Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Curso de Graduação em Engenharia Elétrica

Trabalho de Conclusão de Curso

Reconhecimento de Placas de Automóveis e de seus Condutores para a Abertura Automática de Portões

Vanessa Santos Moura

Campina Grande, PB
Outubro de 2021

Vanessa Santos Moura

Reconhecimento de Placas de Automóveis e de seus Condutores para a Abertura Automática de Portões

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Área de concentração: Redes Neurais Convolucionais

Orientadora: Prof. Luciana Ribeiro Veloso, Dr. Sc.

Campina Grande, PB

Outubro de 2021

Vanessa Santos Moura

Reconhecimento de Placas de Automóveis e de seus Condutores para a Abertura Automática de Portões

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em: ____ / ____ / ____

Orientadora: Prof. Luciana Ribeiro Veloso, Dr. Sc.

Convidado: Prof. Edmar Candeia Gurjão, Dr. Sc.

Campina Grande, PB
Outubro de 2021

Dedico este trabalho aos meus pais, José Evilásio Moura e Osanilde Nogueira Santos, e à meus irmãos, Larissa Mylena Santos de Moura e Érick Jung Santos de Moura.

Agradecimentos

Em primeiro lugar agradeço a Deus por me dar força e coragem para lutar pelos meus sonhos e objetivos. Em seguida, agradeço a meus pais, Osanilde e Evilásio, que não mediram esforços para que eu fizesse graduação à quilômetros de casa e mais cedo do que esperávamos. Por acreditarem e lutarem comigo para que eu estude e me forme. Obrigado por me ensinarem que a educação é capaz de transformar as pessoas e o mundo em que elas vivem.

Agradeço ao meu namorado, Gaëtan Michel, por todo o apoio e companheirismo. Por ser um exemplo de disciplina, motivação e positividade. Eu aprendo todos os dias com suas virtudes.

Aos meus professores por todo o aprendizado e oportunidades que me foram proporcionadas. Eu agradeço especialmente a professora Luciana, que me orientou nesse trabalho e me ajudou a concluir essa etapa.

Agradeço à Fundação CAPES, que graças ao programa CAPES/BRAFITEC, financiou meu intercâmbio na França. A minha experiência durante o intercâmbio foi indescritível para a minha formação humana e profissional.

Por fim, gostaria de agradecer a meus amigos maravilhosos, Alison, Andhré, Arllan, Brenda, Weverton e Wislayne, pelas noites de estudos e conversas sem fim. Obrigada por fazer parte dessa jornada, isso não seria possível sem o apoio e o companheirismo de vocês.

"Seja a mudança que você quer ver no mundo."

(Mahatma Gandhi)

Resumo

O presente trabalho tem como propósito contribuir para assistência na entrada e saída de automóveis em prédios e condomínios fechados. Visando atingir este objetivo, este trabalho foi dividido em duas etapas: desenvolvimento de um algoritmo de rede neural capaz de identificar e reconhecer a placa de um automóvel e desenvolvimento de um algoritmo capaz de identificar e reconhecer o condutor. Para a identificação de placas de automóveis, foi utilizado o modelo de rede pré-treinado do *Tensorflow*, chamado *MobileNet-SSD*, destinado para a detecção de objetos, sendo realizada o afinamento do modelo para que se detecte placas. Para o reconhecimento da placa do automóvel, empregou-se a tecnologia de reconhecimento óptico de caracteres OCR. A implementação do algoritmo foi realizada em Python. A rede neural convolucional profunda apresentou bons resultados de precisão, revocação e *F1-score*, métricas de desempenho de redes neurais, podendo ser melhorada em trabalhos futuros realizando um pré-processamento na base de dados escolhida. Para a identificação e reconhecimento facial do condutor, é empregado histograma de gradientes orientados alimentando uma máquina de vetor de suporte. As bibliotecas *dlib* e *face_recognition* foram utilizadas. Foram obtidos resultados satisfatórios de acurácia, precisão, revocação e F1-score, superiores à 98,6 %, o que demonstra a fidelidade de predição do algoritmo de reconhecimento facial implementado.

Palavras-chave: Assistência, Rede Neural, *MobileNet-SSD*, Histograma de Gradientes Orientados, Máquina de Vetor de Suporte.

Abstract

This work aims to contribute with the assistance in the entry and exit of cars in buildings and private condominiums. Aiming to achieve this goal, this work was divided into two stages: development of a neural network algorithm capable of identify and recognize the license plate of a car and development of an algorithm capable of identify and recognize the driver. For the license plates identification, the pre-trained Tensorflow network model was used, called MobileNet-SSD, intended for object detection. For the recognition of the car's license plate, OCR optical character recognition technology was used. The algorithm implementation was performed in Python. The deep convolutional neural network showed good results of precision, recall and F1-score, neural network performance metrics, which could be improved in future works by performing a pre-processing in the chosen database. For the identification and facial recognition of the conductor, an oriented gradient histogram feeding a support vector machine is used. The dlib and face_recognition libraries were used. Satisfactory results of accuracy, precision, recall and F1-score were obtained, above 98.7%, which demonstrates the predictive fidelity of the implemented facial recognition algorithm.

Keywords: Assistance, Neural Network, MobileNet-SSD, Histogram of Oriented Gradients, Support Vector Machine.

Lista de abreviaturas e siglas

CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
BRAFITEC	*BRasil France Ingénieur TEChnologie* (em português, Brasil França Engenheiro e Tecnologia)
TCC	Trabalho de Conclusão de Curso
OCR	*Optical Caracter Recognition* (em português, Reconhecimento óptico de caracteres)
ReLu	*Rectified Linear Unit* (em português, Unidade Linear Retificada)
CNN	*Convolutional Neural Networks* (em português, Redes Neurais Convolucionais)
RGB	*Red Green Blue* (em português, Vermelho Verde Azul)
GPU	*Graphics Processing Unit* (em português, Unidade de Processamento Gráfico)
SSD	*Single Shot Multibox Detection* (em português, Detector Multibox de Disparo Único)
VGG	*Visual Geometry Group* (em portguês, Grupo de Geometria Visual)
YOLO	*You Only Look Once* (em português, Você Só Olha Uma Vez)
HOG	*Histogram of Oriented Gradients* (em português, Histograma de Gradientes Orientados)
SVM	*Suport Vector Machine* (em português, Máquina de Vetores de Suporte)
TP	*True Positive* (em português, Verdadeiros Positivos)
FP	*False Positive* (em português, Falsos Positivos)
FN	*False Negative* (em português, Falsos Negativos)

TN	*True Negative* (em português, Verdadeiros Negativos)
AP	*Average Precision* (em português, Precisão)
AR	*Average Recall* (em português, Revocação)
F1-score	*F1-score* (em português, Medida F)

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

No Brasil, a maioria dos prédios e condomínios fechados tem a entrada controlada por um porteiro. O porteiro, ou agente de portaria, possui muitas atribuições e responsabilidades. Ele é responsável pela segurança do edifício, a sua vigilância, o controle de acesso dos visitantes, dos automóveis, entre outras tarefas de assistência como a gestão de encomendas, assistência telefônica, etc. Uma forma de auxiliar o porteiro em sua profissão pode ser fornecer-lhe ferramentas de informática que lhe permitam trabalhar em melhores condições. Uma das suas tarefas recorrentes é a gestão da entrada e saída de veículos da residência, verificando os direitos de acesso de cada inquilino de acordo com a sua placa e o condutor. Para tornar seu trabalho mais leve e prático, uma solução possível seria tornar essa tarefa assistida, ou mesmo totalmente automatizada por um sistema computacional inteligente.

Esse sistema inteligente deve ser capaz de reconhecer a placa do veículo e o condutor para, portanto, abrir o portão se o acesso for autorizado. Esse sistema oferece mais comodidade ao inquilino e um controle de acesso do local com um sistema a custo acessível e seguro, para as empresas e estabelecimentos que desejam implantá-la.

Assim, a solução proposta visa identificar os veículos autorizados a entrar em locais de restrito acesso, a partir de imagens captadas por um sistema de câmeras de vigilância. As imagens são processadas por um sistema de redes neurais para identificar de forma inteligente o veículo e o condutor. Sendo os automóveis um dos principais meios de locomoção atualmente, a solução proposta visa fornecer monitoramento automático, rápido e eficiente dos automóveis que entram e saem de um local. Caso sejam detectados veículos em situação irregular, pode ser solicitada a assistência necessária à portaria.

Além disso, o sistema de reconhecimento de placas de automóveis e de seus condutores pode ser aplicado em outras situações, como em pedágios, para o controle da passagem; no monitoramento do tráfego, a fim de conhecer a velocidade média dos veículos em um determinado percurso; em balanças eletrônicas, para a pesagem de veículos; para o controle de fronteiras; para a fiscalização de irregularidades no trânsito, entre outras aplicações.

Portanto, este Trabalho de Conclusão de Curso (TCC) visa desenvolver um algoritmo de reconhecimento de placas de automóveis e de seus condutores para identificar os automóveis que são autorizados a entrar ou sair do imóvel. Técnicas de processamento de imagens e de inteligência artificial são amplamente aplicadas.

1.1 Objetivos

1.1.1 Objetivo Geral

- Criar um sistema inteligente capaz de reconhecer placas de automóveis e seus condutores para a abertura de automática de portões de um local de restrito acesso.

1.1.2 Objetivos Específicos

- Desenvolver um algoritmo para identificar/reconhecer a placa do automóvel;
- Desenvolver um algoritmo para identificar/reconhecer o condutor;

1.2 Metodologia

O algoritmo de reconhecimento de placas de automóveis e seus condutores será desenvolvido em Python. Bibliotecas como OpenCV, *Tensorflow* e Scikit Learn são empregadas. As atividades a serem implementadas foram:

1. aquisição e pré-processamento da base de dados de imagens.
2. localização, seleção e extração da placa do veículo;
3. segmentação dos caracteres da placa;
4. reconhecimento dos caracteres da placa;
5. reconhecimento facial do condutor;

1.3 Estrutura do Trabalho

O presente trabalho está organizado em 4 capítulos. No Capítulo 1 é feita uma introdução do sujeito do trabalho, é apresentado os objetivos gerais e específicos assim como a metodologia adotada para solução de problemas.

No Capítulo 2 são descritos os conceitos básicos necessários para a compreensão de Redes Neurais Artificiais e Redes Neurais Convolucionais, assim como é apresentada a arquitetura de redes neurais *MobileNet* e a técnica de reconhecimento de caracteres OCR, empregadas para a identificação e reconhecimento de placas de automóveis.

O Capítulo 3 trata com detalhes a metodologia adotada e as atividades desenvolvidas objetivando identificar e reconhecer placas de automóveis e seus condutores.

No Capitulo 4 são apresentados os resultados obtidos do trabalho e discussões acerca dos resultados.

No Capitulo 5, as conclusões do trabalho em conjunto com as considerações finais.

2 Redes Neurais Convolucionais profundas

As redes neurais profundas são uma evolução das redes neurais artificiais que permitiram avanços na resolução de problemas de visão computacional. A visão computacional com aprendizado profundo possui aplicações em áreas tais como: medicina, buscando auxiliar os médicos a dar um diagnóstico mais rápido e preciso aos seus pacientes a partir de imagens de radiografia; segurança pública, realizando reconhecimento de gestos e de ações humanas em tempo real para prevenir assaltos em banco ou em outras instituições públicas; setor automobilístico, buscando que carros autônomos sejam uma realidade segura e confiável, baseada em sistemas de reconhecimento de carros, pedestres, placas, animais, entre outros objetos localizados ao redor do automóvel; entre outras aplicações.

O sujeito deste trabalho se baseia em uma aplicação de assistência à entrada e saída de automóveis em locais de restrito acesso de forma automatizada, auxiliando o porteiro à gerir de forma rápida e eficiente a entrada e saída de automóveis por meio de um sistema de reconhecimento de automóveis pela sua placa, assim como de seus condutores por meio de um sistema de reconhecimento facial. Para o desenvolvimento desses sistemas, foi utilizado redes neurais convolucionais profundas. Antes de entrar em detalhes do modelo utilizado, serão apresentados os conceitos básicos de redes neurais artificiais e de redes neurais convolucionais, o modelo de rede neural convolucional profunda utilizado para a detecção de placas e a tecnologia OCR para o reconhecimento dos caracteres da placa.

Esse capitulo é dividido em 4 subtópicos. A seção 2.1 introduz os conceitos de redes neurais, como o neurônio, funções de ativação, camadas e o processo de aprendizagem; a seção 2.2 explana como surgiu as redes neurais convolucionais assim como os conceitos básicos para a compreensão da estrutura de uma rede convolucional; a seção 2.3 apresenta o modelo de rede neural *MobileNet*, empregado na detecção de placas; a seção 2.4 apresenta os conceitos básicos da ferramenta de reconhecimento óptico de caracteres OCR utilizada para segmentar os caracteres da placa de um veículo.

2.1 Redes Neurais Artificiais

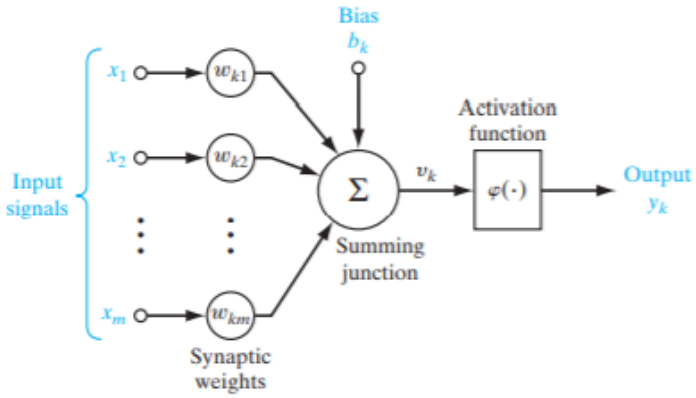
Uma rede Neural Artificial, segundo (HAYKIN; HAYKIN, 2009), pode ser definida como um processador maciçamente e paralelamente distribuído, constituído de unidades de processamento simples, que têm a propensão natural de armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos: o conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem; Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Redes neurais artificiais são comumente apresentados como neurônios interconectados capazes de aprender por exemplos, imitando o comportamento dos neurônios biológicos. Isso é realizado por meio de um processo de aprendizagem interativo que melhora com a experiência, como ocorre com o cérebro humano. A definição de um neurônio em uma rede neural artificial é apresentada na seção que segue.

2.1.1 Neurônio

Um neurônio é a unidade básica de processamento de informações para o funcionamento de um rede neural. A Figura 1 apresenta o modelo de um neurônio.

Figura 1 – Modelo de um neurônio artificial



Fonte: (HAYKIN; HAYKIN, 2009)

O modelo neural pode ser dividido em três elementos básicos, segundo (HAYKIN; HAYKIN, 2009) :

1. Um conjunto de sinapses, ou ligações, cada uma caracterizada por uma entrada x e um peso w;

2. Um somador para somar os sinais de entrada, ponderados pelos respectivos pesos, e somar o bias (b) que representa o quanto o neurônio é fundamental e influencia no resultado da rede neural;

3. Uma função de ativação para limitar a amplitude da saída de um neurônio.

A saída do neurônio é obtida realizando inicialmente a somatória dos sinais de entrada, ponderados pelos respectivos pesos, e do valor do bias a cada resultado. Esse processo é uma transformação linear. Em seguida, na saída do somador é aplicado a função de ativação, que é uma transformação não linear. A função de ativação tem como função limitar a faixa de amplitude permitida do sinal de saída para algum valor finito. Se este nível de atividade exceder um certo limite, a unidade produz uma determinada resposta de saída.

A maioria dos modelos de redes neurais aprendem através de exemplos. Geralmente, é definido alguma regra de treinamento, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados.

2.1.2 Função de ativação

A função de ativação é uma transformação não linear realizada ao longo do sinal de entrada. Quando não se tem a função de ativação, a saída do neurônio é resultado de uma transformação linear dos pesos e bias na entrada. Uma transformação linear é simples de resolver, mas é limitada na sua capacidade de resolução de problemas complexos. Assim, em redes neurais artificiais, as funções de ativação desempenham o papel de aprender a resolver problemas complexos.

As funções de ativação decidem se um neurônio deve ser ativado ou não. Se a informação que o neurônio recebe é relevante para a informação fornecida, o neurônio é ativado por essa função; se não, o neurônio não é ativado. Ou seja, as funções de ativação são essenciais em redes neurais para tornar possível a propagação posterior dos neurônios.

Para isso, é necessário que os gradientes e o erro sejam fornecidos para atualizar os pesos e bias. As funções de ativação permitem que pequenas alterações nos valores dos

pesos e bias causem apenas uma pequena alteração na saída. A atualização dos valores de pesos e bias é importante para que uma rede neural aprenda.

Existem diversos tipos de funções de ativação comumente utilizadas nas diferentes arquiteturas de redes neurais, entre elas a função de ativação ReLu. Essa função zera os valores dos neurônios menores do que zero, e possui derivada unitária pois os valores maiores do que zero são caracterizados por uma função linear. Outras funções de ativação tipicamente utilizadas são as funções de ativação sigmoide, Tanh e Softmax.

A ReLu, comparada com as funções sigmoide, Tanh e Softmax, mencionadas acima, permite um treinamento em uma base de dados grande e complexa de forma rápida e eficiente. Na verdade, a função ReLu realiza apenas operações de comparação, adição e multiplicação, diferente das demais funções de ativação que fazem uso de expoentes, o que explica a rapidez característica desta função de ativação.

2.1.3 Camadas

As redes neurais artificiais são organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior. As camadas são classificadas em:

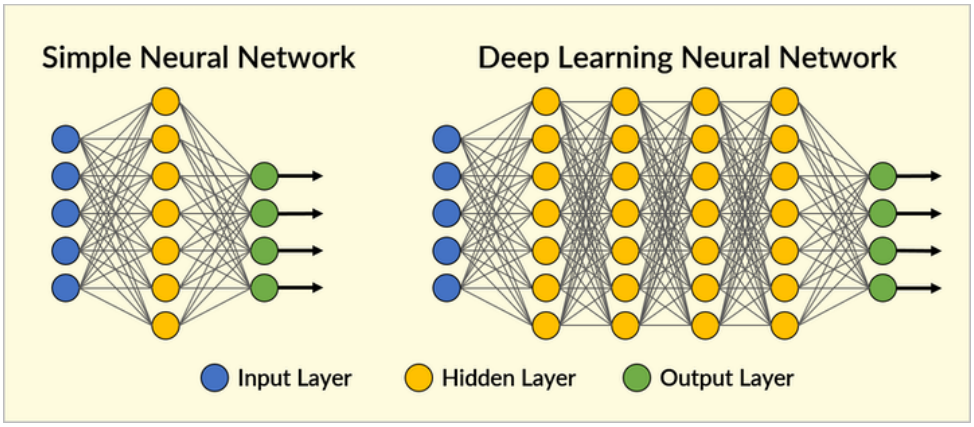
- Camada de Entrada: são os elementos iniciais da rede e onde os padrões são apresentados;
- Camadas Intermediárias (ou ocultas): responsável pelo processamento dos dados. Intervêm entre a camada de entrada e saída fazendo as extração de características.
- Camada de Saída: valor final.

A Figura 2 mostra as camadas de redes neurais artificiais simples (ou rasas) e redes neurais artificiais profundas. Os círculos azuis representam os neurônios da camada de entrada, os círculos laranjas representam os neurônios das camadas intermediárias e os círculos verdes os neurônios da camada de saída. As linhas são conexões entre neurônios que estão interligados de uma camada à outra e as setas indicam as saídas da rede neural. Cada camada é composta de um determinado número de neurônios.

Uma rede neural profunda possuem números maiores de camadas ocultas do que uma rede neural simples, o que melhora o processo de aprendizagem. Em tarefas complexas,

a rede neural profunda possibilita o treino para que as máquinas executem tarefas com maior precisão. (DENG; YU, 2014)conceitua aprendizado profundo como um subconjunto de *machine learning*, no qual, as redes neurais artificiais se adaptam e aprendem a partir de vastas quantidades de dados, utilizando múltiplas camadas para extrair progressivamente características a partir de uma entrada bruta, a fim de aprender a prever e classificar informações.

Figura 2 – Rede neural simples e rede neural profunda *Deep Learning*



Fonte: (AI VS ML VS DL-Let's Understand The Difference, 2021)

2.1.4 Processo de aprendizagem

O processo de aprendizagem em uma rede de aprendizado profundo (*Deep Learning*) só se tornou possível devido o crescimento da quantidade de dados disponíveis nos dias atuais. Para treinar redes *Deep Learning*, é necessário uma alta capacidade de processamento do computador devido à grande quantidade de dados para processamento, além do aumento do número de parâmetros a serem ajustados pelo algoritmo de aprendizado.

O algoritmo de aprendizagem é o responsável pela adaptação dos parâmetros da rede, de maneira que, em um número finito de interações do algoritmo de treinamento, haja convergência para uma solução.

2.2 Redes Neurais Convolucionais

As redes neurais convolucionais (CNN - *Convolutional Neural Networks*) são uma subcategoria de redes neurais. No entanto, as CNNs são especialmente projetadas para processar imagens ("Classez et segmentez des données visuelles", [s.d.]).

Um dos primeiros trabalhos em redes neurais convolucionais foi para o reconhecimento de números manuscritos, em 1998, pelo pesquisador francês Yann LeCun e seus colaboradores (LECUN, et al., 1998). Neste trabalho, foi utilizado *backpropagation* em uma rede com muitas camadas ocultas capaz de reconhecer vários números ao mesmo tempo, mesmo se sobrepostos. As multiplicações de matrizes que eram realizadas em cada camada foram substituídas por convoluções entre imagens. O reconhecedor de números manuscritos foi nomeado LeNet, porém mais tarde a arquitetura da rede foi formalizada sob o nome de redes neurais convolucionais.

O marco da virada para tornar as redes neurais convolucionais popular no mundo foi uma competição que ocorreu em 2012 com objetivo de classificar automaticamente 1000 categoriais de objetos em 1,2 milhões de imagens de entrada, existindo 10 milhões de imagens rotuladas. Dado uma imagem de entrada, o algoritmo tinha como objetivo categorizar quais objetos tinham na imagem, localizar os objetos e atribuir rótulos aos objetos. O algoritmo vencedor teve apenas 16,7 % de erros e o segundo lugar teve 26 %. Com o passar dos anos, o nível de acurácia dos resultados se tornou tão elevado que passou a superar a capacidade humana de identificar ou classificar objetos em imagens.

Figura 3 – ImageNet, a base de dados da competição *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC)

Fonte: (Synced, 2021)

Os próximos subcapítulos apresentam conceitos introdutórios de camadas típicas de redes neurais convolucionais, que são: a camada convolucional, a camada de *pooling*, a camada da função de ativação e a camada totalmente conectada.

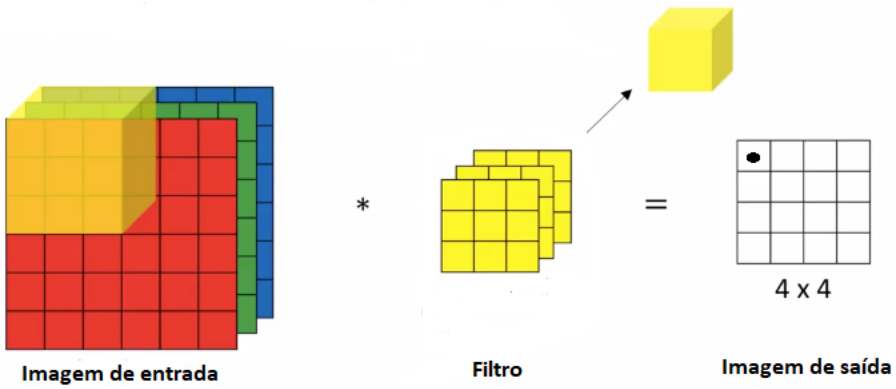
2.2.1 Camada de convolução

A camada de convolução é uma camada fundamental para a construção de uma rede neural convolucional. A função dessa camada é convoluir as imagens de entrada com um conjunto de filtros cujos valores dos pesos são aprendidos durante o processo de treinamento. Os filtros são elementos que extraem certas características das imagens.

As características desejadas não são predefinidas, elas são aprendidas pela rede durante a fase de treinamento. Ou seja, independente do problema que se deseja resolver, a CNN adapta-se ao problema colocado aprendendo a resolver o problema.

Na Figura 4, pode-se visualizar como ocorre a convolução em uma imagem RGB. A imagem de entrada é uma imagem de dimensão 6 x 6 x 3, o filtro tem dimensão 3 x 3 x 3 e a imagem de saída uma imagem de dimensão 4 x 4, sendo a dimensionalidade de uma imagem dada por altura x largura x número de canais. A saída da operação convolucional é obtida colocando inicialmente o filtro na posição mais a esquerda e superior da imagem de entrada, onde é realizada a multiplicação pixel por pixel da imagem de entrada com o filtro e é realizado o somatório desses valores. O resultado desta operação corresponde ao valor do pixel destacado na imagem de saída (pixel mais superior e à esquerda). Para calcular a próxima saída, desliza-se o cubo na imagem de entrada por um e realiza a operação de multiplicação e soma pixel por pixel novamente. Esse processo é realizado continuamente até o filtro ter sido deslocado por toda a imagem de entrada.

Figura 4 – Processo de convolução em imagem RGB



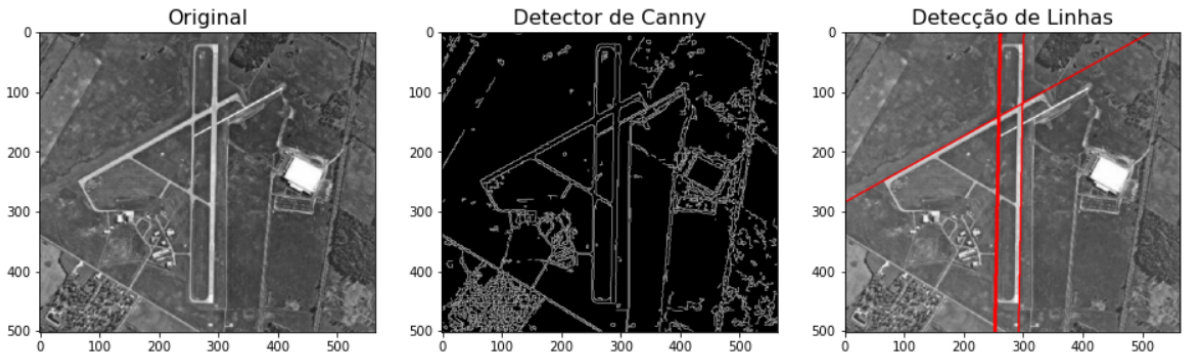
Fonte: (*Convolutional Neural Networks*, [s.d.])

Na camada convolucional, obtém-se para cada par (imagem, filtro) uma imagem de saída, chamada de mapa de características. A camada convolucional, além de ex-

trair as características da imagem, reduz a quantidade de pixeis da imagem de saída, consequentemente diminui a quantidade de dados na rede.

Exemplos de características que se pode desejar obter em uma imagem são contornos ou bordas, regiões, rugosidades, cores e texturas. Para detectar uma ou mais destas características, é necessário projetar filtros específicos para cada característica. Na Figura 5, o filtro de Canny foi aplicado para detectar os contornos da imagem. Este filtro foi projetado para ser sensível aos contornos de uma imagem. Em seguida, foi aplicado um segundo filtro (o filtro de Hough) para detectar linhas específicas da imagem, que correspondem às pistas de um aeroporto. O filtro de Hough foi projetado para realizar o reconhecimento de padrões de uma imagem, como linhas, círculos, elipses, entre outros.

Figura 5 – Detecção de bordas e de linhas referentes às pistas do aeroporto



Fonte: (Autora, 2021)

2.2.2 Camada de *Pooling*

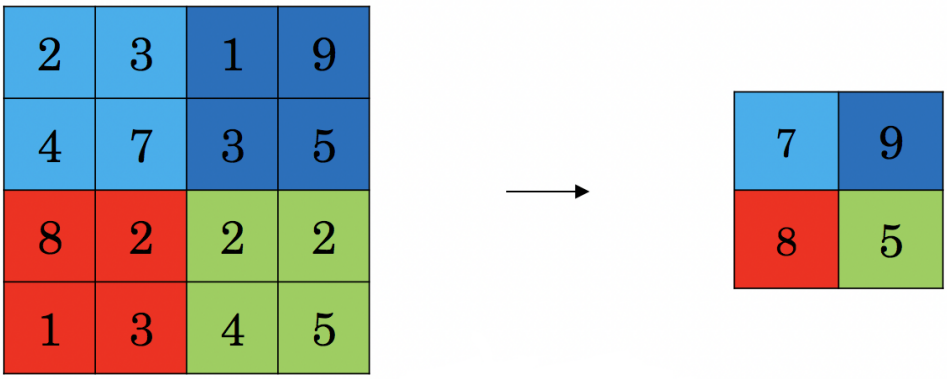
A camada de *pooling* tem como função reduzir o tamanho da imagem para acelerar o processamento da rede mas preservando as suas características importantes. Uma imagem reduzida possui menos parâmetros, logo a quantidade de cálculos na rede diminui, o que melhora a sua eficiência. A saída dessa camada é uma imagem reduzida. Essa camada é tipicamente colocada entre duas camadas de convolução.

O funcionamento do *pooling* é semelhante ao funcionamento da convolução: a imagem de entrada é varrida por um conjunto de neurônios. A principal diferença é que os neurônios de saída são resultado de uma função de *pooling* aplicada sobre a imagem, que pode ser a função máxima, mínimo, média, entre outras funções.

Tomando como exemplo uma função *pooling* máxima, divide-se a imagem de entrada

em regiões, sendo cada uma das saídas o valor máximo da região correspondente da entrada. A Figura 6 exemplifica o funcionamento da camada de *pooling* utilizando a função máxima. A imagem de entrada foi dividida em 4 regiões, representada por 4 cores diferentes e o valor da imagem de saída corresponde ao valor máximo das regiões da imagem de entrada. Os hiperparâmetros da função máximo deste exemplo são um filtro de ordem 2 e *stride* ou passo de 2. Considerando a imagem de entrada contendo um determinado conjunto de características, então um valor grande na saída significa que talvez tenha detectado uma característica particular. Logo, o quadrante superior direito tem essa característica particular. Talvez ele indica a presenta de uma borda vertical ou horizontal da imagem, por exemplo.

Figura 6 – Funcionamento da camada de *pooling* utilizando a função máximo e filtro 2x2 de passo 2

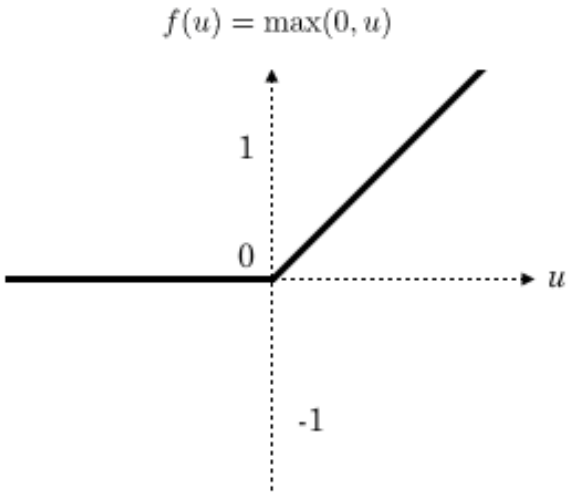


Fonte: (*Convolutional Neural Networks*, [s.d.])

2.2.3 Camada da função de ativação

A função de ativação usada na maioria dos casos atualmente é a função de ativação ReLU (ReLU - *Rectified Linear Unit*), ("Classez et segmentez des données visuelles", [s.d.]). A Figura 7 abaixo mostra o comportamento da função ReLU no tempo. A camada de correção ReLU substitui todos os valores negativos recebidos como entradas por zeros.

Figura 7 – Função ReLU



Fonte: ("Classez et segmentez des données visuelles", [s.d.])

2.2.4 Camada totalmente conectada

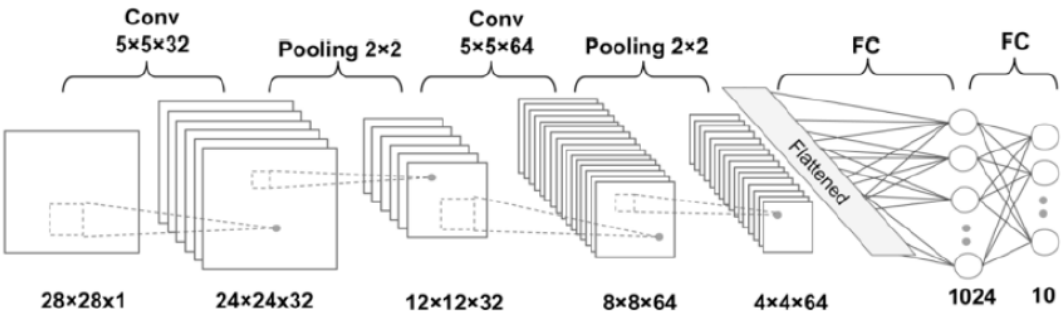
A camada totalmente conectada corresponde à ultima camada de uma rede neural. Ela recebe um vetor de entrada e produz um vetor de saída que é resultado da combinação linear dos valores recebidos na entrada seguidos eventualmente de uma função de ativação desses valores.

O vetor retornado por essa camada tem tamanho N, onde N corresponde ao número de classes de um problema, sendo cada elemento do vetor a probabilidade para a imagem de entrada pertencer a uma classe. Por exemplo, se o problema é um problema de classificação e se deseje diferenciar numa imagem um gato de um cachorro, o vetor seria de tamanho 2, sendo o primeiro elemento a probabilidade da imagem pertencer à classe de gatos e o segundo elemento a probabilidade da imagem pertencer a classe cachorros, ou vice-versa.

As redes neurais convolucionais devem ter a camada de convolução como a primeira, e totalmente conectada como última camada. As camadas intermediárias podem ser organizadas de diferentes formas, a única condição é que a saída de uma camada tenha a mesma dimensão da entrada da próxima camada.

A Figura 8 ilustra uma arquitetura típica de CNN que se utiliza das camadas vistas neste trabalho. Subentende-se que as ativações seguem as camadas convolucionais.

Figura 8 – CNN com camadas de convolução, de *pooling* e totalmente conectadas



Fonte: (RASCHKA; MIRJALILI, 2017)

Esta rede é considerada uma rede neural artificial convolucional profunda. Rede neural artificial pois é composta de neurônios organizados em camadas conectadas, profunda pois é composta de muitas camadas intermediárias entre a camada de entrada e a de saída, e convolucional pois sua estrutura é composta por camadas de convolução.

2.3 *MobileNet*

Proposta por (HOWARD et al., 2017), *MobileNet* é uma arquitetura foi concebida devido a necessidade de modelos mais rápidos e leves, com aplicabilidade em sistemas em tempo real para uso em dispositivos móveis e de borda.

Desde a competição ILSVRC em 2012, a tendência geral foi criar redes mais profundas e complexas a fim de obter maior precisão no desempenho das redes. Em aplicações do mundo real, como robótica, carro autônomo e realidade aumentada, as tarefas de reconhecimento precisam ser realizadas em tempo hábil em dispositivos computacionalmente limitados (HOWARD et al., 2017).

Desta forma, os autores (HOWARD et al., 2017) propuseram uma arquitetura que usa convoluções separáveis em profundidade para construir redes neurais convolucionais profundas.Na Figura 9 encontram-se todas as camadas definidas na rede neural. A maior parte das camadas são formadas de camadas de convolução profunda e pontual.

Em resumo, a arquitetura *MobileNet* funciona da seguinte forma. A rede recebe uma imagem RGB de tamanho 224 x 224, que passa por sucessivas camadas de convolução e que, progressivamente, diminuem a dimensão da imagem de entrada e aumentam a sua profundidade. A imagem que tinha dimensão 224 x 224 x 3, passa a ter dimensão 7 x 7 x

1024. Em seguida, os dados passam por uma camada de *pooling* que reduz ainda mais o tamanho da imagem, de 7 x 7 x 1024 para 1 x 1 x 1024. Neste estágio, os dados passam por uma camada completamente conectada, que realiza uma multiplicação do vetor de dados, de dimensão 1 x 1024 por uma matriz de pesos de dimensões 1024 x 1000, resultando um vetor 1 x 1000. Finalmente, os dados passam por uma camada *softmax*, resultando numa imagem reduzida de tamanho 1 x 1 x 1024. A convolução em profundidade necessita de muito menos multiplicações que a convolução comum, e como a convolução é um processo de alta complexidade computacional, obtém-se uma velocidade de processamento por vezes maior a adoção de modelos baseados em convolução em profundidade, como o *MobileNet*.

Figura 9 – Arquitetura *MobileNet*

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fonte: (HOWARD et al., 2017)

2.4 OCR - *Optical Caracter Recognition*

OCR ou *Optical Caracter Recognition* é uma técnica de reconhecimento de caracteres oriundos de textos escritos à mão, impressos ou digitados. Esta técnica é capaz de segmentar textos de imagens digitalizadas, transformando-os em texto editável. A segmentação depende do padrão, da forma e das características nas quais as palavras foram escritas,

existindo métodos diferentes para realizar a tarefa.

O OCR apareceu inicialmente como uma forma de auxiliar deficientes visuais em meados do século XX. Uma versão moderna do OCR surgiu em meados de 1940 com o desenvolvimento do computador digital (TRIER; JAIN; TAXT, 1995).

Para realizar o reconhecimento de letras e números, o OCR utiliza técnicas de redes neurais artificiais. O reconhecimento de caracteres é uma tarefa difícil, devido as inúmeras variações em tamanho, posição, orientação e forma das palavras, o que dificulta a aplicação de um algoritmo predeterminado para o reconhecimento destes, principalmente em textos manuscritos (TRIER; JAIN; TAXT, 1995).

O OCR tem aplicações em diferentes domínios da visão computacional. Uma aplicação possível e tipicamente utilizada é no reconhecimento de placas de veículos onde é realizada segmentação para separar cada um dos caracteres da placa, reconhecendo-as, como pode ser visto na Figura 10, onde o resultado do reconhecimento da placa foi satisfatório.

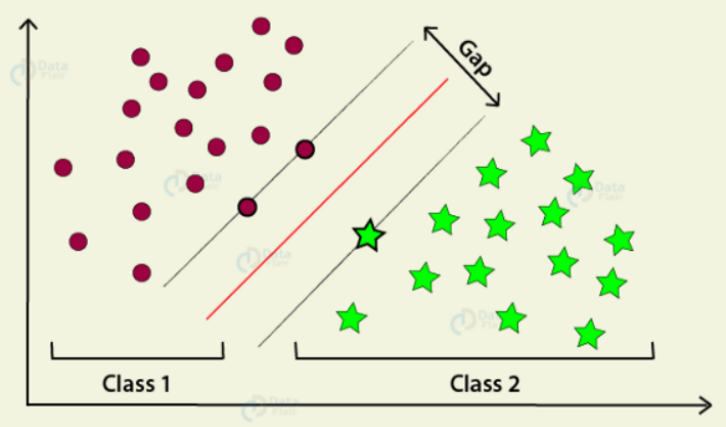
Figura 10 – Aplicação do OCR sobre uma imagem

Fonte: (Autora, 2021)

2.5 SVM e HOG

A máquina de Vetores de Suporte (SVM - *Suport Vector Machine*) é um algoritmo de aprendizado de máquina supervisionado tipicamente utilizado para problemas de classificação e regressão. Quando empregado para classificação, ele classifica algo como pertencente (ou não) a um determinado grupo ou classe. A Figura 11 abaixo ilustra um classificador multiclasse de 2 classes distintas. Multiclasse pois o problema possui mais de uma classe.

Figura 11 – SVM multiclasse



Fonte: (Introdução ao Support Vector Machine (SVM), [s.d.])

Um SVM parte dos dados, representados pelas bolas vermelhas e pelas estrelas verdes da figura, e produz o hiperplano ou a linha que melhor separa as classes. Essa linha é o limite de decisão: qualquer coisa que cair para um lado dela é classificado como uma bola vermelha e tudo que cair para o outro como uma estrela verde.

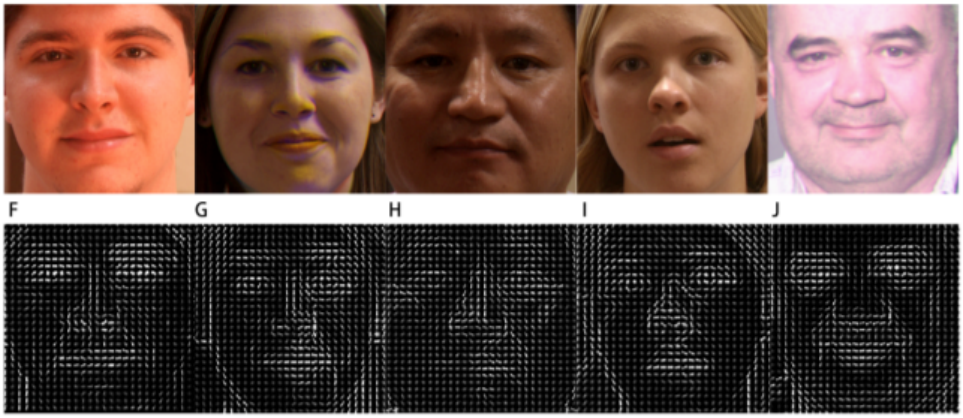
O SVM funciona relativamente bem quando existe uma margem clara de separação entre as classes e quando a base de dados de imagens não é muito grande.

O histograma de gradientes orientados (HOG - *Histogram of Oriented Gradients*) é um descritor de características de imagens empregado para detecção de objetos. A ideia do HOG é usar contornos para descrever o conteúdo de uma imagem. Por exemplo, o contorno de uma pessoa em relação a outra é diferente, assim o HOG quantifica essas diferenças de forma a diferenciar duas pessoas. Os descritores HOG foram utilizados por (DALAL; TRIGGS, 2005) como extrator de características em um SVM.

O cálculo da HOG utiliza o conceito de distribuição da intensidade do gradiente ou direção de cada pixel em uma imagem para descrever um objeto. O gradiente é a medida da variação de uma grandeza, numa determinada direção espacial. A imagem de entrada é dividida em pequenas regiões chamadas células e, para os pixeis dentro de cada célula, um histograma de direções de gradiente é gerado. Ou seja, é calculado a quantidade de gradientes em cada direção principal (quantos apontam para cima, para direita, para a esquerda, etc.). Em seguida, é substituído essa célula da imagem pelas direções das setas que eram as mais fortes. O resultado é a transformação da imagem original em uma representação simplificada do objeto.

A Figura 12 ilustra as características HOG obtidas de imagens de pessoas em diferentes condições de iluminações, expressões, sexos e idades. O resultado é uma representação de rostos com base nas suas características principais. Pode-se observar que as características obtidas para cada pessoa são diferentes, o que permite, adotando o descritor HOG e o classificador SVM, detectar rostos e os atribuir às classes características.

Figura 12 – Características HOG em imagens 3D



Fonte: (ZHENG et al., 2019)

3 Metodologia

Neste capítulo serão apresentadas as principais atividades desenvolvidas para a identificação e reconhecimento de placas de automóveis e de seus condutores. Foram implementados dois algoritmos: um para identificar e reconhecer placas de automóveis e outro para identificar e reconhecer condutores. Para cada algoritmo é detalhado as etapas necessárias para a sua implementação.

3.1 Algoritmo para identificação e reconhecimento da placa de um automóvel

A implementação do algoritmo para identificação de placas de automóveis utiliza redes neurais e é baseada em (RENOTTE, 2021). Para o reconhecimento de placas, foi empregada a tecnologia OCR, onde utilizou-se a rede neural convolucional profunda desenvolvida por Jaided, chamada EasyOCR, disponibilizada em (EasyOCR, 2021).

3.1.1 Instalação e *setup* do ambiente de trabalho

O primeiro passo a ser realizado foi a instalação dos softwares e das bibliotecas necessárias para o projeto. Foi instalado o ambiente de trabalho Jupyter Notebook, a linguagem de programação Python (versão 3.7.3), a biblioteca *Tensorflow* e outras bibliotecas. Embora o ambiente de trabalho escolhido para desenvolvimento de *software* ter sido o Jupyter Notebook, é possível executar o algoritmo utilizando o Google Colaboratory. O Google Colaboratory, ou "Colab", é uma plataforma online que permite escrever código Python sem nenhuma configuração necessária, com acesso gratuito a GPUs e compartilhamento fácil (Google Colaboratory, [s.d.]). A vantagem dessa plataforma comparada com o Jupyter Notebook é a possibilidade de utilizar as Unidades de Processamento Gráfico (GPUs) da Google na hora de executar os códigos, o que é útil para trabalhos com aprendizado de Máquina e Inteligência Artificial que demandam alto processamento computacional. Para o trabalho em questão, foi possível treinar o algoritmo utilizando a própria CPU.

Em seguida, foi realizado o download de modelos pré-treinados do *Tensorflow*, aos quais foi aplicado técnicas de afinamento buscando detectar placas. A prática de se utilizar modelos pré-treinados é comum na resolução de problemas de redes neurais, e está associada a uma técnica chamada *Transfer Learning* (ou transferência de aprendizado). Essa técnica tem como princípio utilizar o conhecimento adquirido por uma rede neural durante a resolução de um problema para resolver outro mais ou menos semelhante. Modelos pré-treinados podem ser encontrados em bibliotecas de *Deep Learning* como o *Tensorflow* e o *Keras*. A transferência de aprendizado, além de acelerar o treinamento da rede, ajuda a prevenir o *overfitting* (ou superaprendizado). Fonte: ("Classez et segmentez des données visuelles", [s.d.]). O modelo de rede pré-treinado escolhido foi o *MobileNet-SSD* do *Tensorflow*. Os detalhes referentes ao modelo são apresentados em uma das subseções que seguem.

3.1.2 Base de dados

Após a instalação e *setup* do ambiente de trabalho, foi necessário escolher uma base de dados (*database*) contendo imagens de veículos e suas respectivas placas. Foi escolhido e utilizado uma base de dados do *Kaggle Data* (Car License Plate Detection, [s.d.]). A base de dados contém 432 imagens de carros de resolução 467 x 300 x 3 e 432 anotações ou *labels* referentes às placas.

A base foi dividida em três partes denominadas de treino, validação e teste. Essa divisão é realizada porque durante o treinamento a rede pode aprender a decorar padrões específicos dos exemplos do conjunto de treino ao invés de padrões genéricos que efetivamente descrevem as classes, fenômeno conhecido como *overfitting*.

O particionamento da base em treinamento, validação e teste foi realizada seguindo a seguinte proporção: 95% das imagens para treino e validação e 5% das imagens para teste. A proporção de particionamento foi à critério da autora. Na verdade, não existe um modelo de divisão considerado mais adequado, ela varia de acordo com a complexidade do problema em que se deseja resolver e com o tamanho da base de dados que se está trabalhando.

A amostra de imagens destinada à treinamento é usada para que o algoritmo treine e aprenda; a amostra de dados destinada à validação é usada para avaliar o desempenho do

algoritmo. Esta avaliação ocorre durante todo o processo de treinamento, onde as imagens de validação são usadas para garantir que os padrões aprendidos não sejam particulares à base de treino. Ao final do processo de treinamento, é realizado uma nova verificação utilizando a partição de imagens do conjunto de teste, onde é realizado uma verificação final com exemplos nunca vistos, garantindo assim que os padrões que a rede aprendeu a reconhecer são os genéricos e não particulares dos conjuntos de treino/validação.

Para evitar o *overfitting*, a separação entre os conjuntos de treino e validação era feito durante o treinamento, sendo 15% das imagens separadas para validar o treino. Desta forma, tornava-se mais difícil que a rede decorasse os padrões.

3.1.3 Modelo de rede utilizado

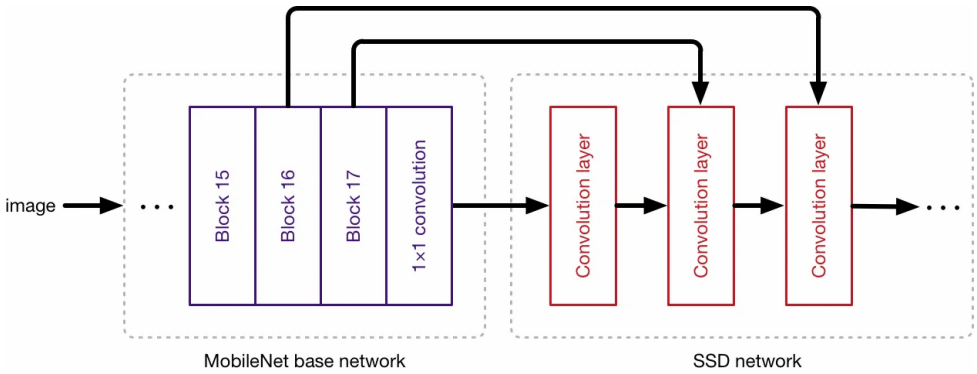
O modelo de rede utilizado foi a rede neural convolucional profunda chamada *MobileNet-SSD*, que trata-se de um modelo pré-treinado de código aberto.

Este modelo de rede consiste em uma rede neural convolucional profunda que funciona como classificador e como detector dos objetos ao mesmo tempo. A primeira parte desta rede é a rede base *MobileNet*, uma rede pré-treinada usada para classificação e reconhecimento. A rede base é classificada como rede muito profunda pois possui um número elevado de camadas e podem classificar centenas de classes.

A segunda parte dessa rede é a rede SSD, do inglês Single Shot Multibox Detection, um detector de objetos *multiBox* de disparo único. O termo *"Single Shot"* implica que a localização e a detecção são realizadas em uma única passagem direta da rede, ou seja, a rede só precisa processar a imagem uma única vez para obter as previsões finais; o termo *"Multibox"* se refere à um método de proposta de coordenadas de caixas delimitadoras rápidas e termo *"Detection"* se refere à rede que detecta objetos da imagem. A abordagem SSD foi proposta por (LIU et al., 2016) e foi desenvolvida pela equipe de pesquisa da Google para atender à necessidade de modelos que podem ser executados em tempo real em dispositivos incorporados. Ela é baseada em uma rede convolucional feed-forward que produz uma coleção de tamanho fixo de caixas delimitadoras e pontuações para a presença de instâncias de classes de objetos nessas caixas. O SSD foi projetado para ser independente da rede de base e, portanto, pode ser executado em qualquer rede de base, como VGG, YOLO e *MobileNet*.

O *MobileNet* foi integrado à estrutura SSD com o objetivo de permitir a execução de redes neurais de alto consumo de recursos e de energia em dispositivos de baixo custo em aplicativos em tempo real. Portanto, quando o *MobileNet* é usado como rede base no SSD, ele se torna o *MobileNet-SSD*. A Figura 13 apresenta uma visão geral das últimas camadas no *MobileNet-SSD*.

Figura 13 – Visão Geral do *MobileNet-SSD*



Fonte: (POKHREL, 2020)

A *MobileNet-SSD* é formada de camadas chamadas convoluções separáveis em profundidade em vez de simples camadas de convoluções para a realizar a detecção de objetos da rede. Segundo (HOWARD et al., 2017), é vantajoso substituir convoluções caras por outras mais baratas, mesmo que isso significasse utilizar mais camadas.

Como pode ser visualizado na Figura 13, a entrada da rede de detecção SSD não é apenas a saída da última camada da rede base *MobileNet*. No caso da detecção de objetos com SSD, é realizada não apenas a leitura da última camada da rede base, mas também das camadas anteriores. Como a detecção de objetos é mais complicada do que a classificação, o SSD adiciona muitas camadas convolucionais adicionais no topo da rede base, o que explica a necessidade de um extrator de características rápidas como o *MobileNet*.

3.1.4 Treinamento

Uma vez que se definiu a base de dados a ser utilizada e o modelo de rede neural, foi realizado o treinamento do modelo de detecção de objetos que detectam placas de carros. Após o treinamento, foi testada a detecção em imagens pontuais do conjunto de teste e em tempo real. Seguida a detecção das placas, foi aplicado a ferramenta OCR *Optical Character Recognition* para reconhecer os caracteres da imagem da placa do veículo.

A avaliação do desempenho do algoritmo de detecção de placas é realizada na seção Resultados e Discussões.

3.2 Algoritmo para identificação e reconhecimento do condutor do automóvel

A implementação do algoritmo para identificação e reconhecimento do condutor de um automóvel é baseada em (GEITGEY, 2021). Para a tarefa de identificação ou detecção de um condutor, foi utilizado o descritor de características HOG. Para o reconhecimento, foi utilizado o classificador SVM.

3.2.1 Instalação e *setup* do ambiente de trabalho

Primeiramente, foram realizadas pesquisas das bibliotecas disponíveis na literatura utilizadas para reconhecimento facial. Foram retidas duas bibliotecas, as bibliotecas *dlib* e *face_recognition*. A biblioteca *dlib* é uma das bibliotecas mais utilizadas para reconhecimento facial (*Face detection with dlib ((HOG and CNN)*, 2021)). A biblioteca *face_recognition* envolve funções de reconhecimento facial do *dlib* de forma simples e fácil de usar. Elas foram desenvolvidas com aprendizado profundo, tendo uma precisão de 99,38% no *Labeled Faces in the Wild*, base de dados de fotos de rostos referência para reconhecimento facial (GEITGEY, 2021). As bibliotecas *dlib* e *face_recognition* foram utilizadas neste trabalho. Além disso, foi utilizada a biblioteca de código aberto *Scikit-Learn*, que permite empregar aprendizado de máquina em Python para problemas de classificação. O algoritmo para identificação e reconhecimento do condutor de um automóvel foi desenvolvido em Python (versão 3.7.3) no ambiente de trabalho Google Colaboratory (ou Colab).

3.2.2 Base de dados

Para este projeto, é utilizado a base de dados *Labeled Faces in the Wild* que, como comentado anteriormente, é uma base de dados de fotos de rostos referência para reconhecimento facial. Esta base de dados consiste em um diretório com pastas, onde cada

pasta contém fotos aleatórias do rosto de uma pessoa. Cada pasta é nomeada com nome da pessoa, para identificação nas etapas de treinamento e reconhecimento.

O conjunto de imagens dessa base de dados contém mais de 13.000 imagens de rostos coletados da internet (GEITGEY, 2021). No entanto, foi observado que a base de dados é desbalanceada, ou seja, existem pastas com poucas imagens de rostos, chamadas classes minoritárias, em comparação com outras (classes maioritárias). Desta forma foi criado *scripts bash* para organizar a base de dados de imagens, balanceando as classes. Foi mantido na base de imagens apenas as pessoas para as quais existia pelo menos 21 imagens de seus rostos. A base de dados original foi reduzida de 13.000 imagens para 1.113 imagens.

Nesta fase temos as imagens de 53 pessoas, sendo cada pessoa com 21 fotos, totalizando as 1.113 imagens, o que nos permite testar se conseguimos detectar e reconhecer cada pessoa na etapa de teste. No entanto, sabe-se que pessoas desconhecidas pela base de dados podem passar na frente da câmera, devendo ser ignoradas e devidamente não reconhecidas pelo algoritmo. Além disso, objetos do plano de fundo da câmera ou até mesmo animais devem ser ignorados pelo algoritmo de reconhecimento.

Dessa forma, para permitir testes nestes cenários foram adicionadas 53 imagens de rostos desconhecidos e 53 imagens de objetos selecionados aleatoriamente do Google Imagens, pois observou-se que a base de dados *Labeled Faces in the Wild* contém apenas fotos de rostos. Assim, a base de dados final utilizada neste trabalho contém no total 1219 imagens.

Para treino, foram consideradas 20 rostos das 53 pessoas conhecidas, totalizando 1060 imagens. Para teste, foram considerados:

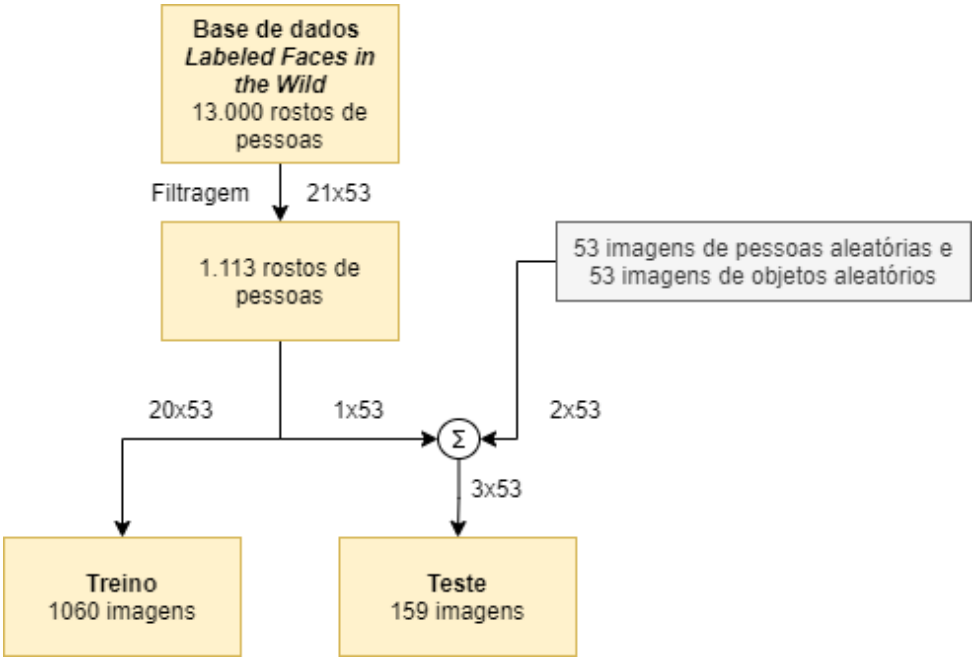
- 53 imagens de rostos de pessoas da própria base de dados.
- 53 imagens de rostos de pessoas desconhecidas pela base de dados, selecionadas aleatoriamente da internet
- 53 imagens de objetos e animais por exemplo. Destas 53 imagens, foram selecionados 28 carros, 10 bicicletas, 6 pássaros, 5 cachorros e 3 árvores, selecionadas aleatoriamente da internet.

Em resumo, a base de dados foi particionada em treino e teste seguindo a seguinte proporção: 1060 fotos para treino e 159 fotos para teste, totalizando 1219 imagens.

O conjunto de treino são formados pelas imagens de rostos nos quais o classificador será inicialmente treinado. O conjunto de teste é utilizado para avaliar o classificador com base nas imagens de rostos que ele nunca viu antes. A divisão total dos dados foi realizada seguindo a proporção: 87% das imagens para treino e 13% das imagens para teste.

A Figura 14 é um esquemático representativo de como os dados foram particionados.

Figura 14 – Esquemático de particionamento dos dados de imagens.



Fonte: (Autora, 2021)

3.2.3 Processo de reconhecimento facial

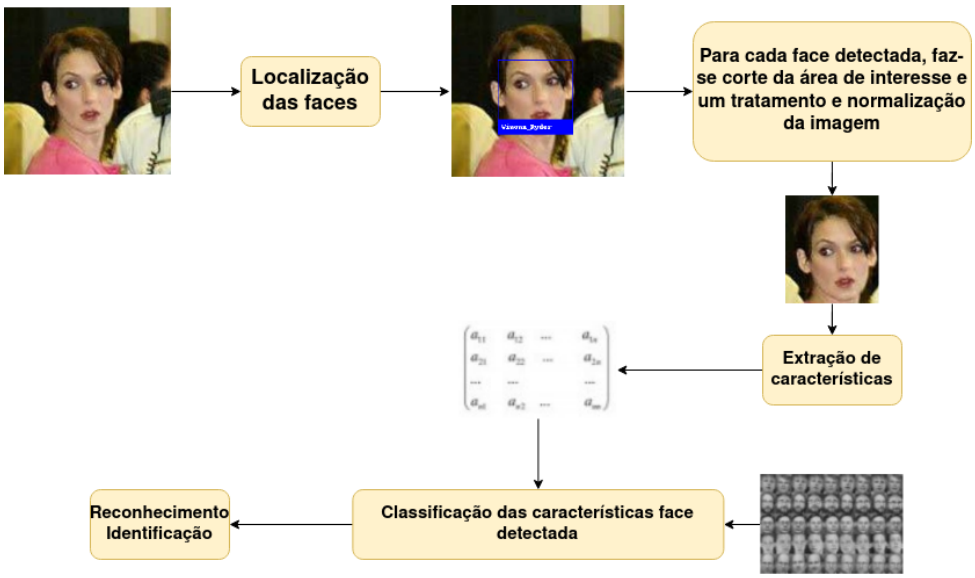
Este trabalho utiliza o algoritmo supervisionado SVM na tarefa de reconhecimento facial. No caso específico deste trabalho, a classificação é realizada entre as 53 classes da base de dados, sendo cada classe referente a uma pessoa diferente.

Dado uma imagem de entrada contendo um rosto, o algoritmo realiza a detecção do rosto desta pessoa, delimitando a região da imagem contendo o rosto com uma caixa delimitadora azul, como é ilustrado na Figura 15. Em seguida é realizado um corte da imagem sobre a região da caixa delimitadora que contém o rosto. É aplicado um extrator de características HOG sobre esta imagem, retornando uma matriz representativa e simplificada das características do rosto da pessoa. Finalmente, é realizado o treinamento

de um SVM para classificar o rosto detectado pelas suas características. Finalmente, é realizado o reconhecimento facial.

A Figura 15 ilustra o processo realizado para identificação e reconhecimento de uma pessoa por uma foto do seu rosto.

Figura 15 – Processo de reconhecimento facial



Fonte: (Autora, 2021)

3.2.4 Treinamento

Para treinar o classificador é fornecido como entrada imagens da base de dados. Se a imagem em treinamento contém um único rosto, o algoritmo o detecta e tem como saída a pessoa que ele identificou e a sua probabilidade de predição; se a imagem em treinamento contém mais de um rosto ou não contém rostos conhecidos pela base de dados, o algoritmo escreve uma mensagem de que não é possível reconhecer aquele rosto.

4 Resultados e discussões

Nessa seção é apresentado algumas métricas de desempenho utilizadas para avaliar um modelo de redes neurais. Nas subseções que se seguem, é apresentado os resultados de desempenho utilizando essas métricas para o algoritmo de detecção de placa de um automóvel e para o algoritmo de identificação e reconhecimento do condutor de um automóvel.

Em um problema de classificação, para avaliar as previsões realizadas pela rede, é medido a quantidade de acertos, quando a classe prevista é igual ao gabarito, e a quantidade de erros, quando a classe prevista não corresponde ao gabarito. No entanto, para visualização do desempenho de um algoritmo de classificação, comumente se utiliza o que se chama de Matriz de Confusão. Ela oferece quatro métricas diferentes e individuais:

- Verdadeiros positivos (TP): Elementos pertencentes à classe analisada que foram reconhecidos como elementos dessa classe;
- Falsos positivos (FP): Elementos identificados como pertencentes à classe analisada sem que essa seja a classe à qual pertencem.
- Falsos negativos (FN): Elementos pertencentes à classe analisada, mas que não foram identificados como pertencentes a ela.
- Verdadeiros negativos (TN): Elementos que não pertencem à classe analisada que foram identificados como elementos de outras classes.

A Figura 16 ilustra uma matriz de confusão.

Figura 16 – Matriz de confusão

Fonte: (Autora, 2021)

Com base nessas quatro métricas, outras métricas podem ser calculadas, oferecendo mais informações sobre como o modelo se comporta:

1. **Precisão ou *Average Precision* (AP)**: Proporção de verdadeiros positivos classificados como positivos. Ou seja, dentre todas classificadas como positivas, essa métrica avalia quantas são realmente positivas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$P = \frac{TP}{TP + FP}, 0 \leq P \leq 1 \tag{4.1}$$

2. **Revocação ou *Average Recall* (AR)**: Número de positivos classificados corretamente em relação ao total de positivos. Dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$R = \frac{TP}{TP + FN}, 0 \leq R \leq 1 \tag{4.2}$$

3. ***F1-Score***: Essa métrica combina as duas métricas anteriores, precisão e revocação, de modo a trazer um número único que indique a qualidade geral de um modelo. Essa métrica também varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$F1\text{-}Score = \frac{P * R}{P + R}, 0 \leq F1\text{-}Score \leq 1 \tag{4.3}$$

4.1 Algoritmo para identificação e reconhecimento da placa de um automóvel

Com base nas métricas de desempenho apresentadas na Seção Resultados e Discussões, é apresentada neste subcapítulo a avaliação do modelo empregado para identificar a placa de um automóvel.

A detecção de placas de automóveis se enquadra como um problema de classificação binária, onde cada imagem de entrada é atribuída a uma de duas classes, existe uma placa ou não existe uma placa.

Neste tipo de problema, é esperado que existam mais falsos positivos e verdadeiros positivos pois a base de dados de imagens continha necessariamente imagens com placas de carro. Ou seja, a probabilidade que o algoritmo detecte uma placa e exista uma placa é grande, assim como a probabilidade que o algoritmo estime que detectou uma placa mas na verdade não é uma placa.

No entanto, existe a possibilidade que o modelo produza falsos negativos, pois algumas imagens da base de dados não apresentavam boa resolução ou apresentavam baixo brilho. Além disso, em algumas imagens a visualização da placa do carro é comprometida pela distância em que se foi capturada a imagem, logo, a detecção também é comprometida.

A tabela abaixo fornece as médias das métricas apresentadas. A performance da *MobileNet-SSD* foi abaixo do desejado. Este resultado pode ser explicado pelo tamanho da base de dados, que é pequena. Uma solução possível seria fazer um pré-processamento da base de dados utilizada, aumentando a quantidade de imagens utilizando *data augmentation*, ou até mesmo trocar a base de dados. O parâmetro de tempo foi calculado pela média dos tempos e processamento de cada imagem.

Tabela 1 – Avaliação do modelo *MobileNet-SSD* para a detecção de placas de carro

Arquitetura	AP	AR	Tempo médio de processamento (s)
MobileNet-SSD	61,13 %	62,65 %	0,035

Fonte: (Autora, 2021).

A decisão de usar precisão ou revocação depende do tipo de problema que está

sendo resolvido. Se o objetivo é detectar todas as amostras positivas (sem se preocupar se as amostras negativas seriam classificadas incorretamente como positivas), é recomendado usar revocação. No problema de detecção de placas de carro de uma imagem, por exemplo, a métrica que melhor avalia o modelo é o revocação. Isso é explicado pois a classificação incorreta do algoritmo de considerar que algum objeto é um carro não é tão problemático, pois o objetivo é detectar todos os objetos alvo.

A precisão é mais utilizada se o problema for sensível à classificação geral de uma amostra como positiva , ou seja, incluindo amostras negativas que foram falsamente classificadas como positivas. Em um problema de detecção de câncer em uma radiografia, por exemplo, a métrica que melhor avalia o desempenho é a precisão. Pois o resultado errôneo de identificação de câncer é muito sensível, devendo ser evitado, logo deve-se ter certeza ao classificar uma imagem como Positiva (ou seja, com câncer).

A Figura 17 ilustra alguns exemplos de imagens de carro pertencentes ao conjunto de testes selecionadas aleatoriamente e as saídas produzidas pela rede *MobileNet-SSD*. As saídas correspondem ao resultado da detecção de placas. A caixa delimitadora verde corresponde à localização das placas na imagem obtida pelo modelo. Nesse subconjunto de imagens do conjunto de teste, a detecção é satisfatória, pois localização e delimitação das placas predito pelo algoritmo correspondente à real localização das placas dos automóveis.

Figura 17 – Detecção de placas da *MobileNet-SSD* para imagens aleatórias

Fonte: (Autora, 2021)

Acima das caixas delimitadoras, é apresentado o limiar de incerteza do algoritmo quanto à sua predição. Nessas imagens sobre teste, a maioria das imagens apresenta um limiar acima de 70%. No entanto, em algumas imagens o modelo realizou algumas predições

com limiar baixo, ou seja, com alta incerteza. Um exemplo é o resultado de predição de 21% da segunda linha e da segunda coluna de imagens. Pode-se concluir que o nível de certeza do algoritmo que se exista uma placa na imagem é baixo, sendo que existe uma placa. Este resultado pode ser explicado pelo distância do carro da câmera no momento em que a foto foi tirada ou pela qualidade baixa das fotos, sendo difícil para o algoritmo reconhecer a placa.

O próximo passo, após a detecção de placas, é identificar e reconhecer os caracteres das placas. No entanto, para melhor visualização das placas isoladas e da leitura das mesmas, foi aplicado uma região de interesse delimitando a placa. Seguido a aplicação da região de interesse, foi então aplicado OCR para o reconhecimento dos caracteres das placas dos automóveis. Na Figura 18 pode-se observar o resultado do reconhecimento dos caracteres das placas obtido dos mesmos carros das imagens acima.

Figura 18 – Reconhecimento dos caracteres da placa aplicando OCR

Fonte: (Autora, 2021)

Podemos concluir, ao fazer uma análise do conjunto de imagens de teste selecionado acima, que o algoritmo reconhece bem a maioria dos caracteres das placas dos carros. No entanto, o algoritmo realizou erros em algumas imagens de placas. Foi observado que para as placas nas quais os caracteres eram escritos na cor branca o algoritmo não fazia um bom reconhecimento. Para a placa da segunda linha e da segunda coluna de imagens, o algoritmo não consegue reconhecer nenhum caractere da placa. Esta placa é do veículo longe da câmera, onde a imagem capturada pela câmera possui baixa qualidade.

4.2 Algoritmo para identificação e reconhecimento do condutor do automóvel

Com base nas métricas de desempenho apresentadas na Seção Resultados e Discussões, é apresentado neste subcapítulo os resultados obtidos do algoritmo para identificação e reconhecimento facial. Inicialmente é feita uma contextualização das etapas utilizadas para finalmente avaliar o algoritmo.

Inicialmente, foi necessário treinar o modelo. A base de dados de treinamento é composta por imagens de rostos de pessoas separadas por arquivos. Ou seja, cada arquivo é composto por múltiplas fotos de uma pessoa. No treinamento, o algoritmo acessa cada arquivo de uma pessoa e, em seguida, sobre cada imagem da pessoa, utiliza um descritor de características objetos HOG seguido de uma máquina de vetores de suporte SVM para treinar o classificador.

Após o treinamento, é realizada a etapa de teste. O teste é realizado em imagens que o algoritmo não conhece, contendo rostos de pessoas conhecidas pela etapa de treinamento, mas também rostos de pessoas desconhecidas pelo algoritmo ou mesmo contendo objetos, o que é importante para avaliar se o algoritmo aprendeu a reconhecer apenas pessoas desejadas e contidas no banco de dados de treinamento.

Neste trabalho, é desenvolvido um algoritmo de predição que foi implementado na etapa de teste. Assim, na etapa de teste, o algoritmo implementa as seguintes tarefas:

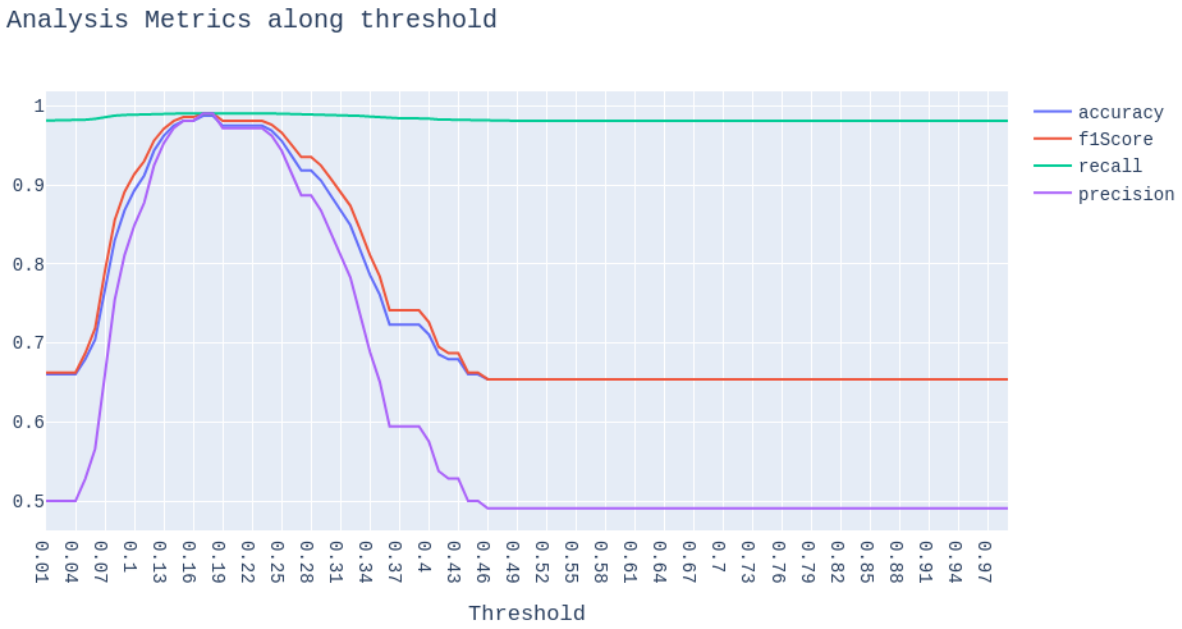
- Detecção facial
- Reconhecimento facial por um modelo SVM
- Definição do limiar que permite aceitar a predição do modelo.

O algoritmo trabalha encontrando similaridades entre a face em análise com as faces contidas no banco de dados de treino, produzindo uma probabilidade de predição. Após esse processo, a face será atribuída a uma das faces do modelo se a probabilidade de predição for superior a um limiar. Isso tenta evitar que faces de pessoas desconhecidas sejam erroneamente reconhecidas. Para isso, é realizado o estudo do limiar em que se aceita a predição realizada pelo modelo.

Seguindo a ordem de tarefas que o algoritmo realiza, a primeira etapa é a detecção facial. A predição só deve ser realizada se na imagem de entrada tiver rostos de pessoas, sejam elas conhecidas ou não. Desta forma, antes da predição, é necessário realizar a detecção de rostos da imagem. Caso exista algum rosto na imagem, seja ela conhecida ou não pelo modelo, o algoritmo faz a predição de reconhecimento utilizando o SVM. Em seguida, o valor predito do algotitmo SVM é comparado com um limiar. Esse limiar corresponde à probabilidade mínima a partir da qual aceitamos que o modelo fez uma boa predição, ou seja, reconheceu corretamente uma pessoa. Abaixo desse limiar, estima-se que o modelo não reconhece bem e, portanto, o algoritmo deve ignorar o reconhecimento e estimar como alguém desconhecido. Desta forma, pode-se garantir segurança em um sistema de abertura automática de portões.

O gráfico da Figura 19 apresenta resultados obtidos de acurácia, revocação, precisão e *F1 score* do algoritmo, considerando as duas situações acima. Observa-se que para um limiar igual a 0,18 as métricas de avaliação são as melhores possíveis. Para o limiar de 0,18, a acurácia é de 98,7 %, a precisão é de 98,8 %, a revocação é de 99,1 % e o F1-score é de 91,1 %.

Figura 19 – Evolução das métricas de desempenho do algoritmo em função do *threshold*



Fonte: (Autora, 2021)

Algumas observações podem ser adicionadas a respeito de cada uma das métricas de desempenho citadas.

1. **Acurácia:** Esta métrica apresentou valor máximo quando o limiar é igual a 0,18. Isso significa que a classe que apresentou melhor acurácia possui a maior probabilidade de predição, que é de 18%. A acurácia reflete a predição correta sobre o conjunto de predições. Assim, para o limiar de 0,18, a acurácia é de 0,987 ou 98,7 %.

2. **Precisão:** A precisão mede o quão bem o algoritmo identificou e reconheceu as pessoas em relação ao total de verdadeiros positivos. No nosso caso, essa métrica é muito importante. Em todas as situações de predição positiva de pessoas, ela avalia o quantas estão corretas. Quanto mais próxima a precisão estiver de 1, menos existem Falsos Positivos, o que é desejado. Observando os resultados de precisão obtidos no gráfico da Figura 19, o valor máximo de precisão é de 98,8 % e ocorre para um limiar de 0,18, o que é pertinente como resultado.

3. **Revocação:** Esta métrica permite medir o numero de positivos preditos corretamente em relação ao total de positivos. Quanto mais a revocação é alta, mais se pode garantir ao utilizador o reconhecimento da pessoa, permitindo que o portão de um estacionamento se abra quando o mesmo aproxima seu rosto da câmera. Observando a Figura 19, para o limiar de 0,18, a revocação é de 0,991 ou 99,1 % aproximadamente. No entanto, ao contrário da precisão, a revocação não é afetada pela variação do limiar, seu valor é sempre muito alto. O que é explicado pela quantidade muito pequena de falsos negativos.

4. ***F1-Score:*** Esta métrica é muito interessante do ponto de vista da nossa análise, pois permite sintetizar as observações de revocação e precisão, representando a média dessas duas métricas. No nosso caso também atinge seu máximo para o limiar 0,18. Para este limiar, o F1-Score é de 0,991 ou 99,1 % aproximadamente.

Com base nestas diferentes métricas podemos concluir que o melhor limiar foi de 0,18 ou 18%. Para tal limiar obtemos resultados de acurácia, precisão, revocação e *F1-Score* satisfatórios e que nos permitem garantir um bom funcionamento do sistema de reconhecimento facial, podendo este sistema ser integrado a um estacionamento para a abertura automática de portões.

A Tabela 2 resume os resultados de desempenho do algoritmo implementado.

Tabela 2 – Avaliação de desempenho do algoritmo de reconhecimento facial

Acurácia	Precisão	Revocação	*F1-Score*
98,7 %	98,8 %	99,1 %	99,1 %

Fonte: (Autora, 2021)

A Figura 20 ilustra alguns exemplos de imagens de pessoas do conjunto de teste selecionadas aleatoriamente para a tarefa de identificação e reconhecimento facial. A detecção dos seus rostos é demarcada pelo retângulo azul. O resultado do reconhecimento de cada pessoa também é identificada com a descrição do seu nome. A detecção e reconhecimento de rostos é satisfatória, possui bons resultados de desempenho e pode ser utilizadas na aplicação proposta neste trabalho.

Figura 20 – Detecção e reconhecimento facial de pessoas aleatórias do banco de teste

Fonte: (Autora, 2021)

5 Considerações finais

Este trabalho se concentrou na implementação de dois algoritmos de visão computacional, um objetivando identificar e reconhecer placas de automóveis e outro objetivando identificar e reconhecer o condutor de um automóvel. O desenvolvimento desses algoritmos exigiu conhecimentos adquiridos ao longo da graduação de Engenharia Elétrica, em especial as disciplinas de Técnicas de Programação e Processamento Digital de Imagens, além dos conhecimentos em Visão Computacional obtidos em projetos realizados durante o intercâmbio BRAFITEC realizado entre 2018 e 2020. Todavia, muitos conhecimentos específicos foram necessários e adquiridos ao longo do projeto, principalmente em Redes Neurais Convolucionais.

Para a identificação e reconhecimento de placas de carros, o algoritmo implementado detecta e reconhece bem a maioria das placas dos carros. No entanto, erros de reconhecimento foram observados em algumas imagens sobre teste. Uma pista da origem do problema é a base de dados utilizada, considerada pequena para uma rede neural convolucional profunda e com algumas imagens com baixa qualidade, tornando difícil a identificação e reconhecimento dos caracteres da placa mesmo para humanos. Uma solução possível seria aumentar a base de dados, realizar um pré-processamento dos dados e utilizar uma câmera de boa qualidade e bem posicionada para a captura das imagens de placas de carros. Os resultados obtidos da avaliação do modelo foram a precisão de 61,13 % e revocação de 62,65 %.

Para a identificação e reconhecimento de rostos de pessoas, foi realizado o balanceamento da base de dados de imagens da *Labeled Faces in the Wild*, que permitiu obter resultados coerentes de predição do algoritmo. Além disso, foi realizado um estudo do limiar que apresenta os melhores resultados das métricas de desempenho. Para o limiar de 0,18, obteve-se acurácia de 97,7 %, precisão de 98,8 %, revocação de 99,1 % e *F1-score* de 99,1 %. Os resultados do desempenho do algoritmo são satisfatórios e ele pode ser utilizado para a aplicação proposta neste trabalho, utilizando reconhecimento facial para permitir o acesso do condutor de um automóvel em um imóvel de restrito acesso.

Com relação à trabalhos futuros, recomenda-se realizar uma comparação do modelo

MobileNet-SSD, utilizado para identificação e reconhecimento de placas, com outros modelos de redes neurais. Esta comparação poderia ser avaliada utilizando as métricas de desempenho apresentadas afim de encontrar o melhor modelo de rede neural pra problemática deste trabalho. Além disso, seria interessante trabalhar na construção de um maior banco de dados e trabalhar no pré-processamento dos dados afim de alcançar melhores resultados de desempenho. Por fim, a integração final dos algoritmos em um sistema eletrônico embarcado poderia ser realizada afim de validar os resultados em um ambiente de teste próximo à realidade, como em um estacionamento em prédios e condomínios fechados.

Referências Bibliográficas

HAYKIN, S. S.; HAYKIN, S. S. **Neural networks and learning machines**. 3rd ed ed. New York: Prentice Hall, 2009.

AI VS ML VS DL. **Let's Understand The Difference**. Analytics Vidhya, 7 jul. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/07/ai-vs-ml-vs-dl-lets-understand-the-difference/>. Acesso em: 22 set. 2021

Classez et segmentez des données visuelles. Disponível em: <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles>. Acesso em: 22 set. 2021.

POKHREL, S. **Real-Time Vehicle Detection with MobileNet SSD and Xailient**. Disponível em: $< https://www.xailient.com/post/real-time-vehicle-detection-with-mobilenet-ssd-and-xailient >$. Acesso em: 27 set. 2021.

Google Colaboratory. Disponível em: $< https://colab.research.google.com/notebooks/welcome.ipynb?hl=pt-BRscrollTo=5fCEDCU_qrC0 >$. Acesso em: 27 set. 2021.

LIU, W. et al. **SSD: Single Shot MultiBox Detector**. arXiv:1512.02325 [cs], v. 9905, p. 21–37, 2016.

RENOTTE, N. **Tensorflow Object Detection Walkthrough**. [s.l: s.n.].

Synced. Disponível em: <https://syncedreview.com/>. Acesso em: 29 set. 2021.

HOWARD, A. G. et al.**MobileNets: Efficient Convolutional Neural Networks for**

Mobile Vision Applications. arXiv:1704.04861 [cs], 16 abr. 2017.

NG, A.; KATANFOROOSH, K.; MOURRI, Y. B. **Convolutional Neural Networks**. Disponível em: $<https://www.coursera.org/learn/convolutional-neural-networksfaq>$. Acesso em: 2 out. 2021.

DENG, L.; YU, D. **Deep Learning: Methods and Applications**. Foundations and Trends in Signal Processing, v. 7, n. 3–4, p. 197–387, 30 jun. 2014.

RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [S.l.]: Packt Publishing Ltd, 2017.

TRIER, Ø. D.; JAIN, A. K.; TAXT, T. **Feature Extraction Methods For Character Recognition - A Survey**, 1995.

DALAL, N.; TRIGGS, B. **Histograms of oriented gradients for human detection**. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Anais... In: 2005 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05). jun. 2005.

Face detection with dlib (HOG and CNN). PyImageSearch, 19 abr. 2021. Disponível em: <https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/>. Acesso em: 5 out. 2021

GEITGEY, A. **Reconhecimento facial**. [s.l: s.n.].

ZHENG, S. et al. **3D texture-based face recognition system using fine-tuned deep residual networks**. PeerJ Computer Science, v. 5, p. e236, 2 dez. 2019.

Car License Plate Detection. Disponível em: <https://kaggle.com/andrewmvd/car-plate-

detection>. Acesso em: 7 out. 2021.

Introdução ao Support Vector Machine (SVM). Disponível em: <https://ichi.pro/pt/introducao-ao-support-vector-machine-svm-225431442261105>. Acesso em: 7 out. 2021.