ÍTALO DELLA GARZA SILVA

RECONHECIMENTO FACIAL COM *DEEP LEARNING* PARA A SEGURANÇA DE CASAS INTELIGENTES

LAVRAS – MG

2020

ÍTALO DELLA GARZA SILVA

RECONHECIMENTO FACIAL COM *DEEP LEARNING* PARA A SEGURANÇA DE CASAS INTELIGENTES

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. DSc. Erick Galani Maziero
Orientador

LAVRAS – MG

2020

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Silva, Ítalo Della Garza
Reconhecimento facial com *Deep Learning* para a segurança de casas inteligentes / Ítalo Della Garza Silva. – 2020.
85 p. : il.

Orientador(a): Prof. DSc. Erick Galani Maziero.
Monografia (graduação) – Universidade Federal de Lavras, 2020.
Bibliografia.

1. Aprendizado de Máquina. 2. Redes Neurais Profundas. 3. Reconhecimento Facial. I. Maziero, Erick Galani. II. Título.

ÍTALO DELLA GARZA SILVA

RECONHECIMENTO FACIAL COM *DEEP LEARNING* PARA A SEGURANÇA DE CASAS INTELIGENTES
FACE RECOGNITION WITH DEEP LEARNING FOR SMART HOME SECURITY

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADA em 3 de Setembro de 2020.

Prof. DSc. Erick Galani Maziero UFLA
Prof. DSc. André Pimenta Freire UFLA
Prof. DSc. Bruno de Abreu Silva UFLA

Prof. DSc. Erick Galani Maziero
Orientador

LAVRAS – MG
2020

AGRADECIMENTOS

Primeiramente agradeço a Deus, por sempre iluminar meu caminho nessa jornada. Agradeço à minha família, que tanto me incentivou e me levantou nos momentos mais difíceis. Agradeço à UFLA, pelo apoio e infraestrutura fornecidos durante toda a minha graduação e ao meu orientador, por me instruir corretamente, me orientar com toda a atenção e paciência e me apresentar oportunidades únicas na vida.

RESUMO

Este trabalho é constituído de um estudo empírico de tecnologias de reconhecimento facial em vídeos, presentes na literatura. Foca-se nos trabalhos que façam uso de Redes Neurais Artificiais Profundas (*Deep Learning*), analisando a possível aplicação em um sistema de segurança de uma casa inteligente. Como resultado do estudo são testadas, para três diferentes arquiteturas de redes neurais convolucionais, diferentes combinações dos métodos analisados para fundamentar a escolha da melhor arquitetura para aplicação em casas inteligentes. Como a arquitetura envolveria tanto identificação facial como verificação facial, as redes neurais foram combinadas com diferentes métodos de classificação (para avaliar a identificação facial) e combinações de métodos de cálculo de distância e limítrofes (para avaliar a verificação facial) para a realização dos testes. Ao final do trabalho, com a arquitetura estabelecida, foi realizada uma simulação de um sistema de segurança para o controle (abertura) automático de uma porta da casa inteligente. O trabalho objetivou também estudar uma possível adaptação da solução para outros sistemas de autenticação. Obteve-se um produto final com potencial para ser aplicado em uma situação real, embora ainda sejam necessárias algumas melhorias em sua precisão e tempo de execução para esse fim.

Palavras-chave: Reconhecimento Facial. Redes Neurais Profundas. Aprendizado de Máquina.

ABSTRACT

This research consists of an empirical study about video face recognition technologies present in literature, focusing on works that make use of Deep Artificial Neural Network (Deep Learning). Also, the possible application into a smart home security system was taken into account. Different combinations of the analyzed methods were tested to discover which is the best architecture for smart home application. As the architecture would involve facial identification, the neural networks were combined with different classification methods (to evaluate the facial identification), and distance calculation methods and triggers combinations (to evaluate the facial verification). At the end of the research, with the established architecture, a simulation of a security system for a smart home door was tested. This research also seems to study a possible adaptation of the solution for other authentication systems. A final product with potential to become a real application was obtained, although some improvements at its precision and execution time are still necessary.

Keywords: Facial Recognition. Deep Learning. Machine Learning.

SUMÁRIO

1 INTRODUÇÃO

O Processamento de Imagens (PI) pode ser compreendido como o conjunto de métodos que envolvem processar uma imagem em um computador, de maneira automática. Segundo Gonzalez e Woods (2006), não há consenso entre os autores na definição de uma separação clara entre PI e outras áreas, tais como Visão Computacional e Análise de Imagens, porém afirmam que a área engloba tais estudos. É possível organizar os processos em imagens em três níveis, sendo eles alto, médio e baixo nível. Os processos de baixo nível são aqueles que executam operações simples sobre as imagens, como redução de ruído e aguçamento de bordas. Os processos de médio nível compreendem aqueles em que são extraídos atributos relacionados à imagem, como contornos em torno de um objeto, por exemplo. Já os processos de alto nível caracterizam-se por estabelecer funções cognitivas sobre imagens e extrair significados de objetos já reconhecidos.

Ao PI estão associados diversos métodos e técnicas, que podem ser divididos de acordo com a tarefa a que se propõem a solucionar. A Aquisição de Imagem engloba os processos necessários para se obter imagens do mundo real, tornando-as digitais. O Realce e Filtragem de Imagens é a tarefa de se manipular uma imagem com o objetivo de adaptá-la à execução de uma determinada operação seguinte. A Restauração de Imagens busca melhorar a qualidade visual de uma imagem. O Processamento Multirresolução trata as imagens em várias resoluções diferentes, para efetuar algum outro tipo de processamento. A Compressão visa reduzir o tamanho em disco de uma imagem para diminuir o custo de armazenamento minimizando a perda de informação. O Processamento Morfológico lida com a extração de informação útil a respeito de uma imagem, que também é realizado no presente trabalho. A Segmentação visa dividir a imagem através dos objetos que a compõem (“objeto”, dentro de PI, se refere a tudo aquilo que é visível e passível de ser identificado e separado em uma imagem. Logo, seres vivos, por exemplo, também são considerados objetos em uma imagem). A Representação e

Descrição busca tratar os resultados da segmentação, extraindo características úteis dos mesmos. O Reconhecimento, objeto de estudo deste trabalho, visa atribuir um rótulo a um objeto em uma imagem com base em seus atributos previamente extraídos. O presente trabalho foca nas tarefas que compõem um sistema completo de Reconhecimento Facial, ou seja, a tarefa de reconhecer rostos em imagens e lhes atribuir rótulos.

O Reconhecimento Facial (RF) é uma tecnologia muito utilizada no mundo moderno. Ganhou bastante atenção nos últimos anos por parte de corporações e organizações governamentais graças ao seu grande nível de segurança e confiabilidade. Um dos motivos desse crescimento se deve à sua aplicação na área de segurança, como câmeras de vigilância e perícia forense. O RF leva vantagem sobre outros sistemas biométricos, já que é possível reconhecer a face de um ser humano sem que ele precise interagir com o sistema, diferentemente de um reconhecedor de digitais, por exemplo, no qual uma pessoa precisa tocar com sua digital para que o reconhecimento seja feito, o que demanda o consentimento daquele indivíduo a ser reconhecido. Além do seu uso na área de segurança, essa tecnologia tem sido utilizada em sistemas de casas inteligentes para o reconhecimento de humor, de forma a aplicar alterações no ambiente a fim de melhorar o humor do usuário (YANG et al., 2018).

RF tem sido um assunto bastante explorado, porém, conforme descrito por Wang e Deng (2018), os estudos sobre o tema começaram a aumentar no início da década de 90, devido ao surgimento do *Eigenface* (TURK; PENTLAND, 1991). Essa arquitetura convertia o conjunto de imagens de face em uma representação sobre um espaço de atributos de baixa dimensão. O grande problema envolvendo essa metodologia era sua fragilidade perante a variações nas faces que desviavam de suas suposições a priori. No início de 2010, surgem técnicas de RF baseadas em atributos locais, quando foi introduzida a ideia de se utilizar filtros em regiões específicas da imagem. Em 2012, o RF tomou um novo rumo: Popularizava-se

a técnica de *Deep Learning* e, através dela, *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) ganhou a competição *ImageNet* com grande vantagem. Em 2014, *DeepFace* (TAIGMAN et al., 2014) conseguiu alcançar o desempenho humano de reconhecimento facial sobre o *benchmark* LFW, treinando um modelo de 9 camadas com 4 milhões de faces. Até hoje, a abordagem *Deep Learning* tem sido a mais usada e aprimorada para RF na literatura (WANG; DENG, 2018).

1.1 Motivação

Graças à evolução tecnológica, as casas inteligentes são hoje uma realidade possível e podem se tornar cada vez mais populares (WILSON; HARGREAVES; HAUXWELL-BALDWIN, 2017). Cada elemento a mais em uma casa inteligente que possa melhorar a qualidade de vida dos seus moradores pode se tornar um assunto de interesse geral e, consequentemente, objetos de estudo para pesquisadores, incluindo aqueles relacionados à segurança. O foco deste trabalho é a criação de um software que simule um sistema de segurança de uma casa inteligente, identificando o rosto de uma pessoa que deseje entrar na casa e abrindo a porta ou não a partir da identificação realizada.

Um sistema assim poderia, com as devidas alterações, ser adaptado a outras tarefas que envolvem RF em suas atividades. Ainda sobre a casa inteligente, por exemplo, um sistema de RF pode ser capaz de reconhecer o humor do morador e, caso o mesmo esteja de mau-humor, a casa pode alterar alguns elementos de sua configuração interna, como iluminação, música ambiente, etc., para que o humor do morador melhore. Outros exemplos de sistemas que podem fazer uso do RF são os sistemas de autenticação de aparelhos celulares, câmeras de vigilância nas ruas, estabelecimentos comerciais, presídios, entre outros exemplos. O estudo realizado no presente trabalho pode ser adaptado para todos esses sistemas.

1.2 Objetivos

O presente estudo visa testar como diferentes arquiteturas de *Deep Learning* se comportam quando inseridas dentro de diferentes métodos de RF em diferentes situações, porém focando-se no RF aplicado à segurança. Também visa avaliar se a melhor solução encontrada atenderia as necessidades de um sistema de autenticação em uma casa inteligente. Dessa forma, este trabalho tem como objetivos os listados a seguir.

- Explorar e aprimorar um sistema de reconhecimento facial, que detectará, reconhecerá e atribuirá um nome cadastrado em uma base de dados ao rosto adquirido de uma imagem ou vídeo.
- Aplicar o sistema aprimorado em um cenário de autenticação de usuário de uma casa inteligente, buscando maximizar a automação e segurança dessa casa.

1.3 Organização do texto

O Capítulo 2 trata da base teórica do estudo, detalhando os métodos de Detecção, Alinhamento e Classificação de Faces presentes na literatura e aqui utilizados, explicando a teoria de Redes Neurais Artificiais com ênfase em Redes Convolucionais e sua aplicação no RF. Apresenta, também, um apanhado de trabalhos relacionados ao presente estudo. O Capítulo 3 explica em detalhes como foi realizado este trabalho, descrevendo a arquitetura geral da solução proposta, a base de dados utilizada e como foram realizados os testes. O Capítulo 4 traz os resultados obtidos pelos testes e levanta uma discussão a respeito dos mesmos. Por fim, o Capítulo 5 encerra este texto com as contribuições teóricas e práticas alcançadas pelo trabalho, evidencia as limitações, e propõe possíveis projetos futuros.

2 REFERENCIAL TEÓRICO

Os métodos que compõem o estudo sobre Reconhecimento Facial presentes neste trabalho estão fundamentados principalmente em certas subáreas de *Machine Learning* (Aprendizagem de Máquina). Este capítulo tem por objetivo apresentar o conhecimento necessário à compreensão da metodologia utilizada no presente estudo. A Seção 2.1 explica a tarefa da Classificação e os algoritmos utilizados neste trabalho. A Seção 2.2 faz uma síntese específica sobre Redes Neurais Artificiais com ênfase em Redes Neurais Convolucionais, que são usadas etapa de extração de atributos do Reconhecimento Facial. A Seção 2.3 explica todas as etapas do Reconhecimento Facial e mostra como o conteúdo até então descrito se organiza dentro de cada etapa, bem como descreve algoritmos específicos de Detecção de Faces e Alinhamento, as duas primeiras etapas do Reconhecimento Facial. Por fim, a Seção 2.4 faz um apanhado de recentes trabalhos relacionados, descrevendo as contribuições e as limitações de cada um.

2.1 Métodos de Classificação

A Classificação é uma das tarefas do *Machine Learning* e constitui-se, basicamente em atribuir um rótulo, também chamado de classe, a uma instância de uma base de dados, a partir de um conjunto de informações referentes à mesma. Ou seja, dado um conjunto de atributos de uma instância, pretende-se predizer um determinado rótulo para a instância. Um exemplo de classificação é a predição de doenças, na qual, a partir de dados a respeito do paciente, tal como pressão sanguínea, temperatura do corpo, entre outros, busca-se predizer se o paciente possui uma doença ou não, inclusive qual. A Classificação é a última etapa de um sistema de Reconhecimento Facial. Existem diversas abordagens no *Machine Learning*, dividindo-se em três principais categorias: supervisionados, não supervisionados e semissupervisionados. Os métodos não supervisionados inferem padrões entre os dados sem nenhuma informação sobre a classe esperada. Os métodos supervi-

sionados e semissupervisionados aprendem por meio de observações do conjunto de dados de entrada e sua respectiva saída esperada, observações essas chamadas de conjunto de treinamento, sendo que a aprendizagem semissupervisionada se responsabiliza por tratar também dos dados imprecisos e não rotulados, durante o aprendizado. Por isso, os métodos de *Machine Learning* supervisionados e semissupervisionados é que são responsáveis pela tarefa de classificação. Serão citados nas próximas subseções os algoritmos de classificação que foram utilizados neste trabalho, conforme descritos em Marsland (2014).

2.1.1 *K Nearest Neighbors*

O Algoritmo dos K vizinhos mais próximos (*K Nearest Neighbors* - KNN) é um método de classificação supervisionado cuja estratégia consiste em classificar uma determinada instância baseando-se nas classes das instâncias com os atributos que mais se assemelham aos dela. Ou seja, supondo um conjunto de dados representados como pontos em um espaço n-dimensional, cada instância i possui o conjunto de atributos $x_i = x_{i,1}, x_{i,2}, ...x_{i,N}$ e uma classe y_i. Para a classificação de uma nova instância$j$, de atributos x_j, é necessário calcular a distância entre o ponto x_j e todos os outros da base. Dentre os K pontos mais próximos, calcula-se a classe mais frequente, que será a classe $\hat{y}_j$ atribuída à instância de atributos x_j.

A distância entre dois pontos no KNN pode ser calculada de várias formas. Uma forma comum é a Distância Euclidiana, que pode ser obtida através da fórmula

$$d_e(x_i, x_j) = \sqrt{\sum_{n=1}^{N} (x_{i,n} - x_{j,n})^2}. \tag{2.1}$$

Outra distância que pode ser usada é a Similaridade de Cossenos, que é a distância angular entre dois pontos a partir da origem, isto é,

$$s_c(x_i, x_j) = \frac{\sqrt{\sum_{n=1}^{N} x_{i,n} * x_{j,n}}}{\sqrt{\sum_{n=1}^{N} x_{i,n}^2} * \sqrt{\sum_{n=1}^{N} x_{j,n}^2}}. \tag{2.2}$$

2.1.2 Naïve Bayes

O método Naïve Bayes é um método de classificação supervisionado baseado na Regra de Bayes, a qual afirma que a probabilidade de um evento A dado que um evento B ocorreu é igual ao produto entre a probabilidade a priori do evento A ocorrer e a probabilidade condicionada $P(B|A)$ dividido pela probabilidade de B ocorrer, ou seja,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.3}$$

No Naïve Bayes, os atributos e as classes de um dado na base são considerados eventos e possuem uma distribuição probabilística associada a cada um, sendo que as inferências são feitas através da Regra de Bayes. Nesse caso, pode-se substituir nessa regra o evento B pelo conjunto de atributos de um determinado dado e A pela classe a se atribuir a esse dado, isto é,

$$P(y|x_1,x_2,\ldots,x_n) = \frac{P(x_1,x_2,\ldots,x_n|y)P(y)}{P(x_1,x_2,\ldots,x_n)}, \tag{2.4}$$

em que y é a classe atribuída ao dado e $x_1,x_2,\ldots,x_n$ é o conjunto de atributos desse dado. Sabe-se que, para dois eventos independentes A_1 e A_2, $P(A_1,A_2) = P(A_1)P(A_2)$ e que $P(A_1,A_2|B) = P(A_1|B)P(A_2|B)$, sendo B um evento qualquer nesse espaço amostral. No Naïve Bayes, os atributos de um dado são supostos independentes entre si. Portanto, a equação pode ser reescrita como

$$P(y|x_1,x_2,\ldots,x_n) = P(y)\frac{P(x_1|y)}{P(x_1)}\frac{P(x_2|y)}{P(x_2)}\ldots\frac{P(x_n|y)}{P(x_n)}. \tag{2.5}$$

A estratégia de classificação do Naïve Bayes é verificar qual é a maior probabilidade para uma dado pertencer a uma classe dado um conjunto de atributos e atribuir ao dado a classe com essa probabilidade, ou seja,

$$\hat{y} = \arg\max_{y}\{P(y|x_1,x_2,\ldots,x_n)\}, \tag{2.6}$$

em que $\hat{y}$ é a classe atribuída ao dado.

Detalhadamente, a maneira através da qual o modelo será treinado e classificado dependerá da natureza dos atributos. Se forem atributos binários, o treinamento e a classificação se baseiam na distribuição de Bernoulli. Caso os atributos sejam discretos, o modelo é baseado na distribuição Multinomial. Nesses dois casos, o treinamento se dá por meio do cálculo das distribuições necessárias encontradas nas fórmulas supracitadas. No caso da presença de atributos reais, que acontece no presente trabalho, ambos o treinamento e a fórmula de decisão são adaptadas para a distribuição Gaussiana. Logo, uma dada probabilidade condicionada de um dado atributo x_i é então obtida através da fórmula

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right), \tag{2.7}$$

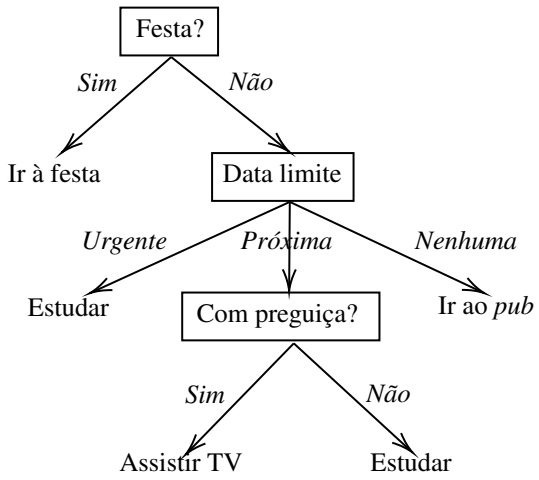
em que μ_y e σ_y são, respectivamente, a média e a variância de y, que devem ser obtidas durante a fase de treinamento.

2.1.3 Árvore de Decisão

As árvores de decisão são modelos supervisionados que classificam um determinado dado através de uma sequência de decisões baseadas nos valores de seus atributos. Cada nó (ramificação) da árvore de decisão armazena uma regra que se refere a um determinado atributo do dado. Para cada valor que esse atributo possa assumir existe uma ligação que leva a um nó sucessor (ou filho), e assim sucessivamente. Os últimos nós (nós-folha) atribuem a classe ao dado. A Figura 2.1 ilustra o conceito de uma árvore de decisão.

A classificação em uma árvore de decisão ocorre da seguinte maneira: partindo-se do nó raiz da árvore e seguindo em direção aos nós folha, o algoritmo faz a escolha do nó por onde seguirá seu caminho baseado em uma regra associada a um determinado atributo no nó atual, sendo que ao fim da execução, no nó folha,

Figura 2.1 – Exemplo de árvore de decisão.



Fonte: Adaptado de Marsland (2014).

a classe é atribuída ao dado. O tipo de regra associada a cada atributo dependerá do tipo de dado do atributo. Caso se trate de um dado discreto (ou nominal), um nó é feito para cada valor que esse dado possa assumir. Para o caso de dados numéricos e reais, são estabelecidas faixas de valores para os quais as decisões são construídas. É possível interpretar a árvore de decisão como uma hierarquia de disjunções lógicas.

O treinamento para a construção das árvores de decisão é feito baseado no conceito de entropia, sendo que existem vários algoritmos para essa tarefa. A fórmula para a entropia, que visa calcular o nível de desordem em um conjunto de dados, é dada por

$$\text{Entropia}(p) = -\sum_i p_i \log_2 p_i, \tag{2.8}$$

onde p_i é a proporção de atributos p com valor i no conjunto de dados. O princípio dos algoritmos de construção se baseia em criar as decisões em cada nó de acordo com o atributo, ainda não usado nos níveis anteriores, mais informativo, isto é, aquele que melhor divide as classes pelos atributos, visando minimizar o número

de níveis da árvore. Dentre os algoritmos mais comuns, destacam-se o ID3, o CART e o C4.5.

2.1.4 Adaboost

O método de Adaboost faz parte de um subconjunto de métodos de aprendizagem de máquina chamadas de "decisão por comitê" (*decision by comittee*), os quais buscam, a partir de um conjunto de classificadores, obter uma única resposta considerando as respostas de cada um deles. No caso do Adaboost, os classificadores são gerados a cada nova iteração, e a resposta é obtida fazendo-se uma média ponderada baseada em um peso α sobre a decisão de tais classificadores.

Os classificadores usados nesse algoritmo são simples e com baixa acurácia quando aplicados sozinhos, também conhecidos como "classificadores fracos" (*weak classifiers*). Porém, quando aplicados nesse tipo de decisão por comitê, a tendência é que o resultado final seja consideravelmente mais acurado.

Inicialmente, a todos os elementos x_n no conjunto de treinamento são atribuídos pesos w_n iguais, sendo que a soma dos pesos é 1. A cada iteração, um novo classificador é treinado e o erro médio ε é computado a partir do somatório dos pesos dos dados incorretamente classificados, i.e.,

$$\varepsilon = \sum_{n=1}^{N} w_n * I(y_n \neq \hat{y}_n), \tag{2.9}$$

em que y_n é a classe correta para x_n, $\hat{y}_n$ é a classe para x_n obtida pelo classificador e $I(c)$ é uma função-indicador, que retorna 1 caso a condição c seja verdadeira e 0 caso contrário. A partir desse erro, o peso de decisão do classificador é obtido pela fórmula

$$\alpha = (1-\varepsilon)/\varepsilon. \tag{2.10}$$

Os pesos de cada dado são calculados para a iteração seguinte baseados no valor α da iteração presente t.

$$w_n^{(t+1)} = w_n^{(t)} * \frac{\exp(\alpha_t * I(y_n \neq \hat{y}_n))}{Z_t}, \tag{2.11}$$

em que Z_t é uma constante de normalização. O término do treinamento se dá em uma das seguintes condições:

- após um determinado número de iterações;
- até que todos os elementos sejam classificados corretamente; ou
- quando um elemento tiver mais que a metade do somatório dos pesos.

A classe atribuída a um novo elemento durante sua classificação é obtida de acordo com o resultado obtido por cada classificador e seu valor α, através da fórmula

$$f(x) = \arg\max_k \{\sum_{t=1}^{T} \alpha_t * I(\hat{y}_t(x) = k)\}, \tag{2.12}$$

em que T é o total de classificadores, k é uma classe e $\hat{y}_t$ é a classificação do atribuída a x pelo classificador t.

2.1.5 *Random Forest*

Assim como o Adaboost, o *Random Forest* também é um método de decisão por comitê. Nele um conjunto de árvores de decisão são gerados e uma classificação final para um dado elemento é a resposta majoritária dada pelo conjunto.

O treinamento do *Random Forest* consiste em separar subconjuntos aleatórios de instâncias do *dataset* e utilizar cada subconjunto para treinar uma árvore de decisão, sendo que cada árvore também é limitada a considerar somente um subconjunto do conjunto de atributos. Dessa forma, uma árvore no *Random Forest* tende a ser ligeiramente diferente de outra. O número de atributos que cada

árvore irá considerar pouco afetará no resultado final, mas as implementações na literatura tendem a atribuí-lo a raiz quadrada do número total de atributos.

2.1.6 Gradient Boosting

O Gradient Boosting é um método de classificação de *boosting*, assim como o *Adaboost*, que busca montar um classificador complexo pela união de vários classificadores simples, sendo que o treinamento busca utilizar as deficiências do classificador anterior para ajustar o seguinte. Porém, o *Gradient Boosting* calcula seu erro baseado na diferença entre o valor obtido e o desejado, e identifica as deficiências utilizando o método de descida do gradiente (*Gradient Descend*).

Uma das implementações mais famosas do *Gradient Boosting* se encontra na biblioteca *XGBoost* (CHEN; GUESTRIN, 2016), possuindo diversas otimizações de *hardware* para melhorar o desempenho.

2.1.7 *Support Vector Machine*

As Máquinas Vetoriais de Suporte (*Support Vector Machines* - SVMs) geram, assim como os neurônios simples de uma Rede Neural Artificial (SUBSEÇÃO 2.2), um hiperplano para separar as classes. Porém, a estratégia da SVM consiste em gerar o que se denomina hiperplano ótimo de separação, o qual consegue separar as instâncias entre classes enquanto se distancia o máximo possível dos pontos mais próximos dele pertencentes à cada classe, que são conhecidos como vetores de suporte (*support vectors*).

O problema matemático que define uma SVM pode ser visto como o mesmo que define o problema do *Perceptron*, porém com a adição de restrições que definam as particularidades do modelo. Primeiramente, existe uma margem M que distancia o hiperplano a se estabelecer de cada classe, ou seja

$$y(x^T w + b) \geq M(1 - \varepsilon) \tag{2.13}$$

para cada x, em que x é o conjunto de atributos de entrada na base de dados, $y \in [-1,1]$ é a classe relativa ao dado de x, w é o conjunto de pesos associados a cada atributo e b é o *bias*. O ε é um pequeno valor também associado ao dado em questão que deve ser levado em consideração na restrição que corrige os casos de não-separabilidade. O problema da SVM é abordado como um problema de otimização e sua solução é implementada geralmente através de programação quadrática. Vale lembrar que a SVM original suporta somente a classificação binária, sendo que para suportar a existência de 3 ou mais classes é necessário adicionar algum complemento no modelo. Hsu e Lin (2002) abordam as principais técnicas para a adaptação do SVM à classificação multiclasse.

Para os casos de não-linearidade, o SVM pode aumentar a dimensão dos dados adicionando a eles mais atributos. Por exemplo, se um dado de atributos x possui dois elementos x_1 e x_2, é possível gerar novos conjuntos de valores a partir desses já existentes, combinando-os de maneira não-linear, como, por exemplo, criando os novos atributos $x_3 = x_1^2$, $x_4 = x_2^2$ e $x_5 = \sqrt{2}x_1x_2$. Com isso, é possível mapear um conjunto de dados de baixa dimensão para um conjunto de dados de alta dimensão. Esse valores podem ser calculados através de adaptações na função-objetivo da otimização (em tempo de execução) e as funções de transformação são conhecidas como funções *kernel*.

2.2 Redes Neurais

Conforme descrito por Haykin (2001), uma Rede Neural Artificial (RNA) é uma técnica de aprendizagem de máquina baseada no funcionamento do cérebro humano. Um modelo de RNA é composto por um conjunto de unidades de processamento simples, conhecidas como neurônios. O primeiro modelo de RNA, o *Perceptron* (ROSENBLATT, 1958) continha somente um neurônio, o qual é composto por três principais componentes:

(i) Um conjunto de entradas x_j, cada uma associada a um peso w_{kj};

(ii) um somador responsável por somar todos os sinais da entrada do neurônio; e

(iii) uma função de ativação que serve para limitar a saída do neurônio a um conjunto específico de valores.

A Figura 2.2 ilustra a composição de um neurônio simples. O neurônio pode ser entendido como uma função não linear, onde para uma dada entrada x uma saída y é gerada, podendo ser descrita pela fórmula

$$y_k = \varphi(w_k x + b_k), \tag{2.14}$$

em que $\varphi(.)$ é uma função de ativação, b_k é bias do neurônio k, e y_k é a saída do neurônio. Existem diversas funções de ativação diferentes para diversas situações, sendo mais comum o uso de funções sigmoides, como

$$\varphi(\upsilon) = \frac{1}{1 + exp(-a\upsilon)}. \tag{2.15}$$

O bias é um peso fixo associado a uma entrada fixada em +1 e serve para modificar a relação entre o potencial de ativação do neurônio e a saída da função de ativação.

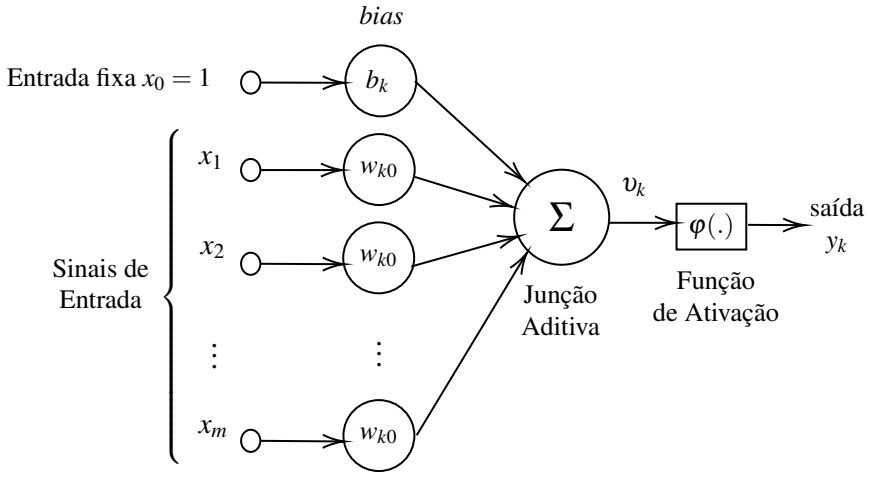
Para um neurônio aprender a partir de um conjunto de treinamento, o método mais utilizado é associar à sua saída uma função de erro (ou função de custo), descrita pela fórmula

$$e_k = d_k - y_k, \tag{2.16}$$

em que e_k é o associado do neurônio k, d_k é a saída desejada e y_k é a saída obtida. Esse erro é usado para atualizar cada peso do neurônio, sendo que o ajuste pode ser obtido pela fórmula

$$\Delta w_{kj} = \eta e_k x_j, \tag{2.17}$$

Figura 2.2 – Modelo de um neurônio simples (*perceptron*).



Fonte: Adaptado de Haykin (2001).

em que Δw_{kj} é a variação do peso w_{kj} e η é a taxa de aprendizagem do neurônio, valor fixo diretamente relacionado à intensidade com que os pesos de um neurônio podem se ajustar.

Desde o surgimento do Perceptron, foram então desenvolvidas várias arquiteturas de RNA. São muito usadas as arquiteturas de rede organizadas por camadas, onde uma camada é um conjunto de neurônios que processam os sinais de entrada e geram um conjunto de saídas, uma para cada neurônio. No caso da existência de apenas uma camada, o processo de aprendizagem é denominado *Shalow Learning*, enquanto na existência de múltiplas camadas o processo é denominado *Deep Learning*. A rede neural de múltiplas camadas, comumente chamadas de *Multilayer Perceptron* (MLP), possui um processo de aprendizagem que propaga corretamente o erro da última camada para as camadas anteriores, conhecido como *Error Backpropagation*. O algoritmo calcula o erro final e o propaga através da derivada de cada peso dos neurônios das camadas anteriores em relação ao erro, gerando um fator de sensibilidade correto para ser usado na correção dos pesos. De acordo com Ponti e Costa (2018), os MLPs podem ser abordados como um conjunto de funções (correspondentes às camadas) aplicadas sucessivamente a uma

entrada, da forma

$$\hat{y} = f(x) = f_1(f_2(f_3(x))), \tag{2.18}$$

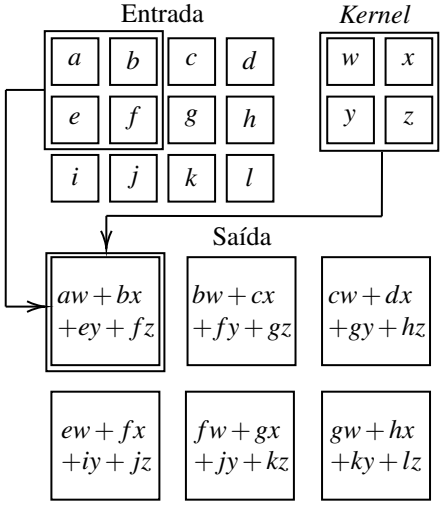
sendo que cada função pode ser alterada de modo a gerar uma combinação de funções que resolvam o problema em questão, o que faz com que surjam diferentes arquiteturas para o *Deep Learning*. Dentre as muitas arquiteturas diferentes, são bastante comuns as Redes Alimentadas Adiante (*Feed Foward*), que possuem uma camada de entrada, uma de saída e uma série de camadas ocultas, sendo que o fluxo de informação é sempre propagado da camada de entrada em direção à camada de saída. Outra arquitetura comum é a Rede Neural Recorrente, que possui um sistema de realimentação no qual a saída da rede é reutilizada como entrada, o que é útil para lidar com dados sequenciais. As redes *Long short-term Memory* (LSTM) são arquiteturas de redes recorrentes capazes de armazenar valores de intervalos arbitrários, muito úteis para tratar dados sequenciais com intervalos de tamanho variáveis. As Máquinas de Boltzmann são redes probabilísticas baseadas em energia, bastante usadas em detecção de fraudes e sistemas de recomendação. Também existem as denominadas *Deep Belief Networks*, que também são redes probabilísticas compostas por várias camadas de variáveis latentes, os *Autoencoders*, para a codificação de dados não-supervisionada, as *Generative Adversarial Networks* (GANs), que consistem de duas Redes Neurais que aprendem através de um sistema de competição entre si, e as Redes Convolucionais, objeto de estudo deste trabalho, descritas na Subseção 2.2.1, dentre outras várias arquiteturas.

2.2.1 Redes Convolucionais

Uma Rede Neural Convolucional (RNC) é um perceptron de múltiplas camadas caracterizado por possuir algumas camadas que realizam operações de convolução sobre suas entradas, objetivando abstrair características relacionadas às mesmas, a fim de extrair as informações mais importantes sobre essas entradas. De acordo com Goodfellow, Bengio e Courville (2016), a saída de uma camada

de convolução é conhecida como *feature map* (mapa de características). As Redes Convolucionais são especializadas em processamento de dados com topologia em grade (e.g. áudios, imagens, etc.), sendo que suas principais vantagens sobre as redes neurais tradicionais são o compartilhamento de parâmetros, as interações esparsas entre os nós da rede e a equivariância nas representações. O compartilhamento de parâmetros refere-se ao que se conhece como *filtro de convolução* (ou *kernel*), que é a matriz de pesos usada para realizar a convolução em uma dada camada. Isso faz com que cada unidade da entrada de uma camada seja visitado várias vezes e a entrada seja analisada como um todo, ao contrário das redes neurais tradicionais. Um exemplo de funcionamento da convolução é ilustrado pela Figura 2.3. As interações esparsas entre os nós fazem com que a rede necessite de menos espaço em memória para ser processada e a eficiência estatística sobre cada entrada aumente. Já a equivariância nas representações faz com que uma dada mudança qualquer na entrada de uma camada altere a saída dessa camada na mesma intensidade, já que cada pixel é igualmente levado em consideração pelo modelo.

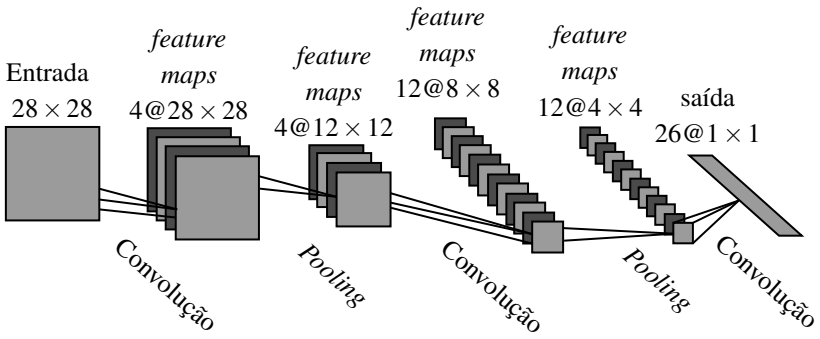
Figura 2.3 – Exemplo de uma convolução bidimensional.



O filtro convolucional (*Kernel*) é aplicado em todas as posições possíveis na entrada, gerando uma saída menor que a entrada a partir das operações descritas. Fonte: Adaptado de Goodfellow, Bengio e Courville (2016).

Outro ponto importante no assunto de Redes Neurais Convolucionais é existência de camadas que realizam a redução das dimensões na saída das camadas convolucionais, reduções essas que buscam minimizar a perda de informação ao passo que diminuem a quantidade de informação a ser processada nas camadas seguintes. Essas camadas realizam operações de *pooling* para tal fim. Um exemplo de operação de *pooling* é o *maxpooling*, que retorna o elemento máximo de uma dada submatriz. Aplicando-se o *pooling* em todas as submatrizes possíveis de uma matriz, temos na saída essa matriz reduzida, com o máximo possível de informação preservada. Um exemplo de arquitetura completa de Rede Neural Convolucional é ilustrado na Figura 2.4.

Figura 2.4 – Exemplo de uma arquitetura de Rede Neural Convolucional.



Fonte: Adaptado de Haykin (2001)

O conteúdo apresentado da Seção 2.1 até este ponto refere-se a algoritmos gerais de *Machine Learning*, que podem ser aplicados, isoladamente ou em conjunto, para resolver inúmeros tipos de tarefas, incluindo Reconhecimento Facial, o foco deste trabalho (SEÇÃO 2.3).

2.3 Reconhecimento Facial

O Reconhecimento Facial pode ser dividido em três etapas principais:

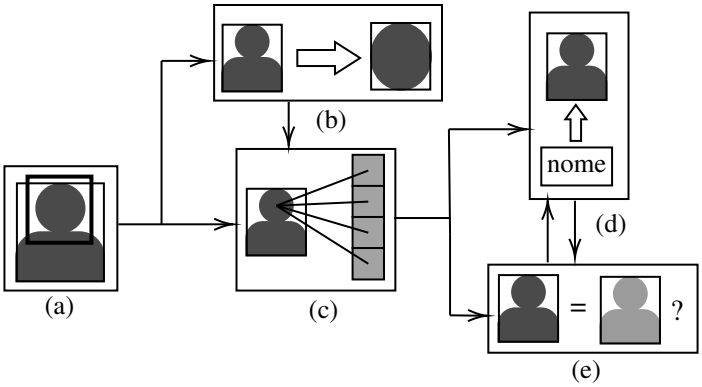
(i) uma primeira etapa de detecção de faces, que deve verificar em uma dada imagem a presença de uma ou mais faces e, caso existam, suas respectivas localizações;

(ii) uma etapa de extração de *features*, que realiza um pré-processamento nas faces encontradas, e;

(iii) uma etapa de classificação e/ou verificação facial.

Para o caso do presente trabalho, há ainda uma etapa extra entre a primeira e a segunda, conhecida como Alinhamento, que se trata de uma normalização das imagens de face. Nela, os pontos-chave das faces são movidos a localizações específicas na imagem (nariz, olhos, etc) e esta é reconfigurada para minimizar as deformidades resultantes. O processo de RF está ilustrado na Figura 2.5.

Figura 2.5 – Processo de Reconhecimento Facial



A figura ilustra o processo de reconhecimento facial, composto por (a) Detecção; (b) Alinhamento; (c) Extração de atributos; (d) Classificação; e (e) Verificação.
Fonte: Do autor (2020).

Os algoritmos utilizados nas etapas de detecção e alinhamento presentes nesse trabalho estão descritos nas Subseções 2.3.1 e 2.3.2, respectivamente. A etapa de extração de atributos, descrita na Subseção 2.3.3 é feita principalmente através de Redes Neurais Convolucionais e sua função é extrair a informação necessária para diferenciar uma imagem de outra, eliminando-se o conteúdo des-

necessário. Essa informação é enviada à classificação e/ou à verificação, etapas descritas na Subseção 2.3.4.

Um ponto importante é que literatura costuma dividir o RF em cenários, dependendo do problema que se deseja resolver. Os cenários geralmente são V2S (*video-to-still*, pesquisar uma face em um vídeo dentro uma base de dados de imagens estáticas), S2V (*still-to-video*, pesquisar uma face em uma imagem estática dentro uma base de dados de vídeos), V2V (*video-to-video*, pesquisar uma face em um vídeo dentro uma base de dados de vídeos) e S2S (*still-to-still*, pesquisar uma face em uma imagem estática dentro uma base de dados de imagens estáticas).

2.3.1 Detecção de Faces com *Haar Cascade*

O algoritmo *Haar Cascade* (VIOLA; JONES, 2001), usado para a detecção de faces, trabalha analisando vários retângulos de um *frame* do vídeo, em diferentes escalas. O objetivo é fazer com que esses retângulos sejam classificados como positivos (há uma face na imagem) e negativos (não há uma face na imagem). Para isso, cada retângulo é inserido em uma série de classificadores em cascata, que utilizam filtros de *Haar* para a extração de *features*.

Os filtros aplicados extraem *features* de três tipos:

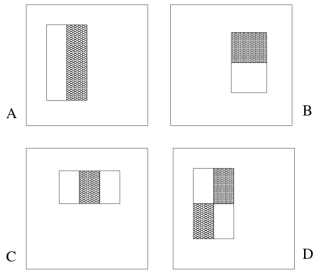
(i) *two-rectangle feature*: diferença entre a soma dos pixels de duas regiões retangulares, as quais tem as mesmas medidas e são verticalmente ou horizontalmente adjacentes;

(ii) *three-rectangle feature*: dadas três regiões retangulares adjacentes, é computada a soma dos pixels das duas regiões opostas subtraída da soma dos pixels da região central; e

(iii) *four-rectangle feature*: computa a soma entre os pares de regiões retangulares diagonais.

A Figura 2.6 mostra exemplos de *features* extraídas com o presente método de detecção.

Figura 2.6 – Exemplo de *features* extraídas com o método.



Fonte: Viola e Jones (2001).

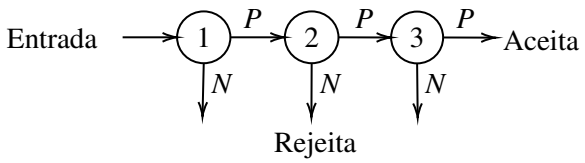
A partir dessa extração, no conjunto de imagens usadas para treinar o modelo, uma variação do algoritmo *Adaboost* foi usada para selecionar, a partir de um conjunto de aproximadamente 180.000 *features* em potencial para cada imagem, o conjunto de *features* com a menor taxa de falsos negativos. Os classificadores simples $h_j(x)$ usados no modelo podem ser descritos pela função

$$h_j(x) = \begin{cases} 1, & \text{se } p_j f_j(x) < p_j \theta_j; \\ 0, & \text{caso contrário,} \end{cases} \tag{2.19}$$

em que p_j é a paridade indicando a direção do sinal de inequação, $f_j(x)$ é a *feature* extraída da imagem em questão, θ_j é um fator limitante e 1 e 0 significa que a imagem é positiva ou negativa, respectivamente.

Uma vez treinados os classificadores e selecionadas as *features* mais significativas, os classificadores são dispostos em cascata, de forma que uma imagem classificada como positiva em um classificador é passada para o classificador seguinte, e se a imagem for desclassificada em algum dos classificadores, não será enviada aos classificadores seguintes (FIGURA 2.7).

Figura 2.7 – Esquema de funcionamento da cascata com três classificadores.



Fonte: Do autor (2020).

2.3.2 Alinhamento

O método para o alinhamento de faces utilizado neste trabalho (KAZEMI; SULLIVAN, 2014) se trata de um conjunto de árvores de regressão que são aplicadas para encontrar a localização de cada pixel que faz a demarcação de áreas importantes na imagem (e.g. boca, nariz, olhos, etc.), e, em seguida, trazer essas demarcações em todas as imagens para uma posição padrão, fazendo com que as imagens fiquem normalizadas antes de passarem pela rede neural convolucional. O conjunto dessas demarcações é conhecido como *shape*. O método é bastante inspirado no já citado *Adaboost* (SUBSEÇÃO 2.1.4).

O algoritmo foi treinado construindo os regressores na ordem em que a imagem passa por eles. Para um conjunto de treinamento $(I_1, S_1), ..., (I_n, S_n)$, em que I_i corresponde a uma imagem de uma face e S_i corresponde ao seu respectivo *shape*. Estabelece-se, então, um conjunto de triplas de treinamento $(I_{\pi_i}, \hat{S}_i^{(0)}, \Delta S_i^{(0)})$ para cada imagem I_{π_i}, em que $i = 1, ..., nR$, R é o número de inicializações por imagem, $\pi_i \in \{1, ..., n\}$, $\hat{S}_i^{(0)} \in \{S_1, ...S_n\} \backslash S_{\pi_i}$ é a estimativa inicial para o *shape* de I_{π_i} (amostrado uniformemente do conjunto de *shapes* e retirado desse conjunto), $\Delta S_i^{(0)} = S_{\pi_i} - \hat{S}_i^{(0)}$ corresponde ao passo de atualização no tempo 0, S_{π_i} é o *shape* correspondente à imagem I_{π_i} e n é o número de imagens do conjunto de treinamento. Esse dado é enviado como entrada ao algoritmo de aprendizado para estabelecer o primeiro regressor r_0, que usa *gradient boosting* com soma de erros quadrados. Para a atualização das triplas de treinamento dos passos seguintes, são utilizadas as seguintes equações:

$$\hat{S}_i^{(t+1)} = \hat{S}_i^{(t)} + r_t(I_{\pi_i}, \hat{S}_i^{(t)}) \tag{2.20}$$

$$\Delta S_i^{(t+1)} = S_{\pi_i} - \hat{S}_i^{(t+1)} \tag{2.21}$$

O processo é então repetido para todos os regressores e é executado em *loop* várias vezes, até que a combinação de todos os regressores atinja a máxima acurácia para o conjunto de treinamento. São estabelecidos T regressores que operam em cascata, similarmente ao que foi descrito na Seção 2.3.2. Ao final, temos um algoritmo que, para uma imagem, a cada passo de regressão, opera de forma a atualizar cada vez mais o *shape* daquela imagem.

Para este trabalho, foi utilizada uma implementação desse algoritmo presente na biblioteca *Dlib* (KING, 2009), que contém um conjunto de métodos de *Machine Learning*, incluindo alguns voltados à manipulação de imagens, implementados em C++ e uma interface em *Python*, o que facilitou a implementação da etapa alinhamento no presente trabalho.

2.3.3 Extração de atributos com RNC

O objetivo desta etapa é transformar uma imagem de face devidamente normalizada em um vetor de atributos conhecido como *embedding vector*. Ou seja, a imagem, sendo uma matriz de *pixels*, é a entrada da Rede Convolucional, e a saída é o *embedding vector*. A distância entre os *embedding vectors* extraídos a partir de imagens de face de uma mesma pessoa devem estar o mais próximos possíveis enquanto os de pessoas diferentes devem ter maior distância possível entre si. Esse fato serve de base para os diferentes métodos de treinamento de RNCs para RF da literatura.

Sendo a extração de atributos uma etapa crucial para a qualidade do RF, foram desenvolvidas diversas arquiteturas de rede para esse fim. Listam-se, abaixo, uma descrição sobre as arquiteturas de RNC utilizadas no presente trabalho, que

foram escolhidas devido ao seu amplo uso na literatura e o fato de ser possível encontrar suas implementações prontas já treinadas em outros projetos:

1. ***Inception Resnet V1***: A família de redes *Inception* foi introduzida por Szegedy et al. (2015) para lidar com os problemas de variação no tamanho do *kernel* de convolução (em função de o dado de interesse estar mais globalmente ou localmente situado na imagem). A solução proposta foi realizar múltiplas convoluções em um mesmo nível na rede, com filtros de resoluções diferentes, fazendo uma concatenação dos resultados antes de enviar à camada seguinte. Cada bloco que fazia uma operação desse tipo era denominado "*Inception block*". Desde a primeira *Inception* criada, denominada Inception V1 (popularmente chamada de *GoogLeNet*), foram criadas diversas versões da rede, cada uma implementando novas melhorias. A *Inception ResNet* foi apresentada juntamente com a *Inception V4* (SZEGEDY et al., 2016) buscando deixar os módulos de convolução até então implementados mais uniformes, já que alguns eram demasiadamente complexos. A rede sofreu então algumas alterações em suas operações iniciais feitas antes da imagem passar pelos *Inception Blocks*. A variação utilizada nesse trabalho, ResNet V1 (HE et al., 2016), inclui operações residuais nos *Inception Blocks*, tendo *stems* e configurações de hiper-parâmetro ligeiramente diferentes da versão ResNet V2.

2. ***OpenFace NN4 Small 1***: Schroff, Kalenichenko e Philbin (2015) utilizaram variações da *Inception V1* com resoluções de entrada e profundidade reduzidas para que pudessem ser executadas em dispositivos móveis, e foram treinadas especificamente para tarefa de gerar os já citados *embedding vectors*. A variação denominada *FaceNet NN4* possui resolução de entrada 96×96, reduzindo consideravelmente o custo computacional. O projeto *OpenFace* (AMOS; LUDWICZUK; SATYANARAYANAN, 2016) alterou então essa variação para trabalhar em uma base de dados reduzida, di-

minuindo o número de parâmetros da rede, resultando na *OpenFace NN4 Small*.

3. ***VGG Face***: Parkhi, Vedaldi e Zisserman (2015) elaboraram um método para a criação de bases de dados de larga escala que pudesse competir com as bases de grandes empresas como Google e Facebook. Em seus testes, foi desenvolvida uma arquitetura de RNC denominada VGG Face (nome baseado em Visual Geometry Group, o grupo da Universidade de Oxford, responsável pelo trabalho). A arquitetura segue o princípio das demais principais da literatura, porém possui alta profundidade, com um total de 22 camadas e 37 unidades profundas. Sua resolução de entrada é 224x224.

2.3.4 Classificação e Verificação Facial

Com a representação adequada das imagens de face, numa maneira que seja possível diferenciá-las, o que será feito em seguida dependerá do problema a ser resolvido e do cenário em questão. A Classificação Facial tem por objetivo atribuir uma imagem de face de uma pessoa a um rótulo relacionado a essa pessoa na base de dados, utilizando para tal fim algoritmos de Classificação (SEÇÃO 2.1). A Verificação Facial é a tarefa de se comparar duas imagens de face, verificando se pertencem à mesma pessoa ou não, geralmente baseando-se na distância entre seus *embedding vectors*.

2.4 Trabalhos Relacionados

Para a obtenção de informações atualizadas sobre o tema envolvendo o presente trabalho, foi feita uma ampla busca por publicações relacionadas ao assunto em questão. Entende-se por "trabalhos relacionados" ao presente tema as publicações que implementaram, de alguma forma, o RF em vídeo usando alguma

arquitetura de *deep learning*. Essa pesquisa foi feita utilizando as ferramentas *ScienceDirect*[1] , *IEEE Xplore Digital Library*[2] , e *Google Scholar*[3] . Foram considerados somente os trabalhos feitos a partir de 2015, priorizando-se os mais recentes sobre o tema. Foram inseridas as strings de busca "*video face recognition*" e "*video facial recognition*". Cada trabalho encontrado foi submetido a uma breve leitura objetivando filtrar somente pelos trabalhos que realizassem RF em vídeo, obtendo um total de 14 trabalhos. Foram excluídos então aqueles que não faziam o uso de *deep learning* em seus métodos (uma vez que o presente trabalho foca-se no uso de diferentes arquiteturas de *deep learning* no contexto de RF), o que resultou então nos 10 trabalhos que compõem o referencial desta seção.

A maioria dos trabalhos a serem citados compartilham algumas características em comum, sendo que uma dessas características é o conjunto de bases de dados comumente usadas, voltadas para a avaliação de métodos de RF. Essas bases geralmente possuem protocolos de avaliação e apresentação de resultados pré-estabelecidos.

As bases de dados YouTube Face Dataset (YTF) (WOLF; HASSNER; MAOZ, 2011) e Youtube Celebrities Dataset (YTC) (KIM et al., 2008) contêm vídeos retirados da plataforma YouTube, sendo que a YTF contém 3.425 vídeos de 1.595 pessoas diferentes, enquanto a YTC compõe-se de somente 1500 vídeos de um total 35 pessoas (todas celebridades) diferentes. Outra base de dados que é composta somente de famosos é a Celebrity-1000 (C-1000) (LIU et al., 2014), composta de 159.726 sequências de vídeos de 1000 pessoas diferentes, cobrindo variações de pose, resolução e iluminação.

[1] Elsevier B.V. **ScienceDirect**. 2020. Disponível em: <https://www.sciencedirect.com>. Acesso em: 19 ago. 2020.

[2] IEEE. **IEEE Xplore**. 2020. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso em: 19 ago. 2020.

[3] Google. **Google Scholar**. 2020. Disponível em: <https://scholar.google.com.br/>. Acesso em: 19 ago. 2020.

Existem inúmeros projetos, como o Janus (IARPA, 2013), da *Intelligence Advanced Research Projects Activity* (IARPA), que fomentam pesquisas relacionadas a RF com objetivos diversos. No caso da IARPA, o interesse é a segurança, e sua parceria com o *National Institute of Standards Technology* (NIST) (IARPA, 2018) resultou no lançamento de desafios de RF sobre bases de dados pré-estabelecidas (NIST, 2015), como é o caso do conjunto de bases de dados *IARPA Janus Benchmark* (IJB). A IJB-A (KLARE et al., 2015) possui 5.712 imagens estáticas e 2085 vídeos de face, sendo que abrange 500 pessoas diferentes. Já a IJB-S (KALKA et al., 2018) simula dados capturados por câmeras de segurança e contém imagens estáticas e vídeos de 202 pessoas diferentes. O NIST também lançou o desafio *Point-and-Shoot Challenge* (PaSC) (BEVERIDGE et al., 2013), cuja base de dados contém 9376 imagens estáticas de 293 pessoas diferentes e 2802 vídeos de 265 do total de pessoas presentes na parte estática da base de dados. Durante a criação dessa base, cada ação das pessoas foi filmada com uma câmera de alta resolução (1920x1080) e uma das três câmeras de baixa resolução (entre 640x480 e 1280x720), dividindo a base de dados, respectivamente, nos conjuntos *control* e *handheld* (1401 vídeos em cada conjunto). Essa divisão é considerada na maioria dos trabalhos feitos sobre essa base.

Uma outra base de dados comum a mais de um trabalho é a COX Face DB (HUANG et al., 2015), que contém, para cada uma das 1000 pessoas registradas, uma foto tirada por uma câmera digital e três vídeos da mesma ação, cada um obtido por uma filmadora diferente. Para três cenários de RF foi estabelecido um protocolo diferente. Cada um dos cenários pode ser subdividido de acordo com a câmera de vídeo que realizou a captura, ou seja, o cenário V2S por exemplo, subdivide-se em V1-S, V2-S e V3-S. Assim ocorre com cada cenário.

Em Herrmann, Willersinn e Beyerer (2016) foi considerada a tarefa de RF no contexto de vídeos de baixa resolução. Para tanto, foi desenvolvida uma arquitetura de rede convolucional adaptada para a situação, denominada *LrfNet*,

que toma como entrada uma imagem de face de resolução 32×32 (em pixels) e gera uma *embedding vector* como saída de tamanho 123. O método de agregação utilizado, denominado LMM (*local mean method*), consiste em obter um número pequeno de centróides via *K-means* para cada conjunto de *embedding vectors* no hiperplano, sendo que a distância entre dois conjuntos de imagens é obtido através da distância mínima entre as centróides dos mesmos. Esse método foi testado na base YTF e em uma base de dados própria, construída a partir de imagens de câmeras de vigilância em três lugares diferentes ao longo de três dias, obtendo-se a 168 filmagens com 71 pessoas diferentes. Para a base de dados própria, a verificação facial resultou em uma acurácia de verificação média percentual de $87{,}1 \pm 1{,}14$, obtida usando apenas uma centróide e distância euclidiana no LMM. Usando essa mesma configuração na base YTF obteve-se a acurácia de verificação percentual de $80{,}3 \pm 1{,}6$. Com 10 centróides, a variância muda para 1,5%.

Um metodo de aprendizagem por reforço foi proposto em Rao, Lu e Zhou (2017). O método, que foi denominado *attention-aware deep reinforcement learning* (ADRL), consiste em converter cada *frame* de vídeo para uma representação temporal através de uma CNN, uma camada recorrente local e uma camada de *pooling* temporal, e em seguida buscar os *frames* que melhor representem os vídeos, denominados "*attentions*", por meio de um processo de decisão de Markov. Foi desenvolvida uma arquitetura própria de rede neural com 3 camadas convolucionais intercaladas com 2 camadas de *max pooling*, sendo que, para melhores resultados, a rede sofreu ajustes finos. A etapa final de verificação compara tais *frames* obtidos e gera o resultado. A classificação foi feita através do método KNN. O algoritmo foi testado nas bases de dados YTF, PaSC e YTC. Para a verificação nas bases de dados YTF, por meio de um método próprio que envolveu o cálculo da similaridade de cossenos, foi obtida as acurácia percentual de $96{,}52 \pm 0{,}054$. Essa configuração obteve, na base PaSC, 95,67% e 93,78% de acurácia de verificação para as porções *Control* e *Handheld*, respectivamente. A similaridade de

cossenos também foi usada para a classificação sobre a base YTC, sendo usada no cálculo das distâncias do método KNN. A acurácia obtida com a classificação foi de 97,82%.

A proposta de solução feita por Bao et al. (2017) foi direcionada para os cenários S2V (*still-to-video*) e V2S (*video-to-still*). Dessa forma, foi estabelecida uma arquitetura de *deep learning* que toma como entrada uma imagem estática e um vídeo e gera um par de vetores de *features* como saída, um referente ao vídeo e o outro à imagem estática. O treinamento dessa arquitetura é dado via *Stochastic Gradient Descend* (SGD), de forma que a função de perda busca minimizar a distância entre os vetores gerados, enquanto maximiza a distância entre diferentes classes. Na etapa de classificação facial foi utilizado o KNN, sendo que os testes foram realizados na base de dados COX Face. Vale a pena citar o resultado obtido para os cenários S2V, onde as acurácias de classificação percentuais foram 43,57$\pm$1,21, 45,71$\pm$1,14 e 64,24$\pm$1,44 para SV-1, SV-2 e SV-3, respectivamente.

Uma nova arquitetura de rede neural foi desenvolvida em Yang et al. (2017) para efetuar a agregação de um vídeo de face e/ou um conjunto de imagens estáticas de face de forma a gerar uma representação compacta desses dados que fosse eficiente para a tarefa de RF. Denominada *Neural Agregation Network* (NAN), essa arquitetura é composta de dois módulos, sendo que o primeiro é uma RNC responsável pela transformação de cada imagem estática (ou *frame* de vídeo) em um vetor de *features* (*embedding vector*) e o segundo é responsável pela agregação desses vetores. A arquitetura RNC do primeiro módulo é uma *GoogleLeNet* com normalização de *Batch*. O segundo módulo, baseado no mecanismo de memória de atenção, atribui de forma inteligente um peso a cada *embedding vector* e soma todos os vetores, gerando assim a representação final. Para a verificação facial, uma abordagem siamesa (que consiste executar a rede neural em um par de imagens e utilizar a saída das duas como entrada em uma outra rede para se obter a saída final) foi utilizada, enquanto para a classificação, foi adicionada uma

camada totalmente conectada e uma operação de *softmax*. O algoritmo foi testado nas bases de dados IJB-A, YTF e Celebrity-1000.

Dos resultados obtidos para Yang et al. (2017), vale citar os provenientes das melhores configurações em cada base. Na IJB-A, para a verificação, a taxa de aceitação verdadeira (*true accept rate* – TAR) percentual, que mede a probabilidade de um verdadeiro positivo, quando limita-se a taxa de falsa aceitação (*false accept rate* - FAR) em 0,1 é de 97,9$\pm$0,4. O TAR percentual para a identificação na mesma base ficou em 98,6$\pm$0,3, acurado em *Rank 10* (sendo que um algoritmo acurado em *Rank K* atribui uma certa probabilidade maior que 0 de o elemento ser atribuído a até K classes diferentes, escolhendo a maior). Esses dois resultados foram obtidos primeiro agregando as imagens em cada mídia e depois agregando as *features* da mídia em um *template*. Para a base YTF, foi obtida a acurácia de verificação percentual de 95,72$\pm$0,64. Para a base C-1000, foram obtidas as acurácias de classificação de 90,44% e 88,76% nas configurações *closed set* (não há elementos desconhecidos na classificação) e *open set* (há elementos desconhecidos na classificação), quando se executa o algoritmo considerando 100 pessoas diferentes dessa base e acurada em *Rank 1*.

Em Parchami, Bashbaghi e Granger (2017) foi proposta uma solução que gerasse resultados satisfatórios no contexto do RF em câmeras de vigilância, sendo consideradas as variações típicas de ambientes não controlados (tais como pose, iluminação, etc.). Desenvolveu-se uma arquitetura de *deep learning*, denominada *HaarNet*, que captura as regiões de interesse (ROIs) do vídeo em questão a as compara com as ROIs de outros vídeos na base de dados. Esse processo se dá por meio de uma rede global de truncamento que aprende a face como um todo e outras três redes responsáveis por aprender atributos assimétricos locais. Todas as redes foram construídas baseadas na arquitetura *Inception* e utiliza filtros *haar*. Para a classificação, foi adicionada uma camada totalmente conectada ao final da rede seguida de uma operação de *softmax*. Para os testes, foram usadas

as bases de dados *COX Face DB* e *ChokePoint* (WONG et al., 2011), sendo que o conteúdo das mesmas foi devidamente adaptado de maneira a serem incluídas as variações consideradas.*Chokepoint* é uma base de dados na qual 54 pessoas foram filmadas passando por dois portões a partir de três câmeras diferentes, sendo que 25 passaram pelo primeiro portão e 29 pelo segundo. As acurácias de classificação percentuais obtidas para *COX Face DB* foram de 98,86±0,37, 97,58±0,77 e 98,97±0,15 para os conjuntos de dados SV-1, SV-2 e SV-3, respectivamente, e de 98,26±0,49, 95,27±0,12 e 99,26±0,69 para os conjuntos de dados V1-S, V2-S e V3-S, respectivamente. A área (em %) sob a curva precisão × revocação (AUPR) média para a base *ChokePoint* ficou, em 99,36±0,59.

Ding e Tao (2017) também fizeram modificações em sua base de dados de forma a melhor simular os ambientes não controlados. Trabalhando no contexto de câmeras de vigilância, foi treinada uma CNN insensível a imagens borradas. Para isso, a algumas imagens das bases de dados utilizadas – PaSC, COX Face DB e YTF – foram borradas artificialmente. Desenvolveu-se uma arquitetura de rede neural, denominada *Trunk-Branch Ensemble CNN* (TBE-CNN), que gera um vetor de *features* para cada *frame* separadamente. Após a transformação pela TBE-CNN, os *frames* são agregados via *average pooling*. Citam-se, a seguir, os melhores resultados obtidos. Na base PaSC, o método foi avaliado no contexto de verificação facial, onde a taxa de verificação obtida foram de 96,23% e 95,85% para os conjuntos *control* e *handheld*, respectivamente. Quando é feita uma "limpeza" manual dos resultados na detecção facial, essas taxas aumentam para 97,80% e 96,12% para os conjuntos *control* e *handheld*, respectivamente. A taxa de classificação média percentual em COX Face DB para os cenários V3-S, S-V3 e V2-V3 foram, respectivamente, 98,96±0,17, 95,74±0,67 e 99,33±0,19. A classificação foi feita seguindo o protocolo em Huang et al. (2015). A taxa de verificação média percentual para YTF ficou em 94,96±0,31

Em Hu et al. (2018) foi proposta uma abordagem *adversarial* para o RF em vídeos inteiros, visando lidar com dados redundantes e o problema de baixa qualidade nos vídeos. Propôs-se então uma arquitetura de rede neural composta de duas partes. Na parte de *adversarial embedding learning* (AEL) a rede extrai *features* dos *frames* de vídeo através de aprendizado *adversarial*, "competindo"com um discriminador responsável por diferenciar *embedding vectors* pertencentes à face de uma mesma pessoa. Na parte de *variational aggregation learning* (VAL) também se faz uso de aprendizado *adversarial*, de forma que uma unidade de inferência variacional infere atributos para gerar a representação que une todos os *embedding vectors* dos *frames* do vídeo, e um outro discriminador compara o vídeo real com a representação gerada. Para a etapa de identificação, Hu et al. (2018) também fez o uso de uma camada totalmente conectada seguida de uma operação *softmax*. Para esse trabalho, vale citar os melhores resultados obtidos para cada base que também foram maiores que os *baselines* comparados. A acurácia de verificação média para YTF foi de 96,84%. Para IJB-A, no cenário de verificação 1 para 1 e considerando um FAR de 0,001 obteve-se um TAR de 94,4%. Para YTC a acurácia de classificação obtida foi de 98,23%. Para C-1000 a acurácia de classificação obtida em um *close set* de 100 sujeitos ficou em 94,82%.

Rao, Lu e Zhou (2018) criaram uma arquitetura de rede neural para a agregação (DAN), que gera um conjunto de faces para representar o melhor possível o vídeo facial em questão. Para isso, foram desenvolvidas uma rede de agregação G, uma rede de discriminação D e uma rede de extração de atributos F. Faz-se o uso de aprendizado *adversarial* entre as redes G e D, no qual a rede G transforma vídeos inteiros em pequenos conjuntos de *frames* e a rede discriminadora verifica se a imagem produzida é real ou sintetizada, maximizando assim a aproximação entre os *frames* gerados e o vídeo de entrada. Por fim, a rede F faz a extração de atributos dos *frames* gerados. Dado que essa rede é totalmente convolucional, alterações bruscas nos vídeos poderiam prejudicar o desempenho do modelo.

Para resolver esse problema foi adicionado um método de pré-processamento nos vídeos denominado V-STN para normalizar os *frames* dos vídeos. A classificação é feita através de um método do vizinho mais próximo. foram testadas as bases de dados PaSC, YTC e MARS (ZHENG et al., 2016), uma base de dados composta de vídeos de 1.261 pedestres capturados no câmpus da Universidade de Tsinghua, por pelo menos 2 câmeras diferentes, gerando um total de 20,715 imagens sequenciais. Destacam-se os seguintes resultados, por serem maiores que os *baselines* comparados pelo trabalho: Para a base PaSC com o FAR fixado em 0,01, foram obtidas, para os conjuntos *control* e *handheld* respectivamente, as taxas de verificação de 94,88% e 92,12%. Para YTC foi obtida a acurácia de classificação percentual de 97,70$\pm$0,72. Ambos os resultados não foram obtidos usando a V-STN. Para MARS, com o uso da V-STN, foram obtidas as acurácias de classificação de 70,23% e 84,65%, acuradas em mAP (*mean Average Precision* - a média das precisões médias) e *Rank* 1, respectivamente.

Uma arquitetura LSTM foi proposta em Gong, Shi e Jain (2019) para a agregação de *frames*, juntamente com um modelo RNC para a realização do RF em vídeos. A arquitetura completa foi denominada *recurrent embedding aggregation network* (REAN), funcionando da seguinte maneira: Para cada *frame* do vídeo é extraído um *embedding vector* pela RNC. O conjunto de features de cada *frame* é inserido sequencialmente na LSTM de forma que a representação agregada seja construída de maneira inteligente, sendo feito o uso informações anteriores, onde um peso é atribuído a cada *frame*, indicando sua importância no contexto do vídeo. O modelo REAN foi treinado na base de dados UMDFaceVideo e avaliado nas bases de dados IJB-S, YTF e PaSC. Foi obtida uma acurácia média de verificação de 96,60% em YTF e uma taxa de verificação média na PaSC de 96,52% (não superou os *baselines*), com a taxa de aceitação de erro (FAR) fixada em 0,01. Na classificação sobre a base IJB-S foram obtidas as taxas de classificação 98,02%, 99,50% e 99,50%, na configuração *closed set* acurados em *Rank* 1, 5 e 10, respectivamente.

Motivado pelo contexto de muitas variações nas expressões, poses e luzes nos vídeos de faces, Mao et al. (2019) estabeleceu uma nova arquitetura de rede neural para a extração de atributos denominada *compact second-order network* (COSONet). Durante o treinamento, a base de dados regional foi aumentada através da geração automática de novas imagens de face, simulando nelas borrões por movimento, borrões por desfoque e diminuição da resolução. Os *embedding vectors* gerados pela COSONet são agregados via *average pooling*. Durante o treinamento, foram testadas as funções de perda *softmax loss* e *ring loss*. O TAR obtido com o FAR fixado em 0,01 para a base PaSC foi de 92,7% e 58,1% para *control* e *handheld* respectivamente. Quando se adiciona o aumento de dados e se usa o *ring loss* os resultados aumentam para 95,1% e 80,5% para *control* e *handheld*, respectivamente. Para a base IJB-A o TAR percentual foi de 85,8$\pm$1,9 com o FAR fixado em 0,01. Quando se adiciona o aumento de dados e se usa o *ring loss* o resultado aumenta para 89,4$\pm$1,4.

Como é possível observar, os trabalhos da literatura geralmente desenvolvem suas próprias arquiteturas de RNC, seja a partir de uma arquitetura pré-existente ou não, e geralmente implementam a agregação de vídeos, o que facilita a tarefa de verificação ou classificação facial em bases de dados que contém vídeos. A Tabela 2.1 mostra os maiores resultados obtidos por cada trabalho citado e a respectiva configuração que o levou àquele resultado.

Tabela 2.1 – Maiores resultados para os trabalhos relacionados

Trabalho	**Arquitetura**	**Base de dados**	**Cenário**	**Métrica**	**Valor (%) ***
Herrmann, Willersinn e Beyerer (2016)	LrfNet	YTF	V2V (verificação)	Acurácia	87,1 ±1,14
Rao, Lu e Zhou (2017)	ADRL	YTC	V2V (classificação)	Acurácia	97,82
Bao et al. (2017)	(sem nome)	Cox Face	SV-3 (classificação)	Acurácia	64,24 ±1,44
Yang et al. (2017)	NAN	IJB-A	Misto (classificação)	TAR	98,6 ±0,3
Parchami, Bashbaghi e Granger (2017)	Haar Net	ChokePoint	V2V (classificação)	AUPR	99,36 ±0,59
Ding e Tao (2017)	TBE-CNN	COX Face DB	V2-V3	Acurácia	99,33 ±0,19
Hu et al. (2018)	(sem nome)	YTC	V2V (classificação)	Acurácia	98,23%
Rao, Lu e Zhou (2018)	DAN	YTC	V2V (classificação)	Acurácia	97,70 ±0,72
Gong, Shi e Jain (2019)	REAN	IJB-S	S2V (classificação)	Acurácia	99,50%
Mao et al. (2019)	COSONet + ring loss	PaSC (Control)	V2V (verificação)	TAR	95,1%

* Os valores obtidos para as métricas não são passíveis de comparação, uma vez que remetem a métricas diferentes obtidas sobre diferentes bases de dados.
Fonte: Do autor (2020).

A partir desse referencial teórico, foi possível estabelecer uma metodologia para o presente trabalho, cuja descrição detalhada consta no Capítulo 3.

3 METODOLOGIA

A presente pesquisa se classifica como sendo de natureza aplicada, com objetivos exploratórios, abordagens quantitativa e qualitativa, através de procedimentos experimentais fundamentados em referências bibliográficas. A natureza aplicada da pesquisa relaciona-se ao objetivo de aplicação dos métodos fundamentados em base teórica visando a criação de uma aplicação de Reconhecimento Facial que possa ser usada por casas inteligentes. Tal objetivo é de caráter exploratório, já que envolve o aperfeiçoamento de modelos já presentes na literatura. Os resultados deste trabalho são avaliados quantitativamente, por meio de métricas já conhecidas na literatura, embora o modelo final venha a sofrer uma análise qualitativa de seu comportamento, sendo comparando ao funcionamento que se espera de uma aplicação desse tipo.

A aplicação e os testes foram desenvolvidos na linguagem Python 3.7, fazendo uso de algumas bibliotecas extras. Uma delas é a OpenCV (BRADSKI, 2000), para obtenção e tratamentos e vídeos e imagens, bem como a detecção e captura das faces. A biblioteca utilizada para o alinhamento é a Dlib (KING, 2009). Keras (CHOLLET et al., 2015) e TensorFlow (ABADI et al., 2015) foram utilizadas para as operações com *Deep Learning*. As arquiteturas de rede testadas, bem como seus pesos, foram obtidos de terceiros. As arquiteturas e pesos dos modelos *NN4 Small* e *VGG Face* foram disponibilizadas por Amos, Ludwiczuk e Satyanarayanan (2016) e Santha (2019), respectivamente. Para a *Inception Resnet V1*, sua arquitetura foi disponibilizada por Sandberg (2017) e seus pesos por Taniai (2019). Os métodos de classificação foram implementados a partir da biblioteca Scikit-learn (PEDREGOSA et al., 2011), exceto o XGBoost, implementado pela biblioteca de mesmo nome (CHEN; GUESTRIN, 2016). O código fonte da avaliação e solução final encontram-se disponíveis para fins de pesquisa[1] .

[1] SILVA Ítalo D. G. **eFaceRecon**. 2020. Disponível em: <https://github.com/italodellagarza/eFaceRecon>. Acesso em: 19 ago. 2020.

A solução final tem sua arquitetura geral explicada na Seção 3.1. Sobre a base de dados descrita na Seção 3.2, foram feitos testes detalhados na Seção 3.3.

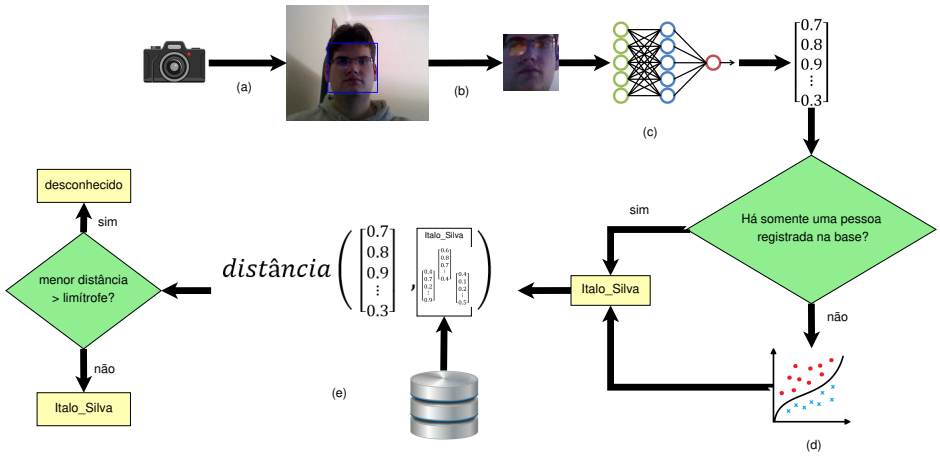
3.1 Arquitetura Geral

A solução final é composta de uma emulação em computador de como seria o funcionamento em uma casa inteligente, em que haveria uma câmera instalada na porta para obter as imagens e um computador para fazer as operações (podendo ser um sistema embarcado). Nessa emulação, faz-se o uso da *webcam* do computador para simular a câmera. O funcionamento da aplicação foi baseado no passo a passo descrito em Rovai (2018), funcionando da seguinte maneira: Primeiramente, uma pessoa solicitaria ao detector que deseja entrar na casa. O detector então captura uma imagem da face dessa pessoa. Um processo de identificação facial é feito a fim de atribuir um nome à pessoa de acordo com os dados registrados na base de pessoas cadastradas (que poderiam entrar na casa). A partir desse momento, surge a necessidade de verificação, isto é, certificar de que esse rótulo foi corretamente atribuído. Para essa etapa, faz-se um cálculo da distância entre os *embedding vectors* da imagem adquirida e de cada uma das imagens relacionadas ao rótulo em questão na base de dados. Se a menor dessas distâncias for maior que uma determinada medida limítrofe, o rótulo da imagem é mudado para "desconhecido". Se só houver uma pessoa registrada na base de dados, a aplicação faz somente a etapa de verificação.

Se o processo de reconhecimento do sistema terminasse realizando somente uma única iteração, qualquer erro ocorrido durante essas etapas poderia afetar negativamente o resultado final, fazendo com que alguém que pudesse entrar na casa fosse impedido ou alguém indesejado invadisse a casa. É fundamental que essa última situação seja evitada, já que se trataria de um sério problema de segurança. Por isso, para aumentar a precisão do reconhecimento, o processo é repetido 30 vezes e o rótulo final é determinado pela maioria dos rótulos atribuídos

a cada iteração. Caso o rótulo que mais se repita dentro do conjunto de rótulos obtidos não se repita ao menos 24 vezes, o indivíduo é classificado como "não identificado". Caso a pessoa seja classificada como "desconhecida"ou "não identificada"a porta não se abre para a mesma. Caso contrário, é permitida a passagem. A simulação implementada não controla porta alguma, mas avisa quando a pessoa foi detectada ou não. Uma iteração dessa solução é ilustrada na Figura 3.1.

Figura 3.1 – Diagrama ilustrando uma única iteração na aplicação principal.



O esquema ilustra a iteração feita para cada imagem pela aplicação, incluindo (a) detecção, (b) alinhamento, (c) extração de atributos, (d) classificação e (e) cálculo das distâncias entre o *embedding vector* obtido e os outros da base de dados.
Fonte: Do autor (2020).

A solução também teria a possibilidade de registro de uma nova pessoa na base de dados. Nesse caso, 50 imagens da face da pessoa a se cadastrar são detectadas e é feito o alinhamento e extração de atributos em cada imagem. O modelo de classificação é então treinado novamente com a presença dessas novas instâncias.

3.2 Bases de dados

A primeira base de dados, doravante neste texto denominada "base de dados de teste", foi construída através de fotografias e vídeos de alguns componentes do grupo de pesquisa em que este trabalho foi conduzido. Totalizam-se 136 fotos e 12 vídeos, sendo que cada um dos 7 componentes tem de 18 a 20 fotos registradas na base e 4 possuem 3 vídeos registrados. Foi pedido para cada componente que cumprisse as seguintes instruções ao gravar os vídeos e tirar as fotos para compor a base de dados:

- Fazer uso de distâncias e fundos diferentes.
- Variar a expressão facial.
- Não inclinar muito o rosto para os lados.
- Registrar somente a parte frontal do rosto, ou seja, não registrar o rosto em perfil.
- Resolução razoável do aparelho de gravação e/ou foto.
- Variação da aparência nas fotos, i.e., barba, penteado, vestimentas, etc..
- Duração máxima de 20 segundos por vídeo, os quais não poderiam ter mudanças bruscas na imagem, i.e., movimentos bruscos com a câmera ao serem gravados.

Essas instruções tinham como objetivo fazer com que os testes fossem o mais realistas possíveis de acordo com o objetivo final da aplicação, ao mesmo tempo em que respeitassem certas limitações dos métodos de detecção e alinhamento utilizados na aplicação. Esses métodos possuem dificuldades em detectar faces que não estejam de frente para a câmera e na posição vertical, bem como imagens com luminosidade e muito baixa ou muito alta e/ou resolução muito baixa.

A segunda base de dados utilizada para os testes, denominada neste texto de "base de dados de famosos"ou "base de dados de pessoas famosas", foi a utilizada na metodologia de Krasser (2018) (SEÇÃO 3.3). Trata-se de um fragmento da base de dados *Labeled Faces in the Wild* (HUANG et al., 2007), contendo 100 imagens de face no total, sendo 10 para cada uma das 10 personalidades famosas registradas na base. A Figura 3.2 mostra um exemplo de cada pessoa nessa parte da base.

Figura 3.2 – Exemplo de imagens da segunda parte da base de dados

Fonte: Adaptado de Krasser (2018)

3.3 Avaliação

Os trabalhos da literatura testam separadamente o processo de classificação facial do processo de verificação. O mesmo se aplicou a este trabalho, pois ambas as etapas têm sua importância na arquitetura definida. O teste deste trabalho, adaptado de Krasser (2018), avaliou a classificação e a verificação facial por meio da combinação entre os seus respectivos modelos e as arquiteturas de redes neurais, a fim de escolher a combinação que gera melhores resultados para serem as que compõem a solução final.

A escolha da combinação para a solução final deste trabalho obedeceu aos seguintes critérios: Primeiramente foi calculada uma média em cada uma das métricas de cada uma das combinações dos modelos, tanto os de verificação quanto

os de classificação. Em seguida uma média ponderada foi feita entre as métricas resultantes de cada combinação, em que a precisão recebeu peso 4 e as outras receberam peso 1. A combinação de maior pontuação dentre as combinações de classificação é escolhida e, dentre as combinações para a verificação que contenham o modelo de RNC do primeiro modelo escolhido, a combinação de verificação será a que tiver a maior pontuação. O motivo de se dar maior importância à precisão é que a mesma avalia melhor a segurança da aplicação final, uma vez que ela informa a quantidade de identificações corretas em uma classe dentre todos os elementos que foram atribuídos a ela, ou seja, mostra o quanto o modelo está excluindo as identificações incorretas.

A solução final também foi avaliada calculando-se a média do tempo de execução para 10 execuções que, juntamente com uma análise qualitativa do programa, considerando principalmente a aplicação no mundo real, serviram de base para discutir sua usabilidade e possíveis melhorias (CAPÍTULO 4). O programa foi testado em um computador com processador *Intel*®*Core*™ i5-3317U com 7,5 Gigabytes de memória RAM e sistema operacional *Linux Ubuntu* 20.04.1 LTS, com 4 componentes previamente registrados.

Os testes feitos nas duas partes da base de dados consideraram o cenário S2S (*still-to-still*), mas somente para a base de dados de teste foram feitas avaliações com o cenário V2V (*video-to-video*). Nesse caso, os vídeos foram transformados em conjuntos de imagens (separados por *frames*). O objetivo da avaliação no cenário V2V foi se aproximar um pouco mais da aplicação real, que fará uso de vídeos para registrar e identificar as pessoas, ao passo que a avaliação do cenário S2S foi testar como a aplicação reagiria a mudanças, tanto na condição das imagens (resolução, luminosidade, etc.) quanto na aparência dos indivíduos (barba, penteado, etc.).

Para as duas bases de dados, foram obtidos os resultados para o cenário S2S. Para a base de dados de teste foi feita uma avaliação com o cenário V2V, com

os vídeos também separados em treino e teste (na mesma proporção estabelecida no cenário S2S) e separados em *frames* antes de passar pelo RF. Para todos os vídeos, a captura da imagem foi realizada a cada 15 *frames*. É importante citar que, para esse cenário, os *frames* em que o detector não conseguiu capturar a face ou, caso detectado, não passou pela etapa de alinhamento, foi descartado na etapa seguinte. Vale citar que todos os *embedding vectors* gerados pela *Inception* sofreram uma normalização L2 antes de serem classificados e verificados.

As métricas para a avaliação usadas neste trabalho são bastante comuns na literatura. São elas: a precisão média, a revocação média, a medida-F média (ou macro-F1) e o percentual de acertos. Cada métrica é calculada por classe e é feita uma média aritmética para todas as classes. Dessa forma, cada uma dessas métricas foram obtidas para os classificadores ou para a combinação de cálculo da distância e valor limítrofe na verificação. Pode-se descrever essas métricas em termos de FP (*false positives* – falsos positivos), FN (*false negatives* – falsos negativos), TP (*true positives* – verdadeiros positivos), TN (*true negatives* – verdadeiros negativos). Entende-se positivo e negativo como um rótulo atribuído ou não à uma classe, respectivamente, no caso da classificação, ou como duas faces entendidas como sendo ou não do mesmo indivíduo, respectivamente, no caso da verificação. As fórmulas para as métricas podem ser descritas por

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.1}$$

$$\text{Revocação} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3.2}$$

$$\text{Medida-F} = 2 * \frac{\text{Precisão} * \text{Revocação}}{\text{Precisão} + \text{Revocação}}, \tag{3.3}$$

$$\text{Percentual de acertos} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{3.4}$$

Vale citar que cada uma das fórmulas foi convertida em percentuais para este trabalho.

3.3.1 Avaliação de classificação

A avaliação de classificação buscou avaliar a melhor combinação entre os métodos de classificação explicados na Seção 2.1 e as arquiteturas de RNC definidas na Subseção 2.3.3. Para ambas as bases de dados, no cenário S2S, as imagens foram primeiramente separadas em treino e teste (sendo 70% das imagens para treino) e as etapas de RF foram executadas em seguida. No cenário V2V, foram separados dois vídeos para treino e um para teste para cada indivíduo, independentemente do número de *frames* em cada vídeo, sendo que a classificação durante o teste para um dado vídeo foi determinado pela maioria das classificações sobre seus *frames*.

3.3.2 Avaliação de verificação

Como a base da etapa de verificação consiste em avaliar se uma pessoa é a mesma que a atribuída pelo método de classificação, a avaliação deve testar diferentes cálculos de distâncias e limítrofes entre pares de imagens. Para isso, foi definido um banco de pares de imagens escolhidas aleatoriamente, porém garantindo que ao menos 40% dos pares teriam imagens da mesma pessoa. Essa avaliação foi feita considerando a combinação entre o tipo de cálculo da distância e valor da distância limítrofe, para cada RNC que extraiu os atributos. Para o cenário V2V, nesse caso, a formação dos pares de vídeos foi feita seguindo os mesmos critérios. A comparação foi feita *frame* a *frame* para cada par de vídeo, sendo que o primeiro *frame* do primeiro vídeo foi comparado ao primeiro *frame* do segundo vídeo, o segundo *frame* ao segundo *frame* do segundo vídeo, e assim por diante, ignorando os *frames* restantes de algum dos vídeos caso o número de *frames* fosse diferente entre os dois. Os resultados foram agrupados para cada vídeo

considerando a maioria dos resultados sobre seus *frames*, como feito na avaliação de classificação.

As duas maneiras de se calcular limítrofes que foram testadas são a Distância Euclidiana e a Similaridade de Cossenos. Ambas são citadas na Subseção 2.1.1. A combinação dessas fórmulas juntamente com o formato de saída dos *embedding vectors* (isto é, seu intervalo de valores) interfere no valor de limítrofe ideal para a verificação facial. Sendo assim, foram testados valores limítrofes diferentes para cada uma das combinações possíveis, sendo que os intervalos de limítrofes foram selecionados manualmente de acordo com seu impacto no resultado das métricas, dando preferência para a Precisão, já que era o que se desejou maximizar para a solução final. Os valores limítrofes escolhidos para o teste em cada uma das combinações estão descritos na Tabela 3.1.

Tabela 3.1 – Valores limítrofes escolhidos para as diferentes combinações entre a RNC de extração de *features* e método de cálculo da distância

RNC	**Cálculo de distância**	**Conjuntos de limítrofes testadas**
NN4 Small 1	Distância Euclidiana	$[0,83; 0,87; 0,89]$
	Similaridade de Cossenos	$[0,305; 0,355; 0,405]$
Inception Resnet V1	Distância Euclidiana	$[1,0; 1,05; 1,1]$
	Similaridade de Cossenos	$[0,4; 0,5; 0,6]$
VGG Face	Distância Euclidiana	$[2,8 \times 10^{-4}; 2,9 \times 10^{-4}; 3,0 \times 10^{-4}]$
	Similaridade de Cossenos	$[1,00 \times 10^{-4}; 1,05 \times 10^{-4}; 1,10 \times 10^{-4}]$

Fonte: Do autor (2020)

Cada valor constante fixado para a execução dos testes neste capítulo, com exceção dos valores para as limítrofes, foi escolhido de maneira aleatória.

Com a metodologia definida em seus detalhes, foram realizados então os testes e a coleta e análise dos resultados, que estão descritas no Capítulo 4.

4 RESULTADOS E DISCUSSÃO

O presente capítulo apresenta e discute os resultados obtidos de acordo com a metodologia descrita no Capítulo 3. Os resultados estão divididos entre os obtidos pelo teste de classificação facial (SEÇÃO 4.1) e verificação facial (SEÇÃO 4.2), estando também subdivididos de acordo com os cenários estudados neste trabalho (S2S para as bases de teste e de famosos e V2V somente para a base de teste). Uma discussão acompanha cada resultado, destacando pontos relevantes e formulando hipóteses para as questões cujas respostas não foram obtidas ou foram obtidas parcialmente.

4.1 Resultados para a classificação

Os resultados para a classificação foram satisfatórios de uma maneira geral. É possível notar, como já era esperado, que alguns métodos de classificação se comportam melhor ou pior dependendo da RNC ao qual foram combinados. Isso provavelmente acontece graças ao intervalo de valores que os elementos dos *embedding vectors* gerados pelas redes pertencem. Um exemplo bastante evidente é a combinação do método SVM com a *VGG Face*, que gera vetores com elementos muito próximos de 0, com intervalos muito pequenos. Essa combinação não funcionou, mesmo com algumas tentativas de normalização dos resultados. Nota-se também que os resultados para a base de dados de pessoas famosas (SUBSEÇÃO 4.1.1) foram melhores que os da base de dados de teste, feita para esse trabalho (SUBSEÇÃO 4.1.2). É possível que a qualidade dos equipamentos tenha influenciado, uma vez que a maioria das fotos da base de pessoas famosas foram tiradas por profissionais da mídia. Esses profissionais geralmente dispõem de equipamentos de alta qualidade, em contraste com as câmeras de celular utilizadas para obter as imagens da base de dados de teste.

4.1.1 Base de dados de famosos

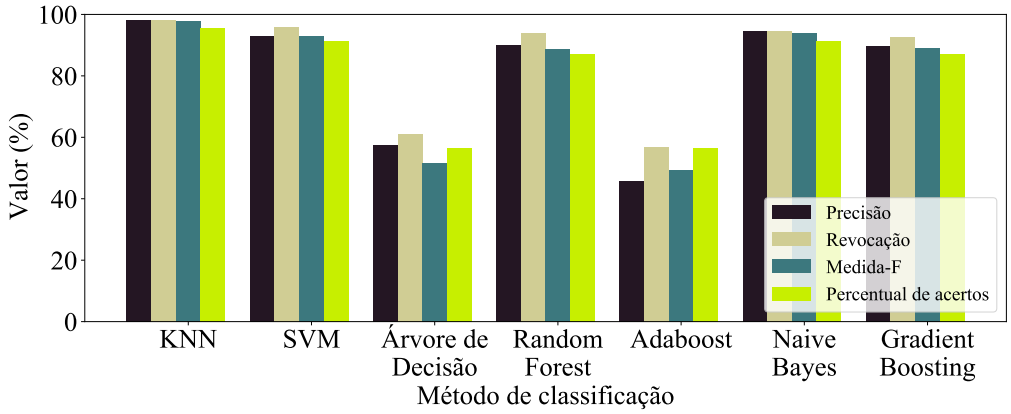
Na base de dados de famosos, as combinações de métodos conseguiram se aproximar de 100% nos resultados. Para esse caso, destacam-se as combinações KNN + *NN4 Small 1*, SVM + *Inception Resnet V1* e KNN + *VGG Face*. Os resultados para essas combinações foram melhores que os demais, se considerarmos a mesma rede neural. Os resultados para a classificação sobre a base de dados de famosos são mostrados, abordando todo o cenário, na Tabela 4.1 e, para cada RNC, ilustrados nas Figuras 4.1, 4.2 e 4.3.

Tabela 4.1 – Resultados para a classificação facial para a base de dados dos famosos.

RNC	**Classificador**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	KNN	**98,00**	**98,00**	**97,78**	**95,65**
	SVM	93,00	96,00	93,06	91,30
	Árvore de Decisão	57,50	61,17	51,67	56,52
	Random Forest	90,00	94,00	88,71	86,96
	Adaboost	45,83	56,83	49,33	56,52
	Naive Bayes	94,67	94,67	93,78	91,30
	Gradient Boosting	89,67	92,67	89,06	86,96
Inception Resnet V1	KNN	87,50	89,00	84,29	82,61
	SVM	**95,00**	**96,00**	**94,17**	**91,30**
	Árvore de Decisão	64,33	63,17	59,57	56,52
	Random Forest	85,00	89,00	83,71	82,61
	Adaboost	86,67	91,50	85,29	82,61
	Naive Bayes	85,67	82,33	77,76	73,91
	Gradient Boosting	70,83	76,50	67,52	73,91
VGG Face	KNN	**94,67**	**96,00**	**94,39**	**91,30**
	SVM	0,43	10,00	0,83	4,35
	Árvore de Decisão	52,50	57,00	48,24	52,17
	Random Forest	90,00	94,00	89,05	86,96
	Adaboost	64,17	66,50	58,88	65,22
	Naive Bayes	78,33	86,00	80,17	86,96
	Gradient Boosting	64,33	67,00	57,17	65,22

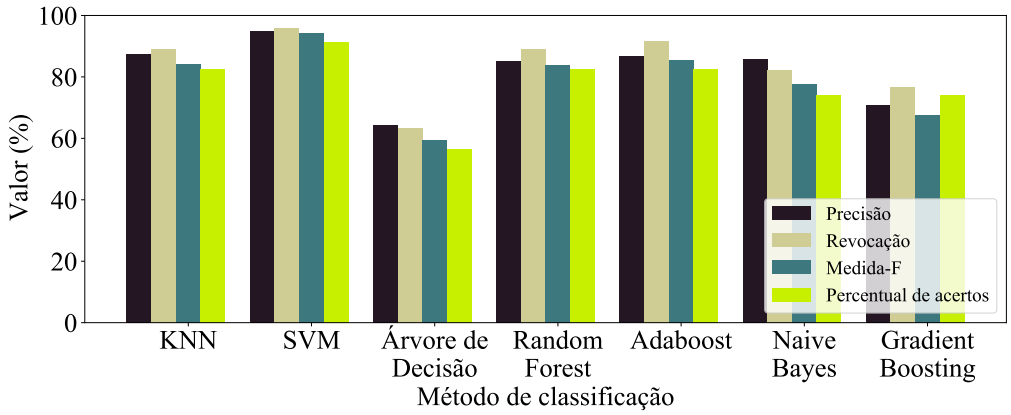
Fonte: Do autor (2020).

Figura 4.1 – Gráfico com os resultados para a classificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *NN4 Small 1*.



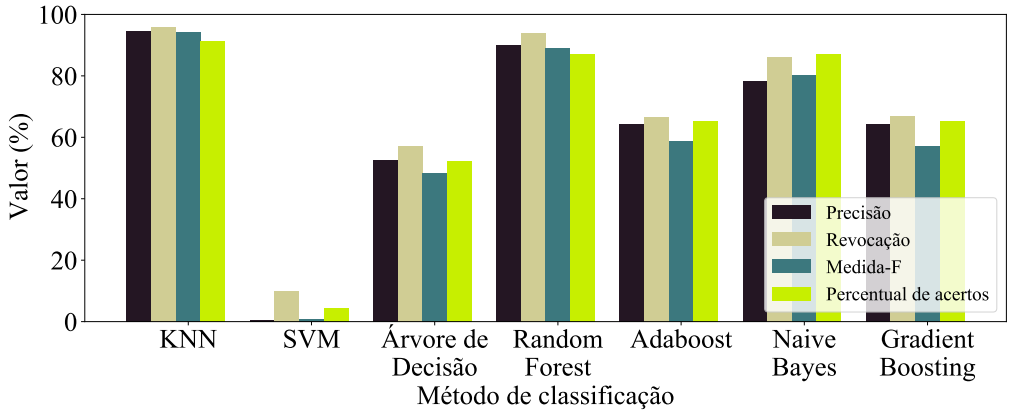
Fonte: Do autor (2020).

Figura 4.2 – Gráfico com os resultados para a classificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.3 – Gráfico com os resultados para a classificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *VGG Face*.



Fonte: Do autor (2020).

4.1.2 Base de dados de teste

Como foi dito na introdução desta seção, os resultados diminuem de uma maneira geral para esta base de dados, embora algumas combinações para o cenário V2V tenham alcançado os 100% em todos os resultados. Como foram poucos vídeos para teste, no entanto, isso pouco pode ser levado em consideração na comparação entre os resultados, servindo mais para se ter uma ideia geral de como as combinações se comportam quando lidam com imagens obtidas em sequência, i.e., vídeos. Além disso, deve-se considerar que, como no cenário V2V a classificação é feita várias vezes e o resultado é definido pela maioria, a tendência é que os resultados tendam a ser mais acurados.

4.1.2.1 Cenário S2S

Para o cenário S2S, os resultados variam bastante. Encontram-se percentuais bastante baixos, um pouco abaixo de 40%, até alguns resultados acima de 70% (desconsiderando-se a combinação SVM + *VGG Face*). Destacam-se, para este caso, as combinações *Random Forest+NN4 Small 1*, KNN + *Inception Resnet V1* e *Naïve Bayes* + *VGG Face* (pela alta precisão). Os resultados para a classifica-

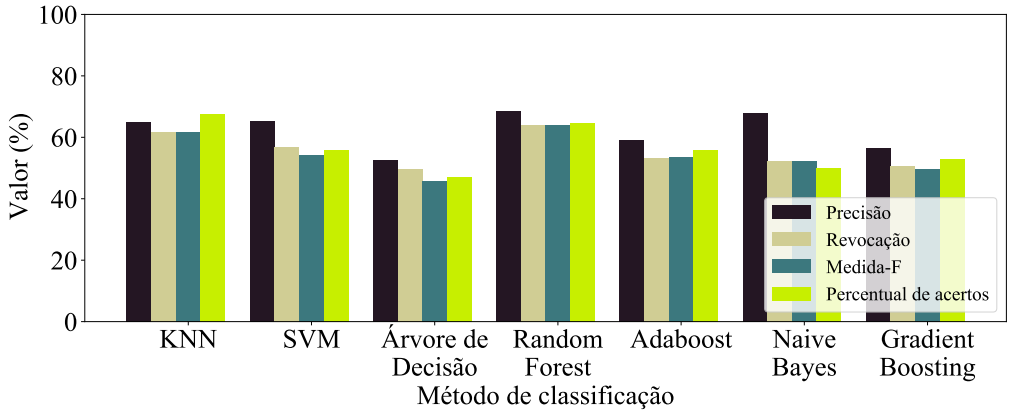
ção sobre a base de dados de teste no cenário S2S são mostrados, abordando todo o cenário, na Tabela 4.2 e, para cada RNC, ilustrados nas Figuras 4.4, 4.5 e 4.6.

Tabela 4.2 – Resultados para a classificação facial sobre a base de dados de teste no cenário S2S.

RNC	**Classificador**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	KNN	64,83	61,67	61,63	67,65
	SVM	65,24	56,90	54,17	55,88
	Árvore de Decisão	52,72	49,76	45,86	47,06
	Random Forest	**68,57**	**64,05**	**64,05**	**64,71**
	Adaboost	59,21	53,33	53,50	55,88
	Naive Bayes	67,84	52,14	52,20	50,00
	Gradient Boosting	56,35	50,48	49,68	52,94
Inception Resnet V1	KNN	**73,16**	**69,52**	**68,34**	**76,47**
	SVM	71,16	67,14	66,92	73,53
	Árvore de Decisão	57,69	44,76	44,36	47,06
	Random Forest	71,63	72,38	70,98	79,41
	Adaboost	72,38	59,29	63,87	64,71
	Naive Bayes	66,63	54,05	55,58	58,82
	Gradient Boosting	65,00	55,24	56,27	58,82
VGG Face	KNN	62,65	58,81	58,10	64,71
	SVM	0,84	14,29	1,59	5,88
	Árvore de Decisão	42,86	39,29	38,23	38,24
	Random Forest	68,57	59,29	61,49	64,71
	Adaboost	65,95	61,67	60,90	61,76
	Naive Bayes	**73,81**	**43,33**	**49,85**	**47,06**
	Gradient Boosting	56,55	56,43	54,20	55,88

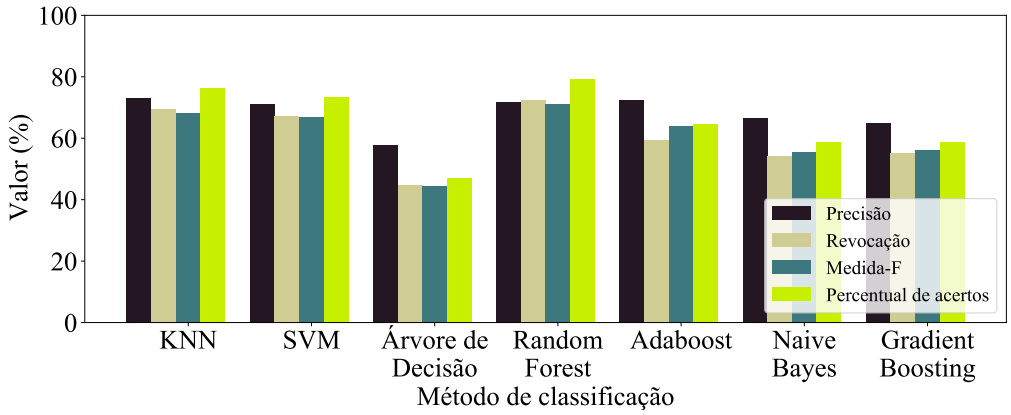
Fonte: Do autor (2020).

Figura 4.4 – Gráfico com os resultados para a classificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *NN4 Small 1*.



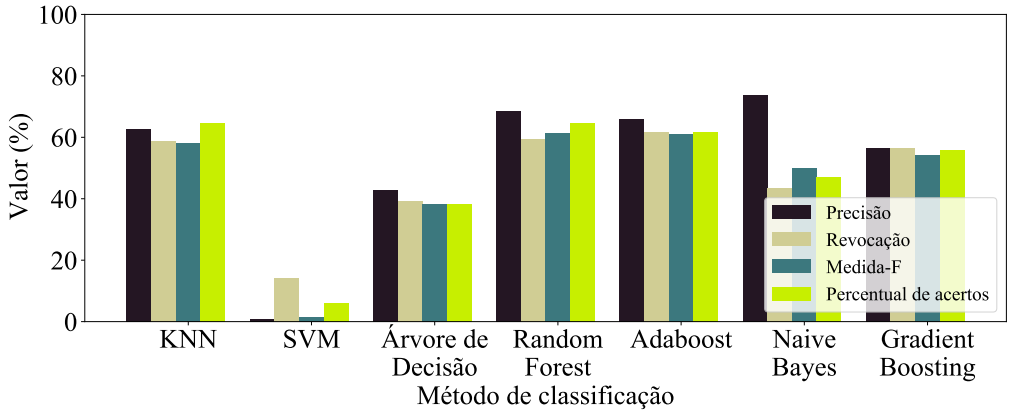
Fonte: Do autor (2020).

Figura 4.5 – Gráfico com os resultados para a classificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.6 – Resultados para a classificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *VGG Face*.



Fonte: Do autor (2020).

4.1.2.2 Cenário V2V

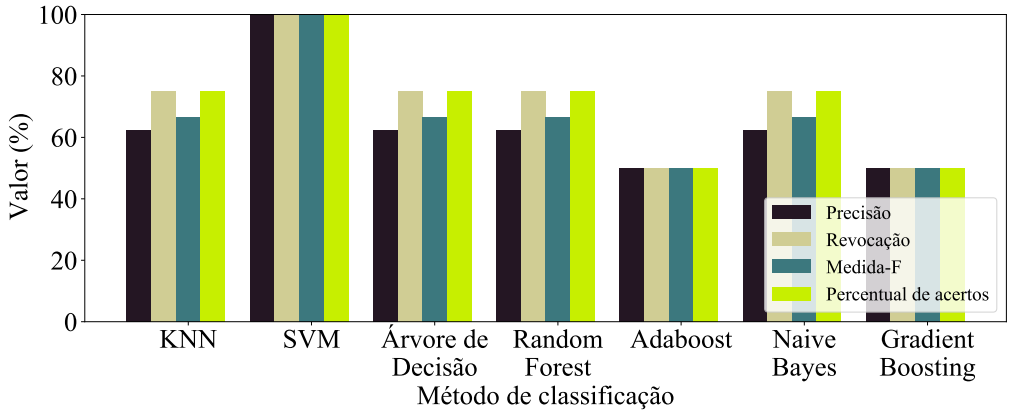
Neste cenário, destaca-se a combinação SVM + *NN4 Small 1*, que obteve 100% em todas as métricas, superando as outras combinações com a mesma rede neural. O máximo nas métricas também se repete para todas as combinações com a rede *VGG Face*, excluindo os métodos SVM e Árvore de Decisão. Os resultados para a *Inception Resnet V1* foram muito similares entre si, com todas as métricas abaixo de 80. Como dito anteriormente, a pouca quantidade de vídeos pode ter influenciado os resultados, tanto os positivos quanto os negativos neste caso. Os resultados para a classificação sobre a base de dados de teste no cenário V2V são mostrados, abordando todo o cenário, na Tabela 4.3 e, para cada RNC, ilustrados nas Figuras 4.7, 4.8 e 4.9.

Tabela 4.3 – Resultados para a classificação facial sobre a base de dados de teste no cenário V2V.

RNC	**Classificador**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	KNN	62,50	75,00	66,67	75,00
	SVM	**100,00**	**100,00**	**100,00**	**100,00**
	Árvore de Decisão	62,50	75,00	66,67	75,00
	Random Forest	62,50	75,00	66,67	75,00
	Adaboost	50,00	50,00	50,00	50,00
	Naive Bayes	62,50	75,00	66,67	75,00
	Gradient Boosting	50,00	50,00	50,00	50,00
Inception Resnet V1	KNN	37,50	50,00	41,67	50,00
	SVM	**62,50**	**75,00**	**66,67**	**75,00**
	Árvore de Decisão	50,00	50,00	50,00	50,00
	Random Forest	**62,50**	**75,00**	**66,67**	**75,00**
	Adaboost	**62,50**	**75,00**	**66,67**	**75,00**
	Naive Bayes	**62,50**	**75,00**	**66,67**	**75,00**
	Gradient Boosting	**62,50**	**75,00**	**66,67**	**75,00**
VGG Face	KNN	**100,00**	**100,00**	**100,00**	**100,00**
	SVM	6,25	25,00	10,00	25,00
	Árvore de Decisão	62,50	75,00	66,67	75,00
	Random Forest	**100,00**	**100,00**	**100,00**	**100,00**
	Adaboost	**100,00**	**100,00**	**100,00**	**100,00**
	Naive Bayes	**100,00**	**100,00**	**100,00**	**100,00**
	Gradient Boosting	**100,00**	**100,00**	**100,00**	**100,00**

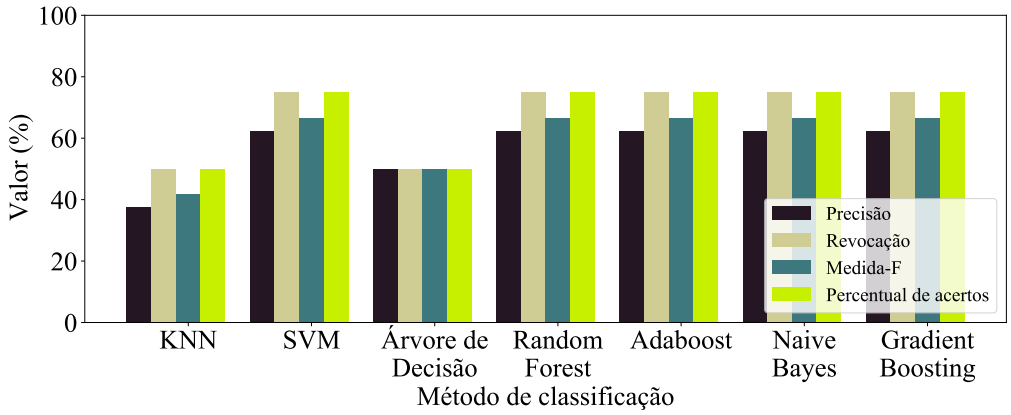
Fonte: Do autor (2020).

Figura 4.7 – Gráficos com os resultados para a classificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *NN4 Small 1*.



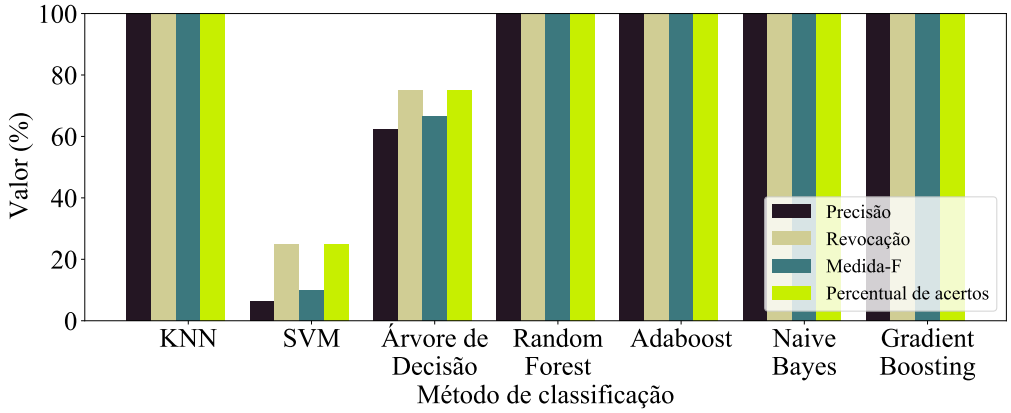
Fonte: Do autor (2020).

Figura 4.8 – Gráficos com os resultados para a classificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.9 – Gráficos com os resultados para a classificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *VGG Face*.



Fonte: Do autor (2020).

4.2 Resultados para a verificação

Os resultados para a verificação facial encontram-se numa faixa de valores um pouco inferior aos resultados da classificação em geral. Nota-se que os intervalos de limítrofes testados pouco variam nos resultados, provavelmente pela

pouca distância entre eles. Novamente, é possível enxergar a discrepância entre a base de dados de famosos e a base de dados de teste.

4.2.1 Base de dados de famosos

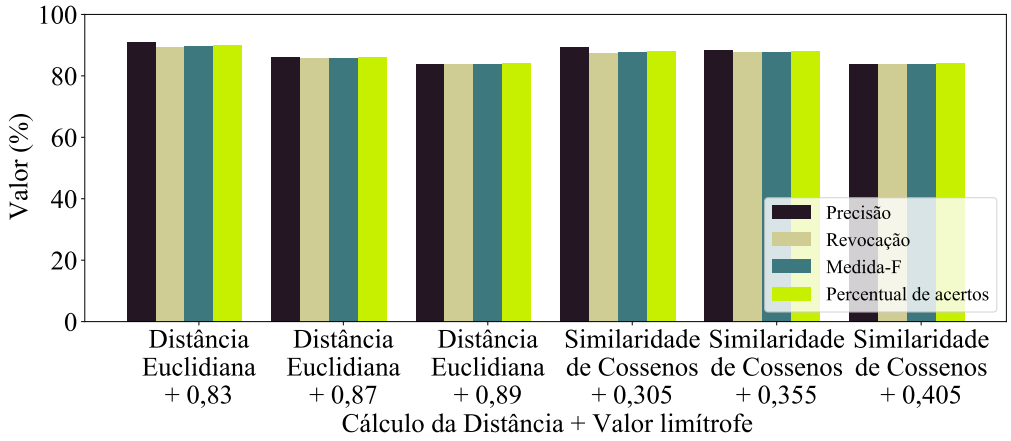
A base de dados de famosos teve resultados satisfatórios, com alguns poucos valores próximos a 60%. A maioria dos resultados variou entre 70% e 90%. Destacam-se, neste caso, as combinações *NN4 Small 1* + Distância Euclidiana com a limítrofe igual a 0,83 e *Inception Resnet V1* + Distância Euclidiana com a limítrofe igual a 1,05. Para a rede *VGG Face* houve três combinações com resultados similares, ou seja, nenhuma combinação se sobressaiu em relação às demais. Os resultados para a verificação sobre a base de dados de famosos são mostrados, abordando todo o cenário, na Tabela 4.4 e, para cada RNC, ilustrados nas Figuras 4.10, 4.11 e 4.12.

Tabela 4.4 – Resultados para a verificação facial sobre a base de dados de famosos.

RNC	**Cálculo da distância**	**Limítrofe**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	Euclidiana	0,83	**90,83**	**89,45**	**89,80**	**90,00**
		0,87	86,04	85,75	85,86	86,00
		0,89	83,90	83,90	83,90	84,00
	Similaridade de Cossenos	0,305	89,30	87,28	87,68	88,00
		0,355	88,34	87,60	87,82	88,00
		0,405	83,90	83,90	83,90	84,00
Inception Resnet V1	Euclidiana	1,00	85,11	80,76	81,08	82,00
		1,05	**86,45**	**82,93**	**83,33**	**84,00**
		1,10	83,51	81,08	81,40	82,00
	Similaridade de Cossenos	0,40	74,70	63,37	60,26	66,00
		0,50	85,11	80,76	81,08	82,00
		0,60	83,51	81,08	81,40	82,00
VGG Face	Euclidiana	$2,8 \times 10^{-4}$	**82,50**	**81,40**	**81,64**	**82,00**
		$2,9 \times 10^{-4}$	77,88	78,02	77,92	78,00
		$3,0 \times 10^{-4}$	77,34	76,81	75,96	76,00
	Similaridade de Cossenos	$1,00 \times 10^{-4}$	**82,50**	**81,40**	**81,64**	**82,00**
		$1,05 \times 10^{-4}$	**82,50**	**81,40**	**81,64**	**82,00**
		$1,10 \times 10^{-4}$	77,88	78,02	77,92	78,00

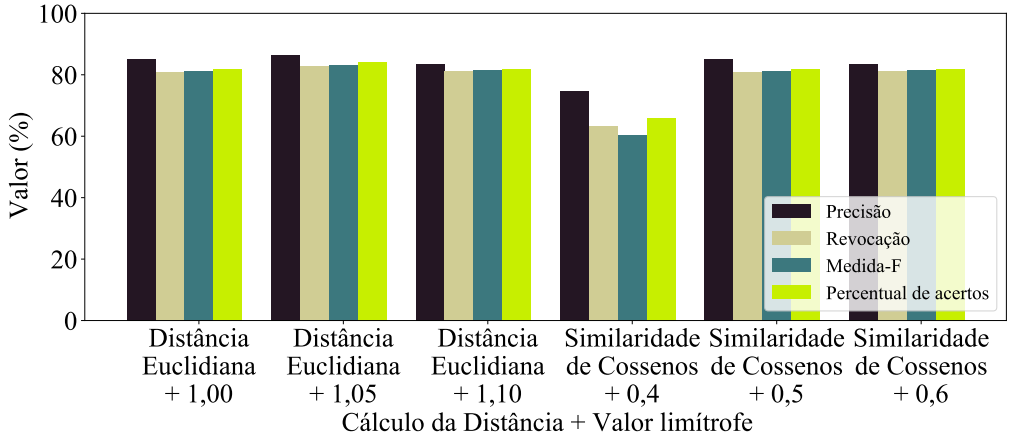
Fonte: Do autor (2020).

Figura 4.10 – Gráfico com os resultados para a verificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *NN4 Small 1*.



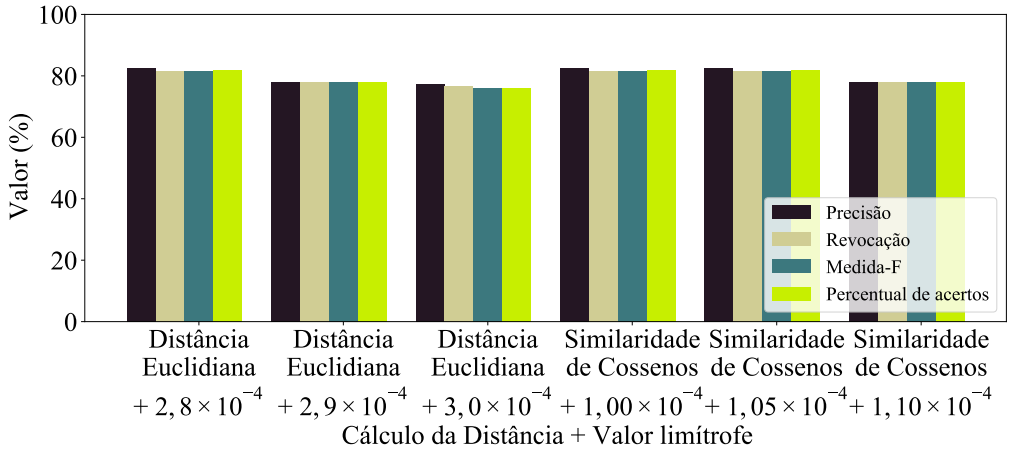
Fonte: Do autor (2020).

Figura 4.11 – Gráfico com os resultados para a verificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.12 – Gráfico com os resultados para a verificação facial sobre a base de dados de famosos com os atributos extraídos pela rede *VGG Face*



Fonte: Do autor (2020).

4.2.2 Base de dados de teste

Para a verificação na base de dados de teste, houve uma grande diferença de resultados entre os cenários S2S e V2V. No entanto, pelo pequeno número de dados no cenário V2V e pela maneira de se calcular os resultados, não fica claro o quanto essa diferença pode ser levada em consideração. No entanto, fica evidente que as condições em que as imagens foram obtidas impactaram nos resultados de verificação de maneira mais significativa que nos resultados de classificação.

4.2.2.1 Cenário S2S

Com resultados pouco satisfatórios, alguns próximos a 50% em alguns casos, o cenário S2S também mostrou que algumas combinações podem ultrapassar 60% em alguns resultados, ainda que não se distanciem muito dessa pontuação. Destaca-se, no presente cenário, a combinação *NN4 Small 1* + Distância Euclidiana com 0,87 de valor limítrofe, a única que se sobressaiu em relação às demais para a mesma rede. Os resultados para a verificação sobre a base de dados de teste

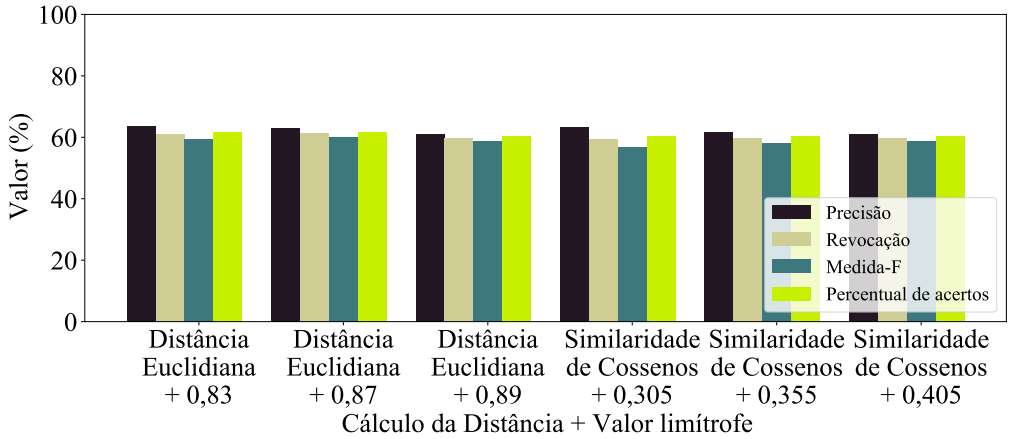
no cenário S2S são mostrados, abordando todo o cenário, na Tabela 4.5 e, para cada RNC, ilustrados nas Figuras 4.13, 4.14 e 4.15.

Tabela 4.5 – Resultados para a verificação facial sobre a base de dados de teste no cenário S2S.

RNC	**Cálculo da distância**	**Limítrofe**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	Euclidiana	0,83	**63,80**	**61,13**	**59,52**	**61,76**
		0,87	63,12	61,21	60,07	61,76
		0,89	61,17	59,78	58,79	60,29
	Similaridade de Cossenos	0,305	63,22	59,52	56,93	60,29
		0,355	61,67	59,70	58,26	60,29
		0,405	61,17	59,78	58,79	60,29
Inception Resnet V1	Euclidiana	1,00	60,15	58,18	56,41	58,82
		1,05	61,67	59,70	58,26	60,29
		1,10	**62,61**	**61,30**	**60,54**	**61,76**
	Similaridade de Cossenos	0,40	55,05	53,64	50,55	54,41
		0,50	60,15	58,18	56,41	58,82
		0,60	**62,61**	**61,30**	**60,54**	**61,76**
VGG Face	Euclidiana	$2,8 \times 10^{-4}$	**61,17**	**59,78**	**58,79**	**60,29**
		$2,9 \times 10^{-4}$	60,53	59,96	59,59	60,29
		$3,0 \times 10^{-4}$	58,80	58,61	58,50	58,82
	Similaridade de Cossenos	$1,00 \times 10^{-4}$	**61,17**	**59,78**	**58,79**	**60,29**
		$1,05 \times 10^{-4}$	**61,17**	**59,78**	**58,79**	**60,29**
		$1,10 \times 10^{-4}$	60,53	59,96	59,59	60,29

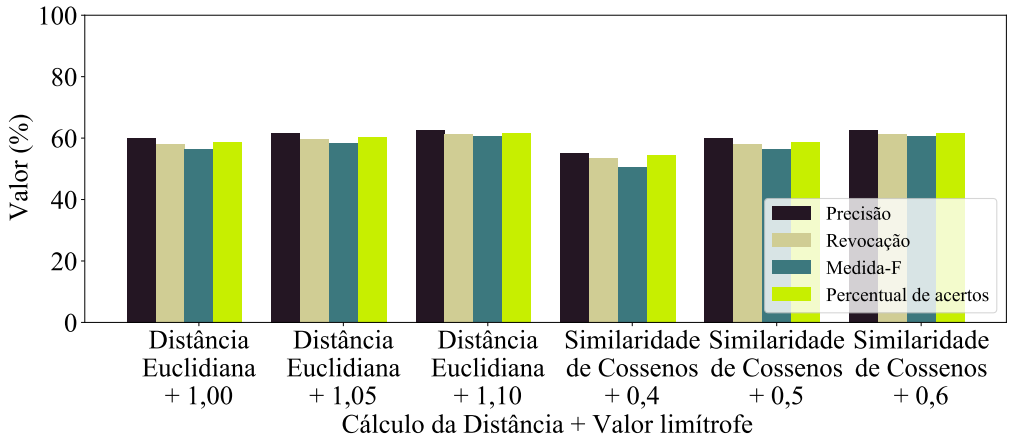
Fonte: Do autor (2020).

Figura 4.13 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *NN4 Small 1*.



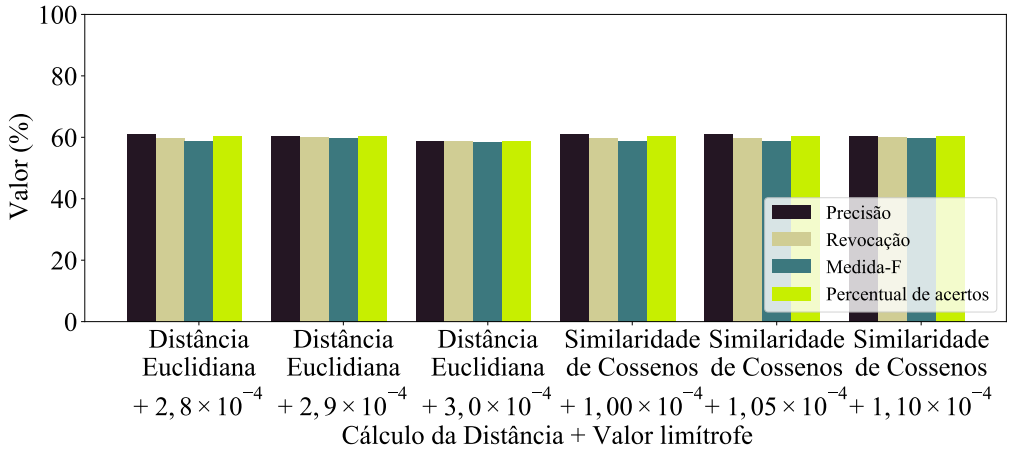
Fonte: Do autor (2020).

Figura 4.14 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.15 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário S2S com os atributos extraídos pela rede *VGG Face*.



Fonte: Do autor (2020).

4.2.2.2 Cenário V2V

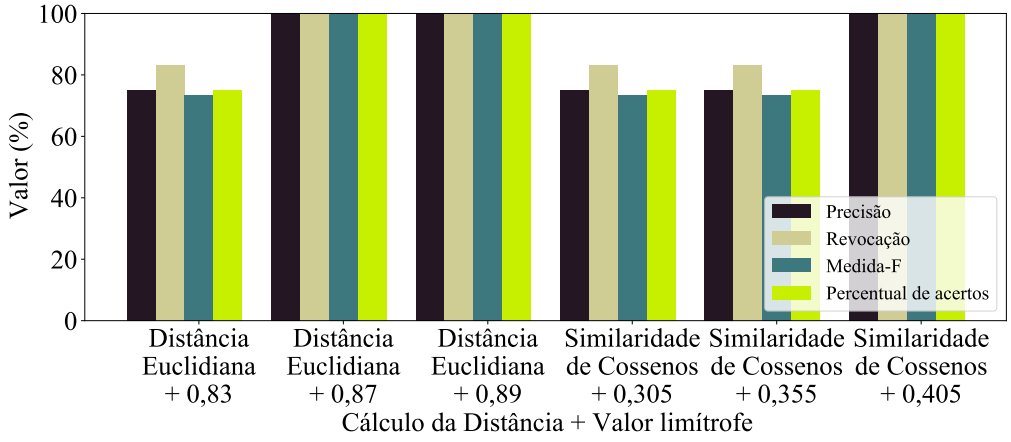
Para o cenário V2V, a condição dos resultados é similar aos resultados de classificação, ou seja, muitos resultados próximos ou iguais a 100%. Não é possível destacar nenhuma combinação em especial para este caso. Os resultados para a verificação sobre a base de dados de teste no cenário V2V são mostrados, abordando todo o cenário, na Tabela 4.6 e, para cada RNC, ilustrados nas Figuras 4.16, 4.17 e 4.18.

Tabela 4.6 – Resultados para a verificação facial sobre a base de dados de teste no cenário V2V.

RNC	**Cálculo da distância**	**Limítrofe**	**Precisão (%)**	**Revocação (%)**	**Medida-F (%)**	**Percentual de acertos**
NN4 Small 1	Euclidiana	0,83	75,00	83,33	73,33	75,00
		0,87	**100,00**	**100,00**	**100,00**	**100,00**
		0,89	**100,00**	**100,00**	**100,00**	**100,00**
	Similaridade de Cossenos	0,305	75,00	83,33	73,33	75,00
		0,355	75,00	83,33	73,33	75,00
		0,405	**100,00**	**100,00**	**100,00**	**100,00**
Inception Resnet V1	Euclidiana	1,00	**100,00**	**100,00**	**100,00**	**100,00**
		1,05	**100,00**	**100,00**	**100,00**	**100,00**
		1,10	**100,00**	**100,00**	**100,00**	**100,00**
	Similaridade de Cossenos	0,40	75,00	83,33	73,33	75,00
		0,50	**100,00**	**100,00**	**100,00**	**100,00**
		0,60	**100,00**	**100,00**	**100,00**	**100,00**
VGG Face	Euclidiana	$2,8 \times 10^{-4}$	75,00	83,33	73,33	75,00
		$2,9 \times 10^{-4}$	75,00	83,33	73,33	75,00
		$3,0 \times 10^{-4}$	75,00	83,33	73,33	75,00
	Similaridade de Cossenos	$1,00 \times 10^{-4}$	75,00	83,33	73,33	75,00
		$1,05 \times 10^{-4}$	75,00	83,33	73,33	75,00
		$1,10 \times 10^{-4}$	75,00	83,33	73,33	75,00

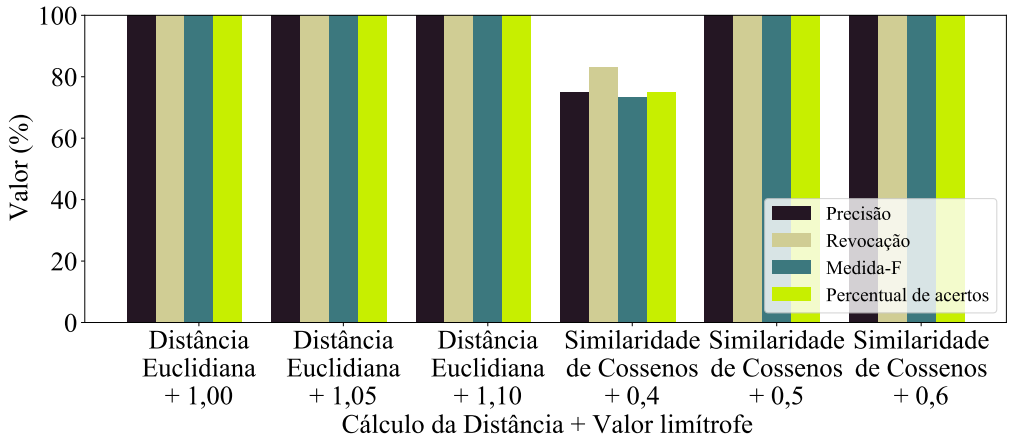
Fonte: Do autor (2020).

Figura 4.16 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *NN4 Small 1*.



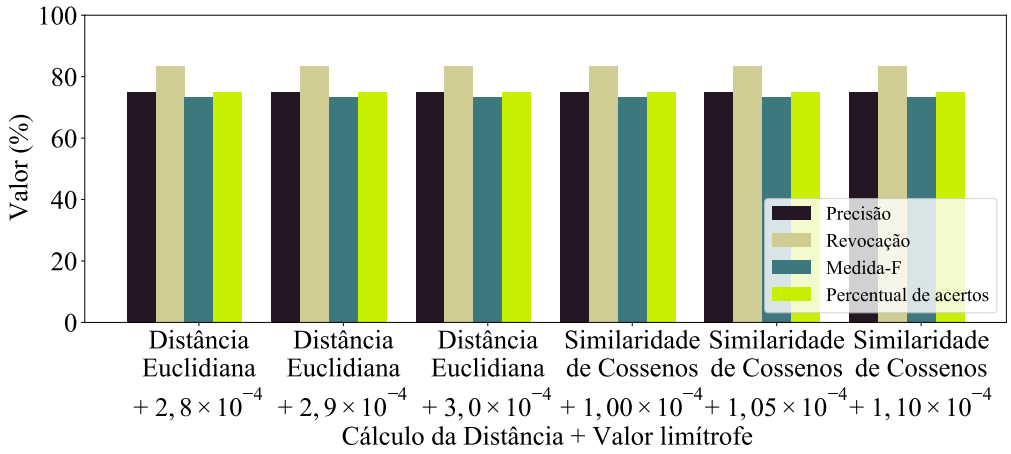
Fonte: Do autor (2020).

Figura 4.17 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *Inception Resnet V1*.



Fonte: Do autor (2020).

Figura 4.18 – Gráfico com os resultados para a verificação facial sobre a base de dados de teste no cenário V2V com os atributos extraídos pela rede *VGG Face*.



Fonte: Do autor (2020).

4.3 Pontuação Geral

A Tabela 4.7 mostra os resultados do cálculo da pontuação geral definido na Seção 3.3 para os resultados da classificação. Através dela, é possível notar que a combinação *VGG Face* + KNN está a frente das demais segundo os critérios adotados no presente trabalho, sendo portanto, a combinação escolhida para compor a solução final. As combinações *VGG Face* + *Random Forest*, *VGG Face* + *Naive Bayes* e *NN4 Small 1* + *SVM* também apresentaram boa pontuação, ficando acima de 80%. Nos resultados apresentados na Seção 4.1 percebe-se que a combinação escolhida apresentou bons resultados na base de dados de famosos e no cenário V2V da base de dados de teste, porém somente sua precisão ultrapassou 60% no cenário S2S da base de dados de teste. Outro ponto a se levar em consideração é o fato de essa rede possuir um numero de nós maior que as outras duas apresentadas neste trabalho, exigindo assim mais recursos de *hardware*.

A Tabela 4.8 mostra os resultados do cálculo de pontuação geral para os resultados de verificação. Dentre as combinações contendo a RNC *VGG Face*, os resultados são bastante semelhantes. A combinação contendo a Distância Euclidiana com limítrofe $2,8 \times 10^{-4}$ (escolhida para compor a solução final) e as combinações contendo o cálculo da distância via Similaridade de Cossenos com as limítrofes $1,00 \times 10^{-4}$ e $1,05 \times 10^{-4}$ foram de melhor desempenho com pontuação igual a 72,87%. Observa-se que essas combinações não foram a de melhor desempenho geral. Levando em consideração as combinações com as demais RNCs, a melhor combinação é *NN4 Small 1* + Distância Euclidiana + 0,87 como valor limitrofe, tendo uma pontuação final de 82,73%. A combinação escolhida, no entanto, se comportou de maneira similar as demais em cada cenário avaliado.

Tabela 4.7 – Pontuação calculada para as combinações de classificação.

RNC	Classificador	Pontuação Geral (%)
NN4 Small 1	KNN	76,21
	SVM	**84,77**
	Árvore de Decisão	58,08
	Random Forest	74,45
	Adaboost	52,17
	Naive Bayes	73,85
	Gradient Boosting	64,57
Inception Resnet V1	KNN	66,88
	SVM	**77,16**
	Árvore de Decisão	54,93
	Random Forest	74,82
	Adaboost	73,82
	Naive Bayes	70,40
	Gradient Boosting	66,58
VGG Face	KNN	**85,36**
	SVM	6,05
	Árvore de Decisão	53,39
	Random Forest	85,23
	Adaboost	75,97
	Naive Bayes	81,04
	Gradient Boosting	73,31

Fonte: Do autor (2020).

4.4 Resultados para a solução final

A solução final, implementada com os métodos descritos na Seção 4.3, gastou um tempo médio de 24,59 segundos para realizar cada reconhecimento e acertou 100% dos reconhecimentos. Provavelmente o tamanho da RNC *VGG Face* influenciou consideravelmente no tempo de execução total, uma vez que o KNN é relativamente veloz em relação aos outros métodos. O rótulo atribuído em cada iteração é quase sempre correto. A solução responde bem a variações na expressão e na iluminação do ambiente.

A implementação do método de detecção utilizada na presente solução só é capaz de detectar faces posicionadas de frente para a câmera e na posição vertical, além de não conseguir detectar faces em imagens muito escuras. Logo, a solução final também herdou essas características. Uma outra limitação do mo-

Tabela 4.8 – Pontuação calculada para as combinações de verificação.

RNC	**Cálculo da distância**	**Limítrofe**	**Pontuação Geral (%)**
NN4 Small 1	Euclidiana	0,83	76,28
		0,87	**82,73**
		0,89	81,47
	Similaridade de Cossenos	0,305	75,31
		0,355	74,92
		0,405	81,47
Inception Resnet V1	Euclidiana	1,00	80,87
		1,05	**81,95**
		1,10	81,55
	Similaridade de Cossenos	0,40	66,61
		0,50	80,87
		0,60	81,55
VGG Face	Euclidiana	$2,8 \times 10^{-4}$	**72,87**
		$2,9 \times 10^{-4}$	71,38
		$3,0 \times 10^{-4}$	70,52
	Similaridade de Cossenos	$1,00 \times 10^{-4}$	**72,87**
		$1,05 \times 10^{-4}$	**72,87**
		$1,10 \times 10^{-4}$	71,38

Fonte: Do autor (2020).

delo ocorre quando se faz a classificação de uma única pessoa na base de dados. O sistema tem dificuldades em distinguir indivíduos fisicamente parecidos, principalmente quando fazem uso de um mesmo tipo de acessório (como óculos, por exemplo). Nesse caso o programa não classifica o indivíduo não registrado como "desconhecido"como deveria. Quando mais pessoas são adicionadas, há uma distinção melhor entre as faces, e o rótulo é corretamente atribuído.

5 CONCLUSÃO

O presente trabalho realizou um estudo de diversas técnicas de reconhecimento facial, com ênfase em reconhecimento facial por vídeos, aplicando e avaliando esse conhecimento em uma simulação o mais próxima possível do mundo real. Pode-se dizer que, a partir de uma arquitetura simples de RF voltada para a aplicação em segurança de casas inteligentes, criou-se um sistema completo e que gera bons resultados, com possibilidade de ser melhorado ainda mais. Esse sistema criado priorizaria, se implementado, a segurança e o bem estar dos moradores de uma casa inteligente. Vale lembrar que algumas limitações no modelo, como o custo computacional e alguns erros de identificação e verificação, podem afetar negativamente alguns casos de uso da solução final, fazendo com que sejam necessárias algumas melhorias antes da aplicação em um sistema real de segurança.

Este capítulo descreve as contribuições teóricas realizadas neste projeto na Seção 5.1, as contribuições práticas na Seção 5.2 e propõe novas possibilidades de trabalhos a serem realizados na Seção 5.3.

5.1 Contribuições teóricas

Neste texto encontram-se diversos conhecimentos presentes na literatura que podem servir como introdução para quem nunca estudou sobre RF, contendo desde uma introdução sobre como RF se classifica dentro de PI, abordando diversas técnicas de Classificação, citando diversas arquiteturas de Redes Neurais com ênfase em RNC e, por fim, mostrando como todo esse conteúdo se aplica em RF e como o tema é tratado na literatura. A união do que foi apresentado no presente trabalho com outros conteúdos vindos da literatura pode aprimorar e criar novas arquiteturas de RF e servir como base para outras aplicações, como o já citado Reconhecimento de Humor.

O trabalho também comparou o desempenho de arquiteturas de RNC e métodos de classificação no contexto de RF, mostrando, nesse contexto, como cada uma se comporta em cada situação e algumas de suas limitações.

5.2 Contribuições práticas

O programa criado neste trabalho pode, com as devidas adaptações e algumas melhorias, ser implementado em um sistema real de segurança em uma casa inteligente ou autenticação de sistemas online. A arquitetura implementada também serve como base para a criação de outros sistemas de segurança envolvendo RF, bem como outros sistemas que envolvam o uso RF.

5.3 Trabalhos futuros

Conforme foi supracitado, a solução final ainda carece de melhorias, sobretudo envolvendo o tempo de execução, que mais pesa na implementação real da aplicação. Isso pode ser resolvido através da inclusão de outras redes e métodos de classificação na metodologia proposta, que podem gerar resultados similares na pontuação geral com um tempo médio de execução reduzido. Essas inclusões também podem se aplicar às outras melhorias, como verificação e identificação mais corretas. Otimizações de *hardware* na solução final e melhorias no próprio *hardware* da máquina que a executa também podem reduzir o tempo final da execução mantendo a mesma combinação para a arquitetura final. Além disso, vale citar que não é necessário que os possíveis trabalhos futuros se limitem detalhadamente à metodologia aqui proposta, podendo alterá-la a fim de escolher uma combinação final de métodos que melhor se adapte à necessidade do contexto. Isso vale também para o funcionamento geral da arquitetura proposta e seus parâmetros, tais como valor limítrofe, número de iterações, método para a obtenção do rótulo final a partir do rótulo de cada iteração, entre outros.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

AMOS, B.; LUDWICZUK, B.; SATYANARAYANAN, M. **OpenFace**: A general-purpose face recognition library with mobile applications. [S.l.], 2016.

BAO, T. et al. Video-based face recognition via convolutional neural networks. In: JIANG, X.; ARAI, M.; CHEN, G. (Ed.). **Second International Workshop on Pattern Recognition**. SPIE, 2017. v. 10443, p. 81 – 86. Disponível em: <https://doi.org/10.1117/12.2280286>.

BEVERIDGE, J. R. et al. The challenge of face recognition from digital point-and-shoot cameras. In: **IEEE Conference on Computer Vision and Pattern Recognition**. Portland, OR, USA: [s.n.], 2013. p. 1–8.

BRADSKI, G. The opencv library. **Dr. Dobb's Journal of Software Tools**, 2000. Disponível em: <https://www.drdobbs.com/open-source/the-opencv-library/184404319>.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2. Disponível em: <http://doi.acm.org/10.1145/2939672.2939785>.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

DING, C.; TAO, D. Trunk-branch ensemble convolutional neural networks for video-based face recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PP, 04 2017.

Elsevier B.V. **ScienceDirect**. 2020. Disponível em: <https://www.sciencedirect.com>. Acesso em: 19 ago. 2020.

GONG, S.; SHI, Y.; JAIN, A. K. **Recurrent Embedding Aggregation Network for Video Face Recognition**. 2019.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing (3rd Edition)**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

Google. **Google Scholar**. 2020. Disponível em: <https://scholar.google.com.br/>. Acesso em: 19 ago. 2020.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman, 2001.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.

HERRMANN, C.; WILLERSINN, D.; BEYERER, J. Low-resolution convolutional neural networks for video face recognition. In: **2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)**. Colorado Springs, CO, USA: [s.n.], 2016. p. 221 – 227.

HSU, C.-W.; LIN, C.-J. A comparison of methods for multiclass support vector machines. **IEEE Transactions on Neural Networks**, v. 13, n. 2, p. 415–425, March 2002.

HU, Y. et al. Adversarial embedding and variational aggregation for video face recognition. In: **2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)**. Redondo Beach, CA, USA: [s.n.], 2018. p. 1–8. ISSN 2474-9680.

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

HUANG, Z. et al. A benchmark and comparative study of video-based face recognition on cox face database. **IEEE transactions on image processing : a publication of the IEEE Signal Processing Society**, v. 24, p. 5967 – 5981, 10 2015.

IARPA. **Janus**. 2013. Disponível em: <https://www.iarpa.gov/index.php/research-programs/janus>. Acesso em: 26 mai. 2020.

IARPA. **IARPA Janus program**. 2018. (4m27s). Disponível em: <https://www.youtube.com/watch?v=61Ihi3VsTHQ>. Acesso em: 26 mai. 2020.

IEEE. **IEEE Xplore**. 2020. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso em: 19 ago. 2020.

KALKA, N. D. et al. Ijb–s: Iarpa janus surveillance video benchmark. In: **2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)**. Redondo Beach, CA, USA: [s.n.], 2018. p. 1 – 9.

KAZEMI, V.; SULLIVAN, J. One millisecond face alignment with an ensemble of regression trees. In: **Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition**. Washington, DC, USA: IEEE Computer Society, 2014. (CVPR '14), p. 1867 – 1874. ISBN 978-1-4799-5118-5. Disponível em: <https://doi.org/10.1109/CVPR.2014.241>.

KIM, M. et al. Face tracking and recognition with visual constraints in real-world videos. In: **2008 IEEE Conference on Computer Vision and Pattern Recognition**. Anchorage, AK, USA: [s.n.], 2008. p. 1–8. ISSN 1063-6919.

KING, D. E. Dlib-ml: A machine learning toolkit. **Journal of Machine Learning Research**, v. 10, p. 1755 – 1758, 2009.

KLARE, B. F. et al. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1931–1939.

KRASSER, M. **Deep face recognition with Keras, Dlib and OpenCV**. 2018. Disponível em: <http://krasserm.github.io/2018/02/07/deep-face-recognition/>. Acesso em: 16 jul. 2020.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. **Neural Information Processing Systems**, v. 25, 01 2012.

LIU, L. et al. Toward large-population face identification in unconstrained videos. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 24, n. 11, p. 1874–1884, 2014.

MAO, Y. et al. Cosonet: Compact second-order network for video face recognition. In: **Computer Vision – ACCV 2018**. Cham: Springer International Publishing, 2019. p. 51–67. ISBN 978-3-030-20893-6.

MARSLAND, S. **Machine Learning: An Algorithmic Perspective, Second Edition**. CRC Press, 2014. (Chapman & Hall). ISBN 9781466583337. Disponível em: <https://books.google.com.br/books?id=6GvSBQAAQBAJ>.

NIST. **Face Challenges**. 2015. Disponível em: <https://www.nist.gov/programs-projects/face-challenges>. Acesso em: 26 mai. 2020.

PARCHAMI, M.; BASHBAGHI, S.; GRANGER, E. Video-based face recognition using ensemble of haar-like deep convolutional neural networks. In: **2017 International Joint Conference on Neural Networks (IJCNN)**. Anchorage, AK, USA: [s.n.], 2017. p. 4625–4632. ISSN 2161-4407.

PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. In: **BMVC**. [S.l.: s.n.], 2015.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. **CoRR**, abs/1806.07908, 2018. Disponível em: <http://arxiv.org/abs/1806.07908>.

RAO, Y.; LU, J.; ZHOU, J. Attention-aware deep reinforcement learning for video face recognition. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. Veneza, Itália: [s.n.], 2017. p. 3951 – 3960. ISSN 2380-7504.

RAO, Y.; LU, J.; ZHOU, J. Learning discriminative aggregation network for video-based face recognition and person re-identification. **International Journal of Computer Vision**, v. 127, 11 2018.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65 – 386, 1958.

ROVAI, M. J. **Real-Time Face Recognition: An End-To-End Project**. 2018. Disponível em: <https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348>. Acesso em: 16 jul. 2020.

SANDBERG, D. **Face Recognition using Tensorflow**. 2017. Disponível em: <https://github.com/davidsandberg/facenet>. Acesso em: 31 jul. 2020.

SANTHA, L. N. **Face Recognition with VGG-Face in Keras**. 2019. Disponível em: <https://medium.com/analytics-vidhya/face-recognition-with-vgg-face-in-keras-96e6bc1951d5>. Acesso em: 31 jul. 2020.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, IEEE, Jun 2015. Disponível em: <http://dx.doi.org/10.1109/CVPR.2015.7298682>.

SILVA Ítalo D. G. **eFaceRecon**. 2020. Disponível em: <https://github.com/italodellagarza/eFaceRecon>. Acesso em: 19 ago. 2020.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. **AAAI Conference on Artificial Intelligence**, 02 2016.

SZEGEDY, C. et al. Going deeper with convolutions. In: **Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1409.4842>.

TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. Columbus, OH, USA: [s.n.], 2014. p. 1701 – 1708.

TANIAI, H. **keras-facenet**. 2019. Disponível em: <https://github.com/nyoki-mtl/keras-facenet>. Acesso em: 31 jul. 2020.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. **J. Cognitive Neuroscience**, MIT Press, Cambridge, MA, USA, v. 3, n. 1, p. 71–86, jan. 1991. ISSN 0898-929X. Disponível em: <http://dx.doi.org/10.1162/jocn.1991.3.1.71>.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: . [S.l.: s.n.], 2001. v. 1, p. I – 511. ISBN 0-7695-1272-0.

WANG, M.; DENG, W. Deep face recognition: A survey. **CoRR**, abs/1804.06655, 2018. Disponível em: <http://arxiv.org/abs/1804.06655>.

WILSON, C.; HARGREAVES, T.; HAUXWELL-BALDWIN, R. Benefits and risks of smart home technologies. **Energy Policy**, v. 103, p. 72 – 83, 2017. ISSN 0301-4215. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030142151630711X>.

WOLF, L.; HASSNER, T.; MAOZ, I. Face recognition in unconstrained videos with matched background similarity. In: **CVPR 2011**. Providence, RI, USA: [s.n.], 2011. p. 529–534. ISSN 1063-6919.

WONG, Y. et al. Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition. In: **CVPR 2011 WORKSHOPS**. Colorado Springs, CO, USA: [s.n.], 2011. p. 74 – 81.

YANG, J. et al. Neural aggregation network for video face recognition. In: . [S.l.: s.n.], 2017. p. 5216–5225.

YANG, R.-P. et al. Intelligent mirror system based on facial expression recognition and color emotion adaptation – imirror. In: **2018 37th Chinese Control Conference (CCC)**. Wuhan, China: [s.n.], 2018. p. 3227–3232.

ZHENG, L. et al. Mars: A video benchmark for large-scale person re-identification. In: **Lecture Notes in Computer Science**. Amsterdam, The Netherlands: Springer, Cham, 2016. v. 9910, p. 868 – 884. ISBN 978-3-319-46465-7.