas, objetiva-se identificar a presença de pessoas na imagem capturada, ignorando quaisquer outros elementos que estejam presentes na imagem e não sejam relevantes à tarefa proposta. Apesar desta ser uma tarefa possível de ser resolvida com modelos de classificação, optou-se primeiramente por experimentar uma abordagem envolvendo apenas a utilização de modelos de detecção de objetos, por permitirem que as pessoas identificadas na imagem sejam destacadas através de *bounding boxes*, fazendo com que seja possível saber a posição relativa das pessoas identificadas na imagem, além da quantidade de pessoas identificadas. Diante dos resultados obtidos com os modelos de detecção de objetos, nos quais fica evidente a limitação decorrente da demanda computacional inerente a estes modelos, é proposta uma segunda abordagem, em que é experimentado um modelo de classificação, como uma primeira etapa da solução, seguido de uma etapa de detecção, caso a imagem tenha sido classificada como um exemplo positivo quanto à presença de pessoas.

Na Figura 7 é mostrado o esquema geral dos experimentos realizados no trabalho. O primeiro experimento proposto (acima) utiliza modelos pré-treinados de detecção de objetos, seleciona a versão de cada modelo com o valor de confiança com melhores resultados em termos de acurácia no conjunto de validação, e avalia os modelos selecionados no conjunto de testes, tanto em termos de acurácia, quanto em termos de FPS. O segundo experimento (abaixo) realiza o treinamento de um modelo de classificação MobileNet, inicializado com pesos de um modelo pré-treinado no conjunto Imagenet, realiza a seleção do modelo com melhor acurácia no conjunto de validação, e combina este modelo com os modelos que apresentaram melhores resultados em termos de acurácia e FPS no conjunto de validação no primeiro experimento. Em seguida, estes modelos híbridos são avaliados em termos de acurácia e FPS no conjunto de testes. As setas que conectam os dois experimentos demonstram que os modelos melhor avaliados no primeiro experimento são utilizados como componentes de detecção no segundo experimento.

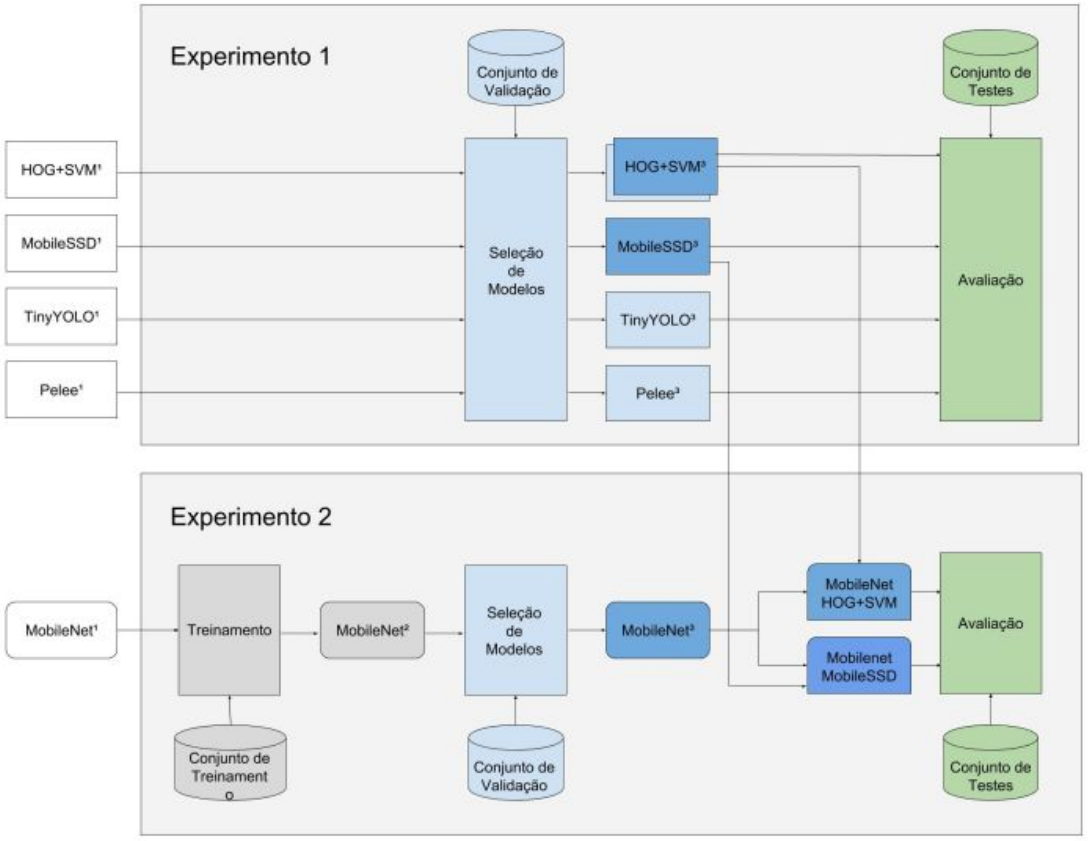

Figura 7: Esquema demonstrando os dois experimentos realizados no trabalho

No primeiro experimento proposto, cujo o intuito é validar a hipótese de que é possível atingir o objetivo geral do trabalho utilizando somente modelos de detecção de objetos pré-treinados, serão comparados quatro modelos de detecção de objetos. Três deles baseados em redes neurais, e outro baseado extração de atributos HOG combinado com um classificador SVM linear, por ser um modelo simples e bastante popular na área de detecção de objetos [1], cuja implementação é disponibilizada na biblioteca OpenCV. Os modelos baseados em redes neurais foram selecionados dentre um conjunto de modelos caracterizados pela eficiência em termos de consumo de memória e demanda computacional: o TinyYolo, por ser baseado em um modelo pioneiro entre modelos de detecção de objetos focados no tempo de processamento e

baixo consumo de memória [18], o MobileSSD, por apresentar os melhores resultados em termos de eficiência computacional de acordo com a literatura [87], e o modelo Pelee [88], [89], que apesar de ser pouco citado em trabalhos relacionados, teve sua implementação disponibilizada pelos seus autores, sendo possível sua utilização neste trabalho. São utilizados modelos pré-treinados, com o conjunto de dados MS-COCO, seguido de um treinamento posterior com o conjunto de dados PascalVOC. Para a avaliação dos modelos, é construído um conjunto de dados composto por 2000 imagens, onde imagens capturadas especificamente para o experimento são utilizadas em conjunto com imagens extraídas do conjunto de dados de imagens de pessoas INRIA. Os modelos são comparados em relação à velocidade em que são capazes de realizar a detecção em um dispositivo Raspberry Pi, em termos de imagens por segundo (FPS - *frames per second*), e a acurácia relacionada com a tarefa de reconhecimento de pessoas na imagem.

No segundo experimento, diante dos resultados em que ficaram claras as limitações dos modelos experimentados, objetiva-se desenvolver uma solução que apresente resultados em termos de processamento e qualidade de detecção superiores aos modelos pré-treinados. Decidiu-se por utilizar uma solução baseada em duas etapas: uma etapa inicial de classificação, onde utiliza-se um modelo de classificação treinado especificamente para o problema, que, ao resultar em um rótulo positivo quanto à presença de pessoas na imagem, é seguido por uma etapa de detecção de objetos, onde o modelo melhor avaliado em termos de velocidade de processamento e o melhor modelo em termos de acurácia no primeiro experimento são comparados,

novamente em termos de acurácia e velocidade de processamento. Para este experimento, complementou-se o conjunto de dados construído para o primeiro experimento, resultando em um total de 11.000 imagens. Os detalhes da construção do conjunto de dados são expostos na Seção 4.3 deste trabalho.

4.1. Contexto - Segurança Residencial

Considerando-se o contexto do presente trabalho como sendo a detecção de pessoas em imagens provenientes de um sistema de segurança residencial, justificam-se as escolhas realizadas dentre as opções existentes para definir os critérios de avaliação e o conjunto de dados utilizados, assim como decisões envolvendo os modelos que fazem parte dos experimentos.

Em relação aos critérios de avaliação, necessita-se medir a capacidade das soluções propostas em detectar corretamente a presença de pessoas nas imagens, uma vez que o sistema de segurança em questão seria acionado mediante a exposição de pessoas nas imagens capturadas, mas não deveria ser acionado quando exposto a outros elementos, como por exemplo, diferentes tipos de animais. Outro requisito igualmente importante é o tempo de processamento das soluções desenvolvidas, pois a necessidade de responder rapidamente é evidente em se tratando de um sistema de segurança, onde ações decorrentes a partir da detecção realizada, como o acionamento de alarmes e mecanismos de defesa, são ainda mais eficientes quando as ações realizadas são imediatas.

O conjunto de dados utilizado também foi produzido levando-se em consideração este contexto, justificando a opção por capturar imagens diretamente a partir de câmeras de segurança utilizadas em um sistema de monitoramento de vídeo real.

4.2. Critérios de avaliação

Como requisitos de ambos os experimentos, espera-se obter uma solução capaz de ser executada em um dispositivo Raspberry Pi 3, atendendo às limitações de memória e processamento do aparelho. As soluções desenvolvidas serão avaliadas de acordo com critérios que demonstrem o tempo de processamento das soluções desenvolvidas, e da qualidade das detecções realizadas.

Para avaliar o tempo de processamento, será medido o tempo de processamento médio de todos os modelos avaliados, ao detectar todos os 2.000 exemplos que fazem parte do conjunto de imagens de teste produzido para os experimentos. A métrica utilizada é a quantidade de *frames* por segundo (FPS) processados por cada modelo.

Para avaliar a qualidade das detecções realizadas, optou-se por utilizar a acurácia do modelo ao detectar se existem pessoas ou não na imagem. Uma das limitações da acurácia está em cenários em que a quantidade de exemplos das diferentes classes é desbalanceada. Neste trabalho, utiliza-se uma mesma quantidade de exemplos de ambas as classes. A acurácia é frequentemente

utilizada para avaliar modelos de classificação, e foi selecionada neste trabalho ao invés de métricas mais comumente utilizadas no contexto de detecção de objetos, como a *mean average precision* (MAP) [18]. Esta decisão foi em decorrência da dificuldade de produzir anotações com as coordenadas das *bounding boxes* a serem detectadas pelos modelos de detecção, o que inviabilizaria a execução dos experimentos. Desta forma, desconsidera-se na avaliação a qualidade das *bounding boxes* produzidas. Espera-se obter uma solução capaz de detectar pessoas nas imagens com acurácia superior ao modelo utilizado como *baseline* (HOG + SVM), que obtiver a maior taxa de FPS entre as configurações de hiperparâmetros experimentadas.

4.3. Conjunto de dados

Para a realização dos experimentos, foi construído um conjunto de dados formado por 11.000 imagens rotuladas, quanto à presença ou não de pessoas, em que 50% possuem o rótulo positivo e 50% rotuladas como negativo. Separou-se o conjunto de dados em 3 frações: o conjunto de treinamento, composto por 7.000 imagens, o conjunto de validação, formado por 2.000 imagens e o conjunto de teste, também com 2.000 imagens.

Deste total, 2.000 imagens foram extraídas do conjunto de dados INRIA Person Dataset [90], com 1.000 imagens positivas e 1.000 negativas. As 9.000 imagens restantes, também formada por 50% de exemplos positivos e negativos, foram capturadas especificamente para este trabalho por seu autor,

onde a maior parte das imagens foi produzida utilizando-se 6 câmeras de um circuito interno de segurança, uma parcela das imagens utilizando-se um Raspberry Pi 3 Model B+ V1.2 [91], capturando a saída de vídeo de uma câmera Câmera Raspberry Pi NoIR v2 8MP [92], além de uma fração composta por imagens capturadas a partir de uma *webcam* de um notebook DELL Latitude 3480.

As imagens originalmente foram produzidas com 300 pixels de altura, 300 pixels de largura e 3 canais de cores. Para utilizá-las nos diferentes modelos de classificação/detecção, foram criadas versões com diferentes dimensões, pelo fato dos modelos exigirem tamanhos específicos de imagens. Além da versão original com 300x300, uma versão com tamanho de 416x416, outra com 304x304 e outra com 224x224 foram produzidas. Optou-se por criar diferentes versões ao invés de redimensioná-las durante sua utilização nos modelos, para evitar que o tempo envolvido no redimensionamento influenciasse nos resultados da avaliação do tempo de processamento.

Nas Figuras 8 e 9 são mostrados exemplos de imagens que fazem parte do conjunto de dados utilizado. Na Figura 8 estão representados exemplos negativos, em que na linha de cima estão exemplos de imagens capturadas pelo autor do trabalho, e na linha de baixo imagens extraídas do INRIA Person Dataset. Na figura 9, são mostrado exemplos positivos, que assim como no caso anterior, na linha de cima estão imagens capturadas pelo autor do trabalho, e na linha de baixo são imagens extraídas do INRIA Person Dataset.

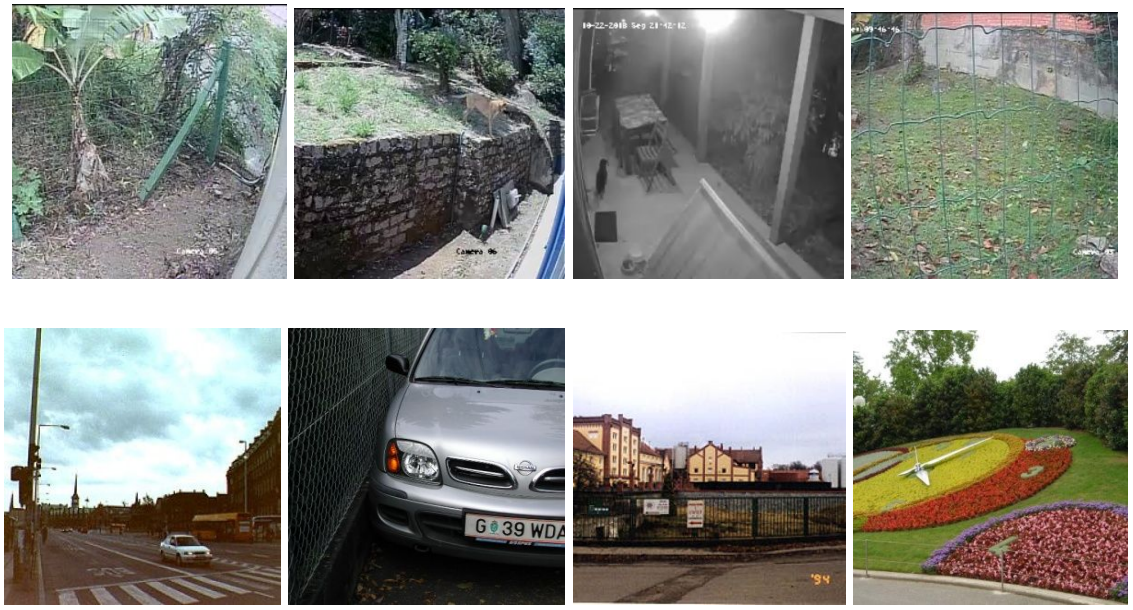

Figura 8: Imagens do conjunto de dados com exemplos negativos quanto à presença de pessoas

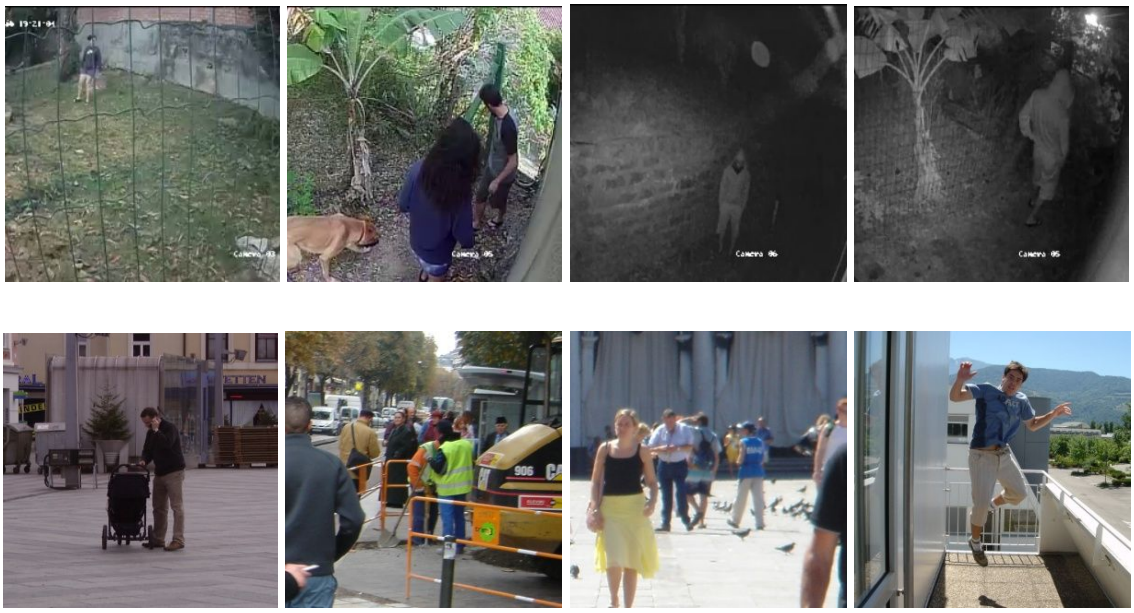

Figura 9: Imagens do conjunto de dados com exemplos positivos quanto à presença de pessoas

4.4. Experimento 1 - Modelos de detecção

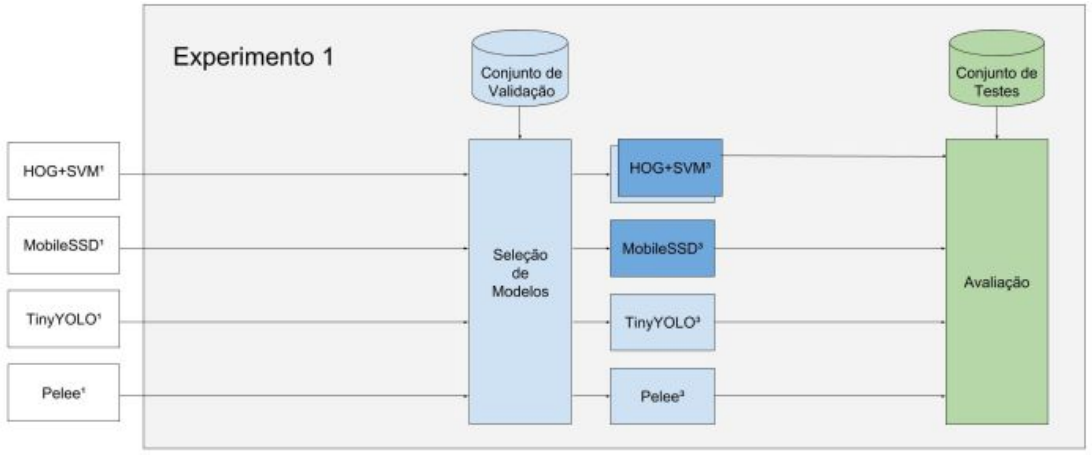

Figura 10: Esquema representando o primeiro experimento com modelos de detecção

O objetivo no primeiro experimento é identificar, dentre um conjunto de modelos de detecção de objetos, aquele que apresentar os melhores resultados em termos de velocidade de processamento, medido em termos de *frames* por segundo (FPS), e acurácia na tarefa de reconhecimento de pessoas nas imagens, com o intuito de validar a hipótese de que seria possível atingir o objetivo proposto pelo trabalho utilizando somente modelos de detecção pré-treinados. Foram selecionados quatro modelos, onde três deles são modelos de redes neurais reconhecidos por sua eficiência em termos de consumo de memória e processamento: os modelos TinyYolo [18], MobileSSD [87], [84] e Pelee [88], [89] além de um modelo baseado em SVM linear combinado com extração de atributos HOG [1], utilizado como referência, por ser um modelo de aprendizagem simples, mas capaz de resolver a tarefa proposta com bom desempenho, tanto em termos de velocidade de

processamento quanto na qualidade das detecções realizadas [1] . Na Figura 10 mostra-se o fluxo de tarefas realizadas durante o experimento.

Em todos os casos, foram utilizados modelos pré-treinados, que primeiramente foram avaliados com diferentes versões de hiperparâmetros relevantes à sua execução. Todos os modelos possuem como parâmetro a confiança para que uma *bounding box* detectada seja considerada relevante. Além da confiança, o modelo baseado em atributos HOG e SVM, que chamaremos a partir daqui simplesmente de Hog+Svm, apresenta outros 2 hiperparâmetros, referentes ao tamanho do passo da janela deslizante que percorre a imagem, e ao *padding* utilizado durante o percorrimento.

A seleção dos hiperparâmetros foi realizada avaliando-se os modelos em relação ao conjunto de validação, descrito na seção 4.1, referente ao conjunto de dados. Para a etapa de seleção dos hiperparâmetros, não é avaliada a performance em termos de *frames* por segundo. Avalia-se apenas a acurácia de cada modelo, para que a versão com os melhores hiperparâmetros seja então avaliada posteriormente em um dispositivo Raspberry Pi, onde então os modelos são comparados em termos de velocidade de processamento, tendo a sua taxa de *frames* por segundo avaliada na execução sobre o conjunto de testes. A descrição de cada modelo, além dos resultados decorrentes da seleção de seus hiperparâmetros são apresentados a seguir.

4.4.1. Hog + SVM

Para comparar a performance dos modelos baseados em redes neurais com abordagens distintas, optou-se por utilizar um modelo de detecção baseado na extração de atributos HOG, em conjunto com um classificador SVM [1]. A extração dos atributos HOG (*Histogram of Oriented Gradients*) e o classificador SVM pré-treinado utilizados fazem parte da biblioteca OpenCV [93].

Para a escolha do melhor conjunto de hiperparâmetros, avaliou-se diferentes combinações destes, baseado em uma busca em grade (*grid search)* [94], onde, além da confiança, foram avaliados o *window stride*, que corresponde ao tamanho do passo, em pixels, que a janela deslizante realiza ao percorrer a imagem e o *padding*, que corresponde à uma margem extra na janela, também em pixels, ao extrair os atributos. Os resultados, avaliados em termos de acurácia sobre o conjunto de validação, são apresentados na Figura 11.

Para a seleção dos hiperparâmetros deste modelo, a detecção foi realizada sobre o conjunto de validação com imagens com dimensões 300x300. O modelo que apresentou maior acurácia foi o modelo com *stride* 2, padding 4 e confiança 0.6, com uma acurácia de 0.6245. Os modelos com maior taxa de *frames* por segundo foram os modelos com stride 8 e padding 8, o que explica-se devido ao fato de um tamanho maior do passo da janela de detecção naturalmente resultar em menos passos computacionais; enquanto o *padding* influencia no tamanho da janela deslizante, fazendo com que um

padding maior também resulte em um número menor de passos computacionais executados.

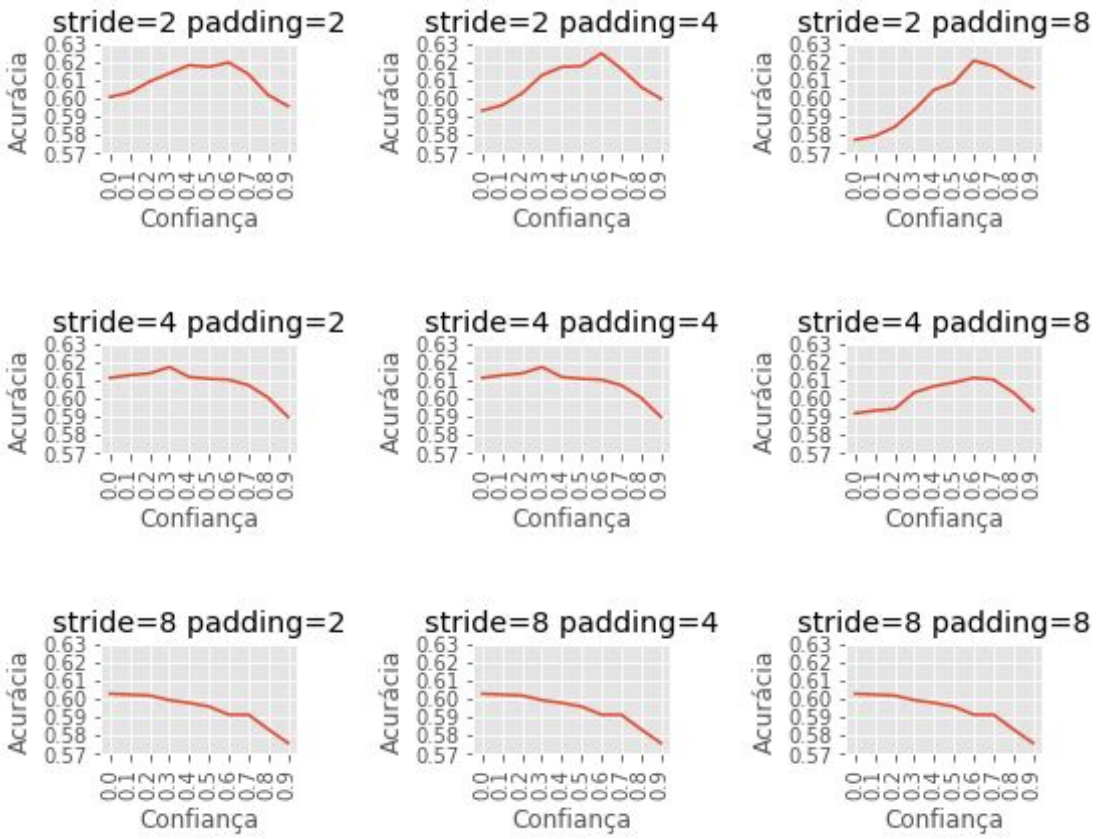

Figura 11: Resultados da seleção de hiperparâmetros do modelo Hog+Svm

Os exemplos das detecções com erros realizados pelo modelo com melhor acurácia são apresentados na Figura 12, onde erros do tipo 1 são apresentados na linha superior e erros do tipo 2 na linha de baixo:

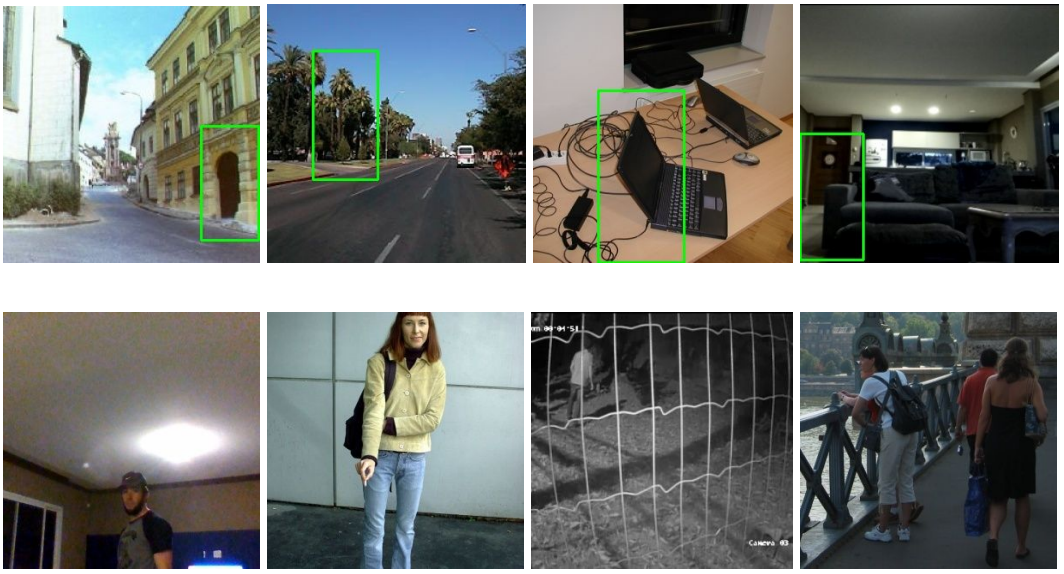

Figura 12: Erros cometidos pelo modelo HOG+SVM v2

Ficam evidentes, a partir do exemplos demonstrados, as limitações do modelo selecionado como *baseline* do estudo. Além disso, na Figura 13 são demonstrados alguns exemplos interessantes em que o modelo detectou a presença de pessoas na imagem. Alguns casos, apesar do modelo detectar a presença de pessoas na imagem, a *bounding box* de detecção não tem relação alguma com a posição da pessoa.

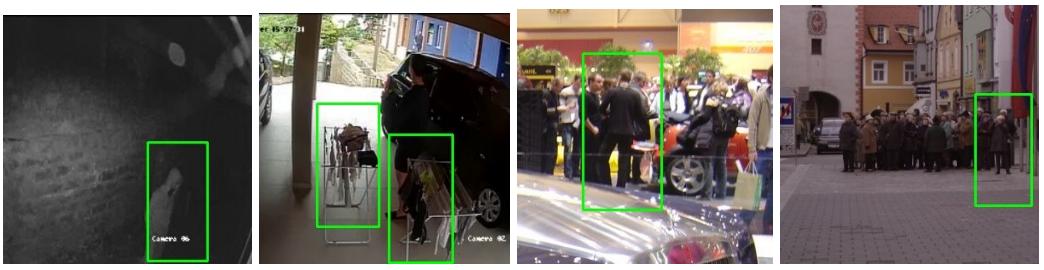

Figura 13: exemplos interessantes de detecções realizadas pelo modelo

O modelo com maior taxa de *frames* por segundo (HOG+SVM v1) obteve acurácia de 59.65%, com uma taxa de 4.97 FPS. O modelo com maior acurácia no conjunto de validação (HOG+SVM v2) atingiu 61.85% de acurácia no conjunto de testes, a com uma taxa de 0.45 FPS. A matriz de confusão apresentada na Tabela 1 contém os resultados do modelo HOG+SVM v1, enquanto a matriz de confusão apresentada na Tabela 2 contém os resultados do modelo HOG+SVM v2.

	classe predita		
classe real		positivo	negativo
	positivo	**271**	**729**
	negativo	**78**	**922**

Tabela 1: matriz de confusão com os resultados do modelo HOG+SVM v1

	classe predita		
classe real		positivo	negativo
	positivo	**306**	**694**
	negativo	**73**	**927**

Tabela 2: matriz de confusão com os resultados do modelo HOG+SVM v2

4.4.2. TinyYOLO

Entre os modelos avaliados, experimentou-se a versão reduzida do modelo Yolo (You Only Look Once), caracterizada por utilizar uma base convolucional eficiente em termos computacionais , baseada na implementação

de referência da rede neural Darknet, chamada por seus autores de TinyYolo [18]. Esta versão é mais rápida, mas com menor acurácia, em relação à rede Yolo, originalmente proposto por seus autores. Utilizou-se os pesos da rede pré-treinada conforme descrito em [18], em que utiliza-se uma base convolucional pré-treinada no conjunto de dados ImageNet, seguido de um treinamento do modelo de detecção sobre o conjunto de dados MS-COCO, posteriormente treinada sobre o conjunto PascalVOC. As imagens utilizadas neste modelo possuem as dimensões 416x416. Os resultados da execução sobre o conjunto de dados de validação são apresentados na Figura 14, onde apresenta-se os valores de acurácia obtidos com diferentes valores de confiança:

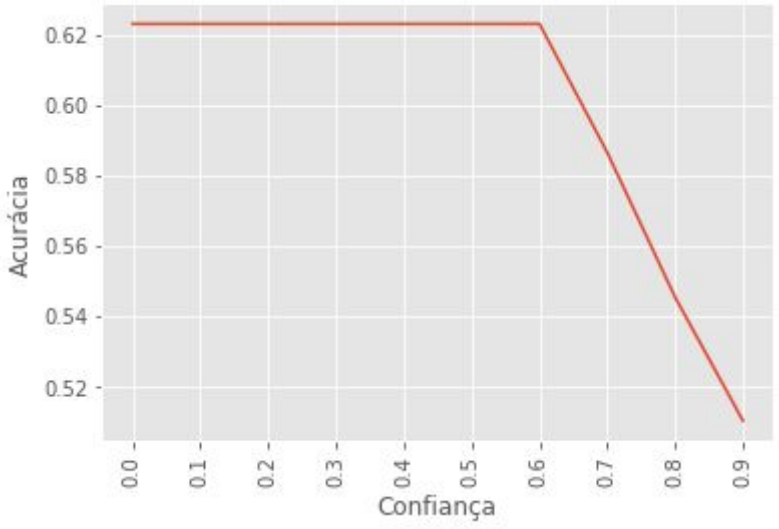

Figura 14: resultados do modelo TinyYolo com diferentes níveis de confiança

Os resultados demonstram que quanto menor a confiança, melhor o desempenho do modelo, quando avaliado quanto ao critério de qualidade das detecções, medido aqui através da acurácia no conjunto de validação. Assim

como nos demais modelos, não avalia-se a qualidade das *bounding boxes* geradas. Este modelo quase não cometeu erros do tipo 1 (falso-positivo). Exemplos de erros de detecção do modelo são apresentados na Figura 15:

Figura 15: Erros cometidos pelo modelo TinyYolo

O modelo com maior acurácia no conjunto de validação atingiu 61.65% de acurácia no conjunto de testes, a com uma taxa de 0.2 FPS. A matriz de confusão com os resultados deste modelo é apresentada na Tabela 3:

	classe predita		
classe real		positivo	negativo
	positivo	**239**	**761**
	negativo	**2**	**998**

Tabela 3: matriz de confusão com os resultados do modelo TinyYolo

4.4.3. MobileSSD

Assim como o modelo TinyYolo, o modelo MobileSSD pertence a um conjunto de modelos caracterizados por exigirem pouco do hardware utilizado

para realizar as detecções [84]. A família de detectores SSD (*Single Shot Detector*) oferece desempenho na detecção inferior, mas comparável, em relação aos modelos considerados estado-da-arte na tarefa de detecção de objetos. A base convolucional do modelo, baseada na arquitetura de rede neural MobileNet [16], desenvolvida especificamente para ser utilizada em ambientes computacionais restritos, possui diferentes versões, de acordo com as dimensões das imagens utilizadas, e no caso do presente trabalho, utilizou-se a versão SSD300, com imagens 300x300. Os resultados para os diferentes níveis de confiança são apresentados na Figura 16:

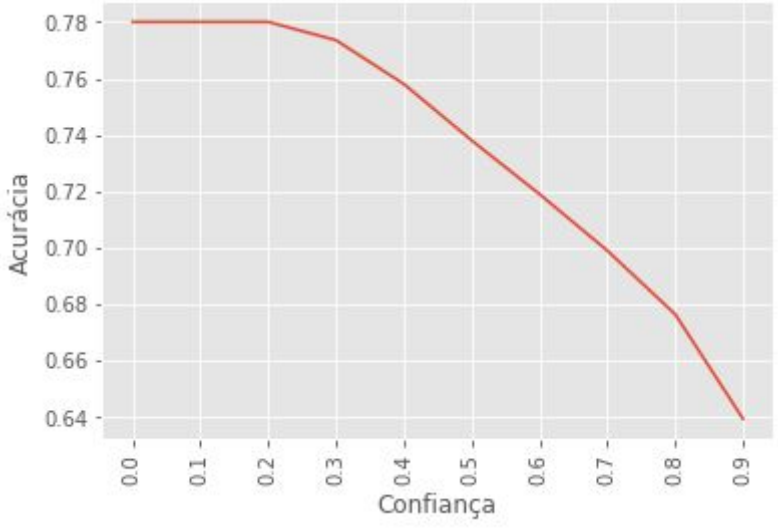

Figura 16: resultados do modelo MobileSSD com diferentes níveis de confiança

Fica evidente, que também no caso deste modelo, assim como no modelo TinyYolo, menores níveis de confiança implicam em melhores níveis de acurácia. Este modelo, ao se analisar qualitativamente as detecções realizadas, apresentou os melhores resultados em relação à sobreposição de

bounding boxes Exemplos de erros cometidos pelo modelo são apresentados na Figura 17, onde na linha de cima são apresentados erros do tipo 1, e abaixo, erros do tipo 2:

Figura 17: Erros cometidos pelo modelo MobileSSD

Apesar deste modelo ter sido o que apresentou maior qualidade das *bounding boxes* produzidas em uma inspeção visual qualitativa, fica claro que a sua simples utilização como solução definitiva do problema proposto ainda está abaixo das exigências de um sistema de segurança residencial comercial.

Figura 18: Exemplos interessantes do modelo MobileSSD

Na Figura 18 são mostrados exemplos que, apesar de não serem erros de acordo com os critérios de avaliação utilizados, demonstram os problemas em utilizar-se o modelo como solução definitiva para o problema.

O modelo com maior acurácia no conjunto de validação atingiu 76.95% de acurácia no conjunto de testes, a com uma taxa de 0.55 FPS. A matriz de confusão com os resultados deste modelo é apresentada na Tabela 4:

	classe predita		
classe real		positivo	negativo
	positivo	**575**	**425**
	negativo	**36**	**964**

Tabela 4: matriz de confusão com os resultados do modelo MobileSSD

4.4.4. Pelee

O modelo Pelee [88] é composto por uma base convolucional que, assim como nos modelos de rede neural avaliados anteriormente, é caracterizada por sua eficiência em termos computacionais, sendo indicado pelos seus autores para utilização em dispositivos móveis. Esta base convolucional, chamada de PeleeNet, possui 66% do tamanho do modelo MobileNet (em termos de quantidade de parâmetros), sendo também combinada com um detector SSD na solução final [88] . Os resultados da avaliação da acurácia sobre o conjunto de validação, com diferentes valores de confiança são apresentados na Figura 19:

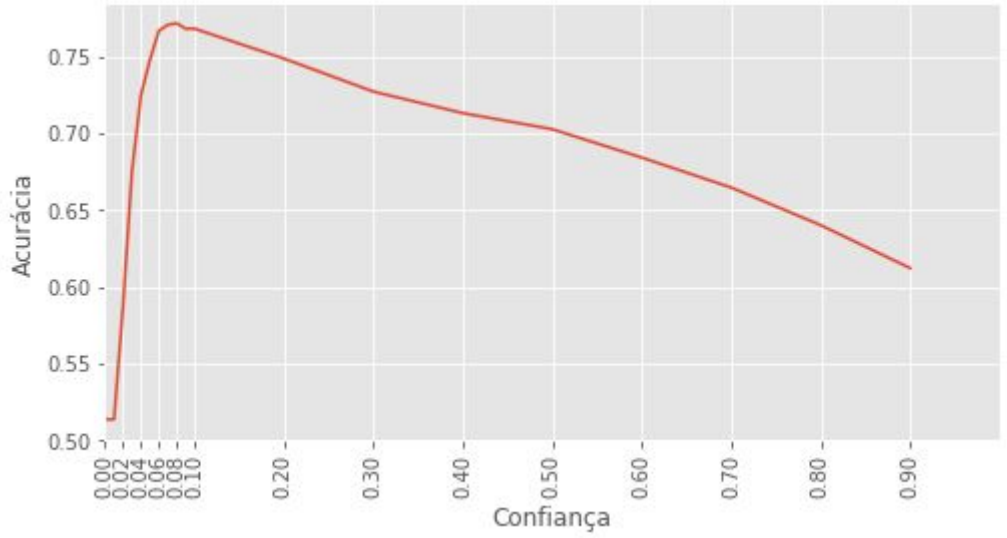

Figura 19: resultados do modelo Pelee com diferentes níveis de confiança

Durante a avaliação deste modelo, foi constatado que o melhor valor de acurácia encontrava-se com confiança 0.1, o que sugeria que melhores valores poderiam ser encontrados experimentando-se valores que variam entre 0.0 e 0.1. Neste caso, incluiu-se esta faixa de valores na avaliação, demonstrando que, de fato, os melhores valores para o parâmetro de confiança estava situado nesta faixa, mais especificamente com confiança de 0.08.

Na Figura 20 são apresentados exemplos de erros cometidos pelo modelo, em que na linha de cima estão os erros do tipo 1, e abaixo, os erros do tipo 2:

Figura 20: Erros cometidos pelo modelo Pelee. Acima: exemplos de falsos-positivos. Abaixo: exemplos de falsos negativos.

Exemplos interessantes, onde o modelo apesar de ter detectado a presença de pessoas, ficam claras as limitações do modelo, são apresentados na Figura 21. Apesar das pessoas terem sido detectadas, várias *bounding boxes* são sobrepostas, inclusive com objetos que não são pessoas.

Figura 21: Exemplos interessantes de detecções realizadas pelo modelo Pelee

Os exemplos apresentados sugerem a possibilidade da utilização da técnica de *non-maximum suppression* [25], evitando que *bounding-boxes* com

alto nível de sobreposição sejam apresentadas no resultado das detecções. Como neste caso não se está avaliando a qualidade das *bounding boxes*, optou-se por não repetir o experimento com a utilização desta técnica.

O modelo com maior acurácia no conjunto de validação atingiu 76.15% de acurácia no conjunto de testes, com uma taxa de 0.56 FPS. A matriz de confusão com os resultados deste modelo é apresentada na Tabela 5:

	classe predita		
classe real		positivo	negativo
	positivo	**574**	**426**
	negativo	**51**	**949**

Tabela 5: matriz de confusão com os resultados do modelo Pelee

4.4.5. RESULTADOS

Definidos os valores ideais para os hiperparâmetros de cada modelo, a melhor versão de cada modelo foi avaliada ao ser executada sobre o conjunto de testes, diretamente em um dispositivo Raspberry Pi. Os valores de acurácia e quantidade de *frames* detectados por segundo são apresentados nas Figuras 22 e 23, respectivamente:

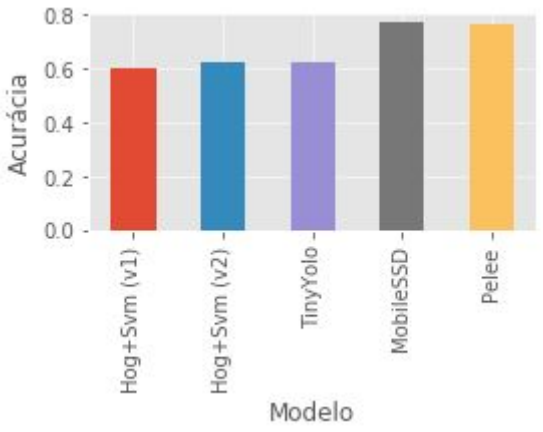

Figura 22: valores de acurácia obtidos no conjunto de testes durante o experimento 1

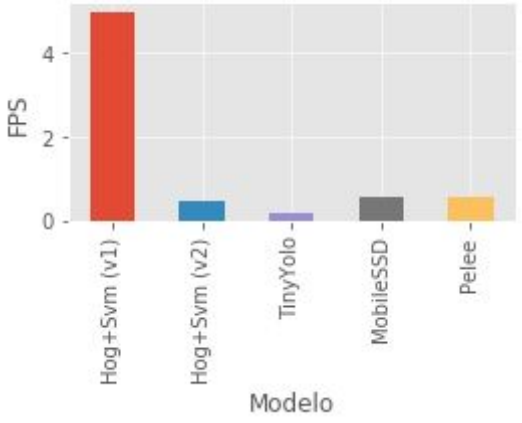

Figura 23: valores de FPS obtidos no conjunto de testes durante o experimento 1

Modelo	Acurácia	FPS
Hog+Svm (v1)	0.5965	4.97
Hog+Svm (v2)	0.6165	0.45
TinyYolo	0.6185	0.20
MobileSSD	0.7695	0.55
Pelee	0.7615	0.56

Tabela 6: resultados no conjunto de testes dos modelos selecionados a partir da avaliação no conjunto de validação, com imagens do tamanho original de cada modelo

Levando-se em consideração o fato dos modelos trabalharem com diferentes tamanhos de imagens, o que afeta o resultado correspondente à taxa de FPS, experimentou-se a utilização dos modelos recebendo como entrada apenas imagens 300x300. Dessa forma, a camada inicial das redes não possui os valores de pesos definidos conforme o treinamento, o que, em tese, faria com que a acurácia seja afetada negativamente, mas em contrapartida pode-se comparar os valores de FPS entre os diferentes modelos. Os resultados são apresentados na Tabela 7:

Modelo	Acurácia	FPS
Hog+Svm (v1)	0.5965	4.97
Hog+Svm (v2)	0.6165	0.45
TinyYolo	0.618	0.20
MobileSSD	0.7695	0.55
Pelee	0.7615	0.52

Tabela 7: resultados no conjunto de testes dos modelos selecionados a partir da avaliação no conjunto de validação, com 300x300

Pode-se observar que os modelos MobileSSD e Pelee apresentaram desempenhos muito semelhantes, tanto em tempo de processamento, quanto na qualidade nas detecções, avaliadas de acordo com a acurácia, e muito superiores aos demais modelo testados. O modelo TinyYolo destacou-se negativamente por apresentar desempenho inferior ao modelo utilizado como *baseline*. O modelo Hog+Svm v1, por utilizar valores altos no tamanho de passo (*stride*) da janela de detecção, apresentou valores muito superiores de FPS em relação aos demais modelos.

Optou-se por selecionar o modelo com melhor acurácia (MobileSSD) e o modelo com melhor taxa de FPS (Hog+Svm v1), como candidatos a fazerem parte da solução híbrida desenvolvida no segundo experimento.

4.5. Experimento 2 - Modelo híbrido

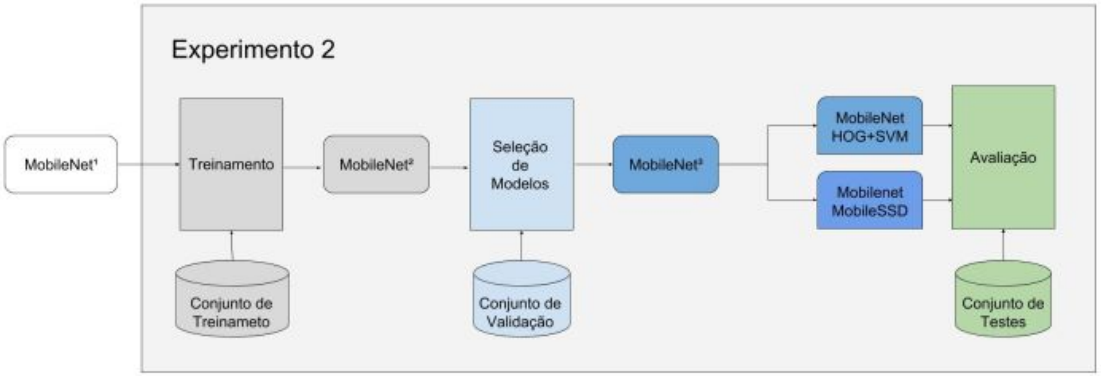

Figura 24: Esquema representando a sequência de passos realizados durante o experimento 2

A partir dos resultados obtidos com o primeiro experimento, nos quais ficam evidentes os valores de acurácia e FPS reduzidos, propõe-se em segundo experimento, com uma solução híbrida, envolvendo dois estágios: um primeiro estágio de classificação, no qual uma rede convolucional, semelhante à base convolucional do modelo MobileSSD do primeiro experimento, é utilizada para identificar imagens com a presença de pessoas, e um segundo estágio, onde o modelo de detecção melhor avaliado em termos de acurácia é executado sobre as imagens classificadas positivamente no primeiro estágio. O resultado é avaliado de acordo com a acurácia e a velocidade de processamento em termos de FPS. São avaliados dois valores distintos de

acurácia: a acurácia do modelo de classificação, e acurácia do modelo de detecção. Este último é avaliado levando-se em conta apenas as imagens classificadas positivamente no primeiro estágio do modelo.

4.5.1. ARQUITETURA

A arquitetura utilizada na rede convolucional que compõe o modelo de classificação segue a arquitetura utilizada na base convolucional do modelo de detecção MobileSSD. Trata-se da rede MobileNet [16], cuja implementação utilizada está presente na biblioteca Keras [95]. Foram retiradas as últimas camadas da rede original, correspondente ao topo do modelo, mantendo-se todas as camadas convolucionais e a camada de AveragePooling. Após a camada de pooling, o topo foi substituído por uma camada de *Dropout*, seguida de uma camada totalmente conectada (*fully-connected*), com ativação Softmax.

O modelo final é composto por uma camada de entrada própria para imagens 224x224x3, seguida por 13 blocos de Depthwise Separable Convolutions [67] (Figura 25), compostos por uma camada convolucional *depthwise* 3x3, uma camada de BatchNormalization, uma camada de ativação ReLU, uma convolução *pointwise* 1x1, uma segunda camada de *BatchNormalization*, seguida de outra ativação ReLU. Após os 13 blocos convolucionais, estão a camada de *AveragePooling*, seguida da camada de *Dropout*, e por fim uma camada *fully-connected*, com ativação *Softmax*, conforme apresentada à direita na Figura 25.

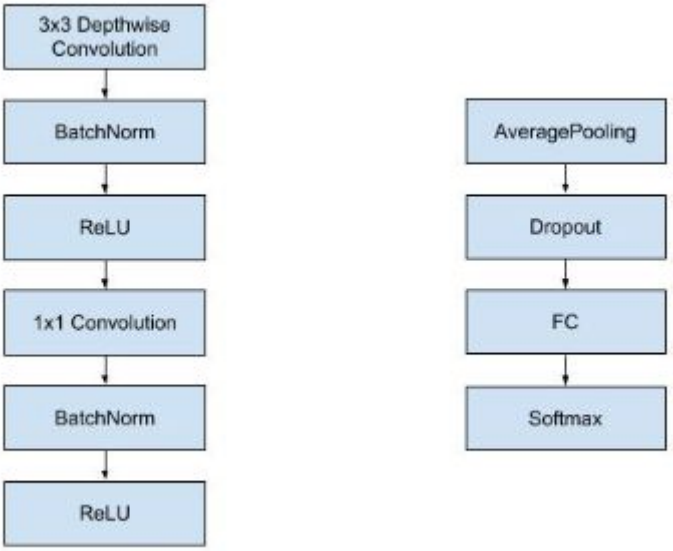

Figura 25: À esquerda: bloco convolucional *Depthwise Separable Convolution*. À direita: camadas finais do modelo, incluídas após a extração das camadas finais do modelo original.

4.5.2. TREINAMENTO

Os pesos da base convolucional da rede foram inicializados com valores resultantes de um pré-treinamento no conjunto de dados ImageNet, fornecidos como opção pela biblioteca Keras. A função-custo utilizada é a da entropia-cruzada (*cross-entropy*). A rede foi otimizada utilizando-se RMSprop, em que o valor da taxa de aprendizagem foi selecionada avaliando-se a acurácia no conjunto de validação. Outros hiperparâmetros selecionados através do conjunto de validação foram a taxa de Dropout e a quantidade de camadas efetivamente treinadas, que não tem os seus pesos mantidos fixos durante o treinamento, no processo de *fine-tuning*.

A busca de hiperparâmetros foi realizada em duas etapas: na primeira etapa foi realizada uma busca em *grid,* experimentando combinações de

valores de taxa de aprendizagem e número de camadas treinadas, durante 10 épocas, em que os resultados estão apresentados nas Figuras 26 e 27; na segunda etapa, os modelos que obtiveram o menor valor de custo nos exemplos de treinamento foram treinados durante 10 epócas, com diferentes valores na taxa de Dropout. O modelo que apresentou melhor desempenho na acurácia no conjunto de validação, continuou a ser treinado por mais 60 épocas, tendo sua taxa de aprendizagem diminuída a cada 20 épocas. As curvas de aprendizagem são apresentadas na Figura 28.

Além da utilização de Dropout para regularização, outra estratégia empregada foi a da Aumentação de Dados (*Data Augmentation*), onde durante o treinamento a imagem passou por transformações como deslocamento horizontal e vertical, ampliação, redução e rotação.

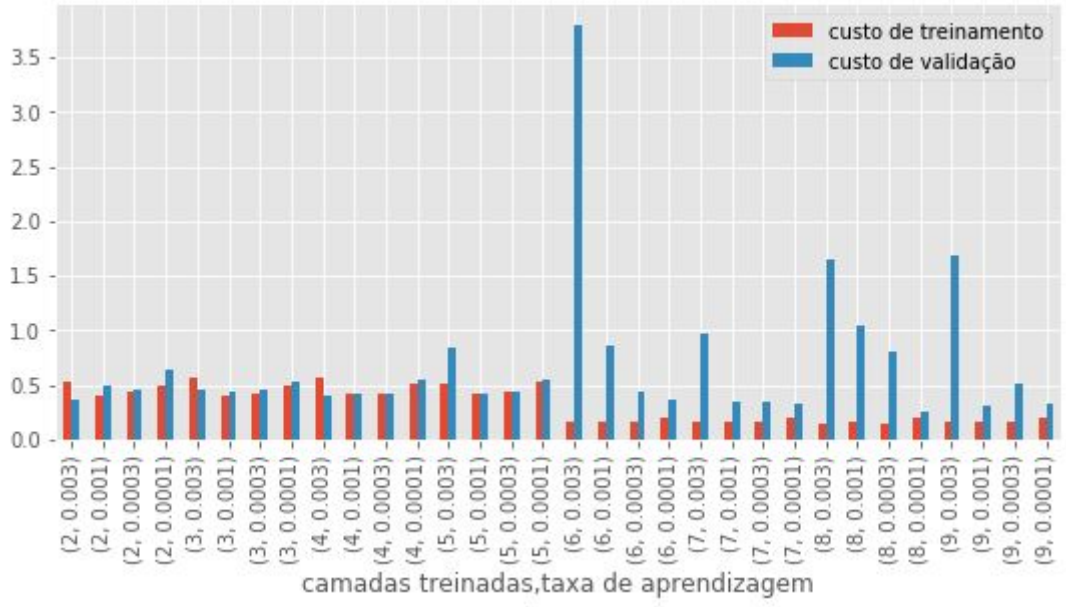

Figura 26: custo de treinamento e custo de validação obtidos por cada combinação de número de camadas treinadas e taxa de aprendizagem.

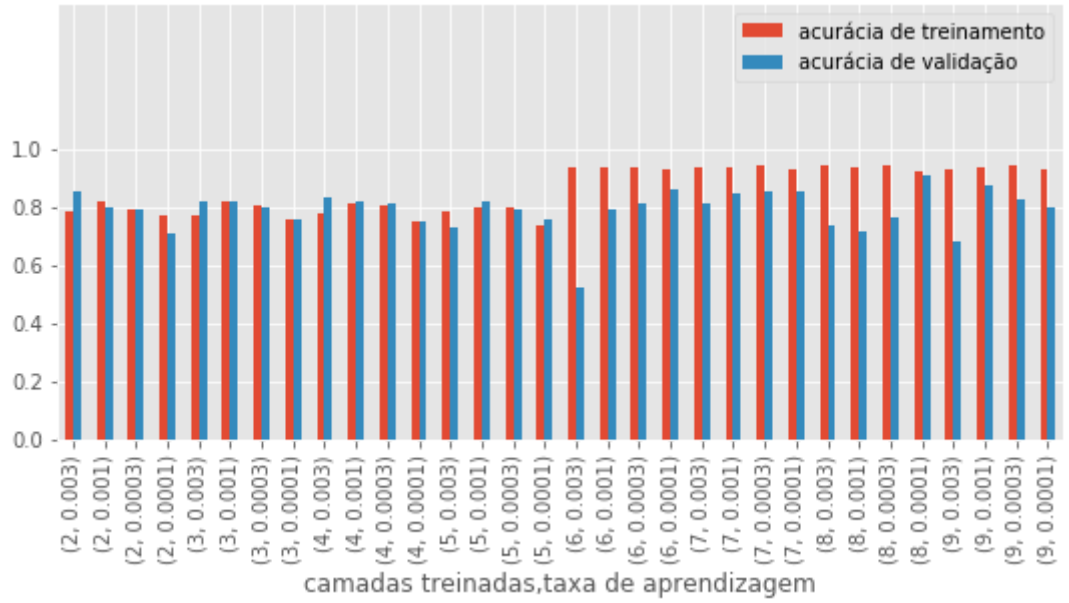

Figura 27: acurácia de treinamento e acurácia de validação obtidos por cada combinação de número de camadas treinadas e taxa de aprendizagem

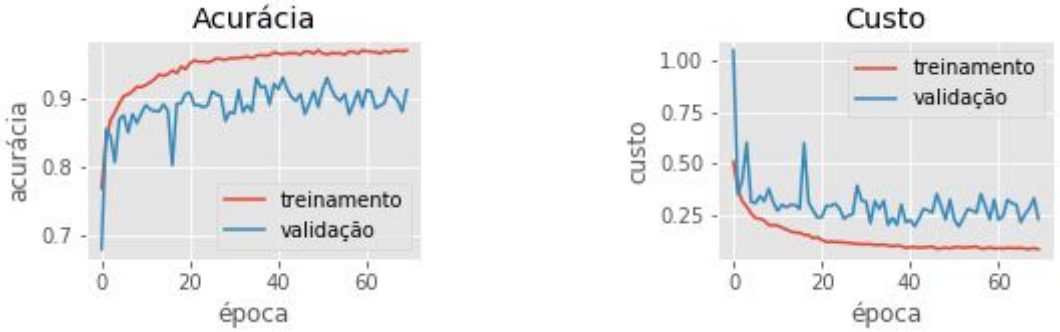

Figura 28: curvas de aprendizagem do modelo de classificação treinado. À esquerda: curva de acurácia. À direita: curva de custo

4.5.3. RESULTADOS

Após realizado o treinamento do componente de classificação presente na solução, experimentou-se utilizá-lo em conjunto aos dois modelos selecionados a partir do primeiro experimento. Os resultados são apresentados

na Tabela 8 e na Figura 29. Pode-se observar que a acurácia na tarefa de classificação é a mesma nos dois modelos, pois o componente responsável por esta etapa é o mesmo nos dois casos. Entre os exemplos classificados como positivos pelo primeiro componente do modelo, avalia-se a acurácia do modelo de detecção, ficando claro que o desempenho em termos de acurácia no modelo baseado no componente de detecção MobileSSD é bem superior ao modelo baseado em Hog+Svm, como já havia ficado claro no primeiro experimento. Na terceira coluna da Tabela 8, estão os valores de FPS da solução final, nos dois casos.

Modelo	Acurácia (classificação)	Acurácia (detecção)	FPS
Classificador Mobilenet, Detector Hog+Svm v1	88.65%	57%	1.96
Classificador Mobilenet, Detector MobileSSD	88.65%	85%	1.29

Tabela 8: Resultados de acurácia e FPS do segundo experimento

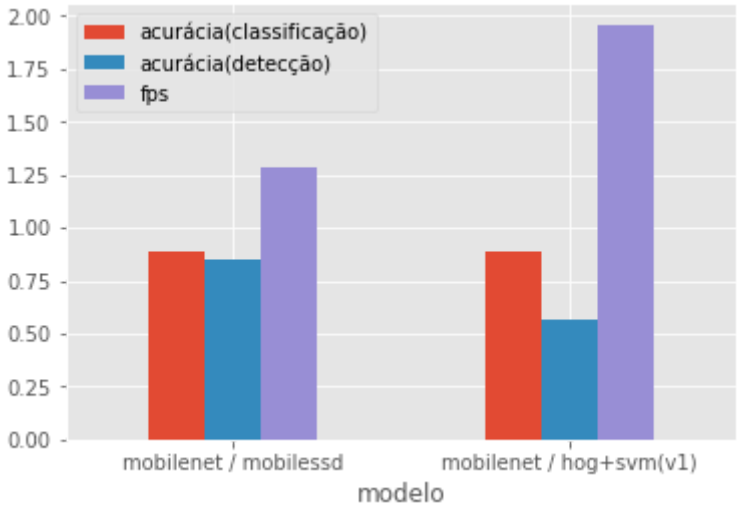

Figura 29: resultados finais das duas soluções avaliadas no experimento 2

	classe predita		
classe real		positivo	negativo
	positivo	**814**	**185**
	negativo	**42**	**958**

Tabela 9: matriz de confusão com os resultados da etapa de classificação

Na Tabela 9 estão os resultados da etapa de classificação. Os resultados da matriz de confusão de cada etapa de detecção não foram avaliados. Diante os resultados obtidos, ficou evidente a superioridade do modelo proposto envolvendo a utilização do classificador MobileNet, em conjunto ao detector MobileSSD, em relação ao modelo de detecção pré-treinado que apresentou melhores resultados em termo de acurácia no primeiro experimento. Atingiu-se um incremento de 15.20% na acurácia, e de 134% na quantidade de *frames* por segundo (FPS). Em relação ao modelo selecionado como *baseline*, o modelo HOG+SVM de melhor taxa de FPS no

primeiro experimento, atingiu-se um desempenho em termos de acurácia 48.61% superior, e em termos de FPS, 60.56% inferior.

5. CONCLUSÕES

O presente trabalho propôs uma solução baseada em redes neurais para o reconhecimento de pessoas em imagens, no contexto de um sistema de segurança residencial, na plataforma de hardware Raspberry Pi. Procurou-se na literatura por modelos de redes neurais capazes de resolver o problema levando-se em conta o ambiente computacional restrito apresentado como requisito do problema. Foram propostos dois experimentos, visando avaliar as limitações e possibilidades de modelos de aprendizagem neste contexto. Para a realização dos experimentos, construiu-se um conjunto de dados com 11.000 imagens, composto por 9.000 imagens capturadas pelo autor do trabalho, junto a 2.000 imagens originárias do conjunto de dados INRIA Person Dataset. Este conjunto foi então separado em um conjunto de treinamento, com 7.000 imagens, um conjunto de validação, com 2.000 imagens e um conjunto de testes, também com 2.000 imagens.

No primeiro experimento avaliou-se a solução do problema de reconhecimento de pessoas utilizando-se apenas modelos de detecção de objetos, na tentativa de se obter uma solução mais simples, com apenas uma etapa, ainda assim capaz de apontar, através de *bounding boxes*, a posição das pessoas na imagem. Foram avaliados 3 modelos de detecção de objetos

baseados em redes neurais, especificamente desenvolvidos para serem utilizados em cenários onde hajam restrições de recursos computacionais: os modelos TinyYolo, MobileSSD e Pelee. Junto a esses modelos, foi avaliado também um modelo baseado em extração de atributos HOG em conjunto com um classificador SVM linear, como referência, por apresentar desempenho considerável em termos de *frames* por segundo e qualidade das detecções. Os resultados do experimento demonstram que uma solução baseada apenas em modelos de detecção de objetos apresenta limitações severas, tanto na avaliação baseada em critérios de velocidade de processamento, através da quantidade de quadros avaliados por segundo, quanto na avaliação baseada no critério de qualidade na tarefa de reconhecimento, neste caso, a acurácia. O modelo que apresentou melhores resultados em termos de acurácia foi o MobileSSD, que ainda apresentou, entre os modelos baseados em redes neurais, o melhor desempenho também quando avaliado em termos de *frames* por segundo. Apesar do modelo utilizado como base de comparação, o modelo baseado em atributos HOG + SVM, em algumas das configurações experimentadas apresentar desempenho superior, em termos de velocidade de processamento, apresenta acurácia inferior a quase todos os modelos de rede neural experimentados, excetuando-se o modelo TinyYolo, que apresentou acurácia e quantidade de *frames* por segundo inferiores.

No segundo experimento, avalia-se uma solução híbrida, baseada em uma etapa inicial onde utiliza-se um modelo de classificação, para identificar a presença ou não de pessoas na imagem, seguida de uma etapa onde são utilizados os modelos de detecção avaliados no primeiro experimento,

mantendo-se a possibilidade de localizar na imagem onde estão as pessoas reconhecidas. Neste experimento foi treinado um modelo de rede neural de classificação, baseado na mesma arquitetura utilizada como base convolucional do modelo MobileSSD, a arquitetura MobileNet. Utilizou-se a técnica de *fine-tuning*, aproveitando-se de um pré-treinamento da rede neural no conjunto de dados ImageNet, substituindo-se as últimas camadas da rede, e complementando-se o treinamento com o conjunto de dados construído para o problema proposto no trabalho. Avaliando-se diferentes configurações dos hiperparâmetros, e combinando-se o modelo treinado com os modelos mais eficientes, em termos de acurácia e quantidade *frames* por segundo entre os modelos avaliados no primeiro experimento, chegou-se a uma solução capaz de classificar as imagens de pessoas, correspondente à primeira etapa da solução, com acurácia de 88.65%, e capaz de detectar as pessoas na imagem, na segunda etapa da solução, com acurácia de 85% e 1.29 FPS, no caso do modelo de classificação combinado com o modelo MobileSSD. Combinando-se o classificador com o modelo baseado em extração de atributos HOG + SVM linear, atingiu-se 57% de acurácia na segunda etapa do modelo, mantendo-se a acurácia da primeira etapa do modelo, que manteve-se o mesmo, com uma taxa de 1.96 FPS.

Durante os experimentos, ficaram evidentes as limitações dos modelos avaliados quanto ao desempenho em termos de velocidade de processamento, onde os melhores modelos, segundo este critério, não foram capazes de realizar o problema proposto com uma taxa de *frames* por segundo que viabilize a sua utilização em cenários críticos, que exijam tempo de resposta

imediato. Quanto à qualidade das detecções realizadas, também ficaram evidentes as limitações quanto à qualidade das *bounding boxes* identificadas, ao se utilizar os modelos de detecção com níveis de confiança que maximizem a acurácia dos modelos. A acurácia final da solução apresentou níveis satisfatórios, diante das restrições presentes no conjunto de dados utilizado, quanto ao número de exemplos e diversidade das imagens utilizadas. Apesar da utilização de *fine-tuning* ter permitido atingir-se uma acurácia de 88.65%, acredita-se que o desempenho possa ser drasticamente melhorado utilizando-se um conjunto de dados maior.

5.1. Trabalhos futuros

Sugere-se como possibilidades de trabalhos futuros explorar a utilização de modelos de classificação e detecção mais exigentes em termos de hardware, em experimentos semelhantes. Este trabalho não realizou de maneira intensiva a possibilidade de utilizar modelos computacionalmente mais exigentes diretamente em dispositivos Raspberry Pi. Da mesma forma, é possível explorar a possibilidade de utilizar modelos que sejam o estado-da-arte na tarefa de detecção em arquiteturas *Fog* [10], delegando-se o processamento pesado para nós de rede presentes em uma rede local, mantendo-se assim a possibilidade de manter uma baixa latência no tempo de resposta. Outro ponto a ser explorado em trabalhos futuros é a utilização de um conjunto de dados com um número maior de exemplos, o que provavelmente

aumentaria substancialmente a acurácia final na primeira etapa no modelo proposto no segundo experimento. Por fim, outra possibilidade seria a construção de um conjunto de dados com anotações próprias para o treinamento de modelos de detecção, com as coordenadas das *bounding boxes* referentes às pessoas das imagens, o que possibilitaria o *fine-tuning* dos modelos experimentados, o que provavelmente acarretaria em resultados superiores, tanto no primeiro experimento realizado, quanto na segunda etapa do segundo experimento.

REFERÊNCIAS

[1] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[2] Dollár, Piotr, et al. "Pedestrian detection: A benchmark." *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009.

[3] Seemann, Edgar, Mario Fritz, and Bernt Schiele. "Towards robust pedestrian detection in crowded image sequences." *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007.

[4] Geronimo, David, et al. "Survey of pedestrian detection for advanced driver assistance systems." *IEEE transactions on pattern analysis and machine intelligence* 32.7 (2010): 1239-1258.

[5] Ahmedali, Trevor, and James J. Clark. "Collaborative multi-camera surveillance with automated person detection." *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*. IEEE, 2006.

[6] Bertozzi, Massimo, et al. "A pedestrian detector using histograms of oriented gradients and a support vector machine classifier." IEEE intelligent transportation systems conference. Vol. 27. 2007.

[7] Zhao, Liang, and Charles E. Thorpe. "Stereo-and neural network-based pedestrian detection." IEEE Transactions on intelligent transportation systems 1.3 (2000): 148-154.

[8] Forsyth, David A., and Jean Ponce. "A modern approach." Computer vision: a modern approach (2003): 88-101.

[9] Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." Advances in neural information processing systems. 2013.

[10] Dastjerdi, Amir Vahid, and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential." Computer 49.8 (2016): 112-116.

[11] Senthilkumar, G., K. Gopalakrishnan, and V. Sathish Kumar. "Embedded image capturing system using raspberry pi system." International Journal of Emerging Trends & Technology in Computer Science 3.2 (2014): 213-215.

[12] Liu, Li, et al. "Deep learning for generic object detection: A survey." arXiv preprint arXiv:1809.02165 (2018).

[13] Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." IEEE CVPR. Vol. 4. 2017.

[14] Ng, Andrew. "Neural Networks and Deep Learning." Coursera. https://www. coursera. org/learn/neuralnetworks-deep-learning 325 (2018).

[15] Karpathy, Andrej. "Cs231n convolutional neural networks for visual recognition." Neural networks 1 (2016).

[16] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[17] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. "Deep learning." 2015(2016).

[18] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[19] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

[20] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[21] Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[22] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[23] Goesele, Michael, et al. "Multi-view stereo for community photo collections." (2007): 1-8.

[24] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.

[25] Agarwal, Shivang, Jean Ogier Du Terrail, and Frédéric Jurie. "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks." arXiv preprint arXiv:1809.03193 (2018).

[26] Smeulders, Arnold WM, et al. "Content-based image retrieval at the end of the early years." IEEE Transactions on Pattern Analysis & Machine Intelligence 12 (2000): 1349-1380.

[27] Lin, Zhe, et al. "Hierarchical part-template matching for human detection and segmentation." Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE, 2007.

[28] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.

[29] Pentland, Alex, Baback Moghaddam, and Thad Starner. "View-based and modular eigenspaces for face recognition." (1994).

[30] Schneiderman, Henry, and Takeo Kanade. "Object detection using the statistics of parts." International Journal of Computer Vision 56.3 (2004): 151-177.

[31] Viola, Paul, Michael J. Jones, and Daniel Snow. "Detecting pedestrians using patterns of motion and appearance." International Journal of Computer Vision 63.2 (2005): 153-161.

[32] Wu, Bo, and Ram Nevatia. "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors." *null*. IEEE, 2005.

[33] Sabzmeydani, Payam, and Greg Mori. "Detecting pedestrians by learning shapelet features." *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007.

[34] Lampert, Christoph H., Matthew B. Blaschko, and Thomas Hofmann. "Beyond sliding windows: Object localization by efficient subwindow search." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.

[35] Felzenszwalb, Pedro F., et al. "Object detection with discriminatively trained part-based models." IEEE transactions on pattern analysis and machine intelligence 32.9 (2010): 1627-1645.

[36] Sadeghi, Mohammad Amin, and David Forsyth. "30hz object detection with dpm v5." European Conference on Computer Vision. Springer, Cham, 2014.

[37] Vaillant, Régis, Christophe Monrocq, and Yann Le Cun. "Original approach for the localisation of objects in images." IEE Proceedings-Vision, Image and Signal Processing 141.4 (1994): 245-250.

[38] Nowlan, Steven J., and John C. Platt. "A convolutional neural network hand tracker." Advances in neural information processing systems (1995): 901-908.

[39] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[40] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

[41] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115.3 (2015): 211-252.

[42] Oquab, Maxime, et al. "Is object localization for free?-weakly-supervised learning with convolutional neural networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[43] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[44] Everingham, Mark, et al. "The pascal visual object classes (voc) challenge." International journal of computer vision 88.2 (2010): 303-338.

[45] Ling, Charles X., Jin Huang, and Harry Zhang. "AUC: a better measure than accuracy in comparing learning algorithms." Conference of the canadian

society for computational studies of intelligence. Springer, Berlin, Heidelberg, 2003.

[46] Samuel, Arthur L. "Some studies in machine learning using the game of checkers." IBM Journal of research and development 3.3 (1959): 210-229.

[47] Mitchell, Tom M. "Machine learning. 1997." Burr Ridge, IL: McGraw Hill 45.37 (1997): 870-877.

[48] Bishop, Christopher M. Neural networks for pattern recognition. Oxford university press, 1995.

[49] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." nature 323.6088 (1986): 533.

[50] McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics 5.4 (1943): 115-133.

[51] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.

[52] Hinton, Geoffrey E. "Learning distributed representations of concepts." Proceedings of the eighth annual conference of the cognitive science society. Vol. 1. 1986.

[53] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[54] Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. "Rmsprop: Divide the gradient by a running average of its recent magnitude." Neural networks for machine learning, Coursera lecture 6e (2012).

[55] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[56] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." AAAI. Vol. 4. 2017.

[57] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[58] Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).

[59] Bergstra, James, et al. "Theano: A CPU and GPU math compiler in Python." Proc. 9th Python in Science Conf. Vol. 1. 2010.

[60] Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning." BigLearn, NIPS workshop. No. EPFL-CONF-192376. 2011.

[61] Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014.

[62] Chen, Tianqi, et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." arXiv preprint arXiv:1512.01274 (2015).

[63] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.

[64] Ito, Yoshifusa. "Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory." Neural Networks 4.3 (1991): 385-394.

[65] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[66] Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." arXiv preprint arXiv:1603.07285 (2016).

[67] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." arXiv preprint (2017): 1610-02357.

[68] Bishop, Christopher M. "Pattern recognition and machine learning (information science and statistics) springer-verlag new york." Inc. Secaucus, NJ, USA (2006).

[69] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

[70] Bertsekas, Dimitri P. Nonlinear programming. Belmont: Athena scientific, 1999.

[71] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of Machine Learning Research 12.Jul (2011): 2121-2159.

[72] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[73] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[74] Chollet, François. "Keras: The python deep learning library." Astrophysics Source Code Library (2018).

[75] Chahal, Karanbir Singh, and Kuntal Dey. "A Survey of Modern Object Detection Literature using Deep Learning." arXiv preprint arXiv:1808.07256 (2018).

[76] Sung, K-K., and Tomaso Poggio. "Example-based learning for view-based human face detection." IEEE Transactions on pattern analysis and machine intelligence 20.1 (1998): 39-51.

[77] Rowley, Henry A., Shumeet Baluja, and Takeo Kanade. "Neural network-based face detection." IEEE Transactions on pattern analysis and machine intelligence 20.1 (1998): 23-38.

[78] Delakis, Manolis, and Christophe Garcia. "text Detection with Convolutional Neural Networks." VISAPP (2). 2008.

[79] Garcia, Christophe, and Manolis Delakis. "A neural architecture for fast and robust face detection." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 2. IEEE, 2002.

[80] Osadchy, Margarita, Yann Le Cun, and Matthew L. Miller. "Synergistic face detection and pose estimation with energy-based models." Journal of Machine Learning Research 8.May (2007): 1197-1215.

[81] Sermanet, Pierre, et al. "Pedestrian detection with unsupervised multi-stage feature learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.

[82] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[83] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[84] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[85] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." IEEE transactions on pattern analysis and machine intelligence (2018).

[86] Figueira, Dario, et al. "The HDA+ data set for research on fully automated re-identification systems." European Conference on Computer Vision. Springer, Cham, 2014.

[87] MobileNet-SSD, 2018. Repositório Github. Disponível em: <https://github.com/chuanqi305/MobileNet-SSD>. Acesso em 18 set. 2018.

[88] Wang, Robert J., et al. "Pelee: A Real-Time Object Detection System on Mobile Devices." arXiv preprint arXiv:1804.06882 (2018).

[89] Wang, Robert J., 2018. Repositório Github. Disponível em: <https://github.com/Robert-JunWang/Pelee>. Acesso em 18 set. 2018.

[90] Dalal, Navneet, and Bill Triggs. "INRIA person dataset." Online: http://pascal. inrialpes. fr/data/human (2005).

[91] Pi, Raspberry. "model B." Raspberrypi.org. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em 18 set. 2018

[92] Pi, Raspberry. "Câmera Raspberry Pi NoIR v2 8MP" Raspberrypi.org. Disponível em: <https://www.raspberrypi.org/products/pi-noir-camera-v2/>. Acesso em 18 set. 2018

[93] OPENCV. OpenCV (Open Source Computer Vision Library) is an open source computer
vision and machine learning software library. Disponível em: <http://opencv.org/>. Acesso em 18 set. 2018.

[94] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1-16.

[95] Chollet, François. "Keras: The python deep learning library." Astrophysics Source Code Library (2018).

Redes Neurais Em Dispositivos Raspberry Pi Para Detecção De Pessoas

Caio Cargnin Cardoso[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

caio.cardoso@grad.ufsc.br

Abstract. *This paper describes the work realized for obtaining an Information Systems Bachelor title, which proposes a machine learning-based solution for person detection in images, in a context of a residential security system application, with the Raspberry Pi hardware platform. The paper describes the two experiments realized, one of them evaluating pre-trained object detection models, and the other, which evaluates an hybrid solution with classification and detection steps.*

Resumo. *Este artigo descreve a realização do Trabalho de Conclusão de Curso de Sistemas de Informação no qual é proposta uma solução baseada em Aprendizagem de Máquina para detecção de pessoas em imagens no contexto de uma aplicação em segurança residencial, a partir da plataforma de hardware Raspberry Pi. São descritos os dois experimentos realizados, um deles envolvendo a avaliação de modelos pré-treinados de detecção de objetos e o outro no qual avalia-se uma solução híbrida de classificação e detecção.*

1. Introdução

Detectar pessoas em imagens trata-se de um problema amplamente explorado na literatura de Visão Computacional. Problemas relacionados com o reconhecimento de pedestres, no contexto de veículos autônomos ou de intrusos, no contexto de sistemas de segurança podem ser resolvidos com algoritmos de aprendizagem de máquina, utilizando-se algoritmos como SVMs e Redes Neurais. Dentre as possibilidades de abordagem, encontram-se modelos de classificação de imagens, que possibilitam a identificação de pessoas na imagem, e modelos de detecção de objetos, que aplicados à pessoas, permitem que identifique-se a quantidade e a posição relativa das pessoas detectadas na imagem. O trabalho apresentado pretende encontrar uma solução para problemas envolvendo a detecção de pessoas em imagens, diretamente à partir de dispositivos computacionais tipicamente utilizados em ambientes de IoT, mais especificamente, a plataforma de hardware Raspberry Pi.

1.1. Objetivos

O trabalho de conclusão de curso tem como objetivo geral propor uma solução de baseada em aprendizagem de máquina, capaz de detectar pessoas utilizando a plataforma de hardware Raspberry Pi, no contexto da aplicação em um sistema de segurança residencial.

Além disso, os objetivos específicos do trabalho envolvem:

- Analisar a fundamentação teórica e o estado da arte no problema de classificação de imagens e detecção de objetos em ambientes computacionais restritos;
- Analisar modelos de aprendizagem de máquina capazes de realizar classificação e detecção de objetos em ambientes computacionais restritos;
- Definir/estabelecer os critérios de qualidade/avaliação dos modelos considerando-se o contexto de segurança residencial;
- Analisar o desempenho, ao realizar a inferência, entre os modelos de aprendizagem selecionados segundo os critérios estabelecidos;
- Propôr uma solução para detectar pessoas em imagens com desempenho superior aos modelos de detecção pré-treinados selecionados, em termos de acurácia e tempo de processamento;

2. Fundamentação Teórica

2.1. Detecção de objetos

Entre os diferentes problemas na área da Visão Computacional, analisar uma imagem e reconhecer os diferentes objetos presentes na cena está entre as tarefas mais complicadas [1]. A tarefa de detecção é uma extensão natural à tarefa de classificação de imagens, apresentando a complexidade adicional de localizar as instâncias dos objetos na imagem. O objetivo na detecção de objetos é identificar a posição na imagem onde estão os diferentes objetos a serem detectados. Além da classe do objeto, deseja-se saber as coordenadas referentes a estes objetos [1]. Resolver o problema da detecção de objetos é um objetivo precursor para a resolução do problema da compreensão semântica do ambiente [2].

2.2. Classificação de Imagens

Na classificação de imagens, o objetivo é inferir a classe de uma imagem, a partir dos seus atributos. Comparada à detecção de objetos, trata-se de uma tarefa mais simples, pois neste caso, considera-se apenas a presença da classe que deseja-se identificar na imagem. Diversos problemas na área podem ser resolvidos a partir de classificação. Como um problema de aprendizagem supervisionada, as etapas envolvidas envolvem a definição, ou construção de um conjunto de dados rotulados, dos quais se extraem atributos, que podem ser os próprios pixels da imagem, ou atributos derivados, como é o caso de atributos HOG e atributos SIFT, para que então se treine um classificador [3].

2.3. Redes Neurais Convolucionais

utilizadas frequentemente em tarefas de Visão Computacional, este tipo de arquitetura de rede neural realiza operações de convolução ao invés de utilizar apenas multiplicações entre as matrizes de pesos e os vetores de ativação da camada anterior [4]. Uma camada convolucional recebe um tensor de entrada, que pode ser uma imagem, com determinada largura, altura e número de canais, sobre o qual operam um conjunto de filtros, ou kernels, de dimensões f x f, e mesmo número de canais, que percorrem este tensor de entrada, gerando um mapa de características (feature map) para cada filtro. Frequentemente utilizadas em problemas envolvendo imagens, fazem com que a quantidade de parâmetros necessários para treinar redes seja reduzido, em decorrência do compartilhamento de parâmetros entre as diferentes regiões de uma imagem, implicando na capacidade em reconhecer padrões independentemente da posição na imagem onde eles ocorram .

3. Trabalhos Relacionados

Considerando-se o contexto da utilização de Redes Neurais Convolucionais Profundas na tarefa de Detecção de Objetos, o trabalho de levantamento no nível de survey apresentado em [1] descreve os modelos utilizados atualmente, além de relacionar um breve histórico de métodos alternativos utilizados anteriormente às redes neurais convolucionais apresentarem-se como abordagem dominante. Além dos modelos utilizados e o estado-da-arte, o trabalho ainda relaciona as diferentes arquiteturas de rede, como single stage detectors, double stage detectors e modelos baseados em partes, detalhando também as técnicas mais comumente aplicadas para o treinamento destes modelos. São apresentadas novas abordagens na área como Redes baseadas em Grafos e Redes Adversárias, assim como também são apresentados os desafios relacionados ao problema, como a variância de escala, variância rotacional, adaptação ao domínio, oclusão, e detecção de pequenos objetos.

Outro trabalho em nível de survey que merece ser destacado é o apresentado em [5], onde também descreve-se o problema, a arquitetura dos principais modelos baseados em redes neurais, assim como as diferentes arquiteturas dos componentes responsáveis pela extração dos atributos, a base convolucional dos modelos. Detalhes referentes ao treinamento destas redes, incluindo aspectos relacionados ao treinamento distribuído, também são abordados no trabalho.

4. Desenvolvimento

O presente trabalho realizou dois experimentos, objetivando compreender as limitações e possibilidades envolvendo a utilização de modelos baseados em redes neurais convolucionais especificamente desenvolvidos para serem utilizados em ambientes computacionais restritos.

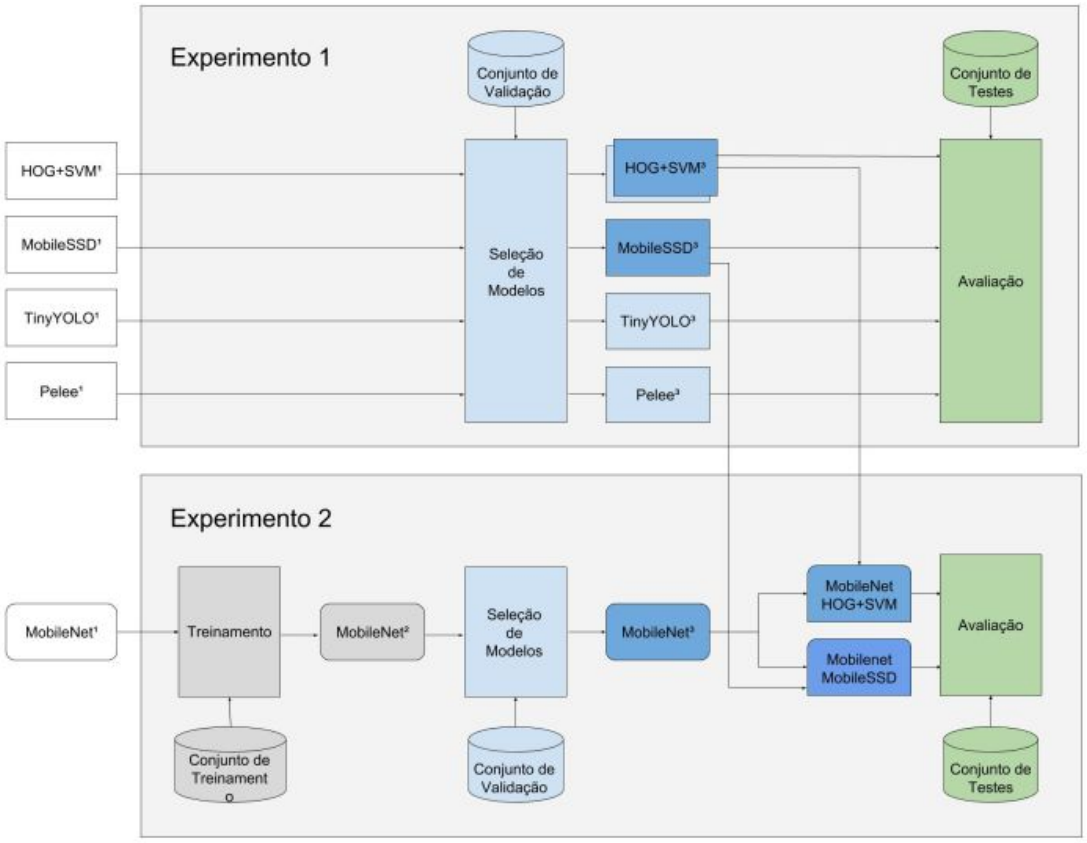

Figura 1. Fluxo geral dos dois experimentos realizados

4.1. Contexto - Segurança Residencial

Considerando-se o contexto do presente trabalho como sendo a detecção de pessoas em imagens provenientes de um sistema de segurança residencial, justificam-se as escolhas realizadas dentre as opções existentes para definir os critérios de avaliação e o conjunto de dados utilizados, assim como decisões envolvendo os modelos que fazem parte dos experimentos.

4.2. Conjunto de dados

Para a realização dos experimentos, foi construído um conjunto de dados formado por 11.000 imagens rotuladas, quanto à presença ou não de pessoas, em que 50% possuem o rótulo positivo e 50% rotuladas como negativo. Separou-se o conjunto de dados em 3 frações: o conjunto de treinamento, composto por 7.000 imagens, o conjunto de validação, formado por 2.000 imagens e o conjunto de teste, também com 2.000 imagens.

Deste total, 2.000 imagens foram extraídas do conjunto de dados INRIA Person Dataset [6], com 1.000 imagens positivas e 1.000 negativas. As 9.000 imagens restantes,

também formada por 50% de exemplos positivos e negativos, foram capturadas especificamente para este trabalho por seu autor, onde a maior parte das imagens foi produzida utilizando-se 6 câmeras de um circuito interno de segurança, uma parcela das imagens utilizando-se um Raspberry Pi 3 Model B+ V1.2 [7], capturando a saída de vídeo de uma câmera Câmera Raspberry Pi NoIR v2 8MP [8], além de uma fração composta por imagens capturadas a partir de uma webcam de um notebook DELL Latitude 3480.

As imagens originalmente foram produzidas com 300 pixels de altura, 300 pixels de largura e 3 canais de cores. Para utilizá-las nos diferentes modelos de classificação/detecção, foram criadas versões com diferentes dimensões, pelo fato dos modelos exigirem tamanhos específicos de imagens. Além da versão original com 300x300, uma versão com tamanho de 416x416, outra com 304x304 e outra com 224x224 foram produzidas. Optou-se por criar diferentes versões ao invés de redimensioná-las durante sua utilização nos modelos, para evitar que o tempo envolvido no redimensionamento influenciasse nos resultados da avaliação do tempo de processamento.

Nas Figuras 1 e 2 são mostrados exemplos de imagens que fazem parte do conjunto de dados utilizado. Na Figura q estão representados exemplos negativos, em que na linha de cima estão exemplos de imagens capturadas pelo autor do trabalho, e na linha de baixo imagens extraídas do INRIA Person Dataset. Na figura w, são mostrado exemplos positivos, que assim como no caso anterior, na linha de cima estão imagens capturadas pelo autor do trabalho, e na linha de baixo são imagens extraídas do INRIA Person Dataset.

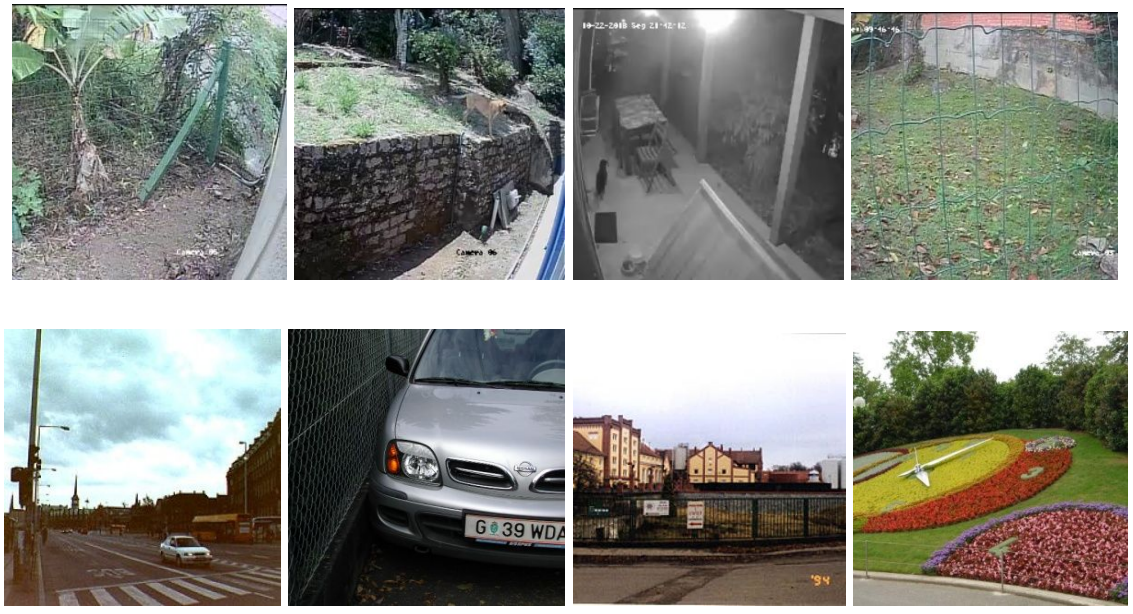

Figura 2. Exemplos negativos do conjunto de dados

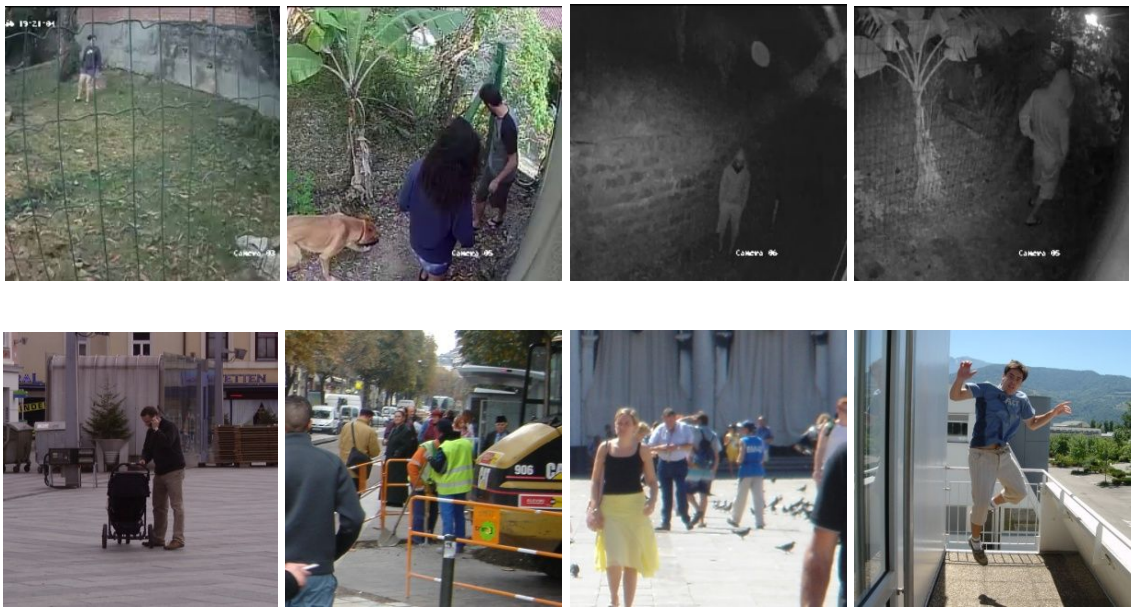

Figura 3. Exemplos negativos do conjunto de dados

4.3. Critérios de avaliação

Como requisitos de ambos os experimentos, espera-se obter uma solução capaz de ser executada em um dispositivo Raspberry Pi 3, atendendo às limitações de memória e processamento do aparelho. As soluções desenvolvidas serão avaliadas de acordo com critérios que demonstrem o tempo de processamento das soluções desenvolvidas, e da qualidade das detecções realizadas.

Para avaliar o tempo de processamento, será medido o tempo de processamento médio de todos os modelos avaliados, ao detectar todos os 2.000 exemplos que fazem parte do conjunto de imagens de teste produzido para os experimentos. A métrica utilizada é a quantidade de frames por segundo (FPS) processados por cada modelo.

Para avaliar a qualidade das detecções realizadas, optou-se por utilizar a acurácia do modelo ao detectar se existem pessoas ou não na imagem. Uma das limitações da acurácia está em cenários em que a quantidade de exemplos das diferentes classes é desbalanceada. Neste trabalho, utiliza-se uma mesma quantidade de exemplos de ambas as classes. A acurácia é frequentemente utilizada para avaliar modelos de classificação, e foi selecionada neste trabalho ao invés de métricas mais comumente utilizadas no contexto de detecção de objetos, como a mean average precision (MAP) [9]. Esta decisão foi em decorrência da dificuldade de produzir anotações com as coordenadas das bounding boxes a serem detectadas pelos modelos de detecção, o que inviabilizaria a execução dos experimentos. Desta forma, desconsidera-se na avaliação a qualidade das bounding boxes produzidas. Espera-se obter uma solução capaz de detectar pessoas nas imagens com acurácia superior ao modelo utilizado como baseline (HOG + SVM), que obtiver a maior taxa de FPS entre as configurações de hiperparâmetros experimentadas.

4.4. Experimento 1 - Modelos de Detecção

No primeiro experimento proposto, cujo o intuito é validar a hipótese de que é possível atingir o objetivo geral do trabalho utilizando somente modelos de detecção de objetos pré-treinados, serão comparados quatro modelos de detecção de objetos. Três deles baseados em redes neurais, e outro baseado extração de atributos HOG combinado com um classificador SVM linear, por ser um modelo simples e bastante popular na área de detecção de objetos [10], cuja implementação é disponibilizada na biblioteca OpenCV. Os modelos baseados em redes neurais foram selecionados dentre um conjunto de modelos caracterizados pela eficiência em termos de consumo de memória e demanda computacional: o TinyYolo, por ser baseado em um modelo pioneiro entre modelos de detecção de objetos focados no tempo de processamento e baixo consumo de memória [9], o MobileSSD, por apresentar os melhores resultados em termos de eficiência computacional de acordo com a literatura [11], e o modelo Pelee [12], [13], que apesar de ser pouco citado em trabalhos relacionados, teve sua implementação disponibilizada pelos seus autores, sendo possível sua utilização neste trabalho. São utilizados modelos pré-treinados, com o conjunto de dados MS-COCO, seguido de um treinamento posterior com o conjunto de dados PascalVOC. Para a avaliação dos modelos, é construído um conjunto de dados composto por 2000 imagens, onde imagens capturadas especificamente para o experimento são utilizadas em conjunto com imagens extraídas do conjunto de dados de imagens de pessoas INRIA. Os modelos são comparados em relação à velocidade em que são capazes de realizar a detecção em um dispositivo Raspberry Pi, em termos de imagens por segundo (FPS - frames per second), e a acurácia relacionada com a tarefa de reconhecimento de pessoas na imagem.

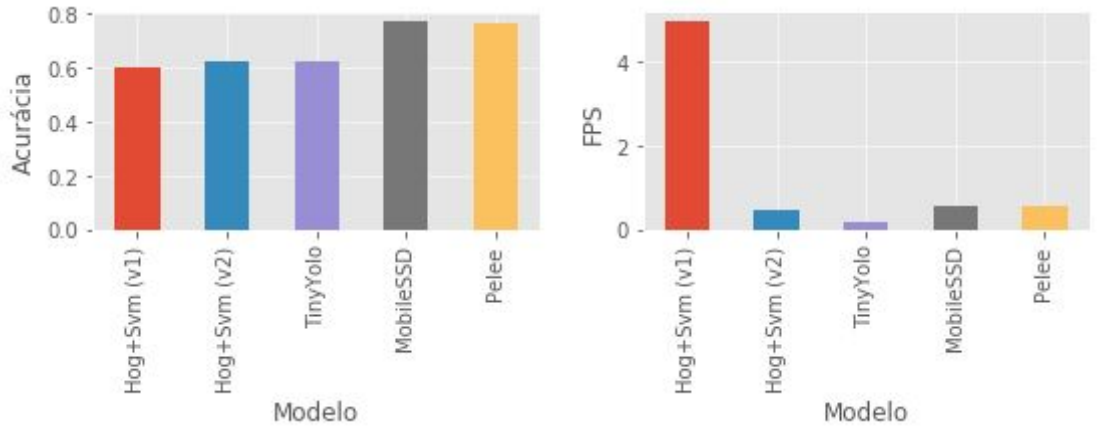

Figura 4. Resultados da avaliação dos modelos de detecção

Modelo	Acurácia	FPS

Hog+Svm (v1)	0.5965	4.97
Hog+Svm (v2)	0.6165	0.45
TinyYolo	0.6185	0.20
MobileSSD	0.7695	0.55
Pelee	0.7615	0.56

Tabela 1. Avaliação com imagens do tamanho original de cada modelo

Modelo	Acurácia	FPS
Hog+Svm (v1)	0.5965	4.97
Hog+Svm (v2)	0.6165	0.45
TinyYolo	0.618	0.20
MobileSSD	0.7695	0.55
Pelee	0.7615	0.52

Tabela 2. Avaliação com imagens de tamanho 300x300

4.5. Experimento 2 - Modelo Híbrido

A partir dos resultados obtidos com o primeiro experimento, nos quais ficam evidentes os valores de acurácia e FPS reduzidos, propõe-se em segundo experimento, com uma solução híbrida, envolvendo dois estágios: um primeiro estágio de classificação, no qual uma rede convolucional, semelhante à base convolucional do modelo MobileSSD do primeiro experimento, é utilizada para identificar imagens com a presença de pessoas, e um segundo estágio, onde o modelo de detecção melhor avaliado em termos de acurácia é executado sobre as imagens classificadas positivamente no primeiro estágio. O resultado é avaliado de acordo com a acurácia e a velocidade de processamento em termos de FPS. São avaliados dois valores distintos de acurácia: a acurácia do modelo de classificação, e acurácia do modelo de detecção. Este último é avaliado levando-se em conta apenas as imagens classificadas positivamente no primeiro estágio do modelo.

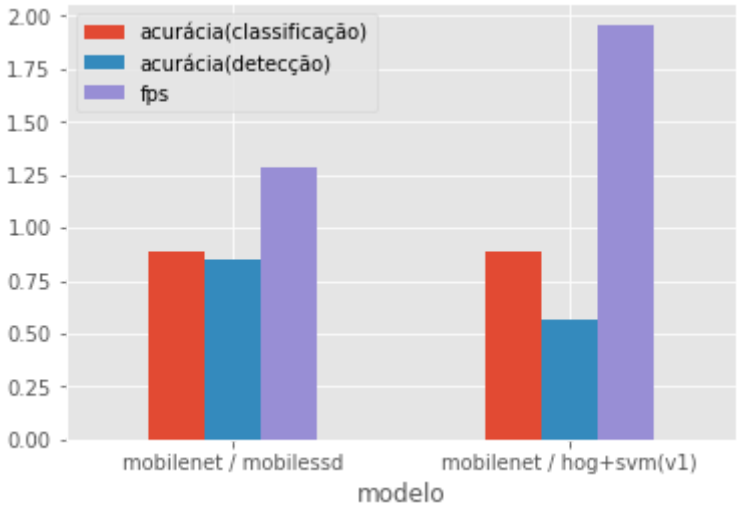

Tabela 3. Resultados da avaliação com os modelos híbridos

Modelo	Acurácia (classificação)	Acurácia (detecção)	FPS
Classificador Mobilenet, Detector Hog+Svm v1	88.65%	57%	1.96
Classificador Mobilenet, Detector MobileSSD	88.65%	85%	1.29

Tabela 3. Resultados da avaliação com os modelos híbridos

	classe predita		
classe real		positivo	negativo
	positivo	**814**	**185**
	negativo	**42**	**958**

Tabela 4. Matriz de confusão da etapa de classificação

Na Tabela 4 estão os resultados da etapa de classificação. Os resultados da matriz de confusão de cada etapa de detecção não foram avaliados. Diante os resultados obtidos,

ficou evidente a superioridade do modelo proposto envolvendo a utilização do classificador MobileNet, em conjunto ao detector MobileSSD, em relação ao modelo de detecção pré-treinado que apresentou melhores resultados em termo de acurácia no primeiro experimento. Atingiu-se um incremento de 15.20% na acurácia, e de 134% na quantidade de frames por segundo (FPS). Em relação ao modelo selecionado como baseline, o modelo HOG+SVM de melhor taxa de FPS no primeiro experimento, atingiu-se um desempenho em termos de acurácia 48.61% superior, e em termos de FPS, 60.56% inferior.

5. Conclusão e Trabalhos Futuros

Durante os experimentos, ficaram evidentes as limitações dos modelos avaliados quanto ao desempenho em termos de velocidade de processamento, onde os melhores modelos, segundo este critério, não foram capazes de realizar o problema proposto com uma taxa de frames por segundo que viabilize a sua utilização em cenários críticos, que exijam tempo de resposta imediato. Quanto à qualidade das detecções realizadas, também ficaram evidentes as limitações quanto à qualidade das bounding boxes identificadas, ao se utilizar os modelos de detecção com níveis de confiança que maximizem a acurácia dos modelos. A acurácia final da solução apresentou níveis satisfatórios, diante das restrições presentes no conjunto de dados utilizado, quanto ao número de exemplos e diversidade das imagens utilizadas. Apesar da utilização de fine-tuning ter permitido atingir-se uma acurácia de 88.65%, acredita-se que o desempenho possa ser drasticamente melhorado utilizando-se um conjunto de dados maior.

Sugere-se como possibilidades de trabalhos futuros explorar a utilização de modelos de classificação e detecção mais exigentes em termos de hardware, em experimentos semelhantes. Este trabalho não realizou de maneira intensiva a possibilidade de utilizar modelos computacionalmente mais exigentes diretamente em dispositivos Raspberry Pi. Da mesma forma, é possível explorar a possibilidade de utilizar modelos que sejam o estado-da-arte na tarefa de detecção em arquiteturas Fog [14], delegando-se o processamento pesado para nós de rede presentes em uma rede local, mantendo-se assim a possibilidade de manter uma baixa latência no tempo de resposta. Outro ponto a ser explorado em trabalhos futuros é a utilização de um conjunto de dados com um número maior de exemplos, o que provavelmente aumentaria substancialmente a acurácia final na primeira etapa no modelo proposto no segundo experimento. Por fim, outra possibilidade seria a construção de um conjunto de dados com anotações próprias para o treinamento de modelos de detecção, com as coordenadas das bounding boxes referentes às pessoas das imagens, o que possibilitaria o fine-tuning dos modelos experimentados, o que provavelmente acarretaria em resultados superiores, tanto no primeiro experimento realizado, quanto na segunda etapa do segundo experimento.

Referências

[1] Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[2] Agarwal, Shivang, Jean Ogier Du Terrail, and Frédéric Jurie. "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks." arXiv preprint arXiv:1809.03193 (2018).

[3] Forsyth, David A., and Jean Ponce. "A modern approach." Computer vision: a modern approach (2003): 88-101.

[4] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. "Deep learning." 2015(2016).

[5] Chahal, Karanbir Singh, and Kuntal Dey. "A Survey of Modern Object Detection Literature using Deep Learning." arXiv preprint arXiv:1808.07256 (2018).

[6] Dalal, Navneet, and Bill Triggs. "INRIA person dataset." Online: http://pascal. inrialpes. fr/data/human (2005).

[7] Pi, Raspberry. "model B." Raspberrypi.org. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em 18 set. 2018

[8] Pi, Raspberry. "Câmera Raspberry Pi NoIR v2 8MP" Raspberrypi.org. Disponível em: <https://www.raspberrypi.org/products/pi-noir-camera-v2/>. Acesso em 18 set. 2018

[9] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[10] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[11] MobileNet-SSD, 2018. Repositório Github. Disponível em: <https://github.com/chuanqi305/MobileNet-SSD>. Acesso em 18 set. 2018.

[12] Wang, Robert J., et al. "Pelee: A Real-Time Object Detection System on Mobile Devices." arXiv preprint arXiv:1804.06882 (2018).

[13] Wang, Robert J., 2018. Repositório Github. Disponível em: <https://github.com/Robert-JunWang/Pelee>. Acesso em 18 set. 2018.

[14] Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." Advances in neural information processing systems. 2013.