Deteção de Face Falsa com Imagem NIR Multiespectral e Proposta de Sistema Biométrico Facial para Controle de Presença

Eduardo Carvalho Nunes, nºa39472

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Professor Doutor Pedro João Soares Rodrigues

Professora Doutora Mauren Louise Sguario

Bragança

2018-2019

Deteção de Face Falsa com Imagem NIR Multiespectral e Proposta de Sistema Biométrico Facial para Controle de Presença

Eduardo Carvalho Nunes, nºa39472

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Professor Doutor Pedro João Soares Rodrigues

Professora Doutora Mauren Louise Sguario

Bragança

2018-2019

Agradecimentos

Primeiramente a Deus que permitiu que tudo isso acontecesse, que em todos os momentos é o maior mestre que alguém pode conhecer.

Agradeço a Universidade Tecnológica Federal do Paraná e o Instituto Politécnico de Bragança pela oportunidade de realizar a dupla diplomação.

Aos meus pais, Renato Nunes e Terezinha de Carvalho Nunes, que apesar de todas as dificuldades, me deram apoio e incentivo nas horas difíceis.

Minha noiva, Francimara Marques, obrigado pelo teu carinho, tua alegria, tua atenção, tua vibração com as minhas conquistas e teu ombro em cada momento difícil que você ajudou a atravessar, sem você essa conquista não teria o mesmo gosto.

Aos meus amigos, aos antigos e novos que o IPB me deu, por compartilharem momentos incríveis comigo.

Ao meu orientador em Portugal, Professor Doutor Pedro João Soares Rodrigues. Muito obrigado pelas risadas, ideias, elogios, conhecimento e experiência transmitidas, que foram fundamentais para o meu crescimento pessoal e profissional.

A minha orientadora no Brasil, Professora Doutora Mauren Louise Sguario que aceitou o convite e orientou esta dissertação.

Agradeço aos professores da UTFPR e IPB, que acompanharam a minha jornada acadêmica de perto e deram muito apoio em sala de aula.

Enfim, a todos os que por algum motivo contribuíram para a realização desta dissertação.

Resumo

Os sistemas de controle de presenças que realizam a autenticação através de faces carecem de detectores de fraudes para que sejam mais confiáveis. Um sistema capaz de executar essa tarefa automaticamente e corretamente vem trazer uma série de vantagens práticas no domínio da autenticação biométrica. Para atender esta carência, um detector de face falsa é desenvolvido e serve como um pré-passo antes do reconhecimento facial. A abordagem proposta para detecção de face falsa é utilizar câmera infravermelha do espectro NIR e *machine learning*, referida de *deep learning*. Neste trabalho foi criado uma base de dados de imagens de faces falsas e reais com auxílio de uma câmera com luz infravermelha NIR. A partir das imagens, foram gerados três *datasets* para implementação dos modelos de *machine learning*: Árvore de Decisão, *Random Forest*, KNN, SVM e MLP. Para a construção do protótipo de reconhecimento facial com detector de face falsa foi utilizado a linguagem Python de programação, as bibliotecas de programação: OpenFace, Scikit-Learn, OpenCV e Flask. A partir destas ferramentas e modelos treinados foi possível ter uma acurácia de 97.50% para detecção de faces falsas e faces reais com o classificador SVM. Para o reconhecimento facial foi definido uma limiar (de 0 a 1) confiável de 0.6 para sistemas que utilizam autenticação no formato 1 para N e limiar 0.2 para formato 1 para 1. Pretende-se que no futuro, o protótipo proposto seja ensaiado numa rede de terminais de marcação de presenças no IPB.

Palavras-chave: Reconhecimento Facial, Detecção de Fraudes, Machine Learning, Deep Learning.

Abstract

Presence control systems that use perform face authentication need fraud detectors more reliable. A system to able to detect this task automatically and correctly brings a number of practical advantages in the field of biometric authentication. For this problem, an anti-spoofing is developed and serves as a pre-step before face recognition. The proposed approach for false face detection is to use NIR infrared camera and machine learning with deep learning. In this dissertation, it was created a database of fake and real face images with an infrared camera. From the images, three datasets were created to implement the machine learning models: Decision Tree, Random Forest, KNN, SVM and MLP. For the construction of the face recognition prototype with anti-spoofing, the Python programming language, the OpenFace, Scikit-Learn, OpenCV and Flask programming libraries were used. From these trained tools and models it was possible to have an accuracy of 97.50% for detection of false faces and real faces with the SVM classifier. For face recognition, a reliable threshold (from 0 to 1) of 0.6 for systems using 1 to N format authentication and 0.25 to 1 to 1 format threshold is set. It is intended that the proposed prototype be tested on a network of attendance at IPB.

Keywords: Face Recognition, Anti-Spoofing, Machine Learning, Deep Learning.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Neste Capítulo 1 é dedicado a uma introdução ao tema do trabalho, seguidos pela motivação, objetivos e a estrutura desta dissertação.

1.1 Contextualização

Sistema de Biometria com Reconhecimento Facial é um tópico ativo de pesquisas nas últimas décadas [1]. Atualmente, organizações estão planejando ou utilizando sistemas de reconhecimento facial, alguns exemplos: o governo Chinês que implantou um sistema de reconhecimento facial nas câmeras de vigilâncias pela cidade em Xinjiang [2]; em Londres no aeroporto de Gatwick, será o primeiro aeroporto do Reino Unido que usará câmeras de reconhecimento facial para identificação de passageiros [3]; no Brasil teve a primeira prisão com ajuda do reconhecimento facial [4]; utilização de um sistema de reconhecimento facial para reduzir fraudes de utilização incorreta em transportes públicos [5]; e por fim, uma prática comum e utilizada pelos usuários de telemóveis é o recurso desbloquear com a face que está disponível em grande parte nos telemóveis atuais.

Porém, alguns sistemas de reconhecimento facial podem falhar na deteção de faces falsas. Um teste realizado pela Consumentenbond (organização holandesa sem fins lucrativos que promove a proteção do consumidor) em Abril de 2019, descobriu que de 110 telemóveis, 42 falharam na deteção de faces falsas. Os pesquisadores utilizaram fotografias

de face impressa, máscaras e cabeças 3D para os testes [6].

Com o advento do *Machine Learning* referida de *Deep Learning*, que é baseada em redes neurais artificiais, tem mostrado ser capaz de atingir níveis de desempenho elevados no que concerne à aprendizagem de máquina a partir de exemplos. Estes exemplos constituem padrões integrados em um problema específico. Os modelos neurais são capazes de extrair informações úteis presente nesses padrões e usá-las para posteriormente responderem adequadamente a estímulos de natureza semelhante a esses exemplos.

Os sistemas de controle de presenças que realizam a autenticação através de faces carecem de detectores de faces falsas para que sejam mais confiáveis. Um sistema capaz de executar essa tarefa automaticamente e corretamente vem trazer uma série de vantagens práticas no domínio da autenticação biométrica.

Desse modo, esta dissertação tem como foco utilizar modelos de *machine learning* para deteção de faces falsas na autenticação de um sistema de biometria facial. Pretende-se que, futuramente, os modelos obtidos sejam ensaiados numa rede de terminais de marcação de presenças a instalar no Instituto Politécnico de Bragança (IPB).

A Figura 1.1 ilustra a arquitetura proposta de um sistema de reconhecimento facial a ser instalado em um sistema de controle de presenças. O sistema se resume em um modelo cliente-servidor, onde o cliente é um Raspberry Pi que envia as informações e uma imagem de face do usuário em formato base64 (codificação de dados para transferências na Internet) para autenticação. O servidor recebe as informações e a imagem de face. Em seguida realiza a decodificação do formato base64 e o processamento para autenticação e reconhecimento. Após o processamento, o servidor irá retornar um resultado da autenticação para o cliente.

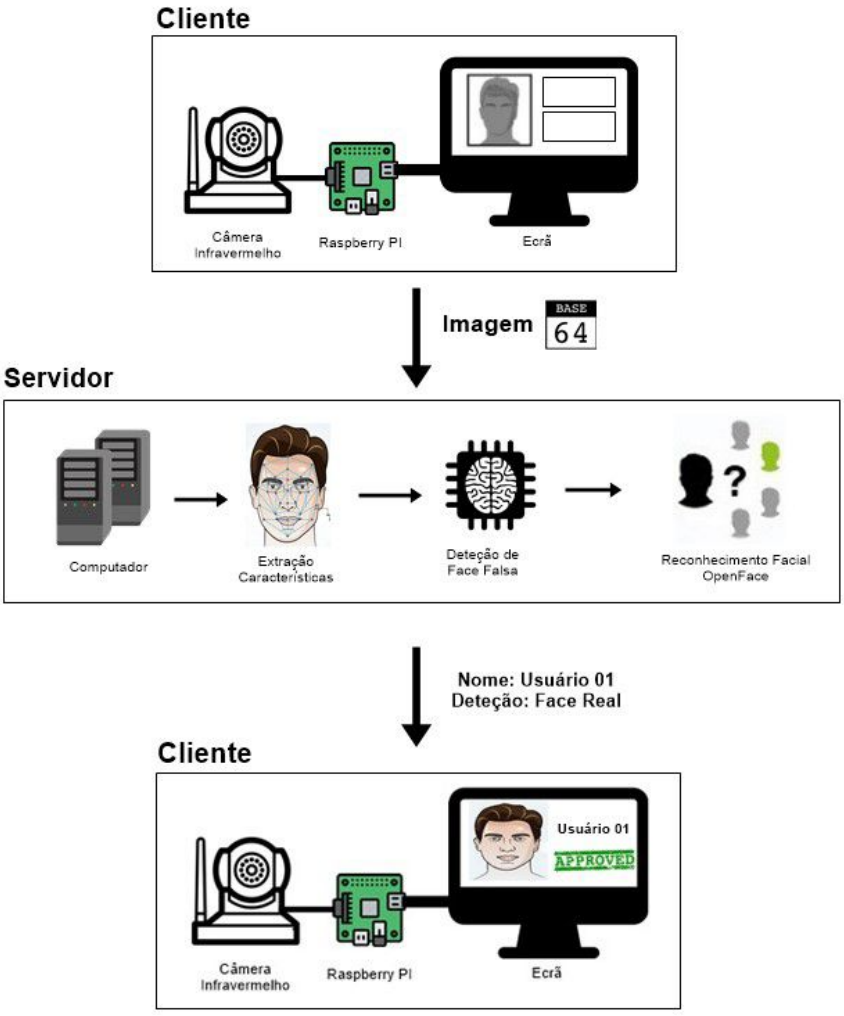


Figura 1.1: Arquitetura Proposta Sistema de Reconhecimento Facial

1.2 Motivação

Para controlar as presenças dos alunos matriculados nas disciplinas, o IPB utiliza um sistema de controle de presença. Esse sistema integram terminais de presença em cada sala de aula, onde o aluno realiza a autenticação a partir de um cartão de identificação por Rádio Frequência (RFID). No entanto, esse sistema pode ser facilmente fraudado por uma pessoa que possui cartões de usuários cadastrados no sistema. Por exemplo, um aluno que tenha em mãos cartões de outros alunos, pode fazer registros de presenças na

ausência dos mesmos.

Dessa forma, a principal motivação dessa dissertação é implementar modelos de *machine learning* para deteção de faces falsas e propor um sistema de reconhecimento facial de baixo custo para integrar no sistema de controle de presença no IPB.

1.3 Objetivos

Os objetivos referentes a esta dissertação serão descritos a seguir. A subsecção 1.3.1 descreve o objetivo geral da dissertação e a subsecção 1.3.2 descreve os objetivos específicos.

1.3.1 Objetivo Geral

O objetivo geral dessa dissertação é utilizar imagens de faces falsas e faces reais obtida por uma câmera infravermelha para detetar fraudes e para reconhecimento facial. Para isto, serão utilizadas técnicas de *Machine Learning*. A partir da deteção de fraudes e reconhecimento facial, será feito um protótipo de modelo cliente-servidor com terminais baseados em Raspberry PI.

1.3.2 Objetivos Específicos

Para satisfazer o objetivo geral, essa dissertação propõe-se apresentar uma solução para os seguintes objetivos específicos:

- Estudar e implementar o *framework* de reconhecimento de faces do OpenFace;
- Construir uma caixa que possa integrar o Raspberry PI, câmera infravermelho e o ecrã;
- Adquirir imagens de faces falsas e faces reais com a câmera infravermelha (projeção na banda de 850nm);
- Com as imagens de faces, extrair características e construir um *dataset* para treinar modelos de *machine learning*;

- Implementar e comparar os modelos de *machine learning* para deteção de faces falsas;
- Implementar um modelo cliente-servidor com terminais baseados em Raspberry PI.

1.4 Estrutura do Documento

A dissertação está estruturada da seguinte forma.

O **Capítulo 2** apresenta a fundamentação teórica sobre Sistema Biométrico Facial, Processamento Digital de Imagem, Aprendizado de Máquina, Técnicas de Aprendizado de Máquina, Rede Neuronal Artificial, *Deep Learning*, Avaliação de Modelos Preditivos e Trabalho Relacionados.

O **Capítulo 3** descreve com mais detalhes as principais metodologias desenvolvidas nesta dissertação.

O **Capítulo 4** apresenta os resultados e discute sobre os modelos de *machine learning*, reconhecimento facial e protótipo desenvolvido.

Por fim, no **Capítulo 5** apresenta as conclusões da dissertação, recomendações e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste Capítulo 2 serão descritos os conceitos necessários para o entendimento da dissertação apresentando inicialmente o Sistema Biométrico Facial, Processamento Digital de Imagem, Aprendizado de Máquina, Rede Neural Artificial, *Deep Learning*, Avaliação de Modelos Preditivos e Trabalhos Relacionados.

2.1 Sistema Biométrico Facial

A Biometria é o estudo de características físicas e do comportamento dos seres vivos [7]. Na área de identificação de pessoas, os sistemas biométricos utilizam as características físicas, biológicas e comportamentais para definir palavra passe, Número de Identificação Pessoal (PIN) e entre outras [8].

Sistema Biométrico Facial mais conhecido como Reconhecimento Facial, envolve o reconhecimento automatizado da identidade do usuário onde são analisadas as características nas imagens da face como olhos, sobrancelhas, queixo, boca, nariz e dentre outras, sendo obtidas através de fotos ou vídeos digitais [9][10].

Faces Falsas

Sistema Biométrico com Reconhecimento Facial pode falhar devido à falsificação de faces feitas por algum invasor não autorizado. O invasor pode encontrar maneiras de falsificar

a face. Os principais ataques são [11]:

- **Ataque Fotográfico Impresso**: O invasor tenta falsificar a face com ajuda de uma fotografia impressa (Figura 2.1(b)) de uma pessoa autorizada;
- **Ataque Fotográfico Digital**: O invasor tenta falsificar a face com uma fotografia digital (Figura 2.1(c)) de uma pessoa autorizada em um dispositivo digital (*tablet*, portátil ou telemóvel);
- **Ataque Reprodução de Vídeo**: O invasor tenta falsificar a face com vídeo digital (Figura 2.1(c)) de uma pessoa autorizada em um dispositivo digital (*tablet*, portátil ou telemóvel);
- **Ataque de Máscara 3D**: O invasor utiliza uma máscara 3D (2.1(d)) de uma pessoa autorizada.

Figura 2.1: (a) Face real. (b) Face falsa fotográfico impresso. (c) Face falsa fotográfico digital. (d) Face falsa Máscara 3D. (reproduzido de Liu [12])

2.1.1 Sistema Facial com Detector de Faces Falsas

O detector de face falsa é a contramedida de falsificação de rosto, onde o principal objetivo é detectar face falsa para proteger o sistema biométrico de pessoa com acesso ilícito. Em um sistema de reconhecimento facial, um rosto é detectado e depois é passado para um detector de face falsa [11]. Se a face de entrada for considerada face real, a imagem

passará para próxima fase do sistema (reconhecimento facial), se for considerada face falsa, terminará o processo com resultado rosto falso. Um diagrama de blocos com detector de face falsa é mostrado na Figura 2.2.

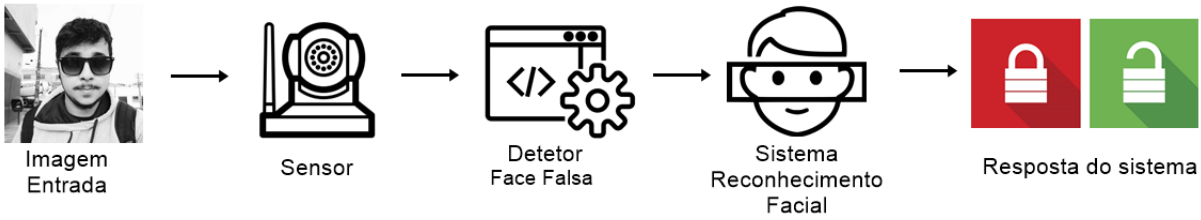


Figura 2.2: Sistema biométrico facial com mecanismo de deteção de faces falsas (adaptado de Transactions [11])

2.2 Processamento Digital de Imagem

Processamento Digital de Imagem é um processo que tem uma imagem como entrada e a saída é um conjunto de números, que podem ou não compor uma outra imagem [13]. A Figura 2.3 é um exemplo de processamento digital de imagem que tem uma imagem colorida como parâmetro de entrada e a saída é uma imagem em tons de cinza como resultado.

Figura 2.3: Conversão de uma imagem RGB para tons de cinza

As próximas subseções: Luz, Infravermelho, Modelo de Imagem Digital, Modelo de

Cor RGB e Modelo L*a*b são referentes a assuntos de Processamento Digital de Imagem utilizados para essa dissertação.

2.2.1 Luz

A luz é o único elemento que pode garantir a percepção da sensação de cor na qual apresenta um ponto mais alto e um ponto mais baixo e uma periodicidade, com isso é possível descrever as características de uma onda eletromagnética pelo seu comprimento ou frequência. O comprimento de onda é especificado em nanômetros (nm) e a frequência indicada em Hertz (Hz) [14].

As cores percebidas pelo ser humano encontram-se na faixa do espectro eletromagnético entre 380nm até 780nm (Figura 2.4) e é através da cor que o ser humano consegue diferenciar, identificar e extrair características de objetos como a luminância, saturação e tonalidade [14][15].

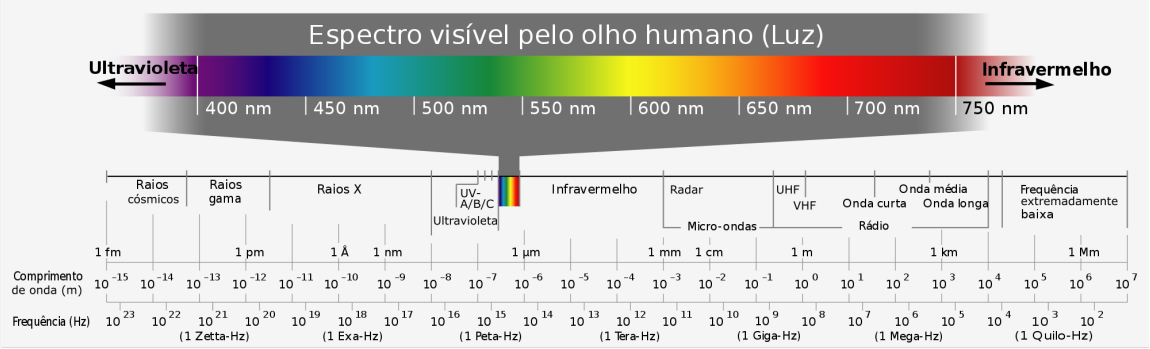


Figura 2.4: Representação do espectro visível pelo olho humano (adaptado de Horst [16])

2.2.2 Luz Infravermelha

A luz infravermelha encontra-se na faixa do espectro eletromagnético entre 750nm até 1mm (milímetros). A frequência do infravermelho é logo abaixo dos comprimentos de ondas da luz vermelha, por isso é chamado de infravermelho [17]. Dentro do espectro infravermelho contém três divisões (Figura 2.5): *Near Infrared* (NIR), *Mid Infrared* (MIR) e *Far Infrare* (FIR).

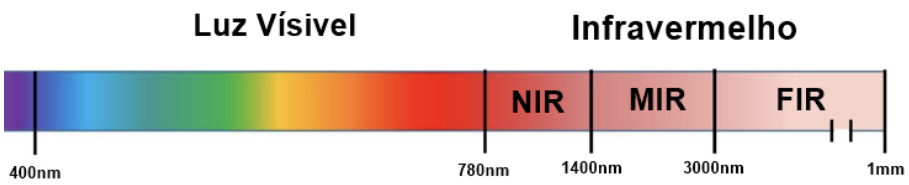


Figura 2.5: Espectro de luz visível e infravermelha

Interação da Luz Infravermelha em Tecido Biológico

O espectro do NIR se encontra na faixa de comprimentos de onda de 650nm a 1400nm, onde pode penetrar nos tecidos biológicos com mais eficiência do que a luz visível[18]. No entanto, comprimentos de onda superiores a 950nm, a eficiência da penetração diminui devido ao aumento da absorção pela água e lipídios.

A Figura 2.6 mostra um resultado da interação da luz infravermelha NIR com um tecido biológico mamário. O eixo horizontal representa o espectro luz em nanômetros e o eixo vertical é o coeficiente da penetração da luz no tecido.

Na figura é possível observar que dentro do espectro do NIR teve uma penetração de luz efetiva na qual o coeficiente efetivo máximo é na banda dos 730nm. Porém, na banda dos 900nm registra-se uma diminuição da penetração comparativamente a uma grande zona espectral da luz visível. Esta caraterística pode ser usada para detectar pele humana quando os sensores de imagem tem uma resposta mais intensa na banda circundante aos 900nm.

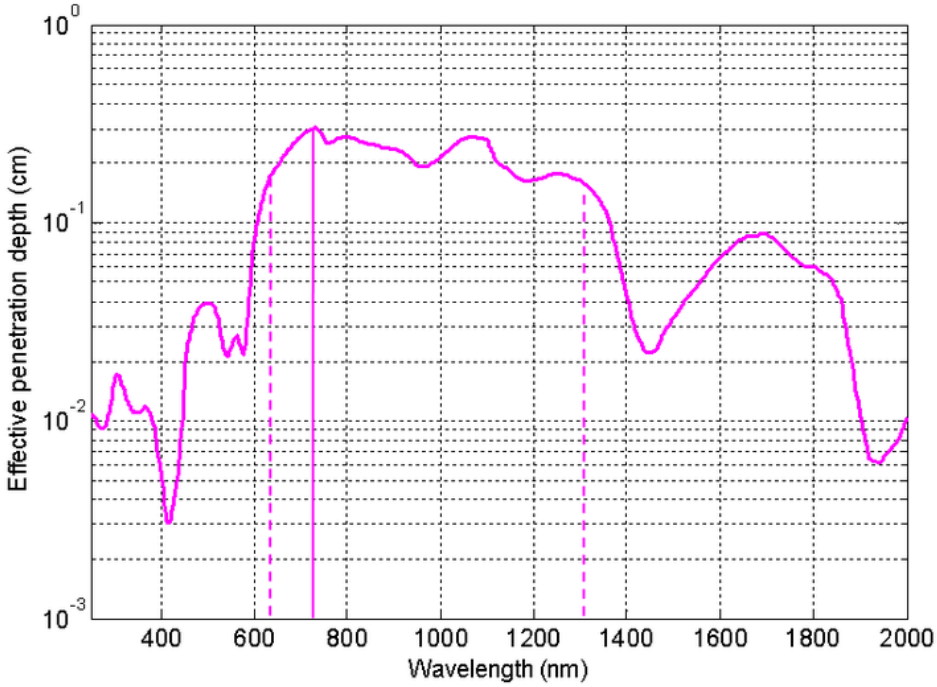


Figura 2.6: Profundidade efetiva de penetração em um tecido biológico mamário (reproduzido de Smith [18])

2.2.3 Imagem Monocromática

Uma imagem monocromática é descrita por um função f(x,y), na qual x e y são coordenadas espaciais e o valor de f em qualquer ponto de (x,y) é proporcional a intensidade luminosa [15]. A Figura 2.7 abaixo mostra uma imagem (tons de cinza) monocromática e a localização de um pixel.

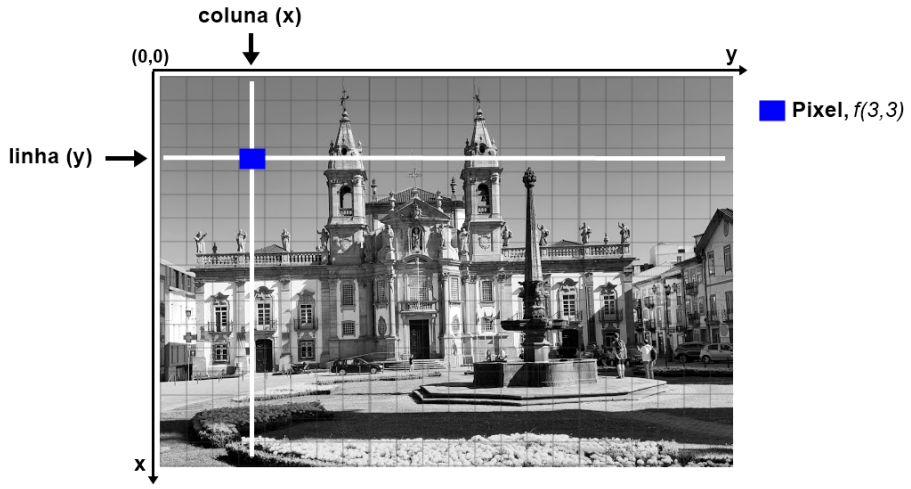


Figura 2.7: Convenção dos eixos para representação de imagem (adaptado de Gonzalez e Woods [15])

2.2.4 Modelo de Cor RGB

Em processamento de imagem a cor é uma característica que muitas vezes simplifica a identificação de objetos [15]. Para representar computacionalmente é necessário um modelo de cor que possa representar alguma forma padronizada. São vários modelos de cores disponíveis na literatura, uma das mais conhecidas é o modelo de cor RGB.

O modelo RGB é baseado em coordenadas onde pode ser representado em um cubo (Figura 2.8). As três vértices são as cores primarias, outras três vértices com cores secundarias, os vértices junto a origem é o preto e o mais afastado da origem corresponde a cor branca [15].

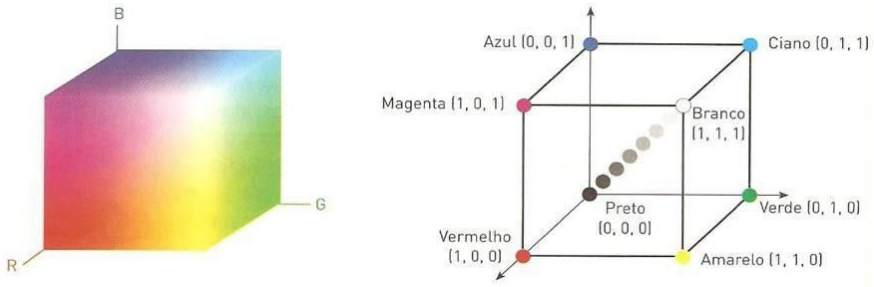


Figura 2.8: Representação modelo de cor RGB (reproduzido de AS [19])

2.2.5 Modelo L*a*b

O modelo L*a*b define três variáveis no espaço tridimensional onde pode ser representado na Figura 2.9 [14]:

- **L***: referente à luminosidade variando entre os valores 0 (escuro) e 100 (luz);
- **a***: referente ao eixo vermelho > verde variando entre -128 e 127, onde os valores positivos são "avermelhados" e os negativos "esverdeados";
- **b***: referente ao eixo amarelo > azul variando entre -128 e 127, onde os valores positivos são os "amarelados" e os negativos "azulados".

Os componentes a* e b* são apropriados para diferenciar as cores referente à luminosidade, isto torna naturalmente útil para este trabalho.

As componentes a e b são adequadas para diferenciar as cores face à luminosidade. O que é naturalmente útil neste trabalho.

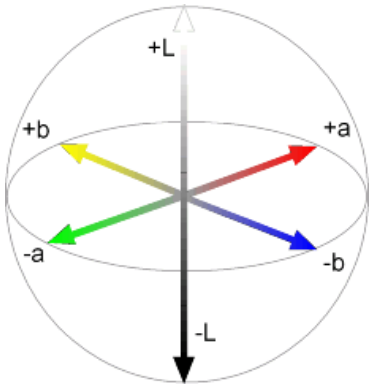


Figura 2.9: Espaço de cor L*a*b (adaptado de Korifi [20])

2.3 Aprendizado de Máquina

Como acontece em qualquer conceito, a definição de Apredizado de Máquina (AM) (do inglês *Machine Learning* (ML)) pode variar um pouco. Samuel define que aprendizado de máquina é o "campo de estudos que fornece aos computadores a capacidade de aprender,

sem ser explicitamente programados" [21]. Para Mohri et al., aprendizado de máquina pode ser amplamente definido como métodos computacionais usando a experiência para fazer previsões, a experiência (normalmente em forma de dados eletrônicos) se refere às informações do passado disponível para aprender [22]. Já Mitchell fornece um pequeno formalismo em sua definição, "um programa de computador é dito para aprender com a experiência E, com relação a alguma classe de tarefas T e medida de desempenho P, se o seu desempenho em tarefas em T, medida pelo P, melhora com a experiência E" [23].

Um exemplo utilizando a definição de Mitchell é um jogo de damas, na qual, E é a experiência de jogar muitos jogos de damas, T a tarefa de jogar damas e P a probabilidade de que o programa irá ganhar o próximo jogo.

Com as definições acima pode-se resumir que o aprendizado de máquina é a ciência de fazer com que os computadores aprendam e ajam como os humanos, melhorem sua aprendizagem ao longo do tempo de forma autônoma, alimentando-os com dados e informações na forma de observações e interações do mundo real.

2.3.1 Modos de Aprendizagem

Segundo Russel e Norvig, para alguns sistemas de aprendizagem é necessário predizer se uma certa ação irá fornecer uma certa saída [24]. Nesse sentido, AM pode ser divida em três importantes subgrupos de algoritmos (Figura 2.10): os que compõe o aprendizado supervisionado, aprendizado não-supervisionado e a aprendizado por reforço.

Aprendizado Supervisionado: tendo um conjunto de exemplos rotulados, ou seja, um conjunto de observações em que a classe (denominada também de atributo meta) de cada exemplo é conhecida, o objetivo é encontrar uma hipótese que seja capaz de classificar novos exemplos entre as classes já existentes [24]. Em sistemas de aprendizagem supervisionado, são classificados problemas de regressão e classificação (Figura 2.11(a)). Problema de regressão tenta prever os resultados em uma saída contínua e um problema de classificação tenta prever os resultados em uma saída discreta [24].

Aprendizado Não-Supervisionado: tendo um conjunto de exemplos não rotulados,

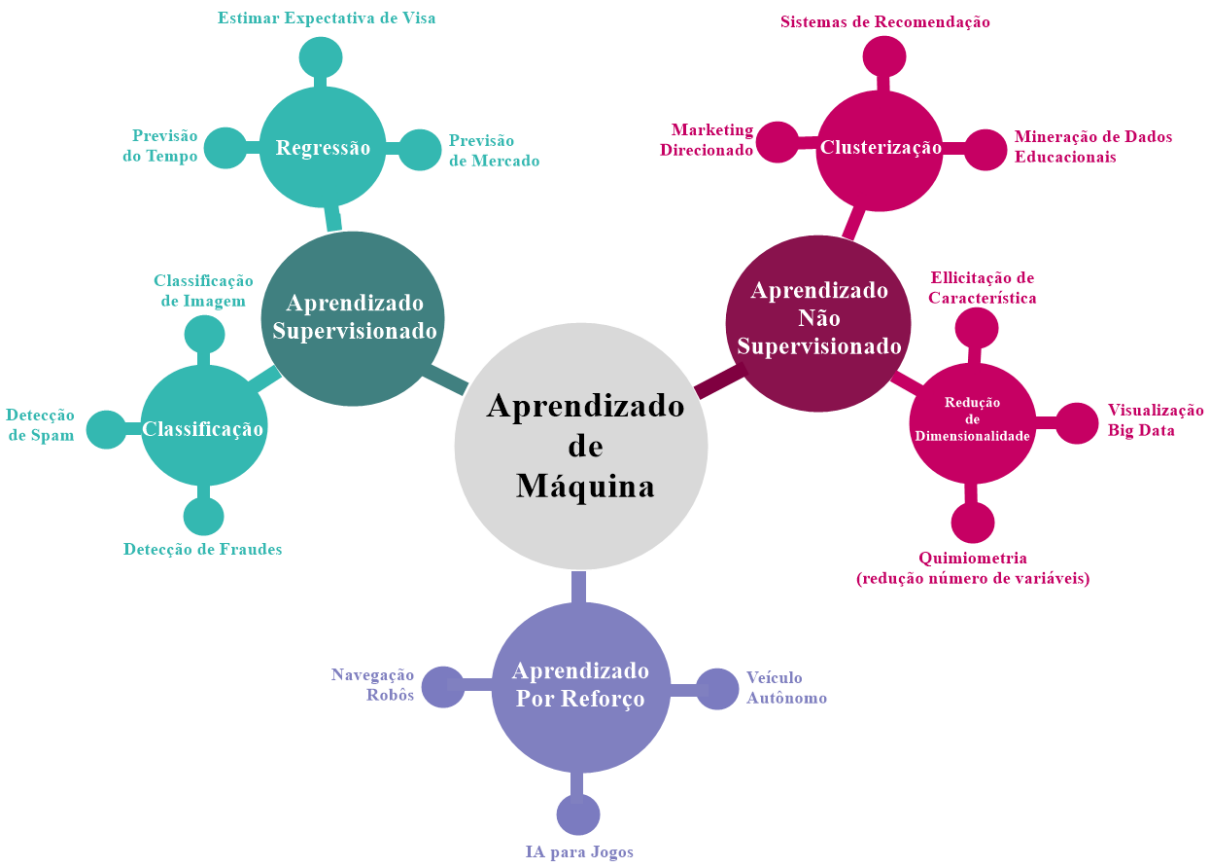


Figura 2.10: Aprendizado de Máquina divido em três grandes categorias: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Reforço (adaptado de Chugh [25])

o objetivo é tentar estabelecer a existência de grupos ou similaridades nesses exemplos (Figura 2.11(b)) [24].

Aprendizado por Reforço: é diferente quando comparado ao aprendizado supervisionado e não-supervisionado, onde é possível ver a relação entre supervisionado e não-supervisionado (a presença ou ausência de rótulos). Em aprendizado por reforço, existe um agente aprendiz que interage com o ambiente que o cerca e aprende uma política de ação por experimentação direta com o ambiente. Dependendo de suas ações, o agente aprendiz é recompensado ou penalizado. O objetivo do agente aprendiz é desenvolver uma política que maximize a quantidade de recompensa recebida ao longo da sua execução (Figura 2.11(c)) [24].

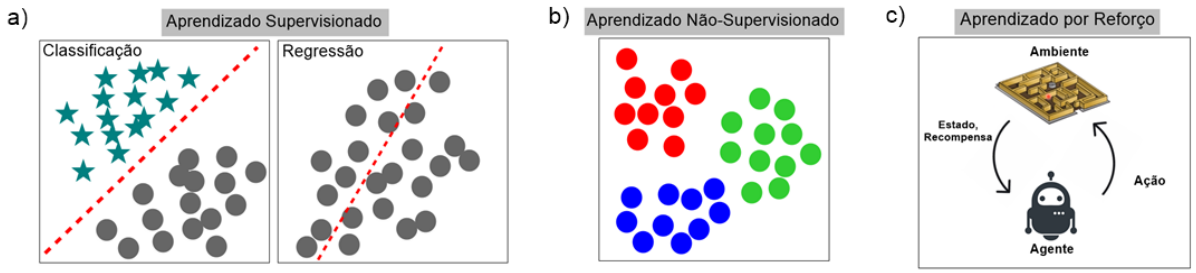


Figura 2.11: (a) Aprendizado Supervisionado (classificação e regressão). (b) Aprendizado Não-Supervisonado. (c) Aprendizado por Reforço.

2.4 Técnicas de Aprendizado de Máquina

As subseções: Árvore de Decisão, *Random Forest*, *K-Nearest Neighbors* e *Support Vector Machine* e *Multilayer Perceptron* irá detalhar as técnicas de AM que foram aplicados nessa dissertação.

2.4.1 Árvore de Decisão

O aprendizado por Árvore de Decisão (AD) é um método que permite a modelagem de sistemas discretos em que o modelo obtido é uma árvore. AD é um dos métodos mais utilizados para a inferência indutiva por ser robusto a ruído nos dados, permitindo uma representação gráfica do modelo gerado [26].

A representação da AD consiste em uma estrutura em que cada nó interno (não folha) corresponde um atributo para ser testado, cada ramo descendente representa uma possibilidade para esse teste e cada folha contém a classe respectiva às instâncias por ela classificadas onde é a decisão obtida após testar os atributos de forma sequencial. O caminho percorrido para chegar à classe corresponde a uma regra de classificação [26].

Um exemplo de AD é demonstrado na Figura 2.12, a decisão sobre qual atividade (folha) fazer no final de semana, pode depender se tiver sozinho ou com amigos (nó raiz), em ambos os casos, a decisão também depende do clima (nós filhos). Se tiver ensolarado e os amigos tiverem disponíveis, pode jogar futebol, se chover, podem ir ao cinema. Se estiver sozinho, não importa o tempo, irá jogar vídeo game.

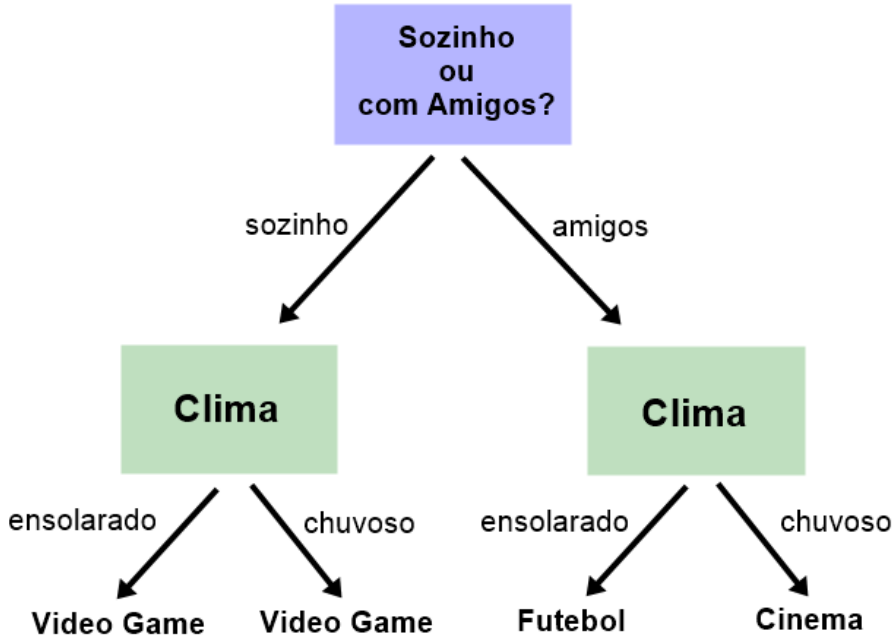


Figura 2.12: Exemplo Árvore de Decisão

2.4.2 Random Forest

Random Forest (RF) é um classificador de AM supervisionada que evolui a partir de árvores de decisão. Smith e Koning [26] definem *Random Forest* como um algoritmo de AM que utiliza várias árvores de decisão para prever um resultado, e essa coleção de árvores é geralmente chamada de conjunto. Cada árvore individual da RF apresenta uma previsão de classe e a classe com mais votos torna-se a previsão do modelo.

A Figura 2.13 ilustra um exemplo de RF. Na RF contem seis árvores e cada árvore faz uma previsão de classe 1 ou classe 0. O resultado final da predição foi a classe 1, pois teve mais votos (4 votos).

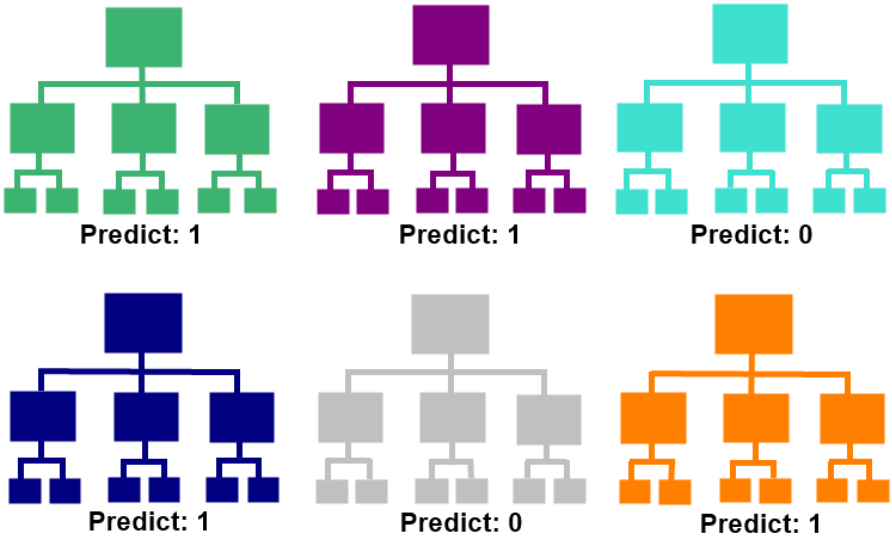


Figura 2.13: Exemplo de RF. Classe 1 teve 4 votos e Classe 0 teve 2 votos. Predição foi a Classe 1

2.4.3 K-Nearest Neighbors

Os *K-Nearest Neighbors* (KNN) é um classificador de AM supervisionado que utiliza técnicas de aprendizado baseado em instâncias, onde calcula-se a distância de um novo exemplo (nova instância) a partir de cada instância pertencente à base de conhecimento [23]. Em outras palavras, o KNN assume que coisas semelhantes estão próximas umas das outras.

O classificador KNN apresenta duas características importantes que antecedem a predição de uma nova instância. A primeira é a definição do número k de vizinhos mais próximos que irão compor a vizinhança da nova instância. A segunda é a escolha da medida de distância responsável por identificar as k observações do conjunto de instâncias mais próximos da nova instância [27].

A Figura 2.14 ilustra um exemplo simples de KNN. Observe-se que no centro da figura contém uma instância (quadrada) que não está classificada e todos outras instâncias estão classificadas (azul e vermelho), cada uma com sua classe (estrela e círculo). Para classificar a nova instância (quadrada), deve-se calcular a distância com todas instâncias,

para identificar as menores distâncias. Para k=3, verifica os três vizinhos mais próximos e a nova instância será classificada como círculo vermelho. Para k=5, verifica os cinco vizinhos mais próximos e a nova instância será classificada como estrela azul.

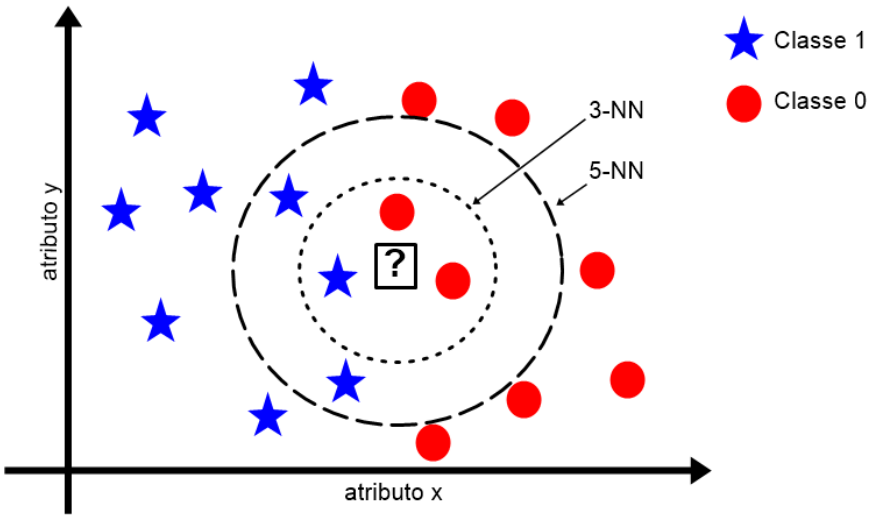


Figura 2.14: Exemplo KNN. Classe 0 são representadas com círculos vermelhos e Classe 1 são representadas com estrelas azuis

2.4.4 Support Vector Machine

O *Support Vector Machine* (SVM) é um classificador de AM supervisionada baseado na teoria de aprendizado estatístico, introduzido por Vapnik [28]. O funcionamento do SVM consiste em maximizar a margem entre a instância de duas classes através de um hiperplano.

Um exemplo simples é ilustrado na Figura 2.15, na qual, contém instâncias bidimensionais. O classificador SVM irá gerar uma linha (hiperplano) que divide o plano em duas partes, em que cada classe fica em cada lado. O classificador SVM usa quatro conceitos fundamentais que são: *kernel* (núcleo), regularização, gama e margem.

Kernels: são utilizados para mapear instâncias de entrada, normalmente de diferentes dimensões, para a fronteira de decisão que possam ser construídas. Exemplos de *kernels* são o Radial (*Radial Basis Function*), Linear e Polinomial.

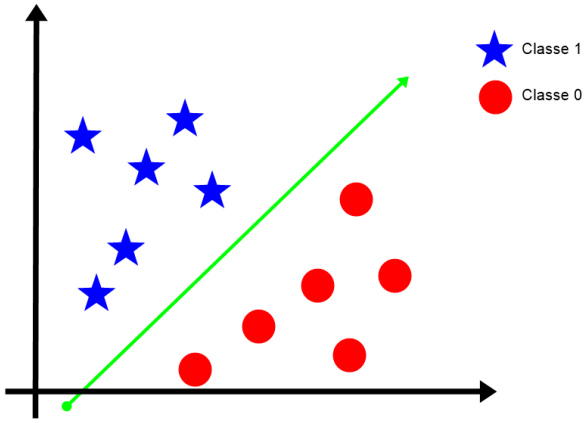


Figura 2.15: Exemplo SVM. Classe 0 são representadas com círculos vermelhos e Classe 1 são representadas com estrelas azuis.

Regularização: informa a otimização do SVM quando deseja evitar erroneamente cada instância do treinamento. Para valores grandes de regularização, a otimização escolherá um hiperplano de margem menor. Por outro lado, um valor muito pequeno de regularização fará com que o otimizador procure por um hiperplano de separação de margem maior, mesmo se esse hiperplano classificar incorretamente mais instâncias. A Figura 2.16 são exemplos de dois parâmetros de regularização diferentes, a esquerda tem um valor baixo de regularização e a direita um valor alto de regularização.

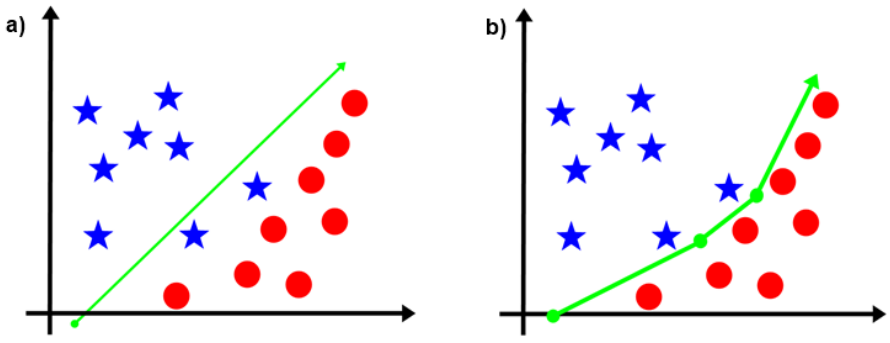


Figura 2.16: (a) Valor baixo de regularização. (b) Valor alto de regularização.

Gama: define até onde chega a influência de um única instância de treinamento. Com valor do gama baixo, pontos distantes da linha de separação plausível são considerados no cálculo da linha de separação. Com valor do gama alto, significa que as instâncias

próximos da linha plausível são considerados no cálculo. A Figura 2.17 são exemplos de valor alto e baixo do gama.

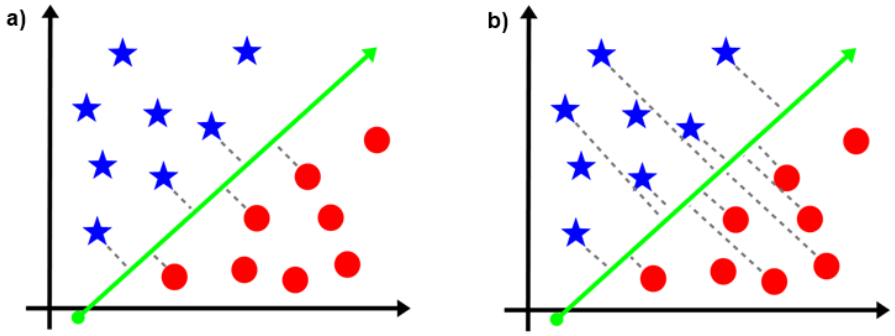


Figura 2.17: (a) Alto gama: apenas pontos próximos são considerados. (b) Baixo gama: pontos distantes também são considerados.

2.5 Rede Neural Artificial

A Rede Neuronal Artificial (RNA) é inspirada nas redes neurais biológicas e possuem variações em suas estruturas, como as *Multilayer Perceptron* amplamente utilizadas para o AM[29]. A RNA pode ser definida como uma estrutura complexa interligada por elementos de processamento simples (neurônios artificiais ou nós), que possuem a capacidade de realizar operações como cálculos paralelos, para o processamento de dados e representação de conhecimento [30]. Antes de compreender o funcionamento de uma RNA é importante entender no que essa técnica está baseada.

2.5.1 Neurônio Biológico

O sistema nervoso biológico possui arquiteturas de muitas complexidades. Esses sistemas complexos são compostos por bilhões de células neurais ou neurônios, que desempenham funções diferentes [30]. A Figura 2.18(a) mostra o esquema de um neurônio biológico que possui um corpo celular com dois tipos de ramos, os dendritos e axônios.

O corpo celular carrega informações sobre suas características, além de um plasma que possui as substâncias moleculares necessárias para o funcionamento da célula. A

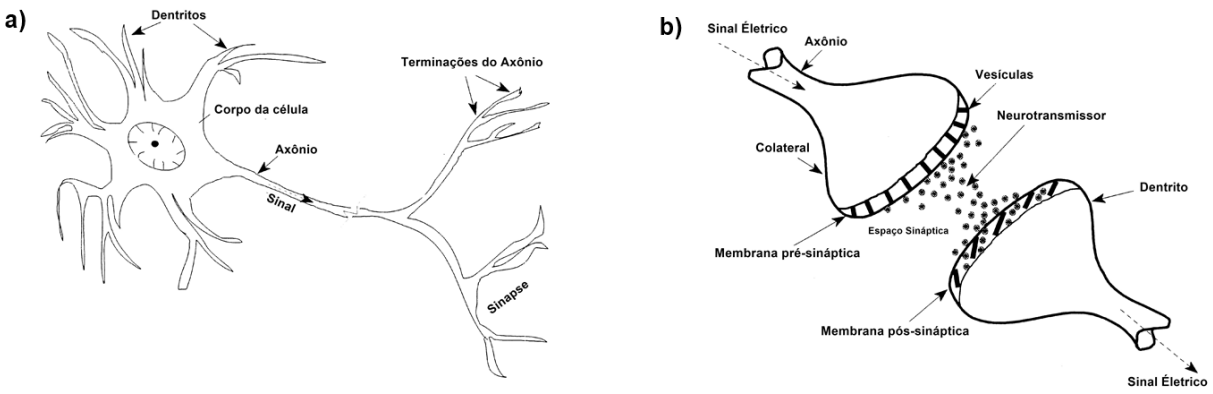


Figura 2.18: (a) Célula nervosa biológica. (b) Mecanismo de comunicação entre dois neurónios biológicos. (adaptado de Basheer e Hajmeer [30])

comunicação entre os neurônios ocorre através de impulsos captados pelos dendritos, responsáveis por receber a informação e repassar para o corpo da célula através do axônio. O axônio que se divide em colaterais, recebe sinais a partir do corpo da célula e os transporta para os dendritos que vão repassar para os dendritos de outros neurônios vizinhos através da sinapse (Figura 2.18(b)) [30].

2.5.2 Analogia entre Redes Artificial e Biológica

A analogia entre a rede artificial e biológica é que ambas possuem axônio e dendrito e modulam por sinapses. A Figura 2.19 ilustra a representação dessa analogia. A letra X representada em ambas figuras são os sinais recebidos e a letra W a força sináptica. O modelo artificial apresentado na Figura 2.19(b) representa a mais simples rede RNA, chamada de Perceptron.

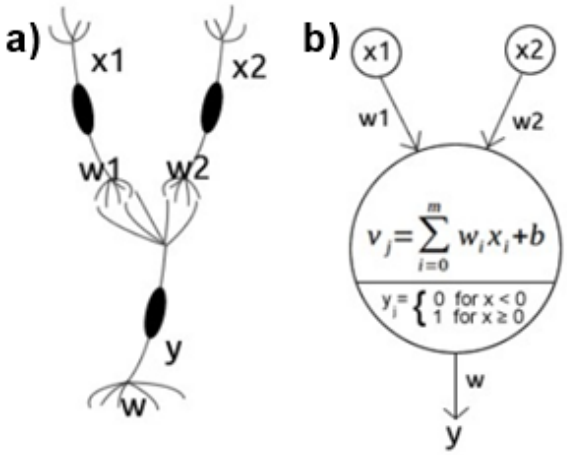


Figura 2.19: Relação entre rede neural biológica(a) e artificial(b) (adapatado de Basheer e Hajmeer [30]).

2.5.3 Perceptron

Em 1958, Rosenblatt introduziu a Rede Neural Artificial Perceptron RNA usada para classificação de padrões linearmente separáveis; ou seja, padrões que estão em lados opostos de um hiperplano. Esse modelo lida com um único neurônio, onde o resultado da classificação é de forma linear.

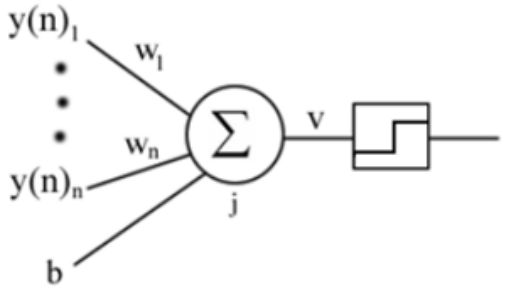


Figura 2.20: Visão geral de uma rede neural artificial (adaptado de Haykin [29]).

Na Figura 2.20, o neurônio artificial é um Perceptron que recebe vários valores de entrada y(n). Essas entradas são multiplicadas pelo peso da sinapse W e no final é somado, formando um conjunto de entrada. O resultado passa por uma função de ativação linear

e transmite na saída (será ativado ou não).

2.5.4 Multilayer Perceptron

Com o objetivo de enfrentar problemas não linearmente separáveis, foram adicionadas camadas de neurônios ocultas no modelo Perceptron, formando assim a Perceptron Multicamadas ou *Multilayer Perceptron* (MLP).

Uma RNA do tipo MLP é constituída por um conjunto de nós onde formam a camada de entrada, uma ou mais camadas ocultas e uma camada de saída (Figura 2.21). Com exceção da camada de entrada, todas as outras camadas são constituídas por neurônios computacional. Esse novo modelo funciona como uma rede *feedforward* (rede progressiva, onde a saída de um neurônio é conectado com outro neurônio da próxima camada, no sentido esquerda para direita), formada por um conjunto de neurônios (nós). A complexidade de uma rede MLP é dada pela quantidade de camadas ocultas e a quantidade de neurônios que essas camadas possuem [29].

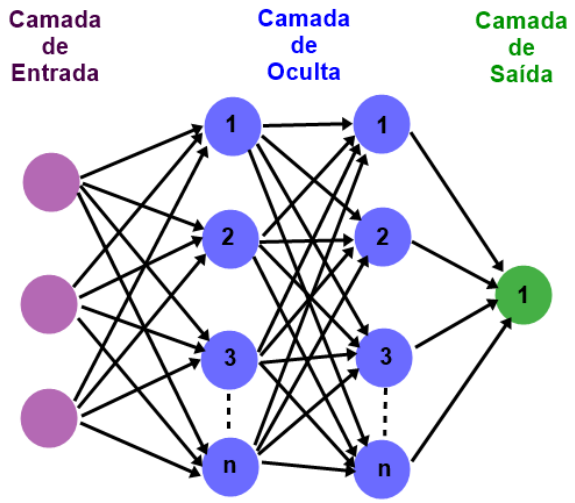


Figura 2.21: Arquitetura de uma rede neural MLP com duas camadas ocultas.

2.5.5 Função de Ativação

Depois do cálculo do somatório ponderado dos sinais, o resultado passa por uma função de ativação. As funções de ativação são um elemento fundamental nas redes neurais, na qual basicamente decidem, se um neurônio vai ser ativado ou não [29]. Porém, não existe somente um tipo de função de ativação, existem vários tipos. A Figura 2.22 ilustra algumas funções de ativação onde a primeira coluna é o nome da função de ativação, segunda coluna é a representação da função em forma de gráfico e a última coluna é a equação da função.

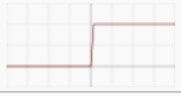
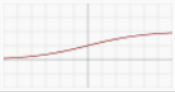
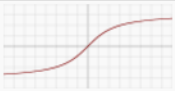
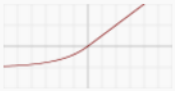
Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

Figura 2.22: Funções de Ativação (reproduzido por Sharma [31])

2.5.6 Backpropagation

Uma vez que a rede neural artificial é inicializada, o conjunto de pesos recebem valores aleatórios e são multiplicados pelos valores recebidos. Pode ocorrer que o conjunto de pesos não atingem os valores desejados. Para corrigir isso, é utilizando um algoritmo chamado retropropagação ou *backpropagation*.

Backpropagation é o processo onde ocorre o aprendizado da rede, na qual é comparado o valor obtido com o valor desejado através de equações matemáticas[29]. Caso o resultado não esteja no padrão aceitável, a rede calcula o erro e propaga a correção para as demais camadas internas até a camada de entrada, ajustando o conjunto de pesos da rede neural.

2.6 Deep Learning

Deep Learning é um subcampo de *machine learning* que utilizam Redes Neuronais Profundas ou *Deep Neural Network* (DNN). As DNNs geralmente são modificações do MLP convencional, onde são adicionadas múltiplas camadas escondidas com conexões parciais [32].

O objetivo das arquiteturas que implementam *deep learning* é identificar abstrações nos dados, dos níveis mais baixos para os mais altos, obtendo novas representações. Com isso características de alto nível são formadas pela composição das características de baixo nível, sendo uma das principais vantagens de *Deep Learning* a extração de características dos dados [33].

Uma das diferenças entre um modelo de *machine learning* e um modelo de *deep learning* está na área de extração de recursos ou características. A extração de recursos é feita por humanos em *machine learning*, enquanto o modelo de *deep learning* é descoberto por si só [34]. Esta diferença é mostrada na Figura 2.23.

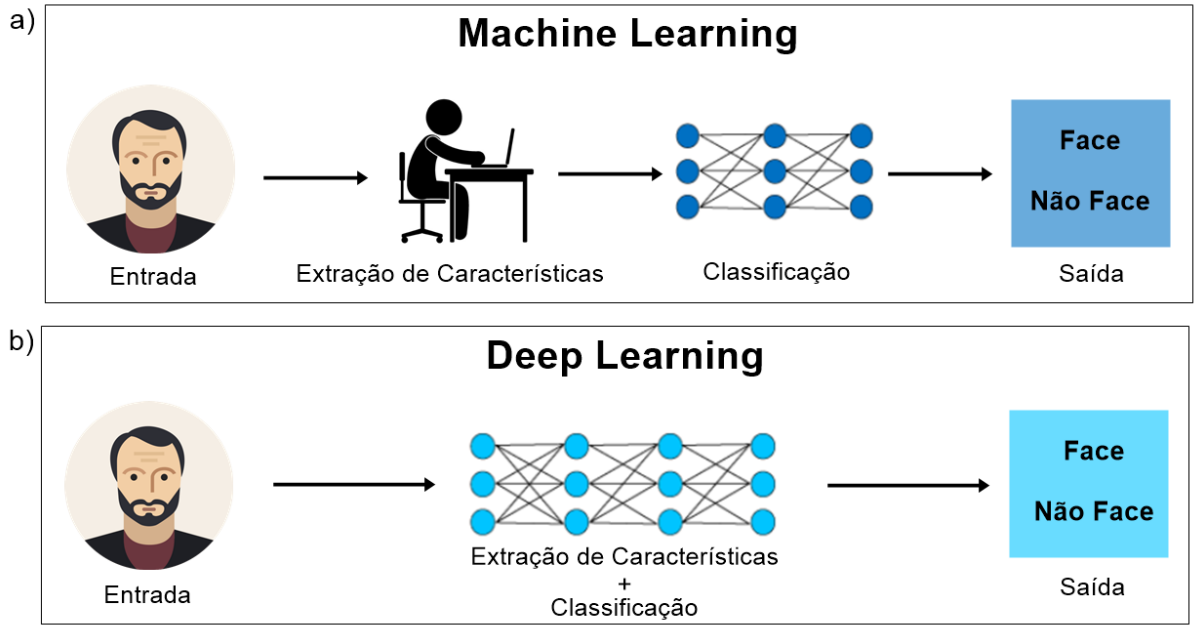


Figura 2.23: (a) Arquitetura simplificada de Machine Learning. (b) Arquitetura simplificada de Deep Learning (adaptado de Sharma [35]).

2.6.1 Redes Neuronais Convolucionais

Em *deep learning*, a Rede Neural Convolucional ou *Convolutional Neural Network* (CNN) é umas das principais redes para o reconhecimento de imagens, detecções de objetos, reconhecimento facial, classificação de imagem e entre outras [33]. Em súmula, a CNN recebe uma imagem como entrada, processa e classifica. No total a CNN é composta por 4 etapas (Figura 2.24): Convolução, *Pooling*, *Flattening* e a *Fully Connected*. As próximas subsecções irão resumir cada etapa.

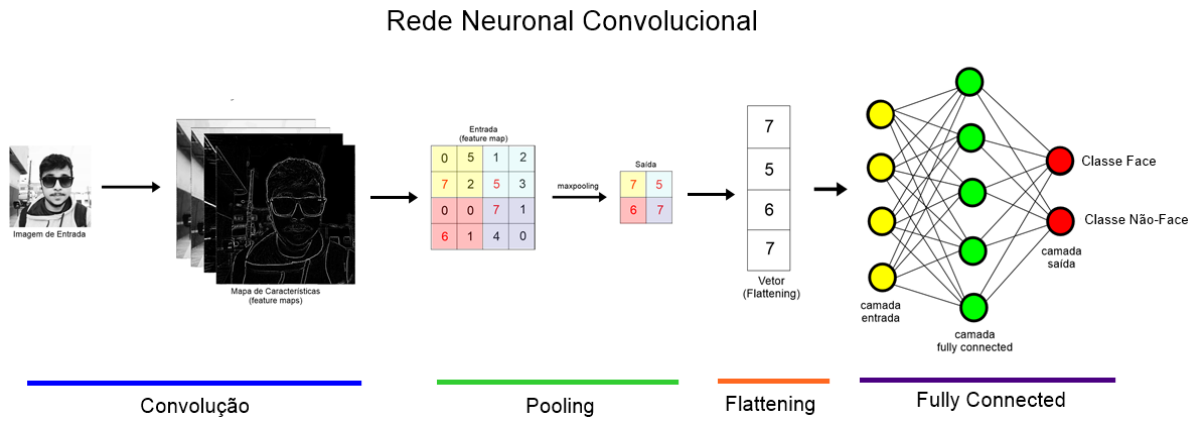


Figura 2.24: Arquitetura da CNN.

Convolução

A convolução é a primeira etapa a extrair recursos da imagem de entrada. O processo de convolução é adicionar cada elemento(pixel) da imagem para seus vizinhos, ponderado por um *kernel* [36]. A Figura 2.25 mostra quatro exemplos de *kernel*, na primeira coluna tem o nome do *kernel*, na segunda coluna é a imagem de entrada, na terceira coluna o *kernel* (matriz do *kernel*) e a última coluna é o resultado da aplicação da convolução.

No contexto da CNN, o *kernel* é chamado de *Feature Detector* (Detector de Características) e o resultado do *kernel* é chamado de *Feature Map* (Mapa de Características). Em *Feature Map* a imagem poderá ficar menor, isto facilitará o processamento durante o treinamento e teste. Outro ponto importante é que neste processo, alguma informação pode ser perdida, porém o principal objetivo desta etapa é detectar as principais características, por exemplo, imagens de faces de pessoas são os olhos, boca, nariz e entre outras [36].

Operação	Entrada	Kernel	Resultado
Sharpen		0 -1 0 -1 5 -1 0 -1 0	
Edge		-1 -1 -1 -1 8 -1 -1 -1 -1	
Emboss		-2 -1 0 -1 1 1 0 1 2	
Sobel (top)		1 2 1 0 0 0 -1 -2 -1	

Figura 2.25: Exemplo de kernel na camada de convolução na CNN.

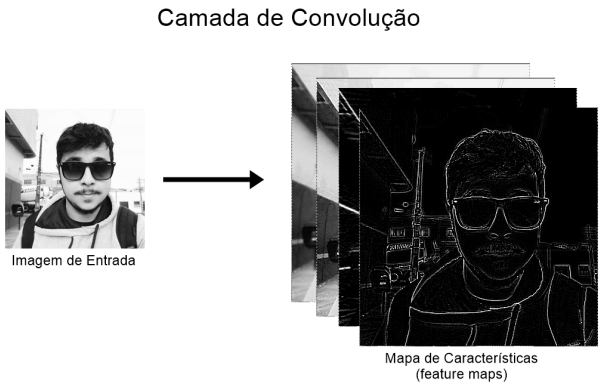


Figura 2.26: Camada de convolução da CNN.

Ainda na etapa de convolução, se encontra a aplicação da função ReLU. Esta função transformará os valores dos elementos (pixels) negativos para zero, caso o valor do pixel for zero ou positivo, a função manterá o valor. Por fim a Figura 2.26 ilustra a camada de convolução onde tem uma imagem como entrada e o resultado é um mapa de características referente a imagem da entrada.

Pooling

A camada *pooling* serve para reduzir *overfitting*, ruídos desnecessários, reduzir a dimensionalidade e enfatizar ainda mais as características de cada *feature map* gerado na camada de convolução. Assim como na camada convolução, é definido uma área, por exemplo uma matriz 2x2, cada área extrairá um único valor. Além disso é preciso escolher um método de sumarização, que pode ser de diferentes tipos como: *Max Pooling*, *Average Pooling* e *Sum Pooling* [36].

A Figura 2.27 mostra um exemplo desta etapa. No exemplo é utilizado o *Max Pooling* com uma área de 2x2. O *Max Pooling* escolherá o valor maior em uma matriz 2x2, o resultado é outra matriz (saída).

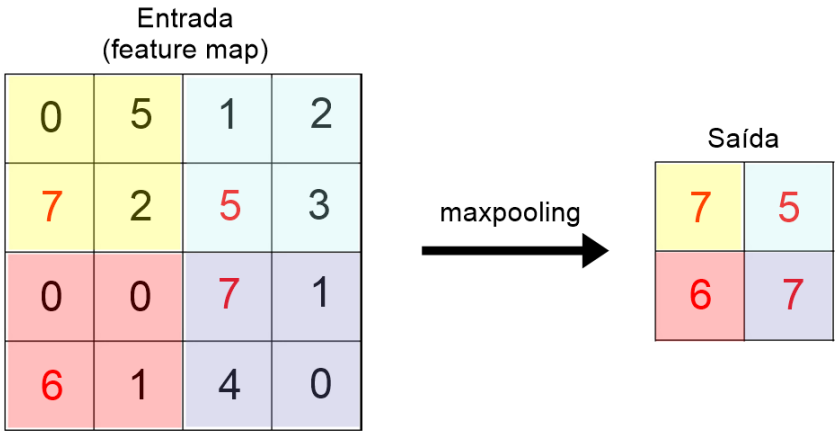


Figura 2.27: Exemplo da etapa pooling usando o max pooling com matriz 2x2

Fully Connected

Nesta etapa que as RNAs e as CNNs se encontram. Observando a Figura 2.28 abaixo, temos a camada de entrada, uma camada Fully Connected e camada de saída [36]. O funcionamento da RNA é descrito na seção 2.5.

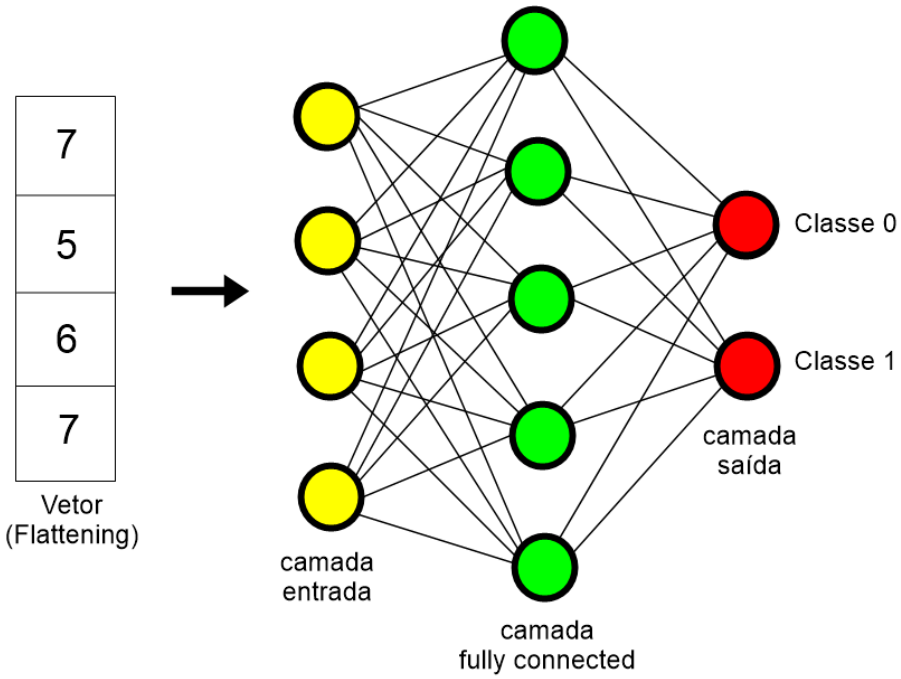


Figura 2.28: Exemplo da etapa fully connected

2.6.2 One Shot Learning com Redes Siamesas

A CNN é um método muito utilizado para classificação e deteção de objetos. Contudo, uma das maiores limitações da CNN é a exigência de muitos dados rotulados. Em determinados problemas, a rotulação de tantos dados não se torna viável, e o *One Shot Learning* propõe-se a solucionar esse problema. Ao contrário de um classificador comum, a classificação utilizando *One Shot Learning* exige somente um exemplo para o treinamento de cada classe (poucos dados rotulados) [37], deste que tenha um mecanismo de transferência de informação anterior (baseado em diversos exemplos).

A RNA do tipo Siamesa é uma classe de arquiteturas de redes neuronais que contêm

duas ou mais sub-redes idênticas. As sub-redes contêm a mesma configuração com os mesmos parâmetros e pesos. As redes neuronais siamesas são populares entre as tarefas que envolvem encontrar semelhança ou relação entre duas coisas comparáveis [37].

A Figura 2.29 mostra uma arquitetura de rede neuronais siamesas. As imagens 1 e 2 são entradas em uma única CNN. A última camada da CNN gera um vetor fixo, como são duas imagens, temos dois vetores fixos de características. A distância absoluta entre os vetores é calculada. Os valores seguem em uma função sigmoide e é gerado um *score* de similaridade. Normalmente, a pontuação de similaridade é calculada entre 0 e 1 usando uma função sigmoide; em que 0 indica sem similaridade e 1 indica similaridade completa. Qualquer número entre 0 e 1 é interpretado de acordo com o problema.

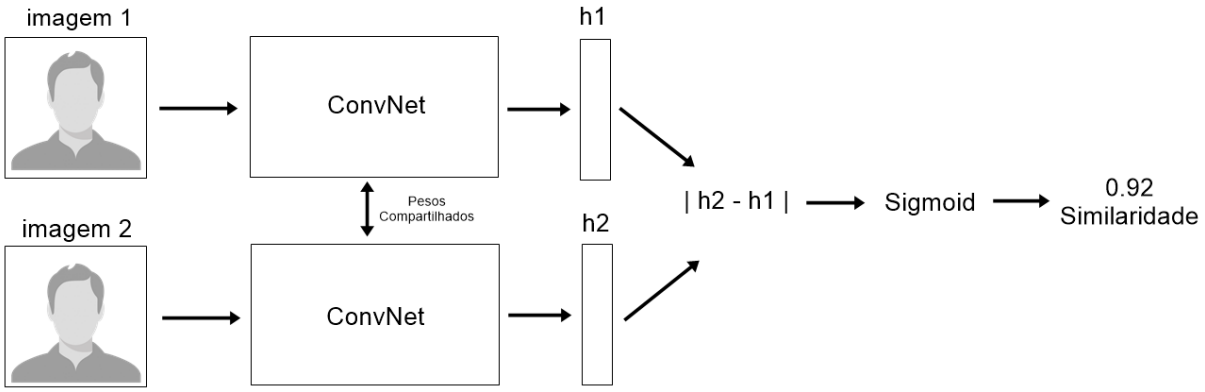


Figura 2.29: Exemplo arquitetura Rede Neural Siamase.

2.7 Avaliação de Modelos Preditivos

A avaliação de um classificador de AM supervisionado é geralmente realizado por meio de análise do desempenho do modelo gerado durante na rotulação de novos objetos, ou seja, a avaliação é feita com base em medidas de desempenho que indicam a taxa de sucesso do modelo preditivo [38].

2.7.1 Amostragem e Holdout

Em determinados casos, tem-se apenas um conjunto de dados ou *dataset*, onde deve ser empregado na indução do preditor e também na avaliação. Nestes casos devem utilizar métodos de amostragem alternativos para obter estimativas de desempenho preditivo mais confiáveis, definindo subconjuntos de treinamento e teste [38].

O método *holdout* (Figura 2.30) consiste em dividir o conjunto total em dois subconjuntos. Um conjunto para treinamento e outro para teste. Uma proporção muito utilizada na literatura é 70% dos dados ao conjunto de treinamento e 30% para o conjunto de testes [39].

Após o particionamento, a estimação do modelo é realizada e, posteriormente, os dados de teste são aplicados e o erro de predição calculado.

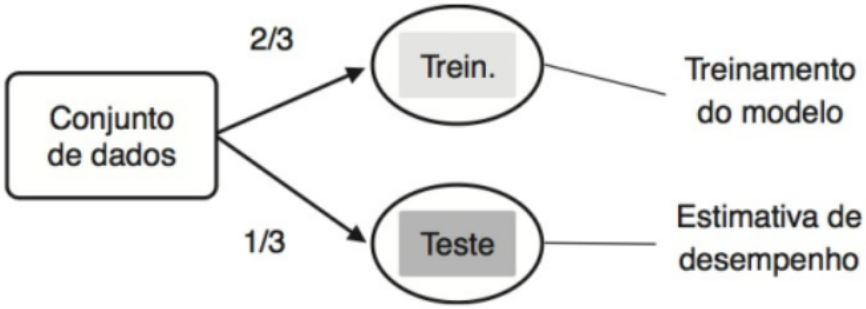


Figura 2.30: Método *Houldout* (reproduzido de Faceli [38]).

2.7.2 Métricas para Classificação

Para visualizar o desempenho de um classificador, pode ser utilizado uma matriz de confusão. Essa matriz ilustra o número de predição corretas (linhas) e incorretas (colunas) em cada classe [38]. Para maior compreensão, imagina-se a classificação binária entre face falsa e face real, então as quatros medidas da matriz de confusão serão:

- **Verdadeiro Positivo (VP)**: verdadeiro positivo é a face que é real e que foi classificada corretamente, face real;

- **Verdadeiro Negativo (VN)**: verdadeiro negativo é a face que é falsa e que foi classificada corretamente, face falsa;

- **Falso Positivo (FP)**: falso positivo é a face que é falsa e que foi classificado equivocadamente como face real;

- **Falso Negativo (FN)**: falso negativo é a face que é real e que foi classificado equivocadamente como face falsa.

Um exemplo da matriz de confusão com face falsa e face real é ilustrado na Figura 2.31.

		Classe predita	
		Face real	Face falsa
Classe real	Face real	**VP**	**FN**
	Face falsa	**FP**	**VN**

Figura 2.31: Exemplo de matriz de confusão para face falsa e face real.

2.7.3 Medidas de Desempenho

A partir da matriz de confusão, é possível avaliar o desempenho do modelo em algumas métricas. As medidas são calculadas das métricas [38]:

- **Acurácia**: calculada pela soma dos valores da diagonal da matriz e divida pela soma dos valores de todos os elementos da matriz (Equação 2.1).

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.1}$$

- **Sensibilidade**: corresponde a taxa de acerto da classe positiva (Equação 2.2).

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

- **Especificidade**: corresponde à taxa de acerto na classe negativa (Equação 2.3).

$$Especificidade = \frac{VN}{VN + FP} \tag{2.3}$$

2.8 Trabalhos Relacionados

Nesta seção serão descritas análises de trabalhos publicados que se referem a deteção de face falsa. Sobre a pesquisa na literatura, existem inúmeros trabalhos sobre deteção de faces falsas utilizando somente câmera visível ou RGB. Porém foram encontrados somente dois trabalhos relacionados a deteção de faces falsas utilizando câmera infravermelha (IR) e *machine learning*. A Tabela 2.1 mostra de forma resumida os trabalhos relacionados.

Tabela 2.1: Título, ano, sensor utilizado, modelo de machile learning, dataset e resultado dos trabalhos relacionados.

Título	**Ano**	**Sensor**	**Modelo**	**Dataset**	**Performance**
Reliable Face Anti-Spoofing Using Multispectral SWIR Imaging [40]	2016	IR	SVM	404; imagens; 137 participantes	Acurácia:99:28%
A Dataset and Benchmark for Large-scale Multi-modal Face Anti-spoofing[41]	2019	RGB, IR e Depth	CNN	5.175.790; imagens; 21000 vídeos; 1000 participantes	TPR(RGB):49.3% TPR(IR):65.3% TPR(Depth):88.3% TPR(RGB+IR+Depth): 96.7%

O artigo de Steiner et al. [40] publicado em junho de 2018, utiliza uma câmera infravermelha no espectro do SWIR e o modelo SVM de *machine learning* para detectar faces falsas. O *dataset* utilizado contém 404 imagens de 137 participantes. O resultado da acurácia para deteção de face falsa foi de 99.28%.

O artigo de Zhang et al. [41] publicado em abril de 2019 utilizou a câmera Intel

RealSense SR300 que contém IR, RGB e *Depth* para capturar faces. O artigo apresenta o maior *dataset* de face falsa da literatura com mais de 5 milhões de imagens e 21 mil vídeos de 1000 participantes. Para deteção de face falsa, os autores utilizaram o modelo de *deep learning* CNN. Como a câmera contém 3 sensores, foram analisadas faces somente com a câmera normal (RGB), câmera IR, câmera de profundidade (*depth*) e com 3 sensores juntos. Utilizando somente a câmera RGB obteve o TPR (Sensibilidade) de 49.3%, com câmera IR o TPR foi de 65.3%, com a câmera de profundidade o TPR foi de 88.3% e por fim, utilizando os 3 sensores o TPR foi de 96.7%.

Capítulo 3

Metodologia e Desenvolvimento

Neste Capítulo 3 serão descritos os matérias e métodos desenvolvidos para essa dissertação apresentando inicialmente as Ferramentas Utilizadas, Metodologia para Aquisição de Imagem, Extração de Características de Faces, o *Dataset* de Faces Falsas e Reais, Metodologia de Treinamento e Testes, os Parâmetros dos Modelos de *Machine Learning* e por fim a Metodologia para Teste de Reconhecimento Facial com OpenFace.

3.1 Ferramentas Utilizadas

Nesta seção mostrará as ferramentas utilizadas para a construção e implementação dessa dissertação.

3.1.1 Computador

O Computador utilizado para este trabalho foi um Acer VX5-591G-5872, 8GB RAM, processador CPU Intel i5-7300HQ 2.5GHz e GPU Nvidia GTX 1050 4GB. O sistema operacional instalado e utilizado foi o Windows 10 64-bit Education N.

3.1.2 Raspberry Pi 3 Modelo B+

O Raspberry Pi (3.1) é um mini-microcomputador que abriga processador, processador gráfico, *slot* para cartões de memória, interface USB, HDMI e seus respetivos controladores. As principais especificações do Raspberry Pi são:

- Processador Broadcom BCM2837B0 64bits ARM Cortex-A53 Quad-Core;
- Clock 1.4 GHz;
- Memória RAM: 1GB;
- Adaptador Wifi 802.11 b/g/n/AC 2.4GHz e 5GHz integrado;
- GPIO de 40 pinos;
- 4 portas USB 2.0.

Figura 3.1: Raspberry Pi utilizado para este trabalho (reproduzido de Raspberry Pi[42]).

3.1.3 Câmera Infravermelho NIR

A câmera se conecta diretamente com o conector CSI no Raspberry Pi e possui dois refletores de LED infravermelho no espectro do NIR. As principais características são:

- Câmera OV5647 de 5 megapixels (sem filtro infravermelho);

- Clock 1.4 GHz;
- 2 LEDs infravermelhos 850 nanômetros de alta potência de 3W:
- Dimensão: 25mm x 24mm;
- Abertura (F): 1.8.

Figura 3.2: Câmera infravermelho utilizada para aquisição de imagens (reproduzido de Boxeletronica[43]).

3.1.4 Impressora 3D

A impressora 3D foi utilizada para construir os *cases* do Raspberry Pi, câmera infravermelho e o *display*. A marca e modelo da impressora 3D utilizada é Fusion3 F400-S (Figura 3.3).

Figura 3.3: Impressora 3D utilizado para construir as *cases* (reproduzido de Fusion [44]).

3.1.5 Python - Linguagem de Programação

A linguagem de programação escolhida para este trabalho foi Python, que possui interface para a biblioteca de programação OpenCV. De acordo com o site da linguagem [45], Python é uma linguagem de programação de alto nível de tipagem dinâmica e forte. Possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Uma de suas principais características é permitir fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo software em outras linguagens.

Figura 3.4: Logotipo Python.

3.1.6 OpenCV - Biblioteca de Programação

No presente trabalho foi empregado a biblioteca de programação OpenCV para o desenvolvimento do *software*. De acordo com o site [46], o OpenCV inicialmente foi desenvolvido pela Intel e é uma biblioteca de programação que está sob a licença BSD (Berkeley Software Distribution).

A biblioteca é totalmente livre ao uso acadêmico e comercial para o desenvolvimento de aplicativos na área de visão computacional. O OpenCV é compatível nos sistemas operacionais Windows, Linux, Mac OS, iOS e Android, e possui interfaces para as linguagens C++, C, Python e Java.

Figura 3.5: Logotipo OpenCV.

3.1.7 Scikit-learn - Biblioteca de Programação

Para a implementação dos classificadores de *machine learning* foi utilizado a biblioteca de programação Scikit-learn. A biblioteca é *open source* e está sob licença BSD. Scikit-learn é compatível nos sistemas operacionais Windows, Linux e Mac OS [47].

Figura 3.6: Logotipo Scikit-learn.

3.1.8 Flask – Biblioteca de Programação

Para a comunicação entre o cliente e servidor foi utilizado o Flask. A biblioteca de programação Flask é um micro-framework de Python que é usado no desenvolvimento da Web e possui uma sintaxe fácil de usar.

Figura 3.7: Logotipo Flask.

3.1.9 appJar – Biblioteca de Programação

Na fase do protótipo, foi utilizado a biblioteca appJar para a construção de uma interface amigável para o usuário. O appJar é uma biblioteca Python de código aberto de para o desenvolvimento de GUIs (interfaces gráficas de usuário).

3.2 Metodologia para Aquisição de Imagem

Para aquisição de imagem foi utilizado uma câmera de infravermelho (utilizando as fontes de luz infravermelha) acoplado em um Raspberry Pi. As faces foram capturadas em um ambiente não controlado onde havia luz natural e artificial.

3.2.1 Distância entre Face Real e Câmera Infravermelha

Antes de capturar as imagens de faces falsas e faces reais, foram realizados alguns testes experimentais para determinar uma distância ideal entre a face e a câmera. Foi observado que quanto mais perto da câmera a face estiver, mais luz de infravermelho a face receberá, porém, se a face estiver muito próxima da câmera, a deteção de face não funcionará.

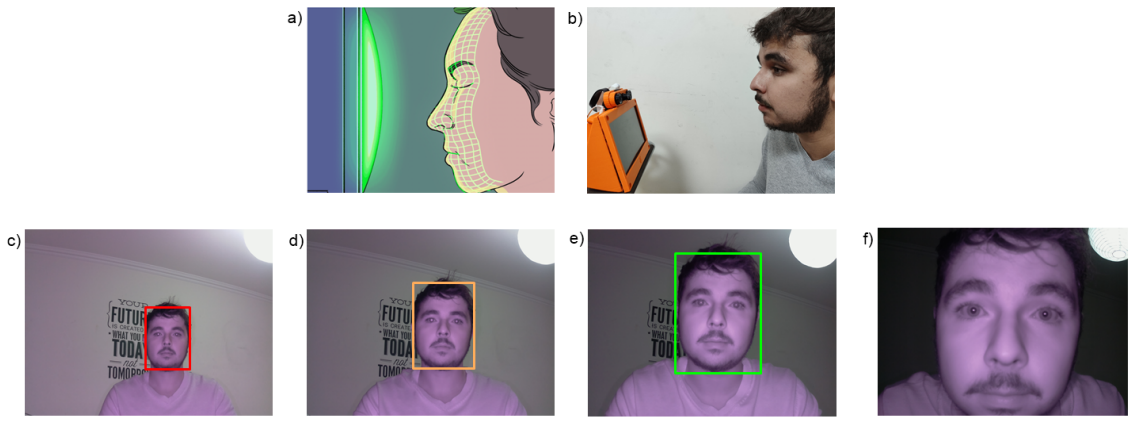


Figura 3.8: (a) Exemplo de uma luz infravermelho sendo emitida em uma face (reproduzido de Furtado [48]). (b) Posição do usuário em frente da câmera infravermelho. (c) Fotografia da câmera infravermelho em uma distância de 1 metro. (d) Fotografia da câmera infravermelho em uma distância de 75 centímetros. (e) Fotografia da câmera infravermelho em uma distância de 50 centímetros. (f) Fotografia da câmera infravermelho em uma distância de 25 centímetros.

Como a luz infravermelha é invisível aos olhos humano, a Figura 3.8(a) tenta simular um exemplo da face recebendo luz infravermelha. A Figura 3.8(b) é um exemplo real ilustrada na Figura 3.8(a). As Figuras 3.8(c), 3.8(d), 3.8(e), 3.8(f) são fotografias da

câmera de infravermelho de uma distância de 1 metro, 75 centímetros, 50 centímetros e 25 centímetros, respectivamente.

Após testes experimentais, as Figuras 3.8(c) e 3.8(d) não foram as ideais, pois, a interação de luz infravermelha na face foi pouca e o detector de face teve dificuldades para reconhecer faces. A Figura 3.8(f) também não foi a ideal, pois, o detector de face não conseguiu detectar a face. A Figura 3.8(e) foi considerada a ideal, pois teve uma interação de luz infravermelha ideal e o detector de face conseguiu detectar a face facilmente.

Foi concluído que a distância ideal para aquisição de imagem das faces reais é entre 35 e 55 centímetros para a potência das fontes de IR usadas.

3.2.2 Interação da Luz Infravermelha da Face Falsa e Real

A Figura 3.9 são exemplos de fotografias capturadas da câmera infravermelha deste trabalho. Na figura pode-se notar uma diferença entre a face real e falsa. A Figura 3.9(a) é uma face real onde é observado uma tonalidade roxa na pele (interação da luz infravermelha com a pele). A Figura 3.9(b) é uma face falsa do tipo digital e nela é possível observar uma tonalidade de cor diferente com a face real. A Figura 3.9(c) é uma face falsa do tipo impresso, porém se comparar (observando visualmente) com a face real, pode-se notar que a tonalidade de cor de ambas são semelhantes.

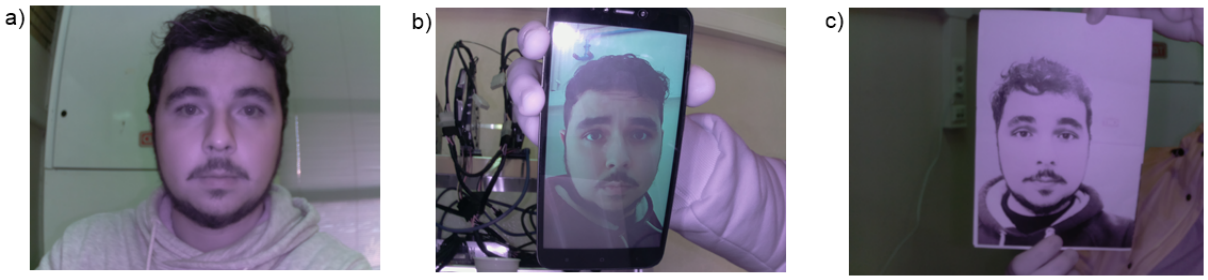


Figura 3.9: (a) Fotografia da face real com luz infravermelha. (b) Fotografia digital da face falsa com luz infravermelha. (c) Fotografia Impressa da face falsa com luz infravermelha.

3.2.3 Procedimento de Aquisição de Imagens de Faces Falsas e Reais

O experimento foi organizado para capturar faces falsas e faces reais. No total teve 15 participantes (13 homens e 2 mulheres) estudantes do IPB, entre 22 e 30 anos de idade. O aplicador (autor) orientou os participantes durante o procedimento.

Faces Reais

O procedimento para aquisição de imagens de faces reais foram os seguintes passos:

- Primeiramente o participante leu e ouviu os procedimentos para a participação do trabalho;
- Em seguida, o participante sentou-se e manteve seu olhar fixado para câmera infravermelha;
- Com uso da câmera infravermelha, o aplicador capturou algumas fotografias da face do participante;
- Depois de capturar as fotografias, o aplicador terminou o procedimento.

A Figura 3.10 mostra a posição ideal do participante durante o procedimento para fotografias de face real.

Faces Falsas

São três tipos de faces falsas: fotografia digital, vídeo digital e fotografia impresso em papel.

Fotografia e Vídeo Digital: as faces falsas foram obtidas através da câmera infravermelha. Como mostra a Figura 3.11(a), a face falsa está no ecrã do telemóvel. O aplicador capturou a face falsa através da câmera infravermelha acoplada no Rasbperrby Pi.

Fotografia Impressa: as faces falsas foram obtidas através da câmera infravermelha. Como mostra a Figura 3.11(b) abaixo, a face falsa está impressa em um papel A4 colorida. O aplicador capturou a face falsa através da câmera infravermelha acoplada no Rasbperrby Pi.

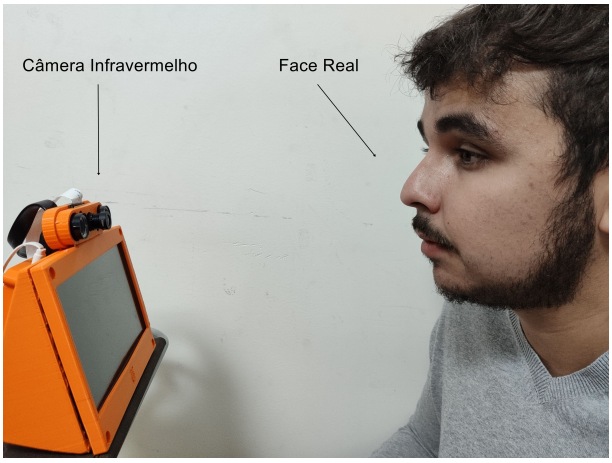


Figura 3.10: Posição ideal do participante para fotografias de Face Real

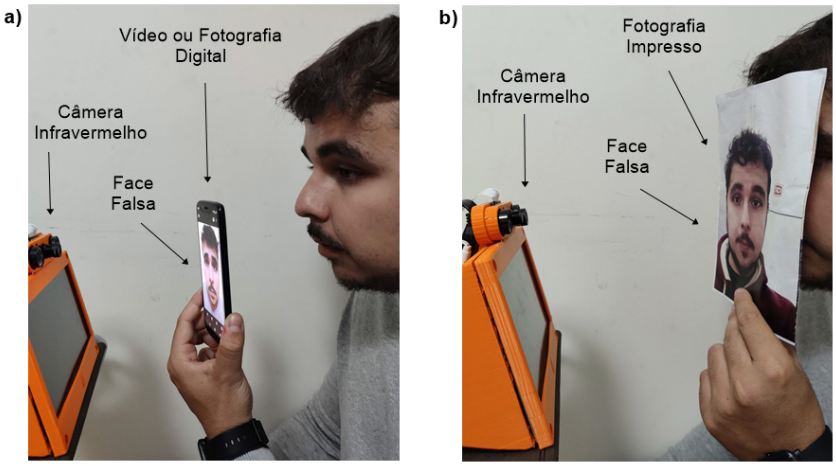


Figura 3.11: (a) Exemplo de Face Falsa Digital. (b) Exemplo de Face Falsa Impresso

Base de Dados de Faces Reais e Falsas

Cada participante tem 10 imagens de faces reais, 10 imagens de faces de fotografia digital e 10 imagens de faces de fotografia impressa, totalizando 150 imagens. As Figuras 3.12,

3.13 e 3.14 são exemplos de imagens de faces de um participante.

Figura 3.12: 10 Imagens de Faces Reais de um Participante.

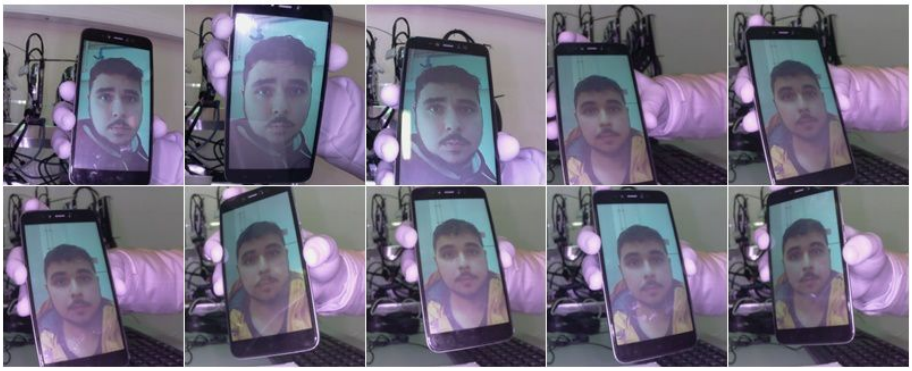


Figura 3.13: 10 Imagens de Faces Falsas Digitais de um Participante.

Figura 3.14: 10 Imagens de Faces Falsas Impressas de um Participante.

3.3 Extração de Características de Faces

Para extrair características das faces, foi utilizado o OpenFace e duas Regiões de Interesses (ROIs). No total foram 132 características faciais, 128 características extraídos pelo OpenFace e 4 características extraídos das ROIs no modelo L*a*b.

3.3.1 OpenFace – 128 características faciais

A biblioteca de programação OpenFace foi um dos métodos utilizado para obter características das faces. De acordo com o site [49], o OpenFace é uma biblioteca de reconhecimento facial com *deep learning*.

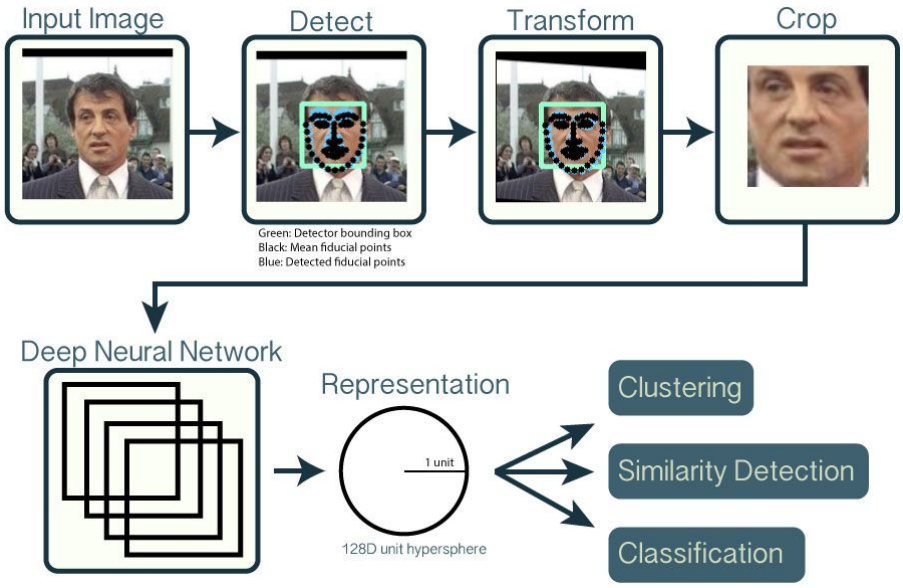


Figura 3.15: Arquitetura OpenFace (reproduzido de OpenFace[49])

O procedimento (que também pode ser visualizado na Figura 3.15) do OpenFace para única imagem de entrada são:

- Detectar face com modelos pré-treinados do dlib ou OpenCV;
- Recortar face detectada e utilizar como entrada na rede neural profunda;
- Usar uma *deep learning* para representar a face em uma hiperesfera unitária de 128 dimensões (características);

- Depois de adquirir as 128 características, aplicar técnicas de *cluster* ou classificação para concluir a tarefa de reconhecimento facial.

A Figura 3.16 abaixo mostra um exemplo com uso do OpenFace, onde a entrada é uma imagem com face e a saída é um vetor com 128 valores que representam as características da face.

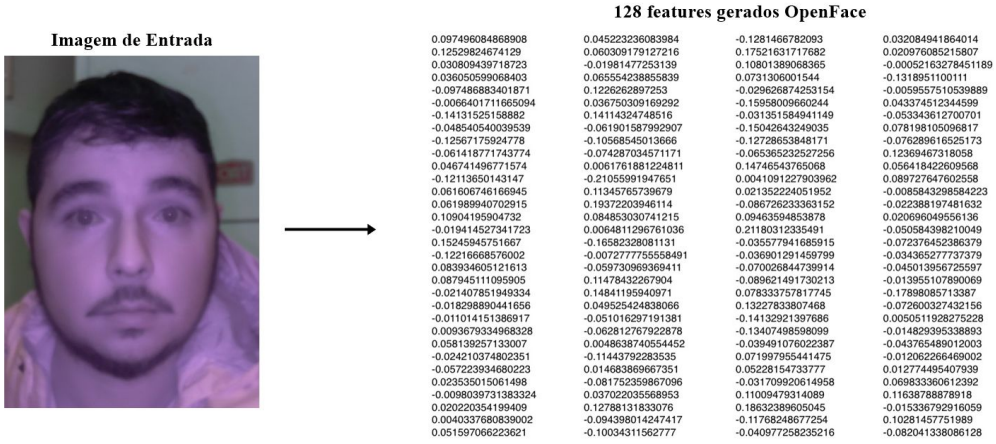


Figura 3.16: Caracteristicas faciais de uma face real utilizando OpenFace

3.3.2 Regiões de Interesses – 4 características faciais

A luz infravermelha interagindo com a pele da face é uma característica determinante para diferenciar face falsa e real desse trabalho. Porém algumas faces podem conter ruídos como maquiagem (batom, pó facial, lápis de olho), barba, bigode, cabelo e entre outras. Com isso foram definidas duas regiões que tem menos probabilidade de ruídos, as regiões são de forma quadricular e se localizam abaixo dos olhos e superior das bochechas.

Para obter as ROIs (ROI Esquerdo e ROI Direito) foi utilizado o método landmarks para detectar os pontos faciais. Na Figura 3.17(a) temos uma imagem de face com luz infravermelha. Pode-se observar que na figura contém pontos verdes, que são os pontos faciais (landmarks) e dois quadrados azuis, que são as ROIs definidas.

Com as ROIs definidas, é feito uma conversão do modelo RGB para L*a*b. Em seguida pega-se os canais A e B de cada ROI, com isso são obtidas quatro imagens ou matrizes que são calculadas as médias de cada uma delas. Essas médias são características faciais para o *dataset*.

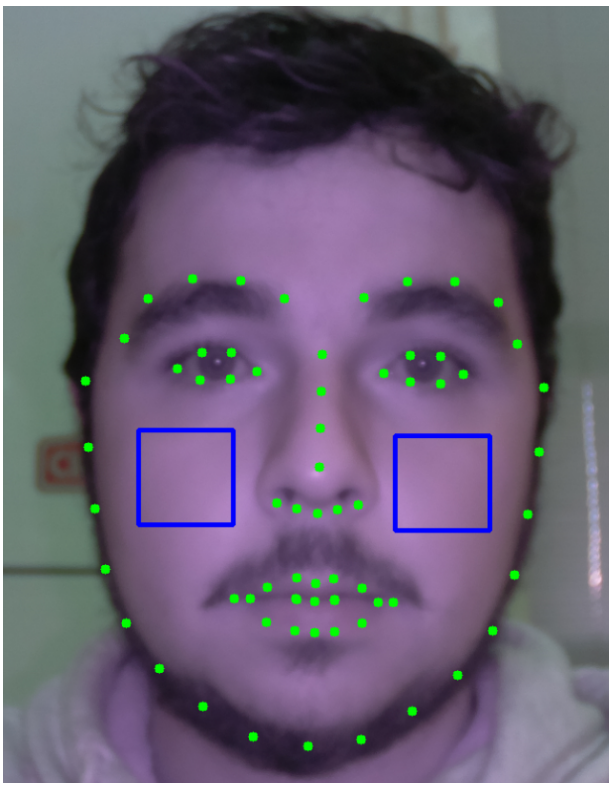


Figura 3.17: ROIs definidas para extração de características

Obtenção das ROIs Direito e Esquerdo

Foram utilizadas seis pontos faciais do landmarks como auxílio da obtenção dos pontos centrais das ROIs definidas na Figura 3.18. O cálculo da obtenção das ROIs são iguais, para exemplificar, será detalhado somente a ROI Direito.

O ponto central da ROI direito (ponto preto no centro da ROI direito na Figura 3.18) são as coordenadas da linha RLD e a coluna RCD (pontos azuis na Figura 3.18). Para obter o valor da linha RLD foi feito médias simples com os valores das linhas dos pontos 15 e 16. A equação para obter o RLD é:

$$RLD = \frac{linhaPonto15 + linhaPonto16}{2} \tag{3.1}$$

Para obter o valor da coluna RCD foi feito uma média simples com valores das colunas dos pontos 15 e 36. A equação para obter o RCD é:

$$RCD = \frac{colunaPonto15 + colunaPonto36}{2} \tag{3.2}$$

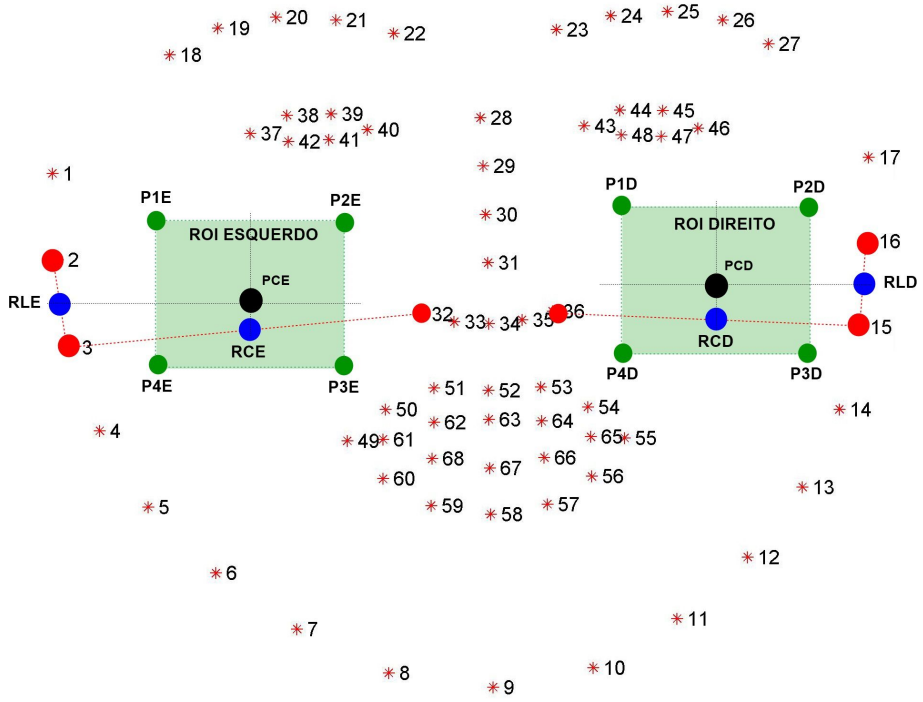


Figura 3.18: ROI Esquerda e Direita destacadas e posicionadas em dois quadrados verdes.

Os formatos das ROIs são quadrados e para obtê-los na biblioteca OpenCV, são necessários quatro pontos. Os tamanhos dos quadrados são ajustados conforme a distância entre a face e a câmara (detalhada na próxima seção). A seguir será descrita as equações dos pontos, onde, a letra D é um valor pré-definido que corresponde a distância entre câmera e face.

As equações para determinar o ponto P1D são:

$$P1D(linha) = linhaPCD - D \tag{3.3}$$

$$P1D(coluna) = colunaPCD - D \tag{3.4}$$

As equações para determinar o ponto P2D são:

$$P2D(linha) = linhaPCD - D \tag{3.5}$$

$$P2D(coluna) = colunaPCD + D \tag{3.6}$$

As equações para determinar o ponto P3D são:

$$P3D(linha) = linhaPCD + D \tag{3.7}$$

$$P3D(coluna) = colunaPCD + D \tag{3.8}$$

As equações para determinar o ponto P4D são:

$$P4D(linha) = linhaPCD + D \tag{3.9}$$

$$P4D(coluna) = colunaPCD - D \tag{3.10}$$

Depois de definir as ROIs direito e esquerdo, são geradas quatro matrizes que serão parâmetros de entradas para o cálculo da média (Figura 3.19). Essas médias se tornarão características faciais para deteção de face falsa e face real.

Média Aritmética dos Canais A e B no modelo L*a*b das ROIs Direito e Esquerdo

O valor da média aritmética é uma característica extraída da face. No total são quatro matrizes que são transformadas em vetor e são calculadas separadamente utilizando a média aritmética simples. A média aritmética simples é determinada pela seguinte expressão:

$$\bar{x} = \frac{x1 + x2 + x3 + x4 + ... + xn}{2} = \frac{1}{n}\sum_{k=1}^{n} xi \tag{3.11}$$

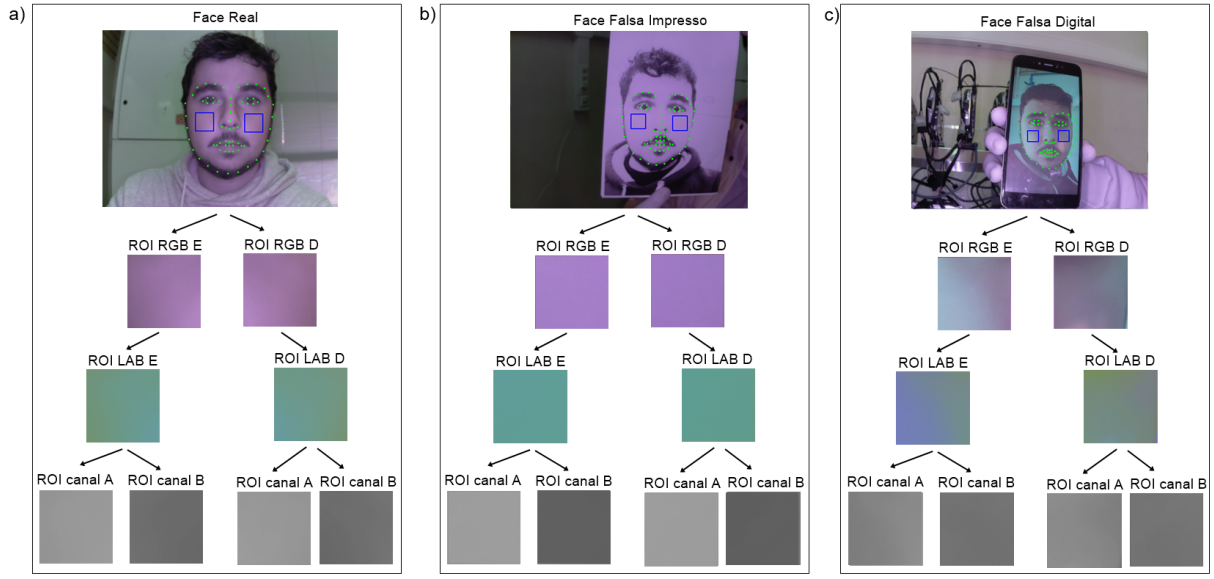


Figura 3.19: Ilustração da obtenção das 4 características faciais.

A Figura 3.20 mostra um resumo do procedimento para obter as quatros características. A entrada é uma imagem com face e são identificadas as ROIs. Cada ROI tem duas matrizes que são do canal A e B e cada matriz resultará num valor. O resultado final são quatros médias e estão em um vetor.

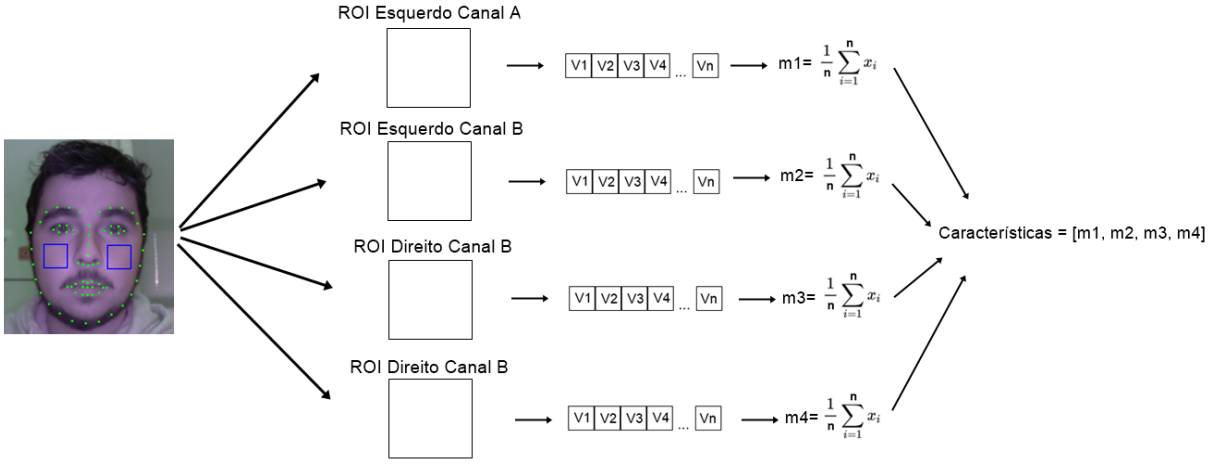


Figura 3.20: Obtenção das 4 características das ROIs

3.3.3 Metodologia do Processo de Extração de Características

Todo processo da extração de características facial é ilustrado no fluxograma (Figura 3.21). Um *script* em Python realiza esse procedimento. O parâmetro de entrada do *script* é uma imagem. Em seguida o detector irá identificar se a imagem de entrada é uma face ou não. Se o detector não identificar uma face na imagem, o *script* encerrará o procedimento, se identificar, o *script* utilizará a biblioteca OpenFace para extrair 128 características faciais e salvar.

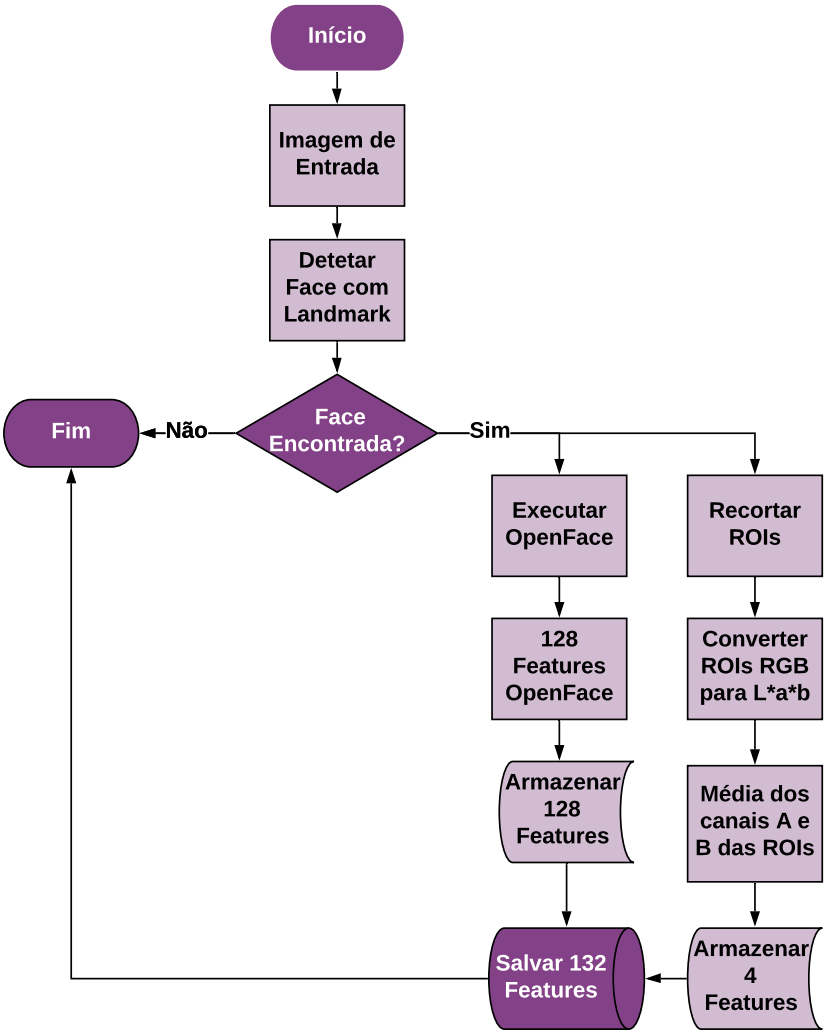


Figura 3.21: Procedimento de extração de características faciais.

Depois de obter as 128 características com OpenFace, o *script* executará o algoritmo para obter as outras 4 características faciais. Com a imagem de entrada, o *script* identificará a ROI direito e esquerdo. Em seguida será feito a conversão do modelo RGB para L*a*b das ROIs, e então, são calculados as médias das ROIs. No final do processo será armazenado em disco as 132 características faciais da face.

3.4 Dataset de Faces Falsas e Reais

Com as faces falsas e reais dos participantes e extração das características, foram geradas 3 *datasets* (Tabela 3.1) para análise com *machine learning*. Para face real (FR) foram obtidas 150 imagens de 15 participantes, para face falsa (FF) foram obtidas 300 imagens, sendo, 150 imagens de fotografia de face falsa impresso (FFI) e 150 imagens de fotografia de face falsa digital (FFD).

Tabela 3.1: Datasets gerados.

Dataset	**nº instâncias**	**nº características**
FR x FF - Openface	450	128
FR x FF - ROIs	450	4
FR x FF - OpenFace-ROIs	450	132

O dataset FR x FF – OpenFace, contém 150 faces reais e 150 faces falsas (75 faces fotografia digital e 75 faces fotografia impresso) com as características gerados do OpenFace.

O dataset FR x FF – ROIs, contém 150 faces reais e 150 faces falsas (75 faces fotografia digital e 75 faces fotografia impresso) com as características gerados das ROIs.

O dataset FR x FF – OpenFace-ROIs, contém 150 faces reais e 150 faces falsas (75 faces fotografia digital e 75 faces fotografia impresso) com as características do OpenFace e ROIs.

Os dados dos *datasets* foram normalizados para terem a mesma ordem de grandeza. Os dados foram normalizados em um intervalo de -1 a 1 pela fórmula [50]:

$$Xnormalizado = \frac{X - Xmin}{Xmax - Xmin} \tag{3.12}$$

3.5 Metodologia de Treinamento e Testes

Cada *dataset* foi dividido em treinamento e teste para os experimentos dos classificadores de *machine learning*. Para evitar um possível *overfitting*, foram separados manualmente as instâncias de faces reais e falsas. Foram 11 participantes para o treinamento e 4 para o teste (Figura 3.22). Desta maneira, a Tabela 3.2 mostra como foram organizados os dados de treino e teste.

Figura 3.22: Dataset separado em 11 participantes para treino e 4 participantes para teste.

Tabela 3.2: Datasets separados em treino e teste

Dataset	**Treino**	**Teste**
-	**nº instâncias**	**nº instâncias**
FR x FF - Openface	220	80
FR x FF - ROIs	220	80
FR x FF - OpenFace-ROIs	220	80

3.6 Parâmetros dos modelos de Machine Learning

Foi realizado um *grid search* para obter modelos com os melhores parâmetros. Os parâmetros dos seguintes classificadores (Tabela 3.3).

Tabela 3.3: Parâmetros dos modelos Machine Learning.

Classificador	**Parâmetros**
Árvore de Decisão	depth=2,3,4,5,6,7,8 e 9
Random Forest	trees=3,5,7,9,11,13 e 15
KNN	K=3,5,7,9,11,13 e 15
SVM	kernel=RBF, C=0.1,5,10 e 100 G=0.001, 0.01, 0.1, 1, 10 e 100.
MLP	Camadas ocultas= [(65,65,65), (65,65), (65)], Ativação camada ocultas=relu, Ativação ultima camada=sigmoide, otimização=adam, épocas=100.

3.7 Metodologia para Teste de Reconhecimento Facial com OpenFace

Em um sistema de reconhecimento facial, cada usuário tem faces cadastradas no sistema. Para essa dissertação, cada usuário tem 10 faces e cada face tem um vetor de 128 valores que representam as características faciais, esses valores são obtidos através da biblioteca OpenFace.

Para o reconhecimento facial, as faces são comparadas (128 características da face de entrada e 128 características da face que esta na base de dados) e é calculada a distancia euclidiana de ambas as faces, na qual, pode ser escrita através da Expressão 3.13.

$$d = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}, \tag{3.13}$$

O resultado da distância euclidiana será um valor entre 0 e 1, onde o valor 0 significa que as faces são iguais e valor 1 significa que as faces são diferentes.

A Figura 3.23 mostra um exemplo de teste para o reconhecimento de um 1 para N, na qual, a face do P01 é comparada com todas as faces da base de dados. Suponha-se um exemplo do mundo real, onde a autenticação em um sistema é feita somente utilizando a face do usuário.

A Figura 3.24 mostra um exemplo de teste para o reconhecimento de um 1 para 1, na

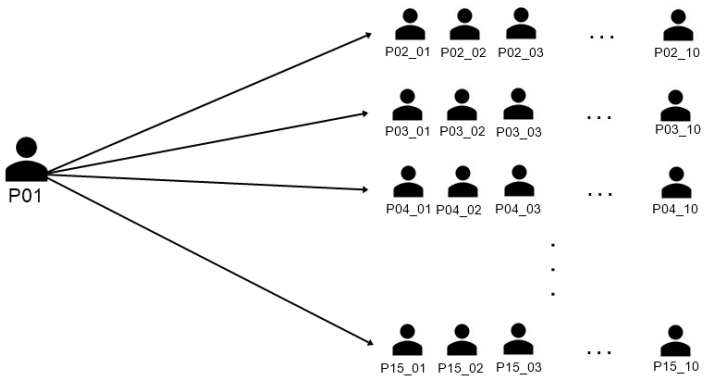


Figura 3.23: Exemplo de teste do reconhecimento 1 para N.

qual, a face do P02 é comparada somente com as faces do usuário P02. Suponha-se um exemplo do mundo real, onde a autenticação em um sistema é feita utilizando a face do usuário e uma palavra passe.

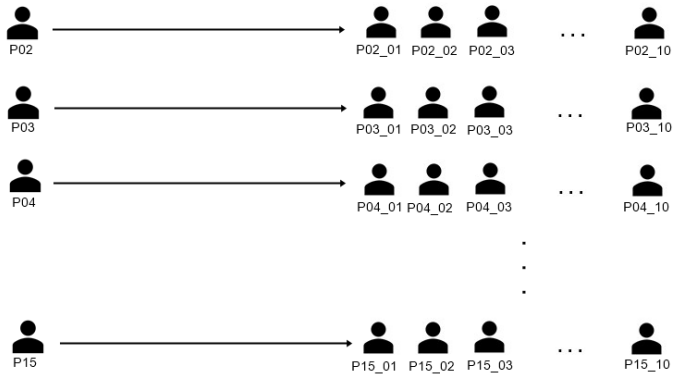


Figura 3.24: Exemplo de teste do reconhecimento 1 para 1.

Capítulo 4

Resultados e Discussão

Neste Capítulo 4 são mostrados os resultados e discutidos a Classificação da deteção de Face Real e Face Falsa, Comparação dos modelos de *Machine Learning*, Reconhecimento Facial com OpenFace e as Telas do Protótipo no Rasberry Pi.

4.1 Classificação da deteção de Face Real x Face Falsa

Nas próximas subseções serão apresentados os resultados em forma de gráficos, tabelas e matrizes de confusão das classificações dos dados de testes utilizando os seguintes modelos: Árvore de Decisão, *Random Forest*, KNN e MLP. Foram apresentados somente os melhores parâmetros para cada teste, analisados e avaliados 3 *datasets* que são:

- **FR x FF OpenFace**: Faces Reais e Face Falsas com 128 características extraídas pelo OpenFace;
- **FR x FF ROIs**: Face Reais x Face Falsas com 4 características extraídas pelas ROIs;
- **FR x FF OpenFace-ROIs**: Face Reais x Face Falsas com 132 características extraídas pelo OpenFace e pelas ROIs.

4.1.1 Árvore de Decisão

Nesta seção são apresentados os resultados do classificador de Árvore de Decisão.

O gráfico (Figura 4.1) mostra as acurácias da Árvore de Decisão com os parâmetros do *grid search*. O eixo horizontal do gráfico representa o nível de profundidade da árvore e o eixo vertical representa a percentagem da acurácia entre 50% e 80%.

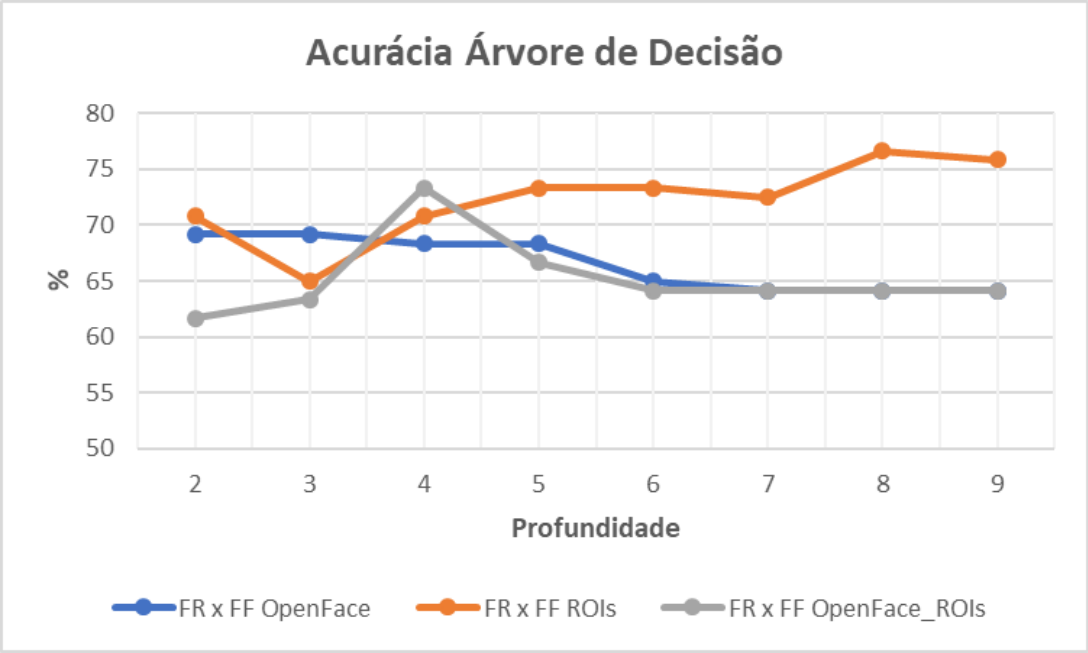


Figura 4.1: Acurácias da Árvore de Decisão na deteção de faces falsas

A partir da análise do gráfico, foi escolhido a melhor acurácia de cada *dataset*. Para o *dataset* FR x FF OpenFace a melhor acurácia foi com a profundidade 3 com 69.16%, a Tabela 4.1 abaixo mostra a matriz de confusão desse *dataset*/modelo. Para 80 faces falsas o modelo classificou 68 corretamente ou 85% de acerto (sensibilidade). Para 40 faces reais classificou 15 corretamente ou 37.50% de acerto (especificidade).

Tabela 4.1: Matriz de Confusão - FR x FF OpenFace - Profundidade 3

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	68	12
	Face Real	25	15

A melhor acurácia do *dataset* FR x FF ROIs foi com a profundidade 8 com 76.66%. A Tabela 4.2 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 64 corretamente ou 80% de acerto (sensibilidade). Para 40 faces reais classificou 21 corretamente ou 52.50% de acerto (especificidade).

Tabela 4.2: Matriz de Confusão - FR x FF ROIs - Profundidade 8

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	64	16
	Face Real	19	21

Por fim, a melhor acurácia do *dataset* FR x FF OpenFace-ROIs foi com a profundidade 4 com 73.33%. A Tabela 4.3 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 58 corretamente ou 72.50% de acerto (sensibilidade).Para 40 faces reais classificou 30 corretamente ou 75% de acerto (especificidade).

Tabela 4.3: Matriz de Confusão - FR x FF OpenFace-ROIs - Profundidade 4

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	58	22
	Face Real	10	30

A partir das matrizes de confusão foram calculados a Acurácia, Sensibilidade e Especificidade de cada *dataset* e são mostrados na Tabela 4.4. Para esse modelo o *dataset* FR x FF ROIs obteve a melhor acurácia com 77.66%, mas obteve uma percentagem de especificidade baixa se comparar com o *dataset* FR x FF OpenFace-ROIs. O *dataset* FR x FF OpenFace não obteve uma classificação satisfatório. Portanto, o melhor *dataset* para deteção de faces falsas e reais utilizando Árvore de Decisão foi o FR x FF OpenFace-ROIs.

Tabela 4.4: Resultados da Acurácia, Sensibilidade e Especifidade com Árvore de Decisão

Dataset	Profundidade	Acurácia	Sensibilidade	Especifidade
FR x FF - Openface	3	69.16%	85%	37.50%
FR x FF - ROIs	8	76.66%	80%	52.50%
FR x FF - OpenFace-ROIs	4	73.33%	72.50%	75%

4.1.2 Random Forest

Nesta seção são apresentados os resultados do classificador *Random Forest*.

O gráfico (Figura 4.2) mostra as acurácias da *Random Forest* com os parâmetros do *grid search*. O eixo horizontal do gráfico representa o número de árvores em uma *Random Forest* e o eixo vertical representa a percentagem da acurácia entre 50% e 90%.

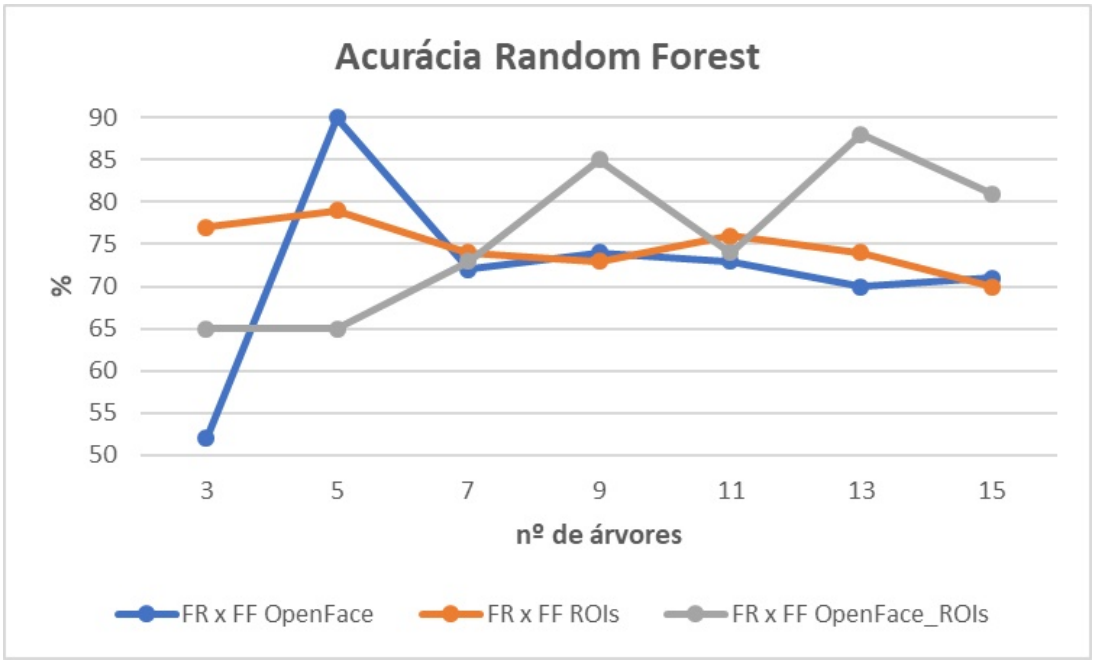


Figura 4.2: Acurácias da Random Forest na deteção de faces falsas

A partir da análise do gráfico, foi escolhido a melhor acurácia de cada *dataset*. Para o *dataset* FR x FF OpenFace a melhor acurácia foi com 5 árvores com 90%, a Tabela 4.5 abaixo mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 72 corretamente ou 90% de acerto (sensibilidade). Para 40 faces reais classificou 36 corretamente ou 90% de acerto (especificidade).

Tabela 4.5: Matriz de Confusão - FR x FF OpenFace - 5 Árvores

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	72	8
	Face Real	4	36

A melhor acurácia do dataset FR x FF ROIs foi com 5 árvores com 79.16%. A Tabela 4.6 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 70 corretamente ou 87.50% de acerto (sensibilidade). Para 40 faces reais classificou 25 corretamente ou 62.20% de acerto (especificidade).

Tabela 4.6: Matriz de Confusão - FR x FF ROIs - 5 Árvores

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	70	10
	Face Real	15	25

Por fim, a melhor acurácia do dataset FR x FD OpenFace-ROIs foi com 13 árvores com 88.33%. A Tabela 4.7 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 71 corretamente ou 88.75% de acerto (sensibilidade). Para 40 faces reais classificou 35 corretamente ou 87.50% de acerto (especificidade).

Tabela 4.7: Matriz de Confusão - FR x FF OpenFace-ROIs - 13 Árvores

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	71	9
	Face Real	5	35

A partir das matrizes de confusão foram calculados a Acurácia, Sensibilidade e Especificidade de cada *dataset* e são mostrados na Tabela 4.8 abaixo. Para esse modelo o *dataset* FR x FF OpenFace obteve a melhor acurácia, sensibilidade e especificidade com 90% respectivamente. Mas deve-se observar que o *dataset* FR x FF OpenFace-ROIs obteve resultados muitos próximos ao FR x FF OpenFace. O *dataset* FR x FF ROIs não obteve resultados satisfatório se comparado aos outros dois.

Tabela 4.8: Resultados da Acurácia, Sensibilidade e Especifidade com Random Forest

Dataset	Árvores	Acurácia	Sensibilidade	Especifidade
FR x FF - Openface	5	90%	90%	90%
FR x FF - ROIs	5	79.16%	87.50%	62.20%
FR x FF - OpenFace-ROIs	13	88.33%	88.75%	87.50%

4.1.3 KNN

Nesta seção são apresentados os resultados do classificador KNN. O gráfico (Figura 4.3) mostra as acurácias do KNN com os parâmetros do *grid search*. O eixo horizontal do gráfico representa o número K vizinhos do classificador KNN e o eixo vertical representa a percentagem da acurácia entre 60% e 90%.

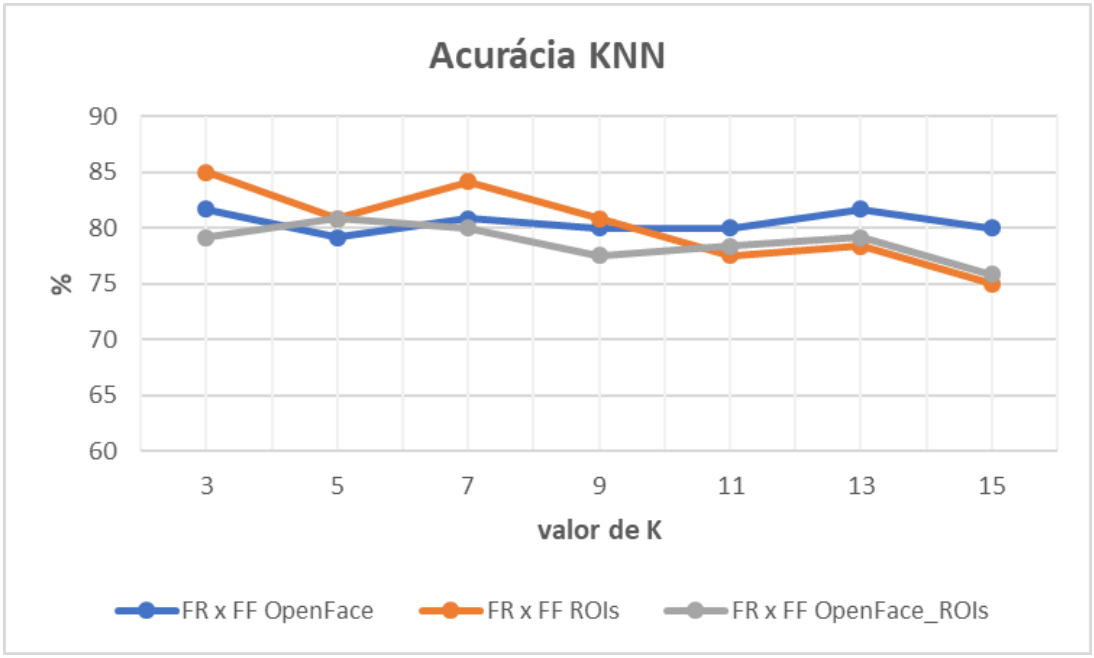


Figura 4.3: Acurácias do KNN na deteção de faces falsas

A partir da análise do gráfico, foi escolhido a melhor acurácia de cada dataset. Para o dataset FR x FF OpenFace a melhor acurácia foi com K=13 com 80%, a Tabela 4.9 abaixo mostra a matriz de confusão desse modelo. Para 80 faces falsas, o modelo classificou 67 corretamente ou 83.75% de acerto (sensibilidade) e para 40 faces reais classificou 29 corretamente ou 72.50% de acerto (especificidade).

Tabela 4.9: Matriz de Confusão - FR x FF OpenFace - K=13

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	67	13
	Face Real	11	29

A melhor acurácia do *dataset* FR x FF ROIs foi com K=7 com 85%. A Tabela 4.10 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 67 corretamente ou 80% de acerto (sensibilidade). Para 40 faces reais classificou 34 corretamente ou 85% de acerto (especificidade).

Tabela 4.10: Matriz de Confusão - FR x FF ROIs - K=7

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	67	13
	Face Real	6	34

Por fim, a melhor acurácia do *dataset* FR x FF OpenFace-ROIs foi com K=5 com 80.83%. A Tabela 4.11 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 67 corretamente ou 83.75% de acerto (sensibilidade). Para 40 faces reais classificou 30 corretamente ou 75% de acerto (especificidade).

Tabela 4.11: Matriz de Confusão - FR x FF OpenFace-ROIs - K=5

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	67	13
	Face Real	10	30

A partir das matrizes de confusão foram calculados a Acurácia, Sensibilidade e Especificidade de cada *dataset* e são mostrados na Tabela 4.12 abaixo. Para esse modelo o *dataset* FR x FF ROIs obteve a melhor acurácia, especificidade e sensibilidade . Os datasets FR x FF OpenFace e FR x FF OpenFace-ROIs obtiveram resultados semelhantes.

Tabela 4.12: Resultados da Acurácia, Sensibilidade e Especifidade com KNN

Dataset	K	Acurácia	Sensibilidade	Especifidade
FR x FF - Openface	11	80%	83.75%	72.50%
FR x FF - ROIs	7	85%	83.75%	85%
FR x FF - OpenFace-ROIs	5	80.33%	87.75%	75%

4.1.4 SVM

Nesta seção são apresentados os resultados do classificador SVM com *kernel* rbf. O gráfico (Figura 4.4) mostra as acurácias do SVM com os parâmetros do *grid search*. O eixo horizontal do gráfico representa os parâmetros gamma e C, e o eixo vertical representa a percentagem da acurácia entre 50% e 100%.

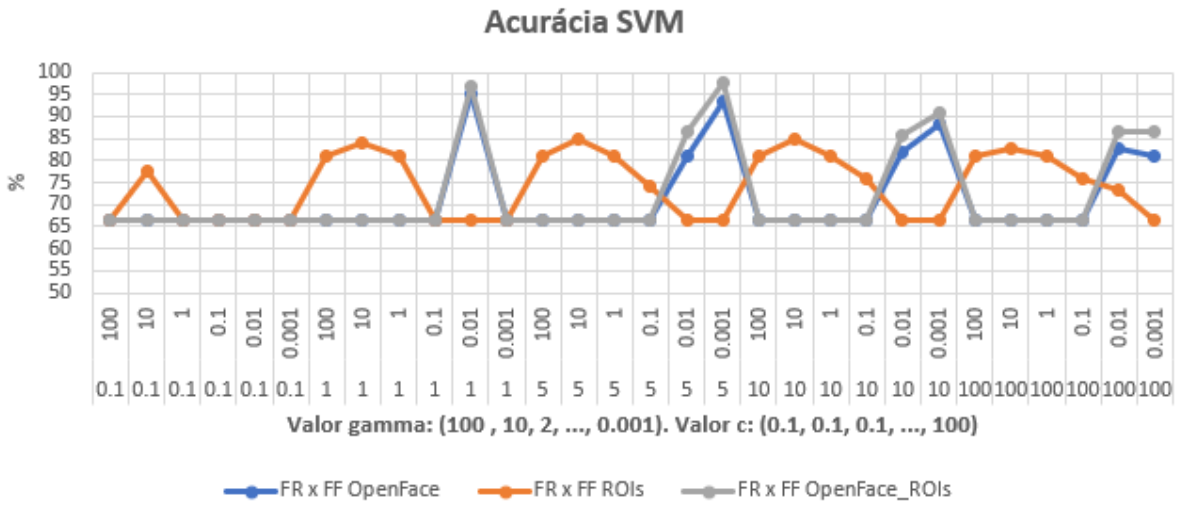


Figura 4.4: Acurácias do SVM na deteção de faces falsas

A partir da análise do gráfico, foi escolhido a melhor acurácia de cada *dataset*. Para o *dataset* FR x FF OpenFace a melhor acurácia foi com C=1 e gamma=0.01 com 95%. A Tabela 4.13 abaixo mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 77 corretamente ou 96.25% de acerto (sensibilidade). Para 40 faces reais classificou 37 corretamente ou 92.50% de acerto (especificidade).

Tabela 4.13: Matriz de Confusão - FR x FF OpenFace - C=1 e gamma=0.01

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	77	3
	Face Real	3	37

A melhor acurácia do *dataset* FR x FF ROIs foi com C=5 e gamma=10 com 85%. A Tabela 4.14 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 72 corretamente ou 90% de acerto (sensibilidade). Para 40 faces reais classificou 30 corretamente ou 85% de acerto (especificidade).

Tabela 4.14: Matriz de Confusão - FR x FF ROIs - C=5 e gamma=10

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	72	8
	Face Real	10	30

Por fim, a melhor acurácia do *dataset* FR x FF OpenFace-ROIs foi com C=5 e gamma=0.001 com 97.50%. A Tabela 4.15 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 80 corretamente ou 100% de acerto (sensibilidade). Para 40 faces reais classificou 37 corretamente ou 92.50% de acerto (especificidade).

Tabela 4.15: Matriz de Confusão - FR x FF OpenFace-ROIs - C=5 e gamma=0.001

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	80	0
	Face Real	3	37

A partir das matrizes de confusão foram calculados a Acurácia, Sensibilidade e Especificidade de cada *dataset* e são mostrados na Tabela 4.16 abaixo. Para esse modelo, o *dataset* FR x FF OpenFace-ROIs obteve a melhor acurácia, sensibilidade e especificidade (*dataset* FR x FF OpenFace obteve o mesmo valor de especificidade). O *dataset* FR x FF OpenFace obteve resultados próximos ao FR x FF OpenFace-ROIs e o *dataset* FR x FF ROIs não obteve resultados satisfatórios.

Tabela 4.16: Resultados da Acurácia, Sensibilidade e Especifidade com SVM

Dataset	C e gamma	Acurácia	Sensibilidade	Especifidade
FR x FF - Openface	C:1, gamma:0.01	95%	96.25%	92.50%
FR x FF - ROIs	C:5, gamma:10	85%	90%	75%
FR x FF - OpenFace-ROIs	C:5, gamma:0.001	97.50%	100%	92.50%

4.1.5 MLP

Nesta seção são apresentados os resultados do classificador MLP. O gráfico (Figura 4.5) mostra as acurácias do MLP com os parâmetros do *grid search*. O eixo horizontal do gráfico representa o número de camadas e a quantidade de neurônios de cada camada do MLP e o eixo vertical representa a percentagem da acurácia entre 50% e 95%.

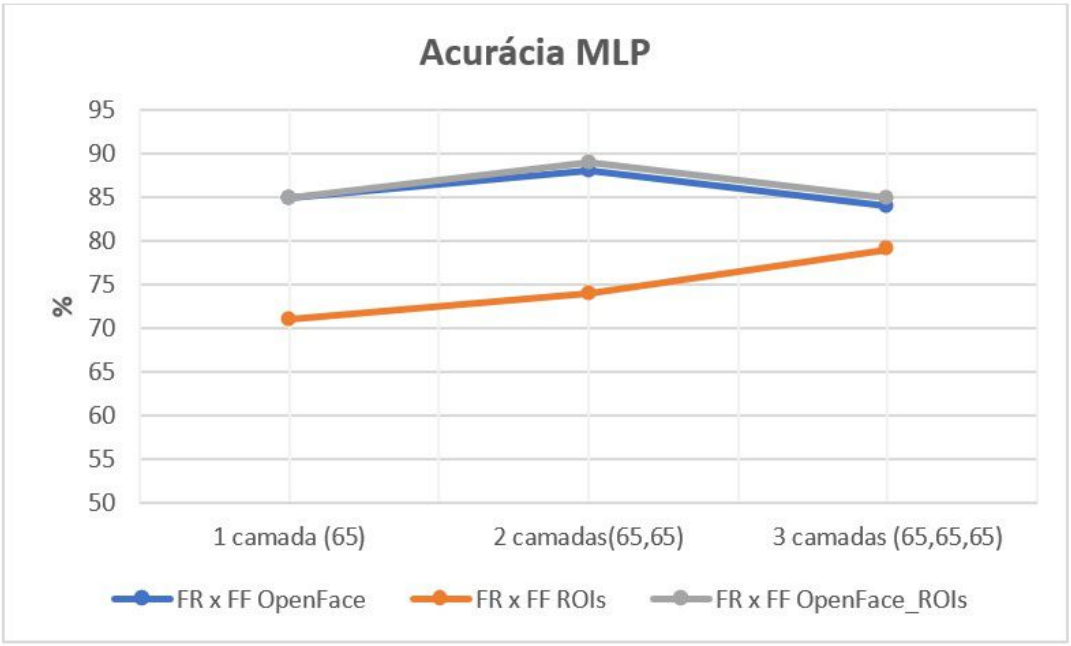


Figura 4.5: Acurácias do MLP na deteção de faces falsas

A partir da análise do gráfico, foi escolhido a melhor acurácia de cada *dataset*. Para o *dataset* FR x FF OpenFace a melhor acurácia foi com 2 camadas com 88.33%. A Tabela 4.17 abaixo mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 73 corretamente ou 91.25% de acerto (sensibilidade). Para 40 faces reais classificou 33 corretamente ou 82.50% de acerto (especificidade).

Tabela 4.17: Matriz de Confusão - FR x FF OpenFace - 2 camadas (65,65)

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	73	7
	Face Real	7	33

A melhor acurácia do *dataset* FR x FF ROIs foi com 3 camadas com 79,16%. A Tabela 4.18 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 69 corretamente ou 86.25% de acerto (sensibilidade). Para 40 faces reais classificou 26 corretamente ou 65% de acerto (especificidade).

Tabela 4.18: Matriz de Confusão - FR x FF ROIs - 3 camadas (65,65,65)

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	69	11
	Face Real	14	26

Por fim, a melhor acurácia do *dataset* FR x FF OpenFace-ROIs foi com 2 camadas com 89.16%. A Tabela 4.19 mostra a matriz de confusão desse modelo. Para 80 faces falsas o modelo classificou 67 corretamente ou 83.25% de acerto (sensibilidade). Para 40 faces reais classificou 40 corretamente ou 100% de acerto (especificidade).

Tabela 4.19: Matriz de Confusão - FR x FF OpenFace-ROIs - 2 camadas (65,65)

		Predita	
	total:120	Face Falsa	Face Real
Real	Face Falsa	67	13
	Face Real	0	40

A partir das matrizes de confusão foram calculados a Acurácia, Sensibilidade e Especificidade de cada uma, e são mostrados na Tabela 4.20 abaixo. Para esse modelo, o melhor *dataset* foi FR x FF OpenFace-ROIs pois obteve os melhores resultados.

Tabela 4.20: Resultados da Acurácia, Sensibilidade e Especifidade com MLP

Dataset	parâmetros	Acurácia	Sensibilidade	Especifidade
FR x FF - Openface	2 (65,65)	88.33%	91.25%	82.50%
FR x FF - ROIs	3 (65,65,65)	79.16%	86.25%	65%
FR x FF - OpenFace-ROIs	2 (65,65)	89.16%	83.75%	100%

4.2 Comparação dos Modelos de Machine Learning na deteção de Faces Falsas

Nesta seção serão comparados os resultados das classificações dos modelos de *machine learning*. A Tabela 4.21 e o gráfico (Figura 4.6) mostra as percentagens da acurácia.

Tabela 4.21: Resultados da Acurácia, Sensibilidade e Especifidade dos datasets.

Dataset	Árvore Decisão	Random Forest	KNN	SVM	MLP
FR x FF - Openface	69.16%	90%	80%	95%	88.33%
FR x FF - ROIs	76.66%	73.16%	85%	85%	79.16%
FR x FF - OpenFace-ROIs	73.33%	88.33%	80.83%	97.5%	89.16%

Para o *dataset* FR x FF OpenFace, o SVM foi o melhor classificador com 95% de acertos. Os classificadores Random Forest e MLP obtiveram resultados interessantes com 90% e 88.13% de acertos, respectivamente.

Para o *dataset* FR x FF ROIs, o SVM e KNN foram os melhores classificadores com 85% de acertos. Os classificadores Árvore de Decisão, Random Forest e MLP não obtiveram resultados satisfatórios onde obtiveram menos de 80% de acertos.

Para o *dataset* FR x FF OpenFace-ROIs, o SVM foi o melhor classificador com 97.50% de acerto. Os classificadores Random Forest e MLP obtiveram resultados interessantes com 88.33% e 89.16% de acertos, respectivamente.

Portanto, o SVM foi o melhor classificador, pois obteve as melhores acurácias em todos os *datasets*. A melhor acurácia de toda a Tabela 4.21 foi de 97.50% com SVM e *dataset* FR x FF OpenFace-ROIs.

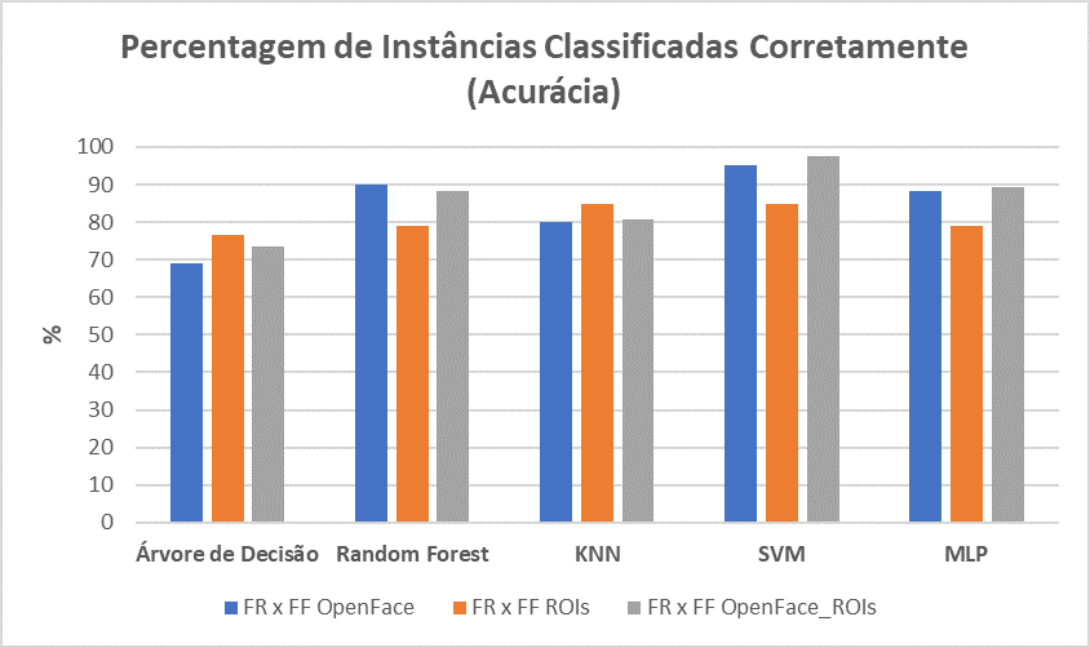


Figura 4.6: Resultado da Acurácia de todos os classificadores e datasets

4.3 Reconhecimento Facial com OpenFace

Nesta seção serão apresentados os resultados do reconhecimento facial dos participantes. A Figura 4.7 mostra o resultado do teste de reconhecimento de 1 para N. Pode-se observar que um limiar rejeitador é uma distância euclidiana de 0.6. Por exemplo, usuário 01 inseriu sua imagem de face como entrada, então será calculada a distância euclidiana da imagem de face de usuário 01 com todas as faces da base de dados. As faces que não pertencem ao usuário 01 terão um valor da distância euclidiana maior que 0.6.

A Figura 4.8 mostra o resultado de teste de reconhecimento de 1 para 1, pode-se observar que um limiar aceitador é uma distância euclidiana de 0.2. Por exemplo, usuário 01 inseriu sua imagem de face como entrada, então será calculada a distância euclidiana da imagem de face de usuário 01 com todas as faces do usuário 01 (se tiver cadastradas). Espera-se que os valores da distância euclidiana das faces comparadas sejam menor que 0.2.

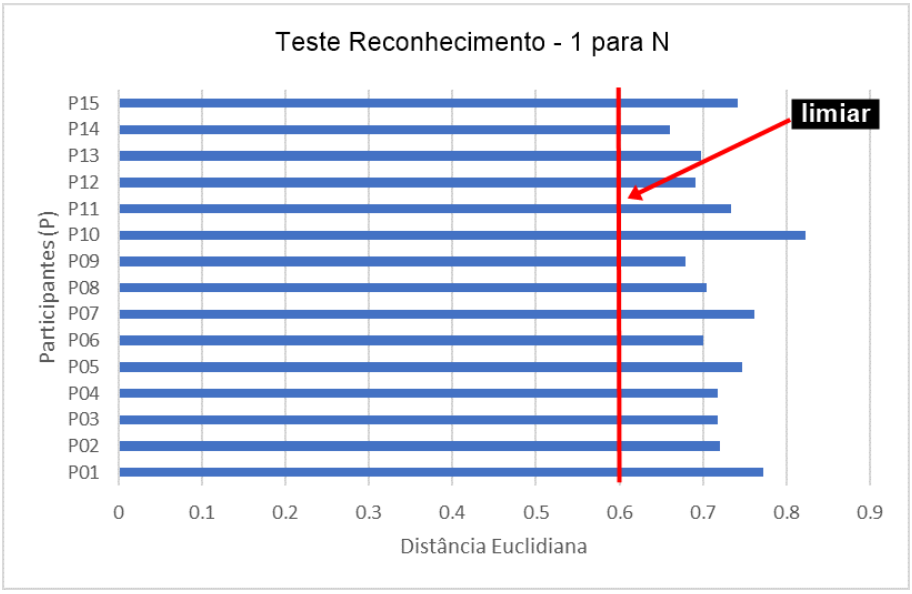


Figura 4.7: Resultado do reconhecimento facial do teste - 1 para N.

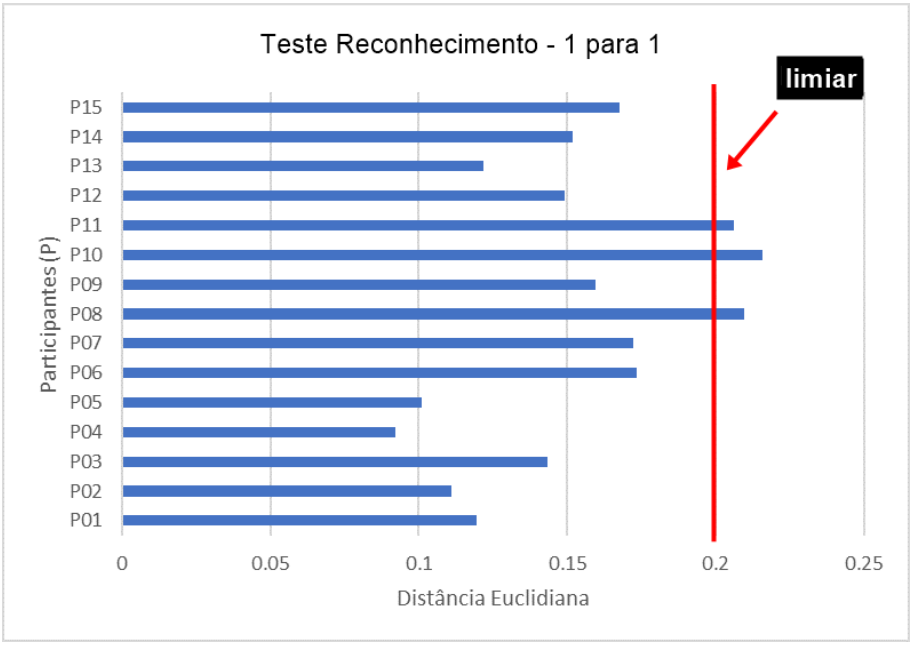


Figura 4.8: Resultado do reconhecimento facial do teste - 1 para 1.

4.4 Telas do Protótipo no Rasberry Pi

Foram construído telas como protótipo para o Raspberry Pi. A Figura 4.9 é a tela inicial do sistema, onde tem quatros botões. As funcionalidades dos botões são:

- **Yes**: iniciará o processo de reconhecimento facial (Figura X);
- **No**: encerrará o programa;
- **More Info**: irá exibir informações sobre o programa;
- **Close**: irá fechar todo o programa.

Figura 4.9: Tela Inicial

A Figura 4.10 exibe a tela de autenticação do sistema de reconhecimento facial onde contêm informações do nome da disciplina, nome do usuário, palavra passe e o resultado reconhecimento facial. Nesta tela o usuário pode inserir a palavra passe ou registro acadêmico. O sistema irá capturar uma imagem do usuário e enviará para o servidor para o processamento. Depois de alguns segundos irá exibir para o usuário o resultado.

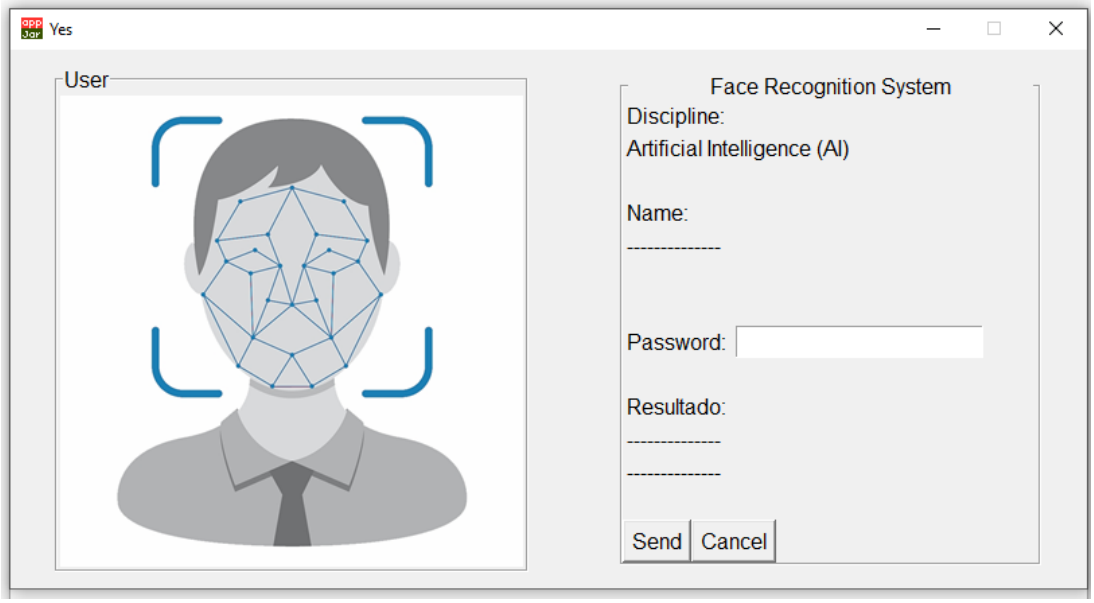


Figura 4.10: Tela da autenticação do usuário.

A Figura 4.11 mostra um exemplo do resultado da autenticação do reconhecimento facial. Pode-se observar que exibiu o nome e o resultado, que neste caso é o usuário Eduardo Carvalho Nunes e uma Face Real.

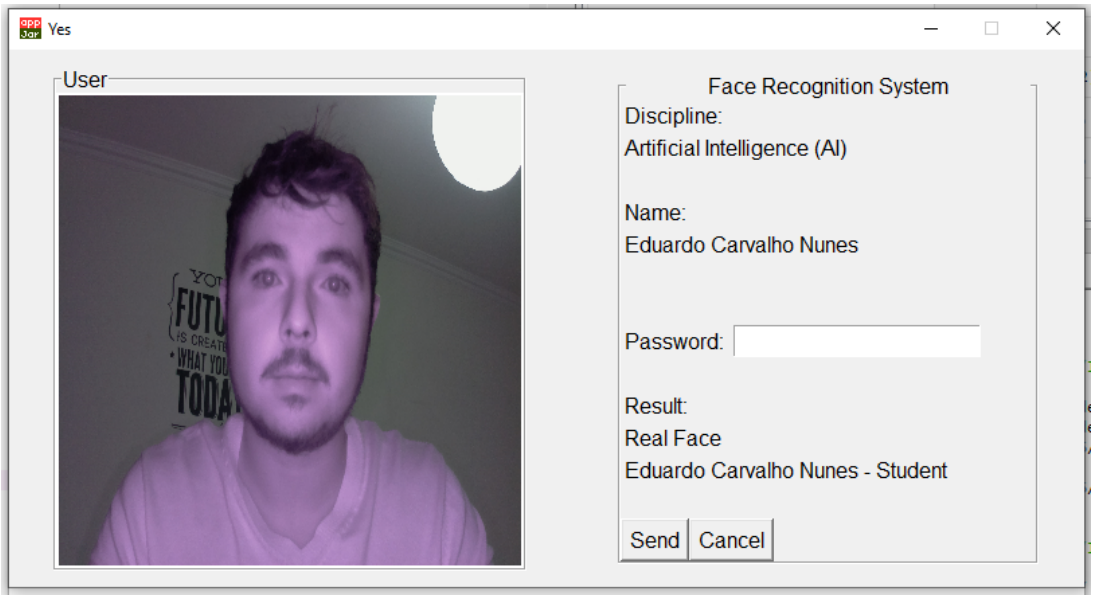


Figura 4.11: Tela da autenticação do usuário com a resposta de FACE REAL e identificação do usuário

A Figura 4.12 mostra um exemplo do resultado da autenticação do reconhecimento facial. Pode-se observar que resultado é uma Face Falsa, neste caso não é necessário efetuar o reconhecimento.

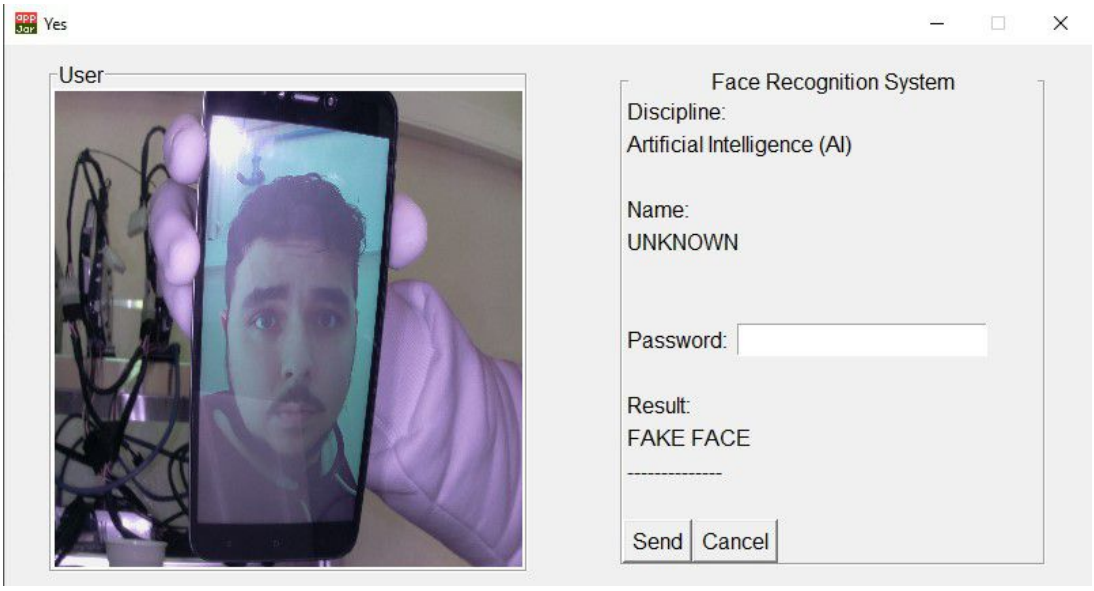


Figura 4.12: Tela da autenticação do usuário com a resposta de FACE FALSA DIGITAL

A Figura 4.13 mostra um exemplo do resultado da autenticação do reconhecimento facial. Pode-se observar que resultado também é uma Face Falsa, neste caso não é necessário efetuar o reconhecimento.

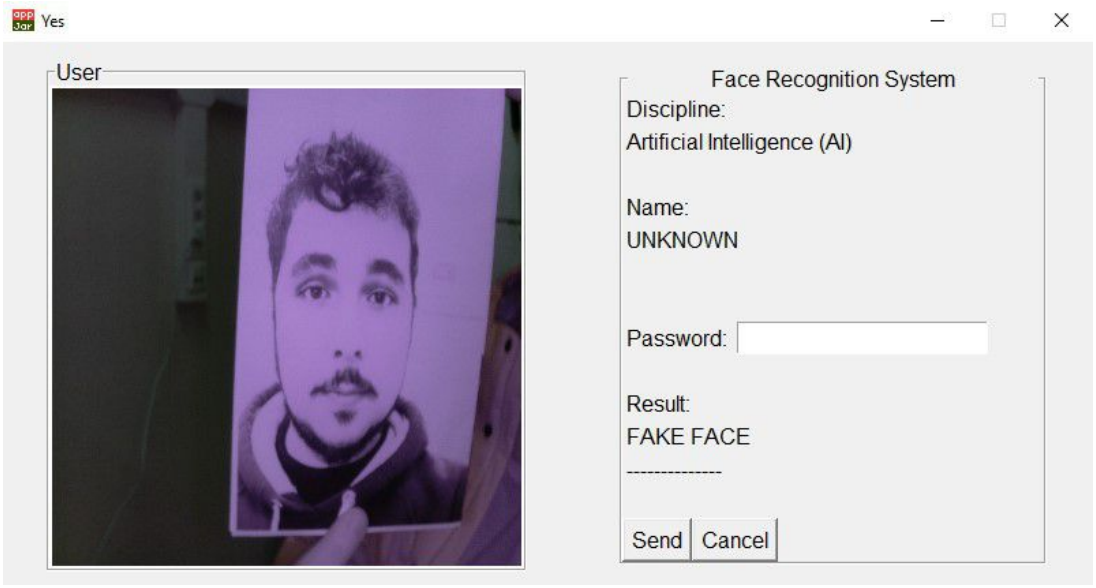


Figura 4.13: Tela da autenticação do usuário com a resposta de FACE FALSA IMPRESSO

Capítulo 5

Conclusões e Trabalhos Futuros

Neste Capítulo 5 serão apresentadas as conclusões dessa dissertação e descrição e sugestões para trabalhos futuros.

5.1 Conclusões

Esta dissertação apresentou um estudo para deteção de faces falsas utilizando *machine learning*, onde pode ser usado como um pré-processamento para um sistema de reconhecimento facial.

Foram criados 3 *datasets* para o treinamento e testes dos modelos de *machine learning*. Para a criação dos *datasets* a câmera com luz infravermelha NIR e o Raspberry Pi 3 demonstraram ser eficientes para aquisição de imagens de faces falsas e faces reais.

Foram implementados cinco classificadores de *machine learning* para detectar faces falsas que são: Árvore de Decisão, *Random Forest*, KNN, SVM e MLP. O modelo que apresentou o melhor resultado para o *dataset* FR x FF OpenFace foi o classificador SVM com 95% de acurácia. Para o *dataset* FR x FF ROIs, dois classificadores apresentaram os melhores resultados, KNN e SVM com 85%. Para o *dataset* FR x FF OpenFace-ROIs o classificador SVM apresentou o melhor resultado com 97.50% de acurácia. Para todos os *datasets* o classificador SVM foi o que mais se destacou comparado com os demais.

O classificador SVM foi o escolhido para deteção de faces falsas deste trabalho com

acurácia de 97.50%. Este resultado é satisfatório se comparado com o trabalho de Steiner [40] com 99.28% de acurácia e Zhang [41] com 65.3% (luz infravermelha) e 96.7% (luz infravermelha, visível e profundidade).

Foi observado que a concatenação das características utilizando a biblioteca OpenFace e as ROIs foi fundamental para diferenciar as faces falsas e faces reais.

Nos testes de reconhecimento facial, a biblioteca OpenFace mostrou-se eficaz para o reconhecimento dos participantes. Para um sistema que utiliza autenticação no formato 1 para N, o limiar recomendado é de 0.6. Para um sistema que utiliza autenticação no formato 1 para 1, o limiar recomendado é de 0.25.

Um protótipo foi desenvolvido para encenar um sistema de biometria facial com detector de faces falsas. Lembrando que todas as bibliotecas de programação utilizadas neste trabalho são *open source* e os *hardwares* são de baixo custo, se comparados com os sistemas de reconhecimento facial implementadas em organizações.

De modo geral, foi concluído que a utilização de uma câmera infravermelha NIR com técnicas de *machine learning* demonstraram que é possível criar um detector de faces falsas, na qual pode ser executada automaticamente em um sistema de reconhecimento facial sendo possível integrar no sistema de controle de presença nos terminais disponíveis no IPB.

5.2 Trabalhos Futuros

Como trabalhos futuros é sugerido aumentar a base de dados com novos participantes ampliando a diversidade do *dataset*. Com o aumento do *dataset* é sugerido novos treinamentos e testes dos modelos de *machine learning* para a deteção de face falsa e face real. Também é sugerido a criação de faces falsas do tipo máscaras 3D.

É sugerida também a criação de uma aplicação *Mobile* no cliente (Raspberry Pi) para interação do usuário no momento da autenticação e um sistema Web para o funcionário responsável de cadastrar novas faces e participantes.

Até o momento (outubro de 2019) está sendo discutido a implementação do protótipo

em uma empresa na cidade de Bragança para fins de testes e simulações em um ambiente real.

Bibliografia

[1] A. K. Jain e S. Z. Li, *Handbook of face recognition*. Springer, 2011.

[2] R. Christina Larson. (fev. de 2018). China's massive investment in artificial intelligence has an insidious downside, URL: https://www.sciencemag.org/news/2018/02/china-s-massive-investment-artificial-intelligence-has-insidious-downside.

[3] F. Payão. (mar. de 2019). Carnaval tem primeiro preso via câmera de reconhecimento facial no Brasil, URL: https://www.bbc.com/news/technology-49728301?intlink_from_url=https://www.bbc.com/news/topics/c12jd8v541gt/facial-recognition&link_location=live-reporting-story.

[4] L. Kelion. (set. de 2019). Gatwick Airport commits to facial recognition tech at boarding, URL: https://www.tecmundo.com.br/seguranca/139262-carnaval-tem-primeiro-preso-via-camera-reconhecimento-facial-brasil.htm.

[5] Globo. (out. de 2019). Biometria facial começa a ser implantada nos ônibus de Belém, URL: https://g1.globo.com/pa/para/noticia/2019/10/14/biometria-facial-comeca-a-ser-implantada-nos-onibus-de-belem-a-partir-desta-segunda-14.ghtml.

[6] P. Kulche. (abr. de 2019). O reconhecimento de rosto no smartphone nem sempre é seguro, URL: https://www.consumentenbond.nl/veilig-internetten/gezichtsherkenning-te-hacken.

[7] C. Pautasso e E. Wilde, "Why is the web loosely coupled?: a multi-faceted metric for service design", em *Proceedings of the 18th international conference on World wide web*, ACM, 2009, pp. 911–920.

[8] S. Nasser, S. Dascalu e G. Vert, "User Interface Design of the Interactive Fingerprint Recognition (INFIR) System.", em *Security and Management*, 2006, pp. 371–377.

[9] J. L. Wayman, "Fundamentals of biometric authentication technologies", *International Journal of Image and Graphics*, vol. 1, n.º 01, pp. 93–113, 2001.

[10] A. Jain, L. Hong e S. Pankanti, "Biometric identification", *Communications of the ACM*, vol. 43, n.º 2, pp. 90–98, 2000.

[11] T. TRANSACTIONS, "Face Spoofing and Counter-Spoofing: A Survey of State-of-the-art Algorithms", 2017.

[12] S. Liu, P. C. Yuen, S. Zhang e G. Zhao, "3D mask face anti-spoofing with remote photoplethysmography", em *European Conference on Computer Vision*, Springer, 2016, pp. 85–100.

[13] M. Marengoni e S. Stringhini, "Tutorial: Introdução à visão computacional usando opencv", *Revista de Informática Teórica e Aplicada*, vol. 16, n.º 1, pp. 125–160, 2009.

[14] M. Sanches, "Previsibilidade na reprodução da cor", 2016.

[15] R. C. Gonzalez e R. C. Woods, *Processamento digital de imagens* . Pearson Educación, 2009.

[16] H. Frank. (fev. de 2006). Electromagnetic spectrum-es.svg, URL: `https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum-es.svg`.

[17] L. Rotava, "Algoritmos de tempo real para melhoramento de imagens capturadas no espectro do infravermelho projetados para sıntese em FPGA", tese de doutoramento, Universidade de São Paulo.

[18] A. M. Smith, M. C. Mancini e S. Nie, "Bioimaging: second window for in vivo imaging", *Nature nanotechnology*, vol. 4, n.º 11, p. 710, 2009.

[19] A. A. Informáticas. (fev. de 2011). Modelo RGB e as suas aplicações, URL: http://asaplicacoesinformaticas.blogspot.com/2011/02/modelo-rgb-e-suas-aplicacoes.html.

[20] R. Korifi, Y. Le Dréau, J.-F. Antinelli, R. Valls e N. Dupuy, "CIEL a b color space predictive models for colorimetry devices–Analysisof perfume quality", *Talanta*, vol. 104, pp. 58–66, 2013.

[21] A. L. Samuel, "Some studies in machine learning using the game of checkers. II—recent progress", em *Computer Games I*, Springer, 1988, pp. 366–400.

[22] M. Mohri, A. Rostamizadeh e A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[23] T. M. Mitchell, *Machine Learning*, 1ª ed. New York, NY, USA: McGraw-Hill, Inc., 1997, ISBN: 0070428077, 9780070428072.

[24] S. J. Russell e P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[25] J. Chugh. (dez. de 2018). Types of Machine Learning and Top 10 Algorithms Everyone Should Know, URL: https://blogs.oracle.com/ai/types-of-machine-learning-and-top-10-algorithms-everyone-should-know.

[26] C. Smith, *Decision trees and random forests: a visual introduction for beginners*. Blue Windmill Media, 2017.

[27] S. Raschka e V. Mirjalili, *Python machine learning*. Packt Publishing Ltd, 2017.

[28] V. Vapnik, *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[29] S. Haykin, *Redes neurais: princıpios e prática*. Bookman Editora, 2007.

[30] I. A. Basheer e M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", *Journal of microbiological methods*, vol. 43, n.º 1, pp. 3–31, 2000.

[31] S. SHARMA. (jul. de 2017). Activation Functions in Neural Networks, URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[32] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning Cambridge*, 2016.

[33] Y. Bengio et al., "Learning deep architectures for AI", *Foundations and trends® in Machine Learning*, vol. 2, n.º 1, pp. 1–127, 2009.

[34] H.-D. Wehle, "Machine Learning, Deep Learning, and AI: What's the Difference?", em *International Conference on Data scientist innovation day, Bruxelles, Belgium*, 2017.

[35] J. Sharma. (jul. de 2018). Insights Of The Machine Learning And The Deep Learning, URL: http://blog.thinkwik.com/insights-of-the-machine-learning-and-the-deep-learning/.

[36] H. H. Aghdam e E. J. Heravi, "Guide to Convolutional Neural Networks", *New York, NY: Springer. doi*, vol. 10, pp. 978–3, 2017.

[37] G. Koch, R. Zemel e R. Salakhutdinov, "Siamese neural networks for one-shot image recognition", em *ICML deep learning workshop*, vol. 2, 2015.

[38] K. Faceli, A. C. Lorena, J. Gama, A. C. P. d. L. Carvalho et al., "Inteligência Artificial: Uma abordagem de aprendizado de máquina", 2011.

[39] R. Kohavi et al., "A study of cross-validation and bootstrap for accuracy estimation and model selection", em *Ijcai*, Montreal, Canada, vol. 14, 1995, pp. 1137–1145.

[40] H. Steiner, A. Kolb e N. Jung, "Reliable face anti-spoofing using multispectral swir imaging", em *2016 International Conference on Biometrics (ICB)*, IEEE, 2016, pp. 1–8.

[41] S. Zhang, X. Wang, A. Liu, C. Zhao, J. Wan, S. Escalera, H. Shi, Z. Wang e S. Z. Li, "A Dataset and Benchmark for Large-scale Multi-modal Face Anti-spoofing", em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 919–928.

[42] R. Pi. (set. de 2019). Raspberry Pi 3 Model B+, URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

[43] B. Electrónica. (set. de 2019). CÂMARAS RASPBERRY PI, URL: https://www.boxelectronica.com/pt/camaras-raspberry-pi/657-camara-para-raspberry-pi-visao-noturna-e-foco-ajustavel.html.

[44] Fusion3. (set. de 2019). Fusion3 F400-S, URL: https://www.fusion3design.com/.

[45] python. (set. de 2019). Python, URL: https://www.python.org.

[46] opencv. (set. de 2019). opencv, URL: https://opencv.org/.

[47] scikit-learn. (set. de 2019). scikit-learn, URL: https://scikit-learn.org/stable/.

[48] F. Furtado. (set. de 2019). Biometria, URL: https://medium.com/tend%5C%C3%5C%AAncias-digitais/biometria-nas-interfaces-fb844e9e8fcc.

[49] OpenFace. (set. de 2019). OpenFace, URL: https://cmusatyalab.github.io/openface/.

[50] C. Albon. (set. de 2019). Chapter 4. Handling Numerical Data, URL: https://www.oreilly.com/library/view/machine-learning-with/9781491989371/ch04.html.