UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

Gustavo Ramos Lima

SISTEMA DE VIDEOMONITORAMENTO AUTÔNOMO COM IDENTIFICAÇÃO DE SUSPEITOS UTILIZANDO BIOMETRIA FACIAL

Vitória-ES

Dezembro/2018

Gustavo Ramos Lima

SISTEMA DE VIDEOMONITORAMENTO AUTÔNOMO COM IDENTIFICAÇÃO DE SUSPEITOS UTILIZANDO BIOMETRIA FACIAL

Parte manuscrita do Projeto de Graduação do aluno Gustavo Ramos Lima, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Engenheiro Eletricista.

Universidade Federal do Espírito Santo

Centro Tecnológico

Departamento de Engenharia Elétrica

Projeto de Graduação

Orientador: Prof. Dr. Patrick Marques Ciarelli

Vitória-ES

Dezembro/2018

Gustavo Ramos Lima

SISTEMA DE VIDEOMONITORAMENTO AUTÔNOMO COM IDENTIFICAÇÃO DE SUSPEITOS UTILIZANDO BIOMETRIA FACIAL

Parte manuscrita do Projeto de Graduação do aluno Gustavo Ramos Lima, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Engenheiro Eletricista.

Prof. Dr. Patrick Marques Ciarelli
Orientador

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo

MSc. Philipe Rangel Demuth
Universidade Federal do Espírito Santo

Vitória-ES

Dezembro/2018

AGRADECIMENTOS

A Deus por proporcionar saúde e força para superar as todas as adversidades. A minha família pelo grande apoio e incentivo nos estudos. A meus amigos por me ajudarem a lidar com os problemas e proporcionarem momentos de alegria e descontração. Ao professor Dr. Patrick Marques Ciarelli pela orientação e apoio no desenvolvimento deste projeto. Agradeço os diversos autores que compartilharam os seus conhecimentos e suas pesquisas visando a disseminação do conhecimento.

RESUMO

Com o aumento dos indicadores de violência, medidas têm sido adotadas no combate aos furtos e roubos em residências e condomínios. Dentre elas, o sistema de videomonitoramento tem se tornado essencial para inibir e registrar a ação de delitos. Por apenas registrar as imagens capturadas pelas câmeras, este sistema geralmente necessita de um operador, que fica responsável por analisar as imagens quando necessário. Realizar a identificação de pessoas ao longo de várias horas de gravações de vídeo pode se tornar uma tarefa exaustiva e demorada, além de aumentar a possibilidade de falha humana. Com a evolução das pesquisas na área de aprendizagem de máquina, sistemas de monitoramento autônomo começaram a ser desenvolvidos utilizando detecção e reconhecimento facial. As redes neurais passaram a ser bastante utilizadas por apresentarem resultados satisfatórios para bases de dados grandes. Por meio do procedimento de transferência de aprendizado, uma rede treinada pode ser reutilizada para outras finalidades como extração de características. Este projeto tem como objetivo utilizar uma rede neural convolucional para automatizar o processo de identificação de indivíduos suspeitos no ambiente residencial. As faces dos indivíduos desconhecidos capturadas, a data e hora das ocorrências foram registradas no sistema para exibição em um *website*. Foram obtidos resultados satisfatórios após a realização de experimentos, que utilizaram uma base de dados contendo vídeos simulando o acesso de indivíduos a dois ambientes internos diferentes.

Palavras-chave: Redes neurais, Videomonitoramento, Reconhecimento facial, Webserver.

ABSTRACT

With the growth of violence levels, actions have been adopted to combat robberies to residences and condominiums. Amongst them, the video surveillance system has become essential to inhibit and register the crime. By only recording the images captured by the cameras, this system usually requires an operator, who is responsible for analyzing the images when necessary. Performing the identification of people in hours and hours of video recordings can become an exhausting and time-consuming task, in addition to increasing the possibility of human failure. With the evolution of research in the area of machine learning, autonomous monitoring alternatives began to be developed using detection and facial recognition. Neural networks have become widely used because they present satisfactory results for large databases. Through the learning transfer procedure, a trained network can be reused for other purposes such as extraction of characteristics. This project aims to use a convolutional neural network to automate the process of identifying suspicious individuals in the residential environment. The faces of the unknown individuals captured, the date and time of occurrences were recorded in the system for display in a website. Satisfactory results were obtained after the experiments, which used a database containing videos simulating the access of individuals to two different internal environments.

Keywords: Neural networks, Video surveillance, Facial recognition, Webserver.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SQL	*Structured Query Language*
HOG	*Histogram of Oriented Gradients*
kNN	*k-Nearest Neighbors*
SVM	*Support Vector Machine*
ReLu	*Rectified Linear Unit*
OpenCV	*Open Source Computer Vision Library*
VGG	*Visual Geometry Group*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Desde a criação do primeiro computador pessoal no início da década de 80, a humanidade tem passado por um processo de desenvolvimento tecnológico que resultou no barateamento e na miniaturização de *hardware*. Com a evolução das máquinas, formas para torná-las mais similares aos seres humanos começaram a serem desenvolvidas. Os sensores buscaram imitar os sentidos, enquanto que a inteligência artificial buscou representar o intelecto humano. Surgiram diversos campos de pesquisa, dentre eles o da visão computacional.

A visão computacional tem como finalidade emular a visão humana, entender a imagem e utilizar as informações obtidas para tomar decisões, diferente do processamento de imagens, que busca melhorar a qualidade das imagens ou facilitar a extração de informações. Na área de visão computacional são estudados métodos de detecção e reconhecimento de objetos em imagens, reconstrução de cenas, análise de movimento, etc. Existem diversas aplicações (NEVES; Vieira Neto; GONZAGA, 2012) na área médica, por exemplo, na análise de imagens para detecção de enfermidades (tumores, lesões, etc.). Na indústria, a visão computacional pode ser utilizada na automatização de processos, como no processo de separação de grãos de café (SZELISKI, 2010). Na área da segurança, estão sendo desenvolvidos sistemas que utilizam a biometria facial, com o objetivo de identificar criminosos em tempo real.

Com o aumento do investimento nesse setor, aplicações foram sendo desenvolvidas visando o barateamento dos sistemas de segurança e a sua popularização. Condomínios e residências passaram a adotar sistemas de monitoramento cada vez mais modernos, uma solução paliativa, mas essencial no combate às práticas criminosas. Com a criação de algoritmos sofisticados e a redução dos custos de *hardware*, um recurso que vem se tornando cada vez mais comum é a biometria facial (WOODWARD et al., 2003), que pode ser adotada para complementar alguns sistemas de monitoramento residenciais que existem no mercado.

A China tem avançado bastante nessa área: bancos, aeroportos, hotéis e até banheiros já contam com sistemas de reconhecimento facial. O governo planeja utilizar a tecnologia para o monitoramento da população na identificação de criminosos, comportamentos suspeitos e na prevenção de crimes (DENYER, 2018). Utilizando um banco de dados contendo imagens de cada um dos 1,4 bilhões de cidadãos a proposta é criar um sistema de grande abrangência no combate à criminalidade no país.

O Brasil sofre com uma crescente onda de violência, com o aumento da taxa de homicídios

e a sensação de insegurança e impunidade. De acordo com (SALGADO, 2018) o Brasil teve em 2018 o número de homicídios 30 vezes maior que a Europa, além disso, segundo relatório do Atlas da Violência (CERQUEIRA et al., 2018) na última década mais de meio milhão de pessoas foram assassinadas. Os roubos e furtos a condomínios sofreram aumento em alguns estados, como em São Paulo que entre 2015 e 2016 registrou um aumento de 172% de casos na sua capital (PRADO; ARCOVERDE, 2017). Segundo estudo do Banco Interamericano de Desenvolvimento, famílias e empresas são responsáveis por 47,9% dos gastos com segurança no país (RESK, 2017). Portanto, a adoção de novas tecnologias na área de segurança têm se tornado cada vez mais necessárias no combate à violência.

1.2 Justificativa

Segundo relatório divulgado pelo Programa das Nações Unidas para o Desenvolvimento (PNUD), o Brasil apresentou, em 2013, a terceira maior taxa de roubo da América Latina (G1, 2013). Com a crescente sensação de insegurança residências e condomínios passaram a adotar sistemas de videomonitoramento de forma a inibir a ocorrência de roubos e furtos. Uma das formas de monitoramento envolve a ação de um operador, que fica constantemente analisando as imagens em busca de atividades suspeitas. Outra maneira é através do armazenamento das imagens das câmeras em um servidor para inspeção de dados, caso seja necessário posteriormente. Entretanto, nessa situação, os vídeos capturados pelas câmeras podem ser analisados apenas horas após a ocorrência do delito, sendo necessário procurar pelo momento do incidente em horas e mais horas de gravação.

Com o desenvolvimento da tecnologia e a redução dos custos de aquisição de componentes eletrônicos, tornou-se possível elaborar sistemas de vigilância mais complexos e inteligentes visando a ação em tempo real que minimizam a necessidade de intervenção do operador. Utilizando algumas técnicas, é possível filtrar os dados gerados pelas câmeras para a obtenção apenas do que é relevante, em um curto intervalo de tempo.

Neste projeto busca-se criar um sistema de vigilância residencial, de baixo custo, para auxílio na prevenção de atividades criminosas, utilizando abordagens adequadas de visão computacional, de forma a permitir a detecção e o reconhecimento facial em tempo real. O sistema proposto usa métodos para a detecção e reconhecimento facial, de forma a realizar a análise em tempo real de imagens provenientes de câmeras de videomonitoramento, e assim identificar e registrar pessoas desconhecidas em um banco de dados.

1.3 Objetivos

O objetivo geral deste trabalho é o desenvolvimento de um sistema que auxiliará na vigilância residencial utilizando a detecção e o reconhecimento facial. Os indivíduos não identificados terão as suas biometrias faciais armazenadas em um banco de dados que será acessível pela internet por meio de um site de gerenciamento, no qual serão exibidos o número de suspeitos identificados e as capturas das faces detectadas.

Os objetivos específicos deste trabalho são: realizar a detecção e o reconhecimento facial utilizando uma rede neural convolucional, classificar indivíduos como suspeitos ou conhecidos, armazenar dados relevantes em um banco de dados em SQL, disponibilizar informações na internet por meio de um servidor web e contabilizar periodicamente o número de indivíduos suspeitos detectados.

1.4 Metodologia da Pesquisa

O desenvolvimento deste trabalho pode ser discriminado em três etapas. Na primeira etapa será utilizada a biblioteca em Python *face_recognition* (GEITGEY, 2017) para a detecção e reconhecimento facial. A rede neural convolucional presente nesse algoritmo será alimentada com novas imagens e serão extraídos na saída vetores de 128 dimensões representando cada face. Os indivíduos serão agrupados em duas categorias: "conhecidos" ou "desconhecidos".

Todas as informações relevantes obtidas na primeira etapa serão armazenadas em um banco de dados SQL, que tornará o sistema mais robusto e confiável.

Na ultima etapa será desenvolvida uma plataforma para exibição das informações aos usuários. Um pequeno servidor será criado para disponibilizar as informações em um *website*.

1.5 Estrutura do Trabalho

A fim de permitir uma melhor compreensão, este trabalho está dividido em cinco capítulos. No Capítulo 1 é realizada a contextualização e a apresentação dos objetivos e justificativas para a realização do projeto. O Capítulo 2 fornece o embasamento teórico necessário ao entendimento do projeto. São abordados assuntos relacionados a *machine learning*, banco de dados, etc. O Capítulo 3 mostra como foi desenvolvido o projeto, exibindo os resultados obtidos em cada etapa. O Capítulo 4 analisa o comportamento do sistema utilizando

diferentes bases de dados. No quinto capítulo é realizada a conclusão baseando-se nos resultados encontrados, além de serem propostas melhorias e sugestões para trabalhos futuros.

2 EMBASAMENTO TEÓRICO

Neste capítulo são apresentados os principais conceitos utilizados no projeto. Primeiramente é abordado o processo de detecção facial que utiliza o Histograma de Gradientes Orientados (HOG). Depois são apresentados dois algorítimos importantes, um que realiza ajustes nas imagens (transformação afim) e o *Face Landmark*, que identifica pontos de interesse da face humana. São apresentados os conceitos de classificação, reconhecimento facial, o funcionamento das redes neurais convolucionais, utilização do banco de dados, as bibliotecas e as linguagens de programação utilizadas.

2.1 Detecção facial

A detecção facial consiste no processo de identificação das características que representam uma face em uma imagem. O processo de detecção permite encontrar quais regiões da imagem contêm faces. Portanto, ele pode se tornar complexo devido ao grande número de elementos (MING-HSUAN; DAVID; NARENDRA, 2002) que interferem na acurácia, como: expressões faciais, diversidade de posicionamento da face (frontal, perfil, etc), a presença ou não de óculos, chapéu ou barba e alterações no ambiente (luz, brilho, etc).

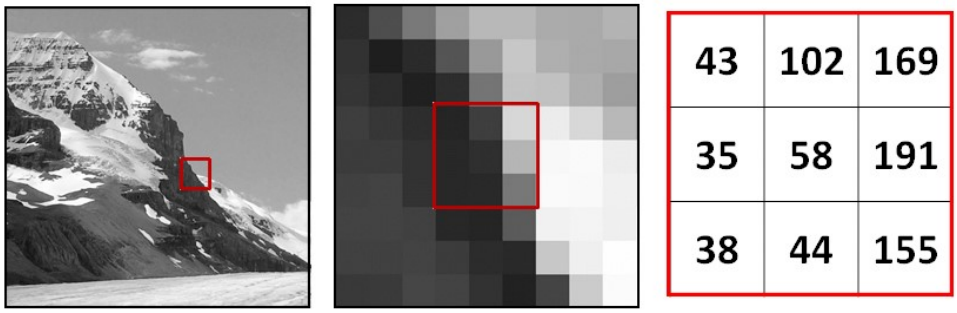
O processo de detecção facial utilizará o Histograma de Gradientes Orientados (HOG), algoritmo proposto por Dalal e Triggs (2005) e conhecido por apresentar bons resultados em ambientes com variação de luminosidade (CERNA; MENOTTI; CÁMARA-CHÁVEZ, 2013).

2.2 Histograma de Gradientes Orientados

O Histograma de Gradientes Orientados (HOG) é um descritor de características que basicamente filtra informações relevantes de um conjunto de dados, sendo utilizado principalmente para a detecção de pessoas (detecção do corpo, da face, etc) e objetos. Baseia-se na análise dos gradientes de variação da intensidade dos pixels em relação aos pixels adjacentes em uma imagem. Antes de aplicar a técnica, a imagem passa por um processo de conversão do formato RGB (red-green-blue) para a escala de cinza com o objetivo de reduzir as dimensões dos dados. Uma imagem em escala em cinza pode ser representada como uma matriz com valores variando de 0 a 255 como mostra a Figura 1.

Após transformar a imagem para a escala de cinza, um filtro (*kernel*) é aplicado para extrair os gradientes horizontal g_x e vertical g_y (Figura 2). Utilizando as Equações 2.1 e

Figura 1 – Quantização dos pixels de uma imagem em escala de cinza.



Fonte: Humboldt State University (2015).

2.2 é obtida a forma polar do gradiente contendo o seu módulo e ângulo. Os gradientes apresentam maior valor em regiões onde há maior variação nos valores dos pixels, que geralmente compreende as regiões de borda da imagem.

Figura 2 – Filtros horizontal e vertical que são aplicados em toda a imagem

Fonte: Mallick (2016)

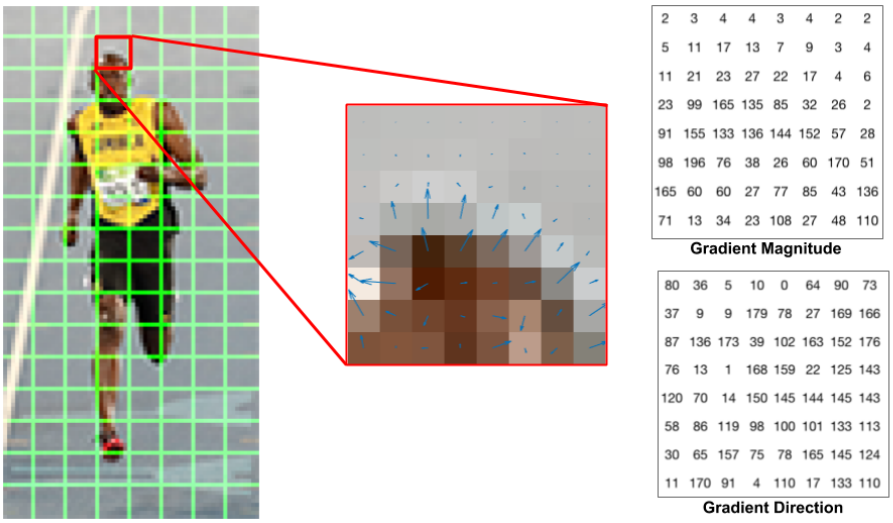
$$g = \sqrt{{g_x}^2 + {g_y}^2} \tag{2.1}$$

$$\theta = arctan\frac{g_y}{g_x} \tag{2.2}$$

Após definir os gradientes para cada pixel a imagem é dividida em pequenas regiões geralmente de dimensão 8×8 ou 16×16 pixels, podendo variar de acordo com as dimensões da imagem e as características que se pretende extrair. Para cada região é obtido um histograma dos gradientes baseando-se na magnitude e direção dos gradientes (Figura 3). Com isso, para cada região é possível saber qual é a direção predominante. Os valores dos gradientes são normalizados antes de serem gerados os histogramas com o objetivo de evitar interferências da variação de luminosidade das imagens.

Para criar um modelo capaz de detectar faces, é preciso aplicar o HOG em um conjunto de imagens de treinamento (Figura 4). Primeiro o HOG realiza a extração de características das

Figura 3 – Esquerda: imagem a ser analisada. Centro: gradientes representados por setas. Direita: módulo e ângulo dos gradientes.

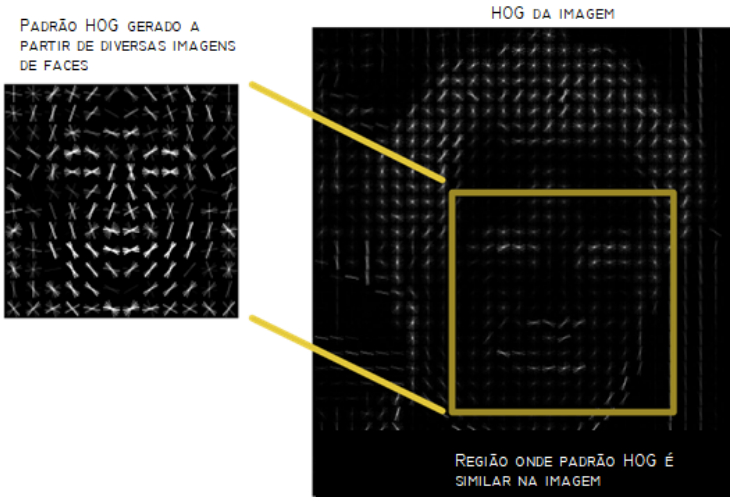


Fonte: Mallick (2016)

faces, gerando vetores de características que são utilizados para treinar um classificador. Para realizar a detecção em uma imagem, uma janela menor do que a imagem a ser analisada é deslocada sobre a imagem, em busca do padrão que representa faces. À medida que a janela é deslocada, são gerados vetores de características que são avaliados pelo classificador. O classificador determina se a janela contém uma face ou não.

A biblioteca Dlib (Seção 2.7.2) disponibiliza dois modelos de detecção facial. O primeiro modelo realiza a detecção frontal de faces utilizando o HOG em conjunto com um classificador SVM. O segundo modelo utiliza uma rede neural convolucional e realiza a detecção da face em diferentes ângulos. Ele apresenta melhores resultados de detecção em relação ao primeiro modelo, contudo necessita de maior poder de processamento. Neste projeto foi utilizado o primeiro modelo por demandar menor poder computacional.

Figura 4 – Padrão HOG utilizado para detectar faces



Fonte: Adaptado de Geitgey (2017)

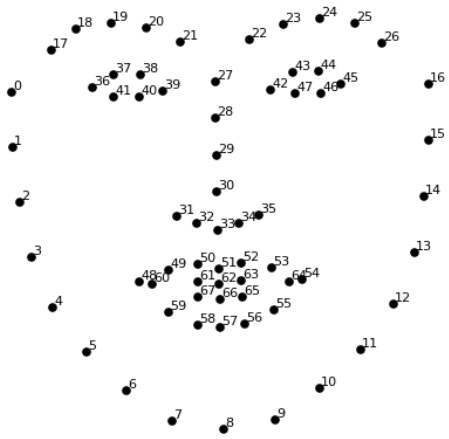
2.3 Algoritmos Utilizados Para Ajustes das Imagens

Nesta Seção são apresentados dois algoritmos utilizados no projeto. O *Face Landmark* que é utilizado para verificar o posicionamento das faces e a transformação afim, que realiza pequenos ajustes nas imagens das faces com o objetivo de padronizar os dados.

2.3.1 *Face Landmark*

O *Face Landmark* é um algoritmo criado por Kazemi e Sullivan (2014) que identifica 68 pontos relevantes na face. Como pode ser visto na Figura 5, ele localiza pontos nas regiões dos olhos, nariz, boca, região das sobrancelhas e contorno do rosto. Com esses pontos é possível verificar de que forma a face está posicionada, se a pessoa está de frente para a câmera no momento da captura ou se está com a face voltada para outra direção. Além disso, é possível determinar o nível de inclinação da face e da abertura dos olhos.

Figura 5 – 68 Pontos que são localizados na face

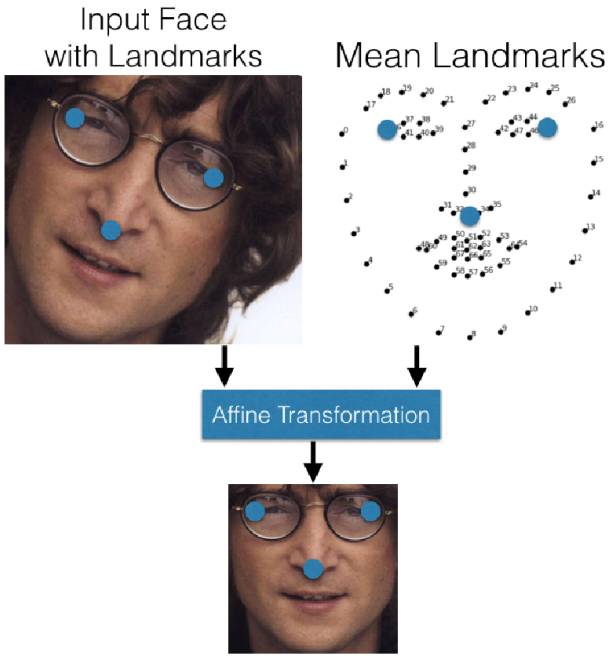


Fonte: Geitgey (2017)

2.3.2 Transformação Afim

Para centralizar e alinhar as faces, baseado nos pontos obtidos pelo *Face Landmark*, é utilizada transformação afim. Tal operação é o conjunto de transformações que distorce, rotaciona e alinha uma imagem com o objetivo de padronizar os dados. A Figura 6 mostra um exemplo de aplicação.

Figura 6 – Ajustes relizados na face utilizando os pontos obtidos pelo *Face Landmark*



Fonte: Amos, Ludwiczuk e Satyanarayanan (2016)

2.4 Reconhecimento facial

O reconhecimento facial permite diferenciar faces de pessoas a partir de traços físicos como posição e características de olhos, nariz, boca, etc. O reconhecimento, diferente da "detecção", necessita de algoritmos mais complexos para que uma pessoa não seja confundida com outra. Os mesmos elementos que interferem na detecção facial como luminosidade e sombra também são de grande impacto no reconhecimento, devendo a base de imagens estar em boas condições para maior acurácia. O reconhecimento facial pode ser utilizado em diversas aplicações (JAFRI; ARABNIA, 2009) como controle de acesso a ambientes, identificação de criminosos em sistemas de videomonitoramento, investigação forense e a autenticação de clientes para realização de operações bancárias.

2.5 Classificadores

O processo de aprendizagem de máquina permite um sistema extrair informações automaticamente, baseado em experiências acumuladas através de soluções de problemas anteriores (MONARD; BARANAUSKAS, 2003). Os classificadores utilizados em uma tarefa são definidos pelas formas de aprendizagem desejadas durante o projeto de um sistema. Existem diversas formas de aprendizado de máquina sendo o objetivo deste trabalho estudar preferencialmente apenas o aprendizado supervisionado e indutivo.

No aprendizado supervisionado aplicado na tarefa de classificação é fornecido ao modelo, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido. A partir da base de treinamento, o classificador consegue rotular os demais elementos a uma das classes disponíveis. Portanto, para um melhor desempenho dos classificadores, estes devem utilizar atributos que maximizem a separação entre as diversas classes utilizadas no treinamento. Algumas técnicas clássicas de classificação são o *Support Vector Machine* (SVM) e o *k-Nearest Neighbors* (kNN). A ideia do kNN é encontrar o rótulo de classificação de uma amostra baseando-se nas amostras vizinhas provenientes de um conjunto de treinamento. São verificadas as distâncias entre a amostra e os vizinhos, onde os k vizinhos mais próximos determinam qual classe a amostra pertence (HARRINGTON, 2012).

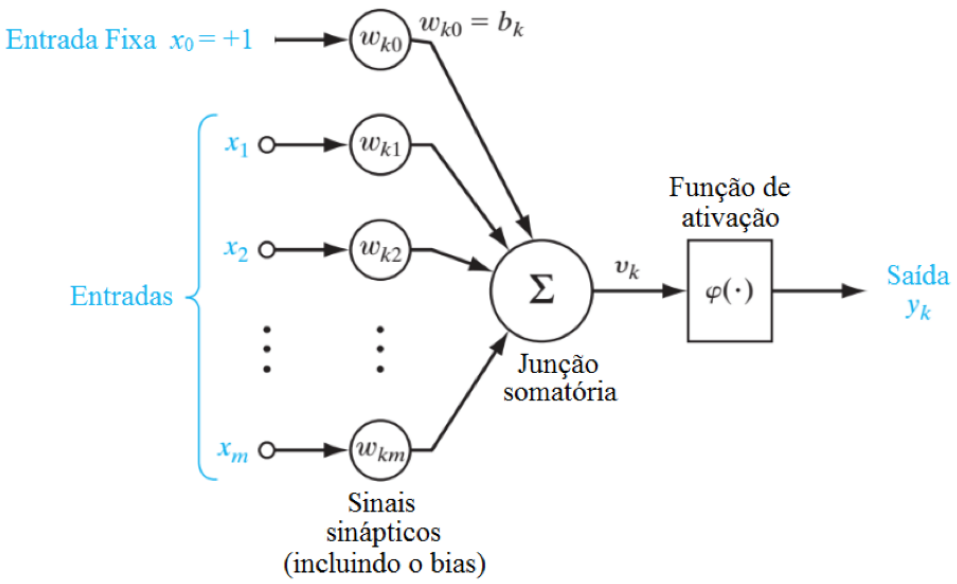
2.6 Redes Neurais Convolucionais

As redes neurais buscam, resumidamente, representar o cérebro humano no ambiente computacional. Uma das características marcantes do cérebro é a capacidade de sofrer constante alteração baseado em experiências vividas. Tal plasticidade resulta em uma

capacidade de realizar atividades complexas com excelência. Nas redes neurais são utilizadas pequenas estruturas análogas aos neurônios (HAYKIN, 2001) e que sofrem alterações à medida que ocorre o processo de aprendizagem na rede.

O neurônio é modelado conforme a Figura 7. Ele é composto por m entradas (x_1 até x_m) que têm seus valores multiplicados por pesos (w_1 até w_m). Além disso, uma entrada fixa x_0 é multiplicada pelo peso w_{k0}, gerando o *bias*. Todos esses elementos são somados gerando a função polinomial v_k (Equação 2.3). O valor de v_k é utilizado como valor de entrada em uma função de ativação (Figura 8), que limita o valor da saída y_k.

Figura 7 – Modelo de neurônio



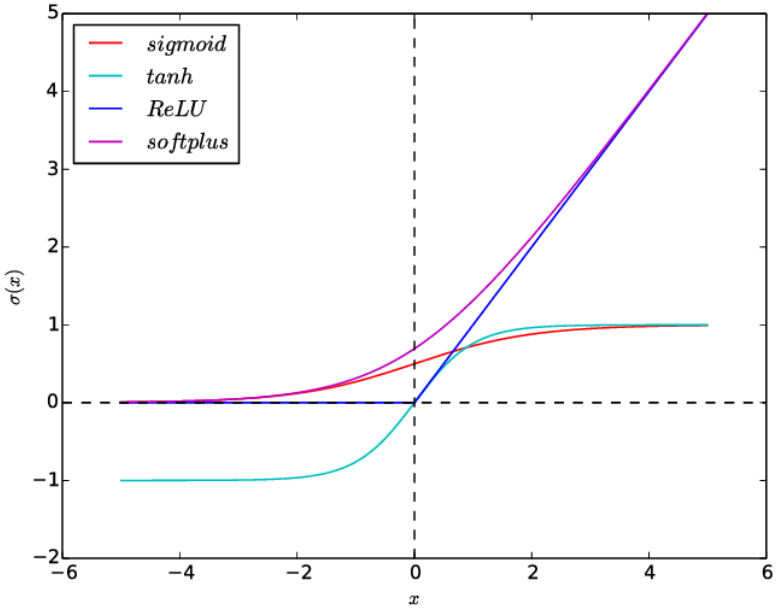
Fonte: Goedert et al. (2017)

$$v_k = w_{k1}x_1 + w_{k2}x_2 + w_{k3}x_3 + ... + w_{km}x_m + b_k = \sum_{j=1}^{m} w_{kj}x_j + b_k \tag{2.3}$$

As redes neurais podem ter os neurônios estruturados em diversas camadas e interconectados de diversas maneiras. As configurações dessas redes geralmente são obtidas empiricamente, após a realização de testes de desempenho. Os valores das entradas e das saídas desejadas são conhecidos durante o processo de treinamento supervisionado. Por meio de algoritmos de otimização como o gradiente descendente estocástico e o método de *backpropagation* vão sendo atualizados os pesos dos neurônios com o objetivo de reduzir o erro entre o valor de saída previsto e o existente (ARAÚJO et al., 2017).

A rede neural convolucional é um tipo de rede utilizada para lidar principalmente com imagens, que passam por diversos filtros, *poolings* e funções de ativação (Figura 9) ao

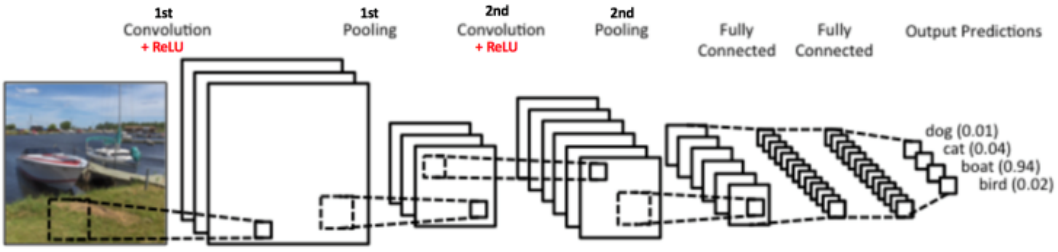
Figura 8 – Alguns tipos de funções de ativação que podem ser utilizadas.



Fonte: Chung, Lee e Park (2016)

longo da arquitetura da rede neural. As operações utilizadas na rede convolucional são as seguintes: convolução, *pooling*, aplicação de uma função de ativação (não linearidade) e classificação utilizando camadas totalmente conectadas.

Figura 9 – Exemplo de uma rede neural convolucional.



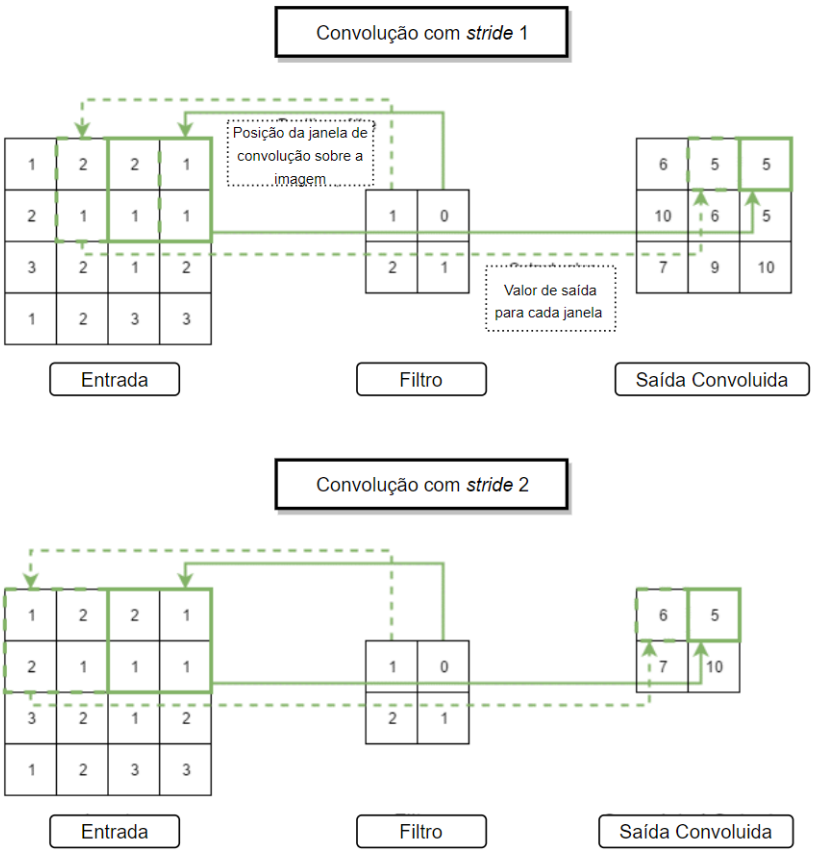
Fonte: Karn (2016)

Convolução

A convolução é um processo que basicamente transforma os dados de entrada de um formato para outro, utilizando filtros (*kernels*) que são pequenas matrizes que tem seus valores alterados durante o treinamento de uma rede. Os filtros são matrizes com dimensão menor que a matriz que representa uma imagem. O Processo de convolução pode ser

compreendido como a soma do produto entre os elementos da matriz do *kernel* e a matriz da imagem. No processo, o *kernel* é deslocado sobre a matriz da imagem gerando desta forma uma nova matriz (matriz de características), operação que é realizada à medida que o filtro é deslocado sobre a matriz da imagem.

Figura 10 – Operação de convolução com *strides* diferentes.



Fonte: Adaptado de Ganegedara (2017)

O processo de convolução é um método de extração de características onde podem ser utilizados vários filtros de forma a se obter diversas matrizes de características. A operação de convolução é caracterizada a partir de três hiperparâmetros (KARN, 2016):

- Profundidade (*Depth*): determina o número de filtros a serem utilizados. No primeiro estágio convolutivo (Figura 9), por exemplo, foram aplicados 3 filtros na imagem, gerando 3 matrizes de características.

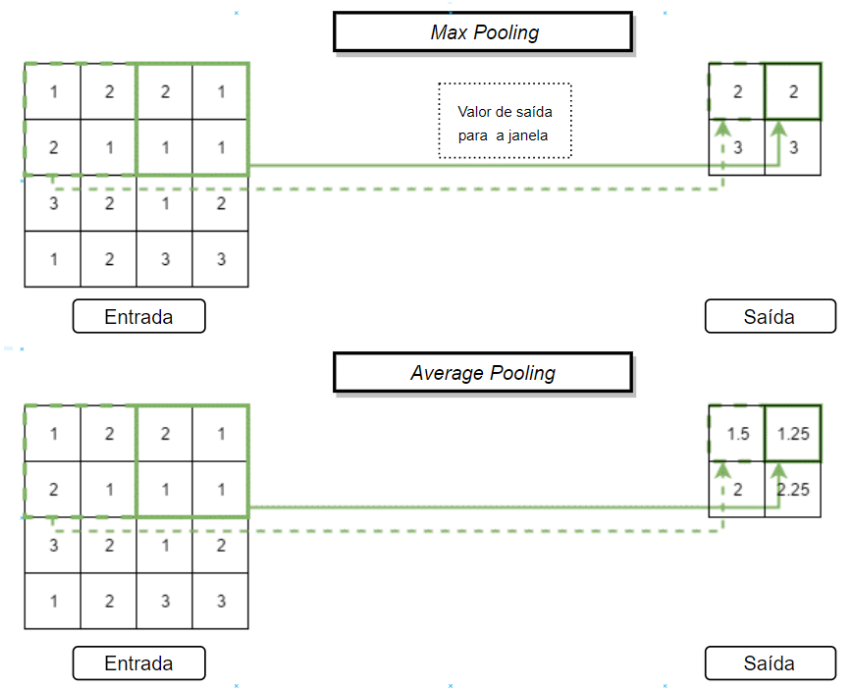
- Passo (*stride*): passo determina o quanto que o filtro se desloca durante o processo de varredura sobre a matriz da imagem. Passos grandes resultam numa maior redução das dimensões da matriz de saída. Na Figura 10, por exemplo, o filtro de dimensão 2×2 realiza a varredura na matriz de dimensão 4×4 utilizando um passo igual a dois e gera uma matriz de saída com metade da dimensão da matriz original.

- Preenchimento (*padding*): insere zeros nas bordas da matriz de entrada para que quando seja aplicado o filtro na imagem sejam considerados os pixels das extremidades.

Pooling

A operação de *Pooling* é utilizada para reduzir a dimensionalidade das matrizes de características, mantendo as informações importantes. Existem diversos tipos de *Pooling: Max, Average, Sum, etc.* O *Max Pooling* obtém o maior elemento de uma matriz em uma janela de determinado tamanho. Por exemplo, na Figura 11 há uma matriz de dimensões 4×4, janela de *Pooling* de dimensão 2×2 e *stride* 2.

Figura 11 – Exemplo de *Max Pooling* e *Average Pooling*.



Fonte: Adaptado de Ganegedara (2017)

Função de Ativação

As funções de ativação são utilizadas para inserir não linearidades na rede. Uma função bastante utilizada é a ReLu (*Rectified Linear Units*), conforme Equação 2.4. Para valores

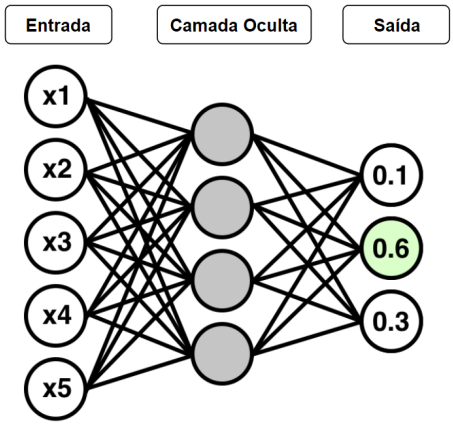
de x menores ou iguais a zero f(x) é zero, enquanto que para x maior que zero f(x) é x.

$$f(x) = Max(0, x) \tag{2.4}$$

Camada Totalmente Conectada

Após as camadas de convolução e de *pooling* frequentemente são adicionadas camadas totalmente conectadas (Figura 12). Isso significa que cada neurônio é conectado a todos os neurônios das camadas adjacentes. Antes da camada de saída os dados passam geralmente pela função *Softmax*, que determina a probabilidade dos dados de entrada pertencerem a cada uma das n classes existentes.

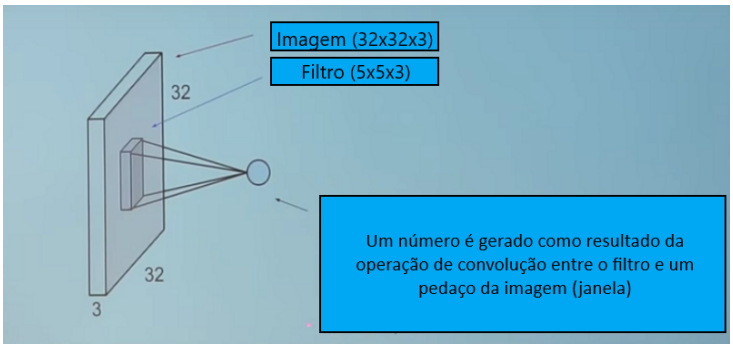
Figura 12 – Exemplo de rede neural totalmente conectada



Fonte: Adaptado de Wong (2017)

É importante ressaltar que a redes neurais convolucionais também apresentam neurônios nos procedimentos de *pooling* e convolução. Durante o processo de convolução, por exemplo, um filtro é deslocado sobre a imagem resultando na soma do produto dos elementos das matrizes, que pode ser representado pela Equação 2.3. A matriz da imagem (Figura 13) representa as entradas enquanto que o filtro representa os pesos. À medida que o filtro é deslocado sobre a imagem são gerados valores escalares que formam uma matriz de saída, na qual é aplicada uma função de ativação. Note que a representação do neurônio é bem parecida com o neurônio das redes neurais convencionais.

Figura 13 – Exemplo de convolução em uma imagem



Fonte: Adaptado de Pokharna (2016)

2.7 Bibliotecas usadas no trabalho

Existem diversas bibliotecas muito úteis na área de reconhecimento facial que facilitam e reduzem o tempo de desenvolvimento das aplicações. Neste projeto foram utilizadas três bibliotecas: Dlib, OpenCV e a Face Recognition.

2.7.1 OpenCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca multiplataforma (ITSEEZ, 2016), totalmente livre ao uso acadêmico e comercial, criada pela Intel em 2000 com o objetivo de tornar acessível a visão computacional à programadores em aplicações, contando com: processamento de imagens, análise estrutural, reconhecimento de padrões, calibração de câmera, reconstrução 3D, análise de movimento, rastreamento de objetos e a criação de GUI (Interface Gráfica do Usuário).

Esta biblioteca foi desenvolvida nas linguagens de programação C/C++. Também fornece suporte a programadores que utilizam Java, Python e Visual Basic.

2.7.2 Dlib

Dlib é uma *tookit* de *machine learning* em linguagem C++, que é utilizada tanto em aplicações industriais quanto acadêmicas. Dentre suas diversas aplicações pode ser utilizada na criação de redes neurais e classificadores, contém diversos algoritmos numéricos de otimização e permite até a criação de interfaces gráficas.

2.7.3 Face Recognition

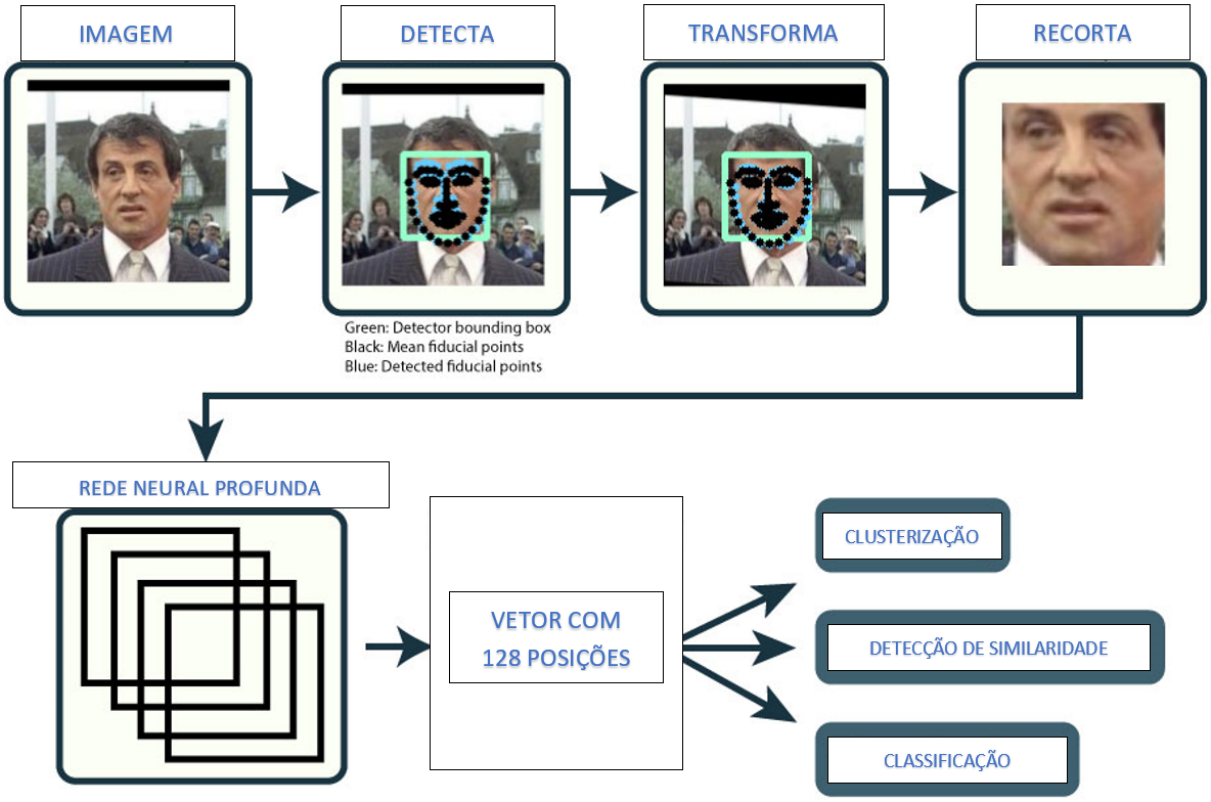
Biblioteca em Python de reconhecimento facial criada por Geitgey (2017) que utiliza um modelo de rede neural convolucional treinada usando o Dlib (KING, 2009), desta forma permitindo o chamado *transfer learning*. Para que sejam obtidos resultados satisfatórios na identificação de pessoas é necessário treinar uma rede neural com um grande volume de dados, isso demanda um grande poder computacional e tempo. Uma maneira de resolver este problema é realizar a transferência de aprendizado, onde os pesos de uma rede treinada são utilizados em outra rede. Com isso, o tempo gasto para gerar os dados de saída da rede é reduzido.

O modelo da rede utilizada neste projeto é uma modificação da ResNet-34 (HE et al., 2016), que foi campeã do desafio de reconhecimento de imagens da ImageNet (RUSSAKOVSKY et al., 2015). A biblioteca utilizou uma variação desse modelo, que teve algumas camadas removidas e o número de filtros por camada reduzidos pela metade. A rede neural conta com 29 camadas convolutivas, que realizam a extração de características das imagens. A rede foi treinada com cerca de 3 milhões de imagens de 7485 pessoas, proveniente das bases de dados *Face Scrub* (NG; WINKLER, 2014) e *VGG Face Dataset* (PARKHI; VEDALDI; ZISSERMAN, 2015). O modelo apresentou no *benchmark Labeled Faces in the Wild* (HUANG et al., 2007) precisão de 99,38% na identificação de pessoas. Segundo Geitgey (2017), a rede pode apresentar acurácia inferior para a identificação de indivíduos asiáticos e crianças, pois a base de imagens de treinamento da rede é composta por indivíduos adultos e de etnia européia. Para realizar o reconhecimento facial são realizadas as etapas (Figura 14) a seguir:

1. Procedimento para detecção da face utilizado o HOG da biblioteca Dlib;
2. Obtenção dos 68 pontos que identificam regiões de interesse na face que destacam as regiões dos olhos, nariz, boca, etc;
3. Utilização da Transformação Afim para corrigir o posicionamento da face;
4. Extração da face detectada da imagem original;
5. A captura da face alimenta uma rede neural convolucional que gera um vetor normalizado de 128 posições;
6. Aplicação de técnicas de classificação, clusterização, etc.

Para realizar a identificação de pessoas é utilizada a distância Euclidiana para verificar a similaridade entre os vetores de 128 dimensões. Ao se comparar dois vetores, caso a

Figura 14 – Procedimentos realizados para a extração das 128 medidas de uma face



Fonte: Adaptado de Amos, Ludwiczuk e Satyanarayanan (2016)

distância entre eles seja menor ou igual a 0,6 (valor de tolerância padrão adotado por Geitgey (2017)) considera-se que os dois vetores representam a face de uma mesma pessoa. É possível melhorar a precisão na identificação ao se reduzir a tolerância de 0,6 para um valor menor, ou então alterando o hiperparâmetro "número de jitters". Esse hiperparâmetro faz com que o Dlib distorça (ampliar, rotacionar, etc) uma imagem n vezes, obtendo n vetores e retornando um vetor com a média dos seus valores.

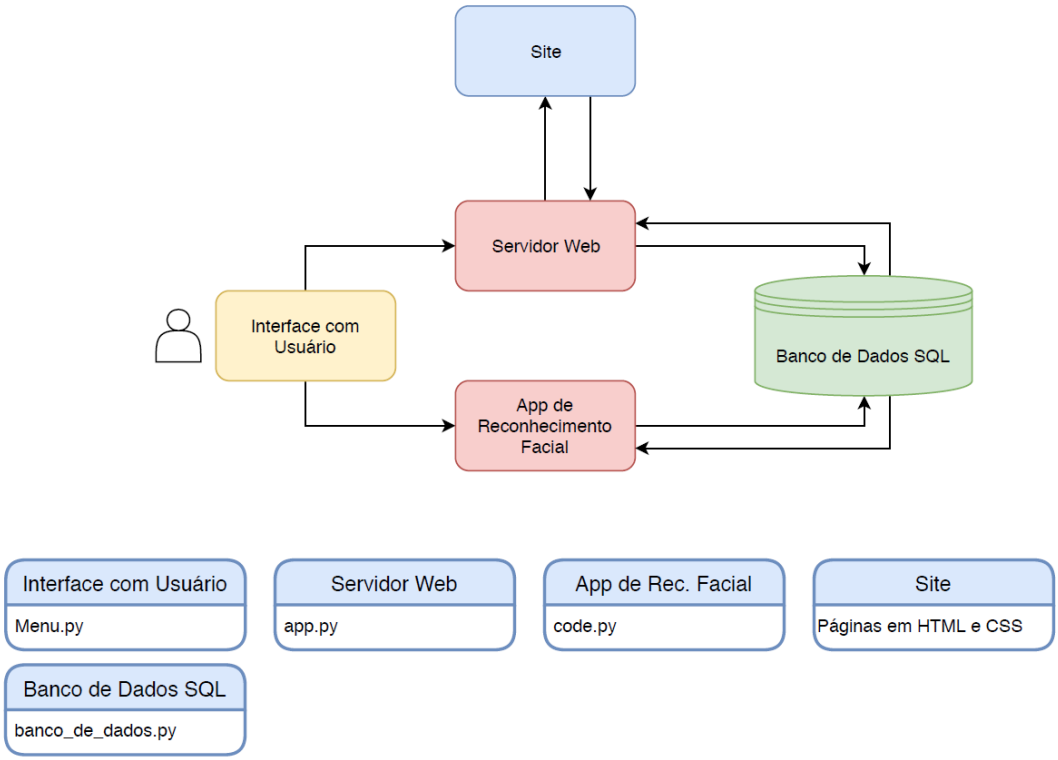
Dependendo do tamanho da face é possível que esta não seja detectada na imagem. Para resolver este problema, a biblioteca *Face Recognition* permite o ajuste do hiperparâmetro *"upsample"* que possibilita, por exemplo, a detecção de faces pequenas na imagem.

3 METODOLOGIA

Neste projeto foi desenvolvido um sistema, aplicável em ambientes residenciais, capaz de realizar o reconhecimento facial de pessoas suspeitas. As pessoas são classificadas em duas categorias: conhecidas e desconhecidas. As pessoas conhecidas são representadas, por exemplo, pelos moradores que têm suas faces armazenadas em um diretório "A", que é acessado pelo programa de reconhecimento facial durante sua inicialização. As pessoas desconhecidas passam por um processo de cadastramento, onde o algoritmo de reconhecimento facial realiza a extração de características da face e armazena a foto destas pessoas em um diretório "B".

O sistema desenvolvido (Figura 15) é composto por dois elementos principais: o aplicativo de reconhecimento facial (code.py) e o servidor web (app.py). O sistema também contém um banco de dados SQL, para armazenamento de informações relevantes, uma interface gráfica para o controle do funcionamento do sistema e um *website* para a exibição de informações aos usuários na internet.

Figura 15 – Esquematização do projeto

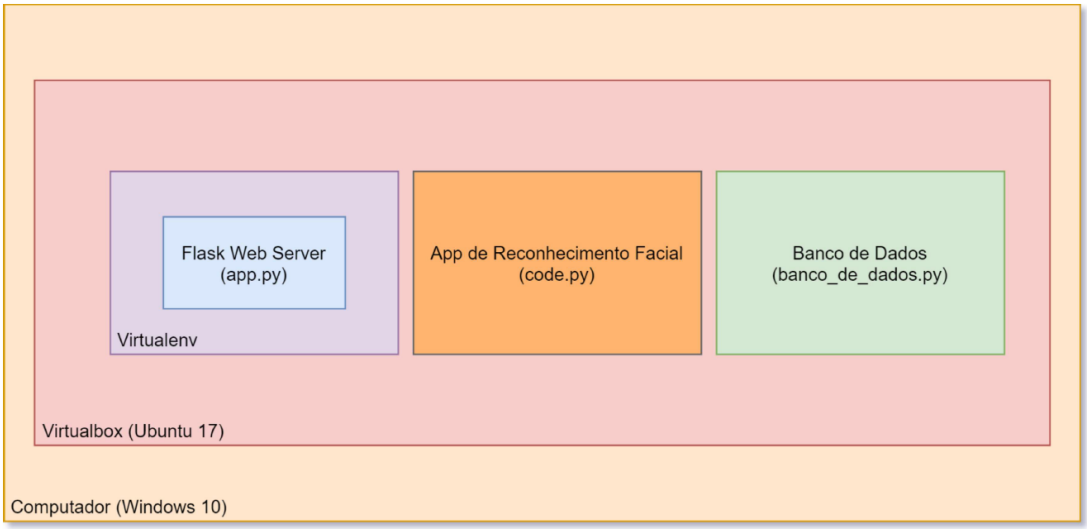


Fonte: Própria do autor

3.1 Organização do Projeto

O projeto foi desenvolvido em uma máquina virtual com o sistema operacional Ubuntu 17. Umas das vantagens de se utilizar máquinas virtuais é a capacidade de executar o sistema em qualquer outra máquina que possua um software de virtualização, isso faz com que não seja necessário instalar novamente os diversos programas e pacotes necessários ao funcionamento do projeto. O software de virtualização utilizado foi o VirtualBox (ORACLE, 2018), que foi utilizado em um computador com Windows 10.

Figura 16 – Organização do projeto



Fonte: Própria do autor

Dentro da máquina virtual (Figura 16) existe um *script* responsável pela leitura e escrita no banco de dados, outro que é utilizado para o reconhecimento facial e por fim o *script* responsável pelo *webserver*, que utiliza o Virtualenv. O Virtualenv é um ambiente virtual que permite trabalhar em um projeto com bibliotecas e dependências em diversas versões sem interferir no sistema operacional e em outros projetos. Ele não é uma máquina virtual, mas uma alternativa que permite trabalhar com bibliotecas em diferentes versões pois altera o "system path", que seria uma lista de diretórios contendo as bibliotecas que são importadas pelos programas em Python. O *Flask Web Server*, exibido na Figura 16, será detalhado na Seção 3.4.

O sistema de diretórios do projeto foi estruturado conforme a Figura 17, onde são exibidos arquivos e pastas. O diretório *RECFACE* contém arquivos relacionados ao processo de reconhecimento facial e o registro de informações no banco de dados, enquanto que *GSERVER* contém arquivos relacionados ao funcionamento do servidor. Os arquivos *menu.py*, *prog1.sh* e *prog2.sh* são responsáveis pela exibição de uma interface gráfica para

o usuário, que será detalhada posteriormente.

Figura 17 – Estruturação de pastas e arquivos

```
PG/
├── RECFACE/
│   └── ...
├── GSERVER/
│   └── ...
├── menu.py
├── prog1.sh
└── prog2.sh
```

Fonte: Própria do autor

3.2 Programação e Banco de Dados

O projeto foi desenvolvido utilizando principalmente a linguagem Python. Para o desenvolvimento do site do projeto foram utilizadas as linguagens HTML, CSS e Javascript. Para a leitura e escrita no banco de dados MySQL foi utilizada a linguagem SQL.

O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto utilizado em grande parte das aplicações gratuitas para gerir suas bases de dados. O serviço utiliza a linguagem SQL (*Structure Query Language* – Linguagem de Consulta Estruturada), que é a linguagem mais popular para inserir, acessar e gerenciar informações armazenadas num banco de dados.

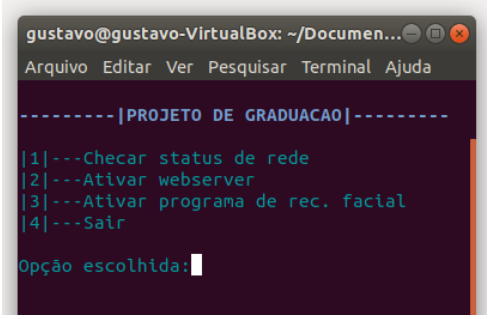
3.3 Interface Inicial

Ao se executar o código *menu.py* é exibido um menu (Figura 18) onde aparecem as seguintes opções:

1. Checar o Status da Rede: verifica se computador possui conexão com a internet;
2. Ativar *webserver*: ativa o servidor que disponibiliza o *website* do projeto na internet;
3. Ativar programa de rec. facial: realiza reconhecimento facial e armazena informações em banco de dados;
4. Sair: fecha aplicação.

Os *scripts prog1.sh* e *prog2.sh* são executados a partir do arquivo *menu.py* e são utilizados apenas para a abertura de janelas do terminal no Ubuntu. O *script prog1.sh* abre uma

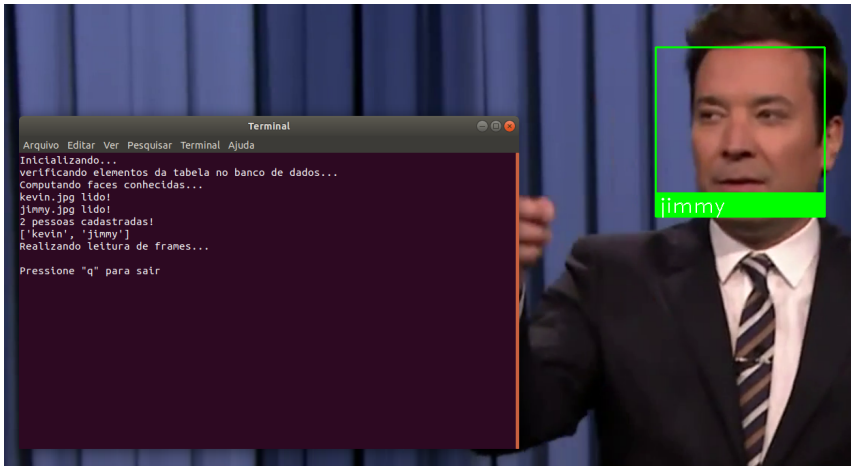
Figura 18 – Menu principal da aplicação



Fonte: Própria do autor

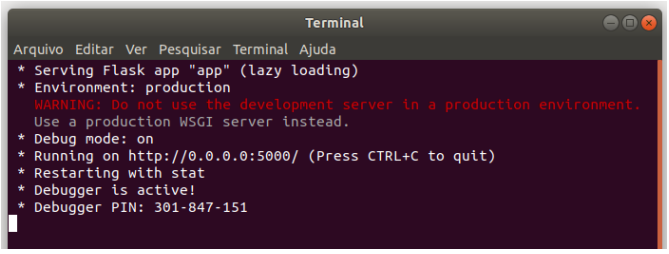
janela que exibe os *frames* obtidos pela câmera e informações adicionais sobre as faces detectadas (Figura 19), enquanto que *prog2.sh* abre uma janela que exibe informações sobre o servidor (Figura 20) e os acessos dos usuários.

Figura 19 – Janela com informações sobre o funcionamento do programa de reconhecimento facial



Fonte: Própria do autor

Figura 20 – Janela com informações sobre atividades no servidor



Fonte: Própria do autor

3.4 Sevidor *Web*

O servidor web (App.py) foi elaborado utilizando Flask, um micro-framework em Python que simplifica a criação de servidores. O servidor fornece informações aos usuários quando solicitado por meio das requisições HTTP. Quando um cliente digita em seu navegador o site ou domínio desejado, o aplicativo python App.py fica responsável por gerar uma página web para o usuário dependendo do endereço digitado. Rotas que associam o domínio digitado com uma página *html* são estabelecidas. Por exemplo, supondo que o usuário digite "www.meusite.com" o servidor deve retornar a página *menu.html* enquanto que se for digitado "www.meusite.com/login" esse deve retornar a página *login.html*.

O *script* é composto pelas seguintes funcionalidades:

- Rotas: determina qual página será retornada ao cliente baseando-se no domínio digitado;
- Leitura de bancos de dados: realiza leitura das tabelas do banco de dados que são incorporados na página web quando renderizada;
- hiperparâmetros de conectividade: são definidos o endereço de IP do servidor, a porta utilizada na aplicação e configurações de segurança.

Uma página web é composta por diversos elementos, que são unidos no processo de renderização. Os arquivos em HTML são a *backbone* da página, contendo texto e ligação com outros tipos de arquivo. Já os arquivos em CSS basicamente definem o design da página, esquemas de cores, tipo de fonte, etc. Os efeitos de animação são criados pelos arquivos em Javascript. Os arquivos são estruturados da seguinte forma (Figura 21):

Figura 21 – Diretório contendo arquivos do servidor

```
GSERVER/
├── amb_virtual/
│   └── ...
└── FlaskApp/
    ├── App.py
    ├── templates/
    │   ├──index.html
    │   ├──index2.html
    │   ├──login.html
    └── static/
        ├── css/
        │   └── ...
        ├── faces/
        │   └── ...
        ├── img/
        │   └── ...
        └── js/
            └── ...
```

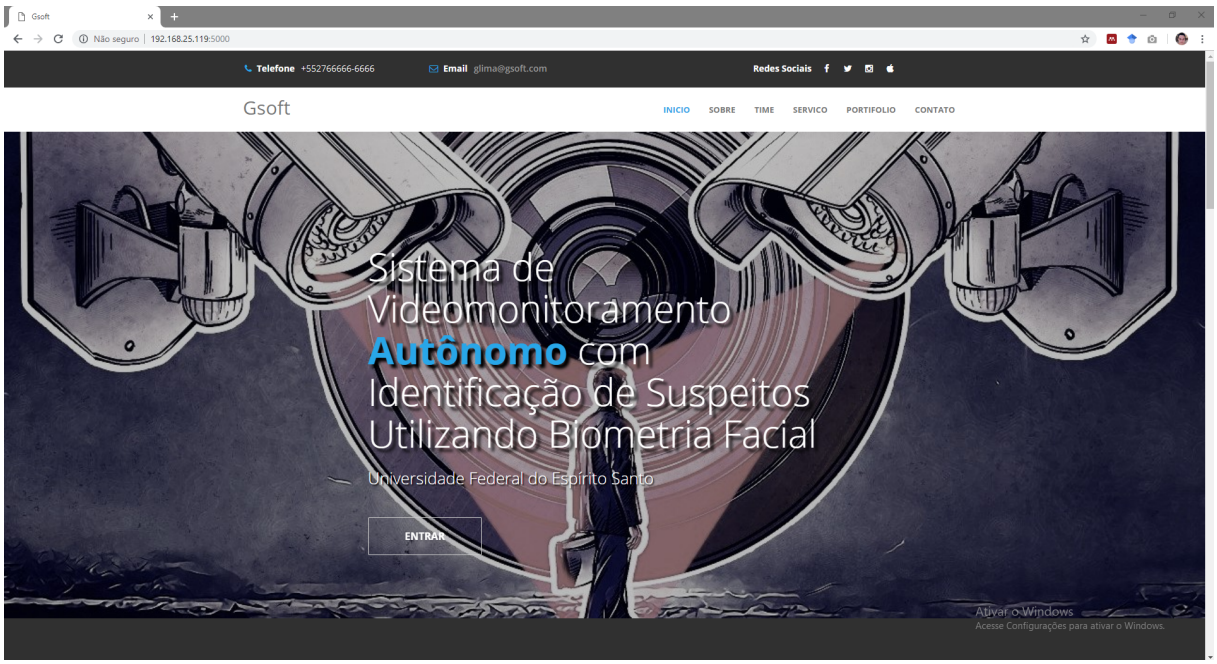
Fonte: Própria do autor

- **amb_virtual:** pasta criada pelo Virtualenv contendo todas a bibliotecas Python que são utilizadas pelo servidor web;
- **templates:** contém páginas em HTML;
- **css:** contém arquivos de estilo;
- **img:** pasta contendo imagens que são utilizadas no website;
- **faces:** pasta onde são salvas as capturas das faces dos indivíduos desconhecidos;
- **js:** contém arquivos em Javascript.

O *website* foi desenvolvido com o intuito de exibir informações do projeto e resultados gerados pelo script de reconhecimento facial. Foram criadas as seguintes páginas:

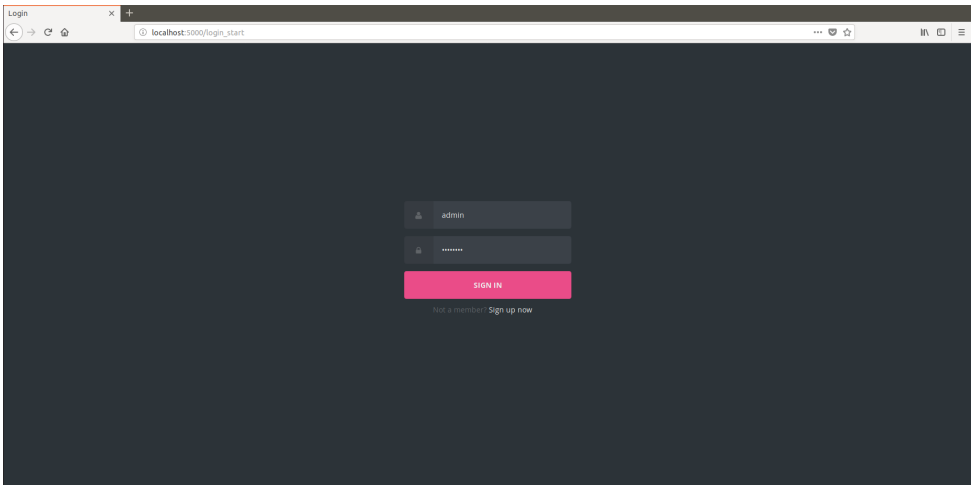
- **index.html:** é a página principal do site. Página que exibe informações sobre o projeto (Figura 22);
- **login.html:** página de login que permite acesso à página index2.html (Figura 23);
- **index2.html:** página que exibe os registros do banco de dados, contendo data e hora das ocorrências, além das faces de indivíduos suspeitos detectadas pelo sistema. Também contém um gráfico exibindo o número de suspeitos detectados em função do tempo (Figura 24).

Figura 22 – Página inicial do *website* (index.html)



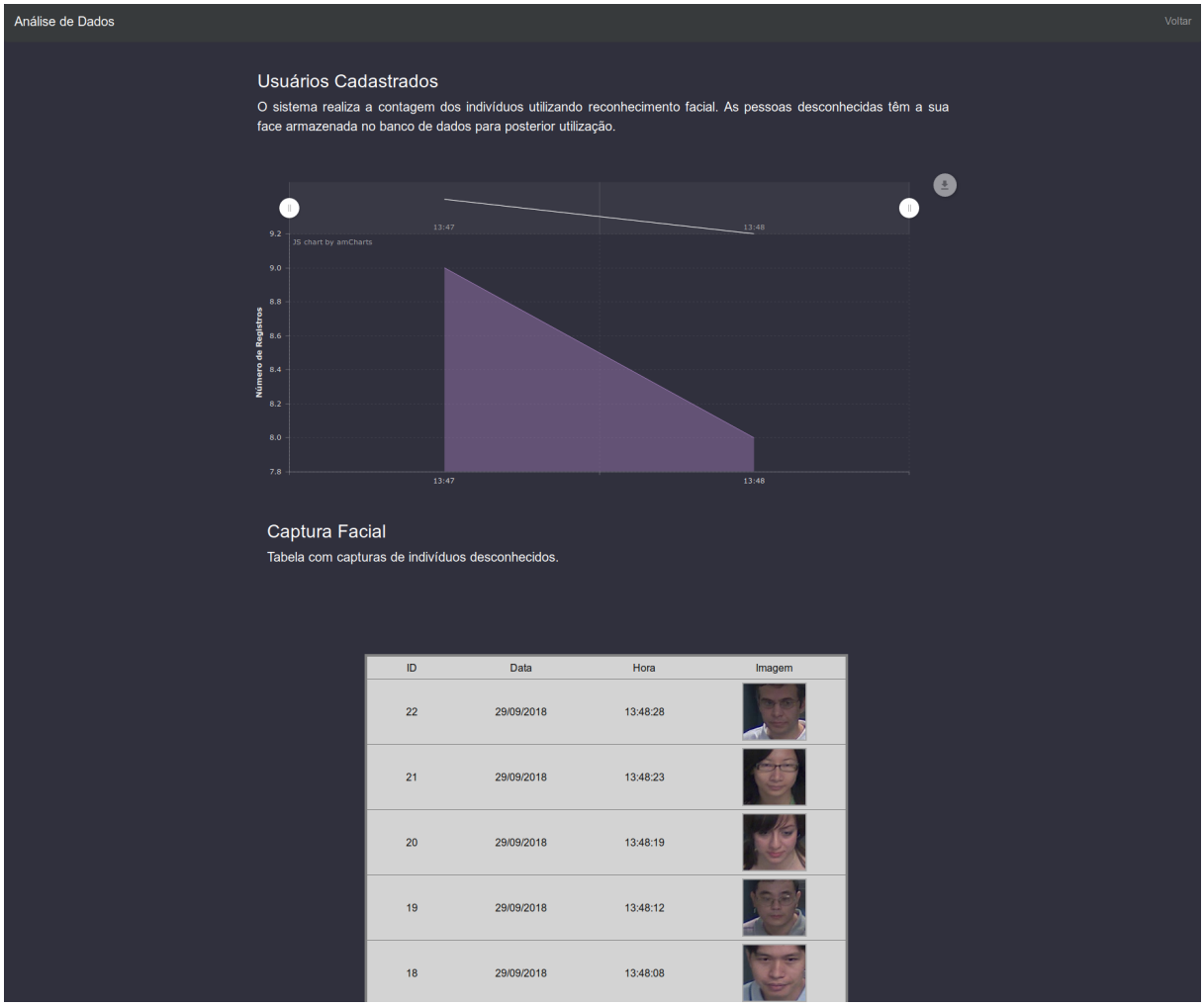
Fonte: Própria do autor

Figura 23 – Página de *login* (login.html)



Fonte: Própria do autor

Figura 24 – Página onde são exibidos os resultados (index2.html)



Fonte: Própria do autor

3.5 Reconhecimento Facial

O procedimento de reconhecimento facial é realizado utilizando o *script* "code.py", que realiza os procedimentos de detecção e reconhecimento facial, além da leitura e escrita nas tabelas do banco de dados (Figura 25). As pessoas são separadas em duas categorias: conhecidas e desconhecidas (suspeito). Os indivíduos conhecidos são, por exemplo, os moradores de uma residência ou condomínio, que frequentemente acessam o local. Já as pessoas desconhecidas são consideradas uma potencial ameaça e têm sua biometria facial salva no sistema ao acessarem um ambiente sob videomonitoramento.

Na etapa de inicialização (Figura 25), é criada a lista chamada de "LISTA_PESSOAS" que contém as medidas dos usuários. Como foi descrito na Seção 2.7.3, a rede neural convolucional utilizada neste projeto gera, para cada face, um vetor de 128 posições. Considerando n pessoas conhecidas, inicialmente temos $LISTA\_PESSOAS = [C_1, C_2, ..., C_n]$.

Ao se realizar a conexão com a *webcam* inicia-se a leitura dos *frames*, após isso, é realizada a detecção facial. Neste processo é possível identificar em que local da imagem foram detectadas faces, podendo ser detectada mais de uma face. As faces detectadas são recortadas da imagem e alimentam a rede neural que fornece as 128 medidas como saída. Essas medidas são então comparadas com as medidas das pessoas conhecidas utilizando a distância euclidiana como métrica de similaridade. A distância euclidiana entre os pontos $P = (p_1, p_2, ..., p_n)$ e $Q = (q_1, q_2, ..., q_n)$ num espaço com n dimensões é dado pela Equação 3.1.

$$D_{euclidiana}(P, Q) = \sqrt{\sum_{k=1}^{n} |p_k - q_k|^2} \tag{3.1}$$

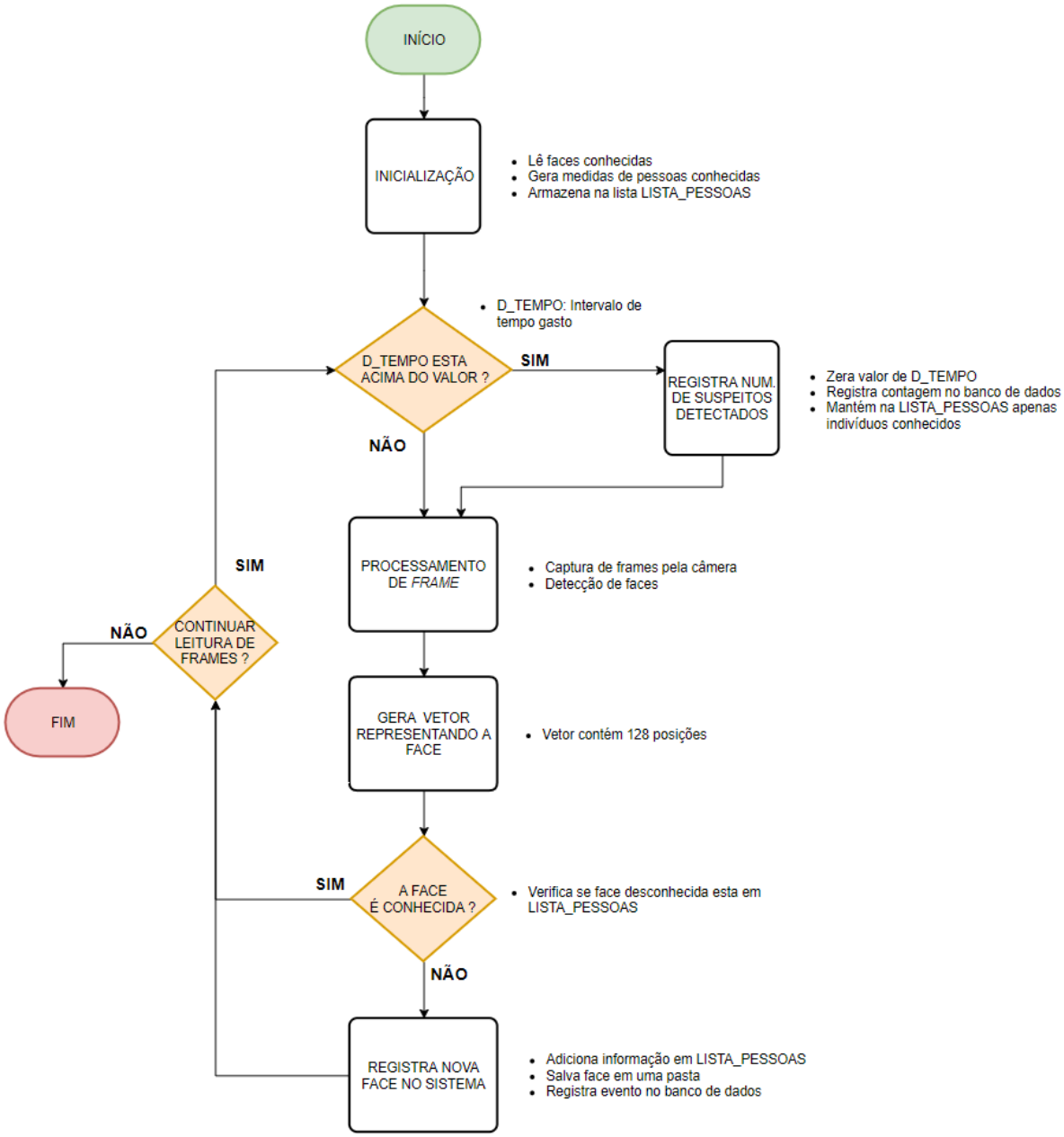
Se identificado determinado grau de similaridade, a pessoa é classificada como conhecida, caso contrário é gerado um rótulo para essa pessoa e suas medidas são armazenadas na "LISTA_PESSOAS". Além disso, a captura da face é armazenada em uma pasta e o evento é registrado no banco de dados.

Ao finalizar o processamento do *frame* é verificado se o usuário pressionou alguma tecla para finalizar o *script*, caso contrário é realizada a leitura do próximo *frame*.

A cada intervalo de tempo (horas, minutos, etc) é realizada a contagem do número de indivíduos desconhecidos registrados no sistema. Por exemplo, supondo que m indivíduos suspeitos foram detectados pelo sistema $LISTA\_PESSOAS = [C_1, C_2, ..., C_n, D_1, D_2, ..., D_m]$

onde "C" indica as pessoas conhecidas, e "D" indica as desconhecidas (suspeitos). Os m indivíduos são contabilizados (informação é registrada no banco de dados), o contador é zerado e o registro destas pessoas são removidos da lista, ou seja, $LISTA\_PESSOAS = [C_1, C_2, ..., C_n]$.

Figura 25 – Lógica de funcionamento do sistema



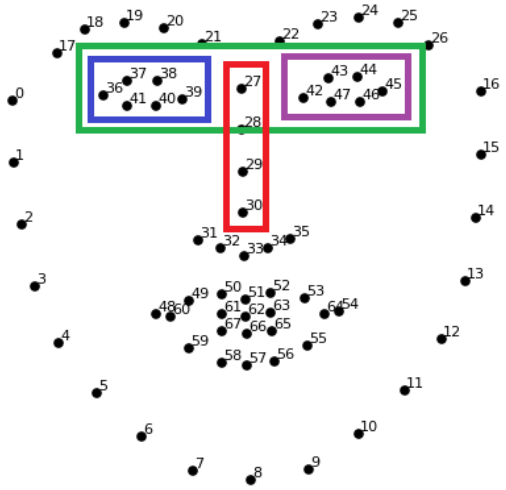
Fonte: Própria do autor

3.6 Utilização do *Face Landmark*

Para reduzir os "ruídos", ou seja, a identificação incorreta dos indivíduos devido a má qualidade dos dados, é preciso realizar uma filtragem das faces antes armazená-las no sistema. O procedimento de filtragem tem como finalidade obter imagens onde os indivíduos apresentam um bom posicionamento da face (posicionamento frontal) e um grau aceitável de abertura dos olhos. Conforme explicado no capítulo anterior, o *Face Landmark* obtém 68 pontos que identificam regiões de interesse na face. Por meio destes pontos é possível realizar a filtragem das imagens conforme os procedimentos a seguir:

- Com os valores dos pontos 27 e 30 (Figura 26) é obtida uma reta, cujo coeficiente angular é utilizado para identificar o posicionamento do nariz. Valores elevados do coeficiente angular indicam um bom posicionamento do nariz e indicam o posicionamento frontal da face. Este hiperparâmetro utilizado se chama *coef_ang_nose*;
- A partir dos pontos 37 e 43 é possível verificar a inclinação do rosto. Valores baixos do coeficiente angular da reta, formada por esses pontos, indicam um bom posicionamento da face. Este hiperparâmetro a ser utilizado se chama *coef_ang_eye*;
- Para determinar se os olhos direitos estão abertos é calculada a distância entre os pontos 43 e 47. Esta é dividida pela distância entre os pontos 42 e 45, gerando desta forma um valor normalizado que indica o nível de abertura dos olhos (hiperparâmetro *dist_right_eye*). O processo é realizado de forma análoga no olho esquerdo, onde a distância entre os pontos 37 e 41 é dividida pela distância entre os pontos 36 e 39 (hiperparâmetro *dist_left_eye*).

Figura 26 – Conjunto de pontos de interesse



Fonte: Adaptado de Geitgey (2017)

3.7 Banco de dados

O banco de dados utilizado é composto de duas tabelas, que armazenam informações que são utilizadas tanto pelo *script* de reconhecimento facial, quanto pelo servidor web.

3.7.1 A tabela *gsoft_events*

Quando uma face desconhecida é detectada, para registrá-la no sistema são realizadas as seguintes etapas: criação de um rótulo (nome) para a face, inserção de informações na tabela do banco de dados e o salvamento da face no diretório "faces" (Figura 21). O rótulo é criado ao se verificar o maior valor da coluna "id_evento" (Figura 27), gerando dessa forma um nome no seguinte formato:

"usuário_" + <maior valor da coluna "id_evento" + 1>

Por exemplo, na tabela abaixo, na coluna "imagem" o próximo nome será "usuário_7". A tabela é composta por quatro colunas:

- id_evento: contém número de identificação sequencial único para a linha criada na tabela;
- data_evento: registra data do evento no formato DD/MM/AAAA;
- hora_evento: registra o tempo em horas, minutos e segundos;
- imagem: contém o nome do arquivo salvo na pasta "faces" (Figura 21) com a extensão *.jpg*.

Figura 27 – Tabela *gsoft_events*

id_evento	data_evento	hora_evento	imagem
1	16/11/2018	10:32:23	usuario_1.jpg
2	16/11/2018	10:32:35	usuario_2.jpg
3	16/11/2018	10:32:40	usuario_3.jpg
4	16/11/2018	10:32:47	usuario_4.jpg
5	16/11/2018	10:32:55	usuario_5.jpg
6	16/11/2018	10:33:01	usuario_6.jpg

Fonte: Própria do autor

3.7.2 A tabela *num_pessoas*

Esta tabela registra a contagem do número de suspeitos detectados pelo sistema em um intervalo de tempo. Apesar da similaridade das colunas data_evento e hora_evento com as colunas da tabela *gsoft_events*, elas apresentam uma função diferente, que é apenas computar quando os valores da coluna "contagem" foram adicionados no banco de dados. A tabela é composta por quatro colunas:

- id_evento: contém número de identificação sequencial único para a linha criada na tabela;
- data_evento: registra data no formato DD/MM/AAAA;
- hora_evento: registra o tempo em horas, minutos e segundos;
- contagem: contém o número de faces desconhecidas detectadas em um intervalo de tempo. Na Figura 28 observe, por exemplo, que para as três primeiras linhas da tabela o intervalo de tempo entre os registros foi de 1 minuto, contudo este intervalo é bem pequeno e recomenda-se adotar um tempo maior (hora em hora, uma vez ao dia, etc) entre os registros das contagens.

Figura 28 – Tabela *num_pessoas*

id_evento	data_evento	hora_evento	contagem
37	28/07/2018	16:13:24	1
38	28/07/2018	16:14:24	1
39	28/07/2018	16:15:24	1
40	10/09/2018	21:34:02	2

Fonte: Própria do autor

4 EXPERIMENTOS E RESULTADOS

Neste capítulo são realizados experimentos com o objetivo de avaliar o algoritmo de identificação e reconhecimento facial. Inicialmente as informações sobre a base de dados utilizada são detalhadas e depois são realizados diferentes testes com o objetivo de avaliar o desempenho do algoritmo. Por fim, são discutidas as dificuldades encontradas durante os testes e o desenvolvimento do projeto.

4.1 Bases de dados

Para verificar o funcionamento do sistema foi utilizada a base de dados "Chokepoint Dataset" (WONG et al., 2011), que contém sequências de vídeo que tentam representar um sistema de videomonitoramento real. O *dataset* tem, no total, 48 sequências de vídeo e 64204 imagens de faces. Para os experimentos foram utilizadas duas sequências de vídeo (Figura 29), contendo cada uma, 25 pessoas (19 homens e 6 mulheres) acessando ambientes internos. Os vídeos apresentam resolução de 800×600 pixels a uma taxa de 30 quadros por segundo. Nas sequências de vídeo as faces dos indivíduos não eram rotuladas. O primeiro vídeo (ambiente A) tem duração de 1min18s e contém 2369 *frames*. O segundo vídeo (ambiente B) tem duração de 1min58s e contém 3558 *frames*. Foram utilizadas três câmeras para obter o registro em diferentes ângulos (Figura 30), entretanto, neste projeto apenas os dados das câmeras em que os indivíduos apresentam o posicionamento frontal da face foram utilizados.

Figura 29 – *Ambientes analisados. Esquerda: ambiente A. Direita: ambiente B*

Fonte: Wong et al. (2011)

Figura 30 – Alguns *frames* do *ChokePoint Dataset*

Fonte: Wong et al. (2011)

4.2 Análise do *ChokePoint Dataset*

Foram realizados dois experimentos com o objetivo de verificar o processo de reconhecimento facial em um ambiente interno e com um fluxo moderado de pessoas. No primeiro experimento foi utilizado o vídeo do "ambiente A", no qual foram analisados a detecção facial e dois métodos de classificação. O segundo experimento analisou o procedimento de identificação de pessoas no "ambiente B", ao se utilizar dados faciais de cinco indivíduos extraídos do "ambiente A".

Os experimentos foram realizados em uma máquina virtual contendo o sistema operacional Ubuntu 17 (ver Seção 3.1) com as seguintes especificações:

- Sistema Operacional: Ubuntu 17;
- Processador: Intel Core i7-7700HQ 2.80GHz (2 núcleos utilizados);
- Memória RAM: 4GB;
- Memória de Vídeo: 128MB;
- HD: Tamanho dinâmico;
- Rede: Configuração do adaptador em modo Bridge.

A máquina hospedeira, que contém o programa de virtualização Virtualbox, tem as seguintes especificações:

- Sistema Operacional: Windows 10;

- Processador: Intel Core i7-7700HQ 2.80GHz;
- Memória RAM: 16GB;
- GPU: Nvidia Geforce GTX 1060 (6GB);
- HD: 1TB.

4.2.1 Experimento 1

Foi utilizada uma sequência de vídeo do "ambiente A" (Figura 29), com 2369 *frames*, de 25 pessoas (Figura 32) com o objetivo de avaliar os processos de detecção e reconhecimento facial. No vídeo, os indivíduos, um de cada vez, acessavam uma sala ao passar por um portal, que tinha uma câmera instalada.

Detecção facial

Para o experimento, as faces detectadas durante a leituras dos *frames* do vídeo foram armazenadas em uma pasta. Com esses dados foi realizada uma contagem do número de indivíduos detectados pelo sistema. No experimento foram detectadas as faces dos 25 indivíduos. Note que no vídeo, apenas uma face era detectada por *frame* (Figura 31), contudo, o sistema é capaz de detectar múltiplas faces em um mesmo *frame*.

Figura 31 – Exemplo de detecção facial durante os experimentos

Fonte: Própria do autor

Reconhecimento facial

O experimento teve como objetivo verificar a capacidade de reconhecimento facial e registro de suspeitos no sistema, além de analisar a acurácia do classificador utilizado.

Na primeira etapa, todos os 25 indivíduos (Figura 32) foram considerados desconhecidos, ou seja, não apresentavam registro biométrico no sistema. Isso significava que, inicialmente, "LISTA_PESSOAS" (Seção 3.5) era uma lista vazia. A medida que as faces desconhecidas eram identificadas, suas medidas (vetor de 128 posições) eram adicionadas a "LISTA_PESSOAS", portanto, caso ocorresse uma nova aparição do indivíduo registrado este já seria considerado identificado, não sendo necessário realizar os procedimentos de registro e cadastro no sistema novamente.

Figura 32 – Faces das 25 pessoas do *ChokePoint Dataset*

Fonte: Própria do autor

Desta forma, o algoritmo utilizado deveria detectar as faces e registrá-las. Conforme a Seção 3.6, para que uma face fosse registrada no sistema ela deveria ser de boa qualidade, ou seja, deveria apresentar um posicionamento frontal e um desejável grau de abertura dos olhos. Durante os experimentos foram utilizados os hiperparâmetros conforme a Tabela 1.

O processo de identificação funciona a partir da comparação entre os vetores (128 posições ou medidas) das pessoas desconhecidas com os vetores presentes em "LISTA_PESSOAS". Para realizar a comparação é utilizada a distância euclidiana, que gera um valor normalizado da proximidade entre os vetores. Durante a comparação, se o valor da distância for inferior

Tabela 1 – Requisitos para registro da face no sistema

Hiperparâmetros	Valor
coef_ang_nose	>=20
coef_ang_eye	<0.04
dist_right_eye	>0.15
dist_left_eye	>0.15

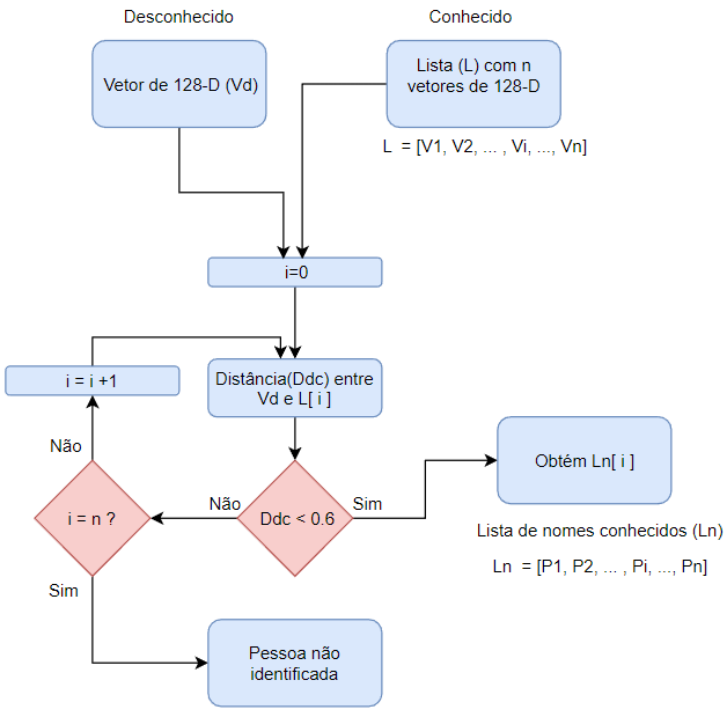
a 0,6 (valor padrão, que pode ser alterado) o sistema considera que o vetor é de uma pessoa registrada, caso contrário, ela é desconhecida e deve passar pelo processo de registro, onde o vetor é adicionado a "LISTA_PESSOAS". Cada indivíduo identificado tem sua face armazenada no diretório "../RECFACE/pessoas" (Figura 17). Apenas uma face é armazenada por indivíduo.

Foram utilizados dois métodos durante a comparação entre os vetores (Figuras 33 e 34): Os métodos "A" e "B". No Método "A" ocorre a comparação entre o vetor representando as medidas da pessoa a ser identificada (V_d) e os elementos de "LISTA_PESSOAS", representado na Figura 33 pela letra L. Os vetores presentes na lista L são comparados, um a um com V_d até que o valor da distância entre eles, V_{dc}, seja menor que 0,6 (tolerância), neste momento a busca termina. No Método "B", V_d é comparado com todos os elementos da lista L, diferente do Método "A". O índice i da posição com menor valor no vetor V_{dc} é utilizado para obter $L_n[i]$, que é o menor valor de V_{dc}. Foi constatado que o Método "A" é menos dispendioso computacionalmente, entretanto, ele é menos preciso que o Método "B".

Na segunda etapa do experimento foi feita uma análise para verificar se as faces eram classificadas corretamente. As k pessoas identificadas na etapa anterior foram adicionadas a "LISTA_PESSOAS" durante a inicialização do programa (esses indivíduos foram considerados conhecidos) e o vídeo foi analisado novamente. O objetivo foi verificar se as faces detectadas em cada *frame* eram identificadas como sendo da pessoa correta. É importante ressaltar que durante a inicialização, o programa utiliza as faces salvas no diretório "../RECFACE/pessoas" para gerar os vetores associados a cada indivíduo e os adiciona a "LISTA_PESSOAS".

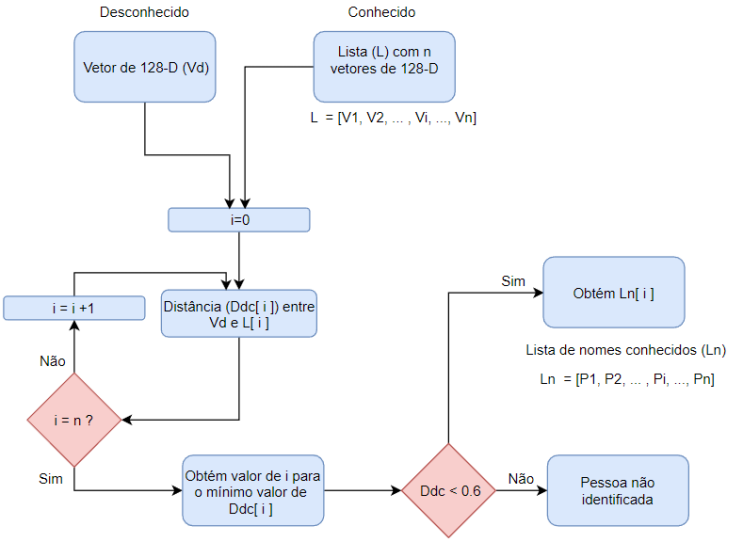
Para avaliar o desempenho da classificação dos indivíduos, foi utilizada a matriz de confusão. Esse tipo de matriz é estruturada da seguinte forma: os rótulos na linha vertical representam as classes reais, enquanto que os rótulos na linha horizontal representam as classes que foram preditas pelo classificador. Os elementos da matriz indicam o número de faces associadas a cada classe. O classificador, que pode ser o método A ou B, utiliza a distância euclidiana para comparar os vetores que representam as faces, associando a elas um rótulo. Se o rótulo real for o mesmo que o predito, a classificação está correta,

Figura 33 – Fluxograma do Método A



Fonte: Própria do autor

Figura 34 – Fluxograma do Método B



Fonte: Própria do autor

caso contrário houve erro. Por exemplo, supondo que uma face fosse da classe “P1” e o classificador a considerasse como pertencente à classe “P2”, nesse caso haveria um erro de predição.

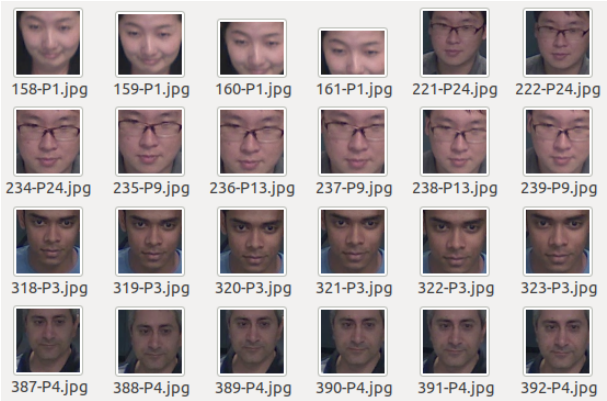
Os valores para a elaboração da matriz foram obtidos através da análise de dados similares

aos apresentados na Figura 35, que mostra algumas faces capturadas dos *frames* do vídeo. Cada imagem continha um rótulo com o seguinte formato:

"<número do *frame*> + "-" + <nome do usuário>.jpg "

Quando uma face não era devidamente identificada, o campo "nome do usuário" passava a se chamar *unknown* (desconhecido), semelhante ao visto na Figura 31. Para montar a matriz de confusão as faces foram comparadas com os seus rótulos associados. Uma matriz com valores mais elevados na diagonal principal indicavam um bom desempenho do classificador, enquanto que matrizes mais dispersas e com valores pequenos na diagonal principal mostravam uma menor acurácia do classificador.

Figura 35 – Tipo de dados utilizados para gerar a matriz de confusão



Fonte: Própria do autor

Classificação utilizando o Método A

Durante o experimento foram identificados e cadastrados 22 dos 25 indivíduos. Os indivíduos P2, P14 e P17 (Figura 32) foram detectados, mas não foram cadastrados por não apresentarem os requisitos da Tabela 1. Dos 2369 *frames* do arquivo de vídeo, 615 continham faces identificáveis, ou seja, que poderiam passar pelo processo de classificação. Apesar de ter identificado 22 indivíduos, este método apresentou uma precisão variável, pois a ordem dos elementos da lista L interferiam nos valores da matriz de confusão (Tabela 2).

Como pode-se ver na Tabela 2, P2 apresentou uma predição mais dispersa, que expõe as limitações do modelo da rede neural utilizada para indivíduos de etnia asiática conforme foi explicado na Seção 2.7.3. Já em P14 existe uma quantidade reduzida de *frames* associados. Infere-se que houve dificuldade na detecção facial devido ao posicionamento

facial desfavorável. P17 apresentou problema de identificação ao ser associado aos indivíduos P8 e P16.

Para calcular a acurácia foi realizada a divisão entre a soma dos elementos da diagonal principal da matriz (Tabela 2) pela soma total dos elementos da matriz. Foi obtida acurácia de 76,42%. O tempo de execução do programa foi de 2min16s, ou seja, foram analisados 17,42 *frames* por segundo.

Tabela 2 – Matriz de confusão dos 25 indivíduos (Método A)

Classe Real \ Classe Prevista	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25
P1	21	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0
P3	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P4	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
P5	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P7	11	0	0	0	0	0	0	0	0	0	0	8	1	0	0	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P9	8	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	19	0	0	0
P10	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P11	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P12	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
P13	5	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	1	0	0	0	0	0	0
P16	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
P17	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0
P20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0
P21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0
P22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0
P23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0
P24	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P25	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	29

Classificação utilizando o Método B

Durante o experimento foram identificados e cadastrados 22 dos 25 indivíduos. Os indivíduos P2, P14 e P17 (Figura 32) foram detectados, mas não foram cadastrados pelo mesmo motivo descrito no experimento com o Método A. Na segunda etapa da análise os 22 indivíduos foram adicionados à lista de "pessoas conhecidas" e foi executado novamente o algoritmo com o objetivo de analisar a quantidade de *frames* que foram identificados corretamente. Dos 2369 *frames* do arquivo de vídeo, 613 continham faces identificáveis.

Foi observado uma maior quantidade de predições corretas, pois a matriz de confusão apresentava valores maiores na diagonal principal. P2, P14 e P17 apresentaram os mesmos problemas do experimento com o Método A.

A partir da matriz de confusão (Tabela 3) foi calculada a acurácia, que consiste na razão entre o número de predições corretas (soma dos elementos da diagonal principal) e o número

total de predições. Foi obtida acurácia de 91,84%. O tempo de execução do programa foi de 2min20s, ou seja, foram analisados 16,92 *frames* por segundo.

Tabela 3 – Matriz de confusão dos 25 indivíduos (Método B)

Classe Real \ Classe Prevista	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25
P1	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	9	0	0	0	6	0	0	0	0	0	0	0	0	1	0	9	0
P3	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P4	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P5	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P9	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P11	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P12	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0
P16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0
P17	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0
P20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0
P21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0
P22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0
P23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0
P24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0
P25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35

O Método B se mostrou mais viável por apresentar resultados satisfatórios tanto na acurácia quanto em relação à taxa de processamento de *frames* que teve pouca variação em relação ao Método A.

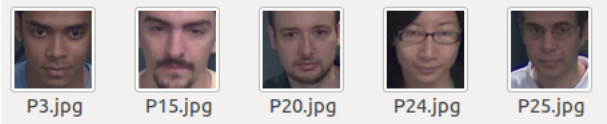
4.2.2 Experimento 2

O objetivo do experimento foi verificar o desempenho do sistema ao se utilizar capturas faciais de pessoas obtidas no "ambiente A" para a identificação dessas mesmas pessoas no "ambiente B". A sequência de vídeo do "ambiente B" (Figura 29) é composta por 3558 *frames*, de 25 pessoas (Figura 32) . No vídeo, os indivíduos, um de cada vez, saiam de uma sala passando por um portal, que tinha uma câmera instalada.

O primeiro passo realizado foi extrair as faces de cinco pessoas identificadas no "ambiente A". As faces dos indivíduos (P3, P15, P20, P24 e P25) foram identificadas e salvas, conforme a Figura 36. O programa analisou o arquivo de vídeo do "ambiente B" considerando os cinco indivíduos "conhecidos" e os demais como "desconhecidos". O Método "B" foi utilizado durante a etapa de classificação, por ter apresentado melhores resultados no experimento 1.

Todos os 25 indivíduos tiveram suas faces detectadas pelo sistema. Foram identificadas 491 faces durante a execução do programa, sendo que 153 não eram identificáveis (faces

Figura 36 – Faces dos cinco indivíduos conhecidos obtidas do ambiente "A"



Fonte: Própria do autor

detectadas possuíam rótulo "unknown"). Os resultados da análise foram organizados na Tabela 4 da seguinte forma:

- indivíduo: nome (rótulo) do indivíduo;
- faces: número de faces do indivíduo (a soma dos valores das colunas "faces corretas", "faces incorretas" e "faces não identificadas");
- faces corretas: número de faces do indivíduo que foram identificadas corretamente;
- faces incorretas: número de faces do indivíduo que não foram classificadas corretamente;
- faces de outros: número de faces de outras pessoas que receberam o rótulo do indivíduo;
- faces não identificadas: número de faces do indivíduo que não puderam ser identificadas (receberam o rótulo "unknown").

Tabela 4 – Número de faces identificadas e não identificadas no experimento 2

indivíduo	faces	faces corretas	faces incorretas	faces de outros	faces não identificadas
P3	18	18	0	0	0
P15	25	25	0	0	0
P20	26	24	0	0	2
P24	13	13	0	69	0
P25	26	24	0	2	2

De acordo com a Tabela 4, o indivíduo P24 teve todas as suas faces identificadas corretamente, contudo, foi verificada a dificuldade na identificação de asiáticos devido ao problema descrito na Seção 2.7.3, pois outras pessoas (P1, P2, P7, P9, P12, P13 e P21) tiveram algumas de suas faces identificadas como sendo P24. Isto explica o valor elevado da coluna "faces de outros". Os indivíduos P20 e P25 tiveram alguns registros na coluna "faces não identificadas", entretanto, como os valores foram pequenos eles foram considerados aceitáveis. Nenhum dos cinco indivíduos teve suas faces identificadas de forma incorreta conforme a coluna "faces incorretas".

Durante este experimento foi constatado que faces capturadas em um vídeo podem ser utilizadas na identificação de pessoas em outros vídeos.

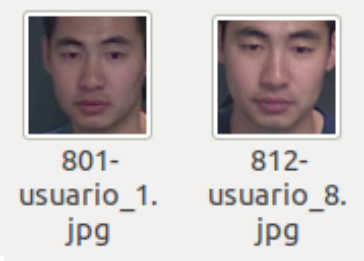
4.3 Dificuldades Encontradas

Desenvolver um sistema de reconhecimento facial eficiente é uma tarefa complicada, já que diversos fatores podem atrapalhar no processo de identificação. O posicionamento das câmeras e as condições de iluminação do ambiente podem interferir na qualidade dos dados. Além disso, capturar faces em diferentes posições podem gerar erros de classificação. Nesta Seção são apresentadas as principais dificuldades encontradas durante o desenvolvimento do projeto.

4.3.1 Múltiplos registros e filtragem dos dados armazenados

Para evitar que uma mesma pessoa fosse registrada múltiplas vezes, representando indivíduos diferentes (Figura 37), foi necessário realizar os ajustes dos hiperparâmetros que determinam quais faces detectadas são computadas no sistema (apresentados na Tabela 1). Utilizar estes hiperparâmetros com valores muito restritivos acabam limitando o que é registrado, portanto, devem ser ajustados de forma apropriada baseando-se no tipo de ambiente onde o sistema será utilizado e o posicionamento das câmeras. O ajuste foi realizado de forma empírica, ao se analisar desempenho da identificação de pessoas variando-se os valores dos quatro hiperpâmetros apresentados na Seção 3.6.

Figura 37 – Faces iguais classificadas como sendo de indivíduos diferentes



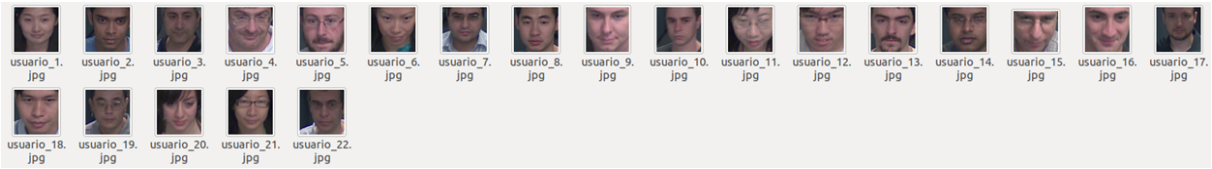
Fonte: Própria do autor

4.3.2 Ajuste da Tolerância

A tolerância é um parâmetro muito importante e é basicamente o módulo da distância entre dois vetores. Ela é utilizada para determinar se uma face é conhecida ou não. Foi realizado um teste para verificar o número de indivíduos que são identificados pelo sistema ao se alterar o valor desse parâmetro em passos de 0,05. Os testes foram executados utilizando o vídeo do “ambiente A” com o Método “B”, que apresentou melhores resultados. De

forma similar ao experimento 1, todos os 25 indivíduos (Figura 32) foram considerados desconhecidos, ou seja, não apresentavam registro biométrico no sistema. Durante a execução do programa as pessoas eram identificadas e a captura da face era salva pelo sistema (Figura 38). Ao se alterar o valor a tolerância foi observada uma variação no número de indivíduos identificados, além disso, alguns indivíduos foram registrados mais de uma vez (Figura 37). Os resultados do teste foram organizados na Tabela 5. A coluna "erro",

Figura 38 – Faces dos indivíduos identificadas e salvas pelo sistema considerado tolerância de 0,6



Fonte: Própria do autor

presente na tabela, indica o número de vezes que uma pessoa foi registrada indevidamente, ou seja, quando uma mesma pessoa é registrada como sendo outra. Observe que valores baixos de tolerância aumentam a precisão da identificação mas reduzem a acurácia.

Tabela 5 – Número de indivíduos identificados do *Chokepoint Dataset* para diferentes valores de tolerância

Num. Identificados	Tolerância	Erro
23	0,5	3
23	0,55	1
22	0,6	0
19	0,65	0
16	0,7	0

4.3.3 Erros de Contagem

Conforme foi dito na Seção 3.7.2, periodicamente eram verificados o número de pessoas desconhecidas registradas no sistema. Esta informação era exibida através de um gráfico (Figura 24) no site do projeto. Contudo, foi verificado que essa contagem poderia apresentar erros devido aos seguintes problemas:

- um indivíduo poderia ser registrado como sendo outra pessoa, portanto, esse seria contabilizado mais de uma vez (Figura 37);
- o indivíduo poderia não ser detectado ou identificado pelo sistema.

5 CONCLUSÃO

Este projeto teve como objetivo criar uma aplicação capaz de automatizar a identificação de suspeitos nos sistemas de videomonitoramento. Utilizando uma rede neural previamente treinada foi possível identificar e registrar indivíduos suspeitos. As informações dos eventos foram salvas em um banco de dados, o que permitiu um maior controle sobre os dados gerados. O banco de dados funcionava como intermediador entre as aplicações de reconhecimento facial e o servidor web, responsável por exibir informações sobre os eventos em um *website*.

Foram aprofundados os conhecimentos do autor sobre aprendizagem de máquina, o gerenciamento e controle de dados. Foi necessário compreender diversos conceitos, como a utilização de redes neurais convolucionais para a biometria facial, a manipulação de dados em um banco de dados relacional e a configuração de um servidor para a disponibilização das informações na internet. Integrando esses conceitos foi possível desenvolver um sistema de monitoramento autônomo, que utiliza a biometria facial para facilitar o processo de identificação de suspeitos.

Apesar dos resultados satisfatórios durante os testes com o *Chokepoint Dataset*, que obteve acurácia de 91,84% na classificação (Método B) durante o primeiro experimento, foram identificadas limitações na rede neural, que apresentou dificuldade na diferenciação entre indivíduos de determinados grupos étnicos, evidenciando a necessidade de um conjunto de dados mais diversificado de treinamento para a rede.

Uma das preocupações durante o projeto foi com o tempo de processamento dos *frames*, já que o armazenamento de dados no banco de dados ou nos diretórios do projeto poderiam causar lentidão na execução dos *scripts*. No primeiro experimento (Método B) foi obtida uma taxa de 16,92 *frames* por segundo, que é uma taxa aceitável.

Uma das dificuldades encontradas durante a execução do projeto foi na maneira utilizada para limitar quais faces seriam cadastradas no sistema, visto que os problemas de posicionamento facial geravam resultados indesejáveis, como a criação de várias classes para um mesmo indivíduo. A solução adotada foi utilizar os pontos gerados pelo algoritmo *Face Landmark* para filtrar os dados, reduzindo desta forma os erros de predição.

Em pesquisar futuras, recomenda-se retreinar a rede neural com um conjunto de dados de faces mais amplo e diversificado, buscando uma possível melhora na identificação facial. Além disso, verificar o desempenho do sistema ao utilizar o modelo de detecção facial baseado em redes neurais convolucionais. Incluir a curva ROC (*Receiver Operating*

Characteristic Curve) na análise do desempenho da classificação. Poderia haver a integração de uma base de dados de criminosos com o sistema, com o objetivo de facilitar o reconhecimento dos suspeitos e agilizar a tomada de decisão. Para isso seria necessário a disponibilização dos dados dos criminosos pelas autoridades policiais.

Apesar de algumas limitações, o sistema baseado em reconhecimento facial pode se tornar uma ferramenta poderosa no combate às atividades criminosas permitindo a celeridade e facilidade na identificação de suspeitos.

REFERÊNCIAS BIBLIOGRÁFICAS

AMOS, B.; LUDWICZUK, B.; SATYANARAYANAN, M. *OpenFace: A general-purpose face recognition library with mobile applications*. [S.l.], 2016. Citado 2 vezes nas páginas 20 e 29.

ARAÚJO, F. et al. Redes Neurais Convolucionais com Tensorflow: Teoria e Prática. In: III ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ. *Livro Anais - Artigos e Minicursos*. [S.l.], 2017. p. 382–406. Citado na página 22.

CERNA, L. R.; MENOTTI, D.; CÁMARA-CHÁVEZ, G. Face Detection: Histogram of Oriented Gradients and Bag of Feature Method. In: *Proceedings of the International Conference on Image Processing Computer Vision and Pattern Recognition (IPCV). The Steering Committee of The World Congress in Computer Science Computer Engineering and Applied Computing (WorldComp)*. [S.l.: s.n.], 2013. Citado na página 16.

CERQUEIRA, D. et al. Atlas da Violência 2018. *Ipea*, 2018. Citado na página 13.

CHUNG, H.; LEE, S.; PARK, J. *Deep neural network using trainable activation functions*. 2016. 348-352 p. Citado na página 23.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I, p. 886–893, 2005. ISSN 1063-6919. Citado na página 16.

DENYER, S. *In China, facial recognition is sharp end of big data drive for total surveillance*. 2018. Disponível em: <https://www.washingtonpost.com/news/world/wp/2018/01/07/feature/in-china-facial-recognition-is-sharp-end-of-a-drive-for-total-surveillance/?utm{_}term=.b15d4e4dc>. Citado na página 12.

G1. *Brasil tem a terceira maior taxa de roubos da América Latina, diz Pnud*. 2013. Disponível em: <http://g1.globo.com/mundo/noticia/2013/11/brasil-tem-terceira-maior-taxa-de-roubos-da-america-latina-diz-pnud.html>. Acesso em: 4 nov 2016. Citado na página 13.

GANEGEDARA, T. *Convolutional Neural Networks: Mayor of the Visionville*. 2017. Disponível em: <http://www.thushv.com/computer{_}vision/convolutional-neural-networks-mayor-of-the-visionvil>. Citado 2 vezes nas páginas 24 e 25.

GEITGEY, A. *Face Recognition: The world's simplest facial recognition api for Python and the command line*. 2017. Disponível em: <https://github.com/ageitgey/face{_}recognit>. Citado 6 vezes nas páginas 14, 19, 20, 28, 29 e 39.

GOEDERT, M. L. et al. *Computação natural: conceitos e aplicações da computação inspirada na natureza*. 2017. Disponível em: <http://www.revistaespacios.com/a17v38n34/17383431.html>. Citado na página 22.

HARRINGTON, P. *Machine Learning in Action*. Greenwich, CT, USA: Manning Publications Co., 2012. 18–36 p. ISBN 1617290181, 9781617290183. Citado na página 21.

HAYKIN, S. *Redes Neurais: Princípios e Práticas*. 2. ed. Ontário: Bookman, 2001. Citado na página 22.

HE, K. et al. Deep residual learning for image recognition. In: CVPR2016. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 2016. p. 770–778. Citado na página 28.

HUANG, G. B. et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. [S.l.], 2007. Citado na página 28.

ITSEEZ. *OpenCV API Reference*. 2016. Disponível em: <http://docs.opencv.org/2.4.13/modules/core/doc/intro.html>. Citado na página 27.

JAFRI, R.; ARABNIA, H. R. A Survey of Face Recognition Techniques. *Journal of Information Processing Systems*, v. 5, n. 2, p. 41–68, 2009. ISSN 1976-913X. Disponível em: <http://koreascience.or.kr/journal/view.jsp?kj=E1JBB0{&}py=2009{&}vnc=v5n2>. Citado na página 21.

KARN, U. *An Intuitive Explanation of Convolutional Neural Networks*. 2016. Disponível em: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>. Citado 2 vezes nas páginas 23 e 24.

KAZEMI, V.; SULLIVAN, J. One Millisecond Face Alignment with an Assemble of Regression Trees. *27th IEEE Conference on Computer Vision and Pattern Recognition*, n. August, p. 1867–1874, 2014. ISSN 978-1-4799-5118-5. Disponível em: <http://www.csc.kth.se/{~}vahidk/papers/KazemiCVPR14{_}poste>. Citado na página 19.

KING, D. E. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, v. 10, p. 1755–1758, 2009. Citado na página 28.

MALLICK, S. *Histogram of Oriented Gradients | Learn OpenCV*. 2016. Disponível em: <https://www.learnopencv.com/histogram-of-oriented-gradients/>. Citado 2 vezes nas páginas 17 e 18.

MING-HSUAN, Y.; DAVID, J. K.; NARENDRA, A. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 1, p. 34–58, 2002. Citado na página 16.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas Inteligentes-Fundamentos e Aplicações*, v. 1, n. 1, 2003. Citado na página 21.

NEVES, L. A. P.; Vieira Neto, H.; GONZAGA, A. (Ed.). *Avanços em Visão Computacional*. 1. ed. Curitiba, PR: Omnipax, 2012. 406 p. ISBN 978-85-64619-09-8. Citado na página 12.

NG, H.-W.; WINKLER, S. A data-driven approach to cleaning large face datasets. In: ICIP. *Proc. IEEE International Conference on Image Processing*. [S.l.], 2014. p. 343–347. Citado na página 28.

ORACLE. *Oracle VM VirtualBox*. 2018. Disponível em: <https://www.virtualbox.org>. Citado na página 31.

PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep Face Recognition. In: *British Machine Vision Conference*. [S.l.: s.n.], 2015. Citado na página 28.

POKHARNA, H. *The best explanation of Convolutional Neural Networks on the Internet!* 2016. Disponível em: <https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8>. Citado na página 27.

PRADO, G.; ARCOVERDE, L. *Roubos e furtos a condomínios crescem 172% em São Paulo, aponta levantamento*. 2017. Disponível em: <https://g1.globo.com/sao-paulo/noticia/roubos-e-furtos-a-residencias-crescem-172-em-sao-paulo-aponta-levantamento.ghtml>. Citado na página 13.

RESK, F. *Brasil gasta mais com segurança privada que pública, diz BID*. 2017. Disponível em: <https://brasil.estadao.com.br/noticias/geral,brasil-gasta-mais-com-seguranca-privada-que-publica-diz-bid,70001652523>. Citado na página 13.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 28.

SALGADO, D. *Atlas da Violência 2018: Brasil tem taxa de homicídio 30 vezes maior do que Europa*. 2018. Disponível em: <https://oglobo.globo.com/brasil/atlas-da-violencia-2018-brasil-tem-taxa-de-homicidio-30-vezes-maior-do-que-europa-22747176>. Citado na página 13.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2010. Citado na página 12.

WONG, A. *Coding up a Neural Network classifier from scratch – Towards Data Science*. 2017. Disponível em: <https://towardsdatascience.com/coding-up-a-neural-network-classifier-from-scratch-977d235d8a24>. Citado na página 26.

WONG, Y. et al. Patch-based Probabilistic Image Quality Assessment for Face Selection and Improved Video-based Face Recognition. In: *IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, 2011. p. 81–88. Disponível em: <http://arma.sourceforge.net/chokepoint/>. Citado 2 vezes nas páginas 42 e 43.

WOODWARD, J. D. et al. *Biometrics: A Look at Facial Recognition*. SANTA MONICA , CA, USA: RAND CORP, 2003. 3–31 p. ISBN 978-0833033024. Citado na página 12.