INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
IFCE *CAMPUS* ARACATI
COORDENADORIA DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

AUGUSTO FRANCO SOARES DE MOURA

SISTEMA PARA RECONHECIMENTO FACIAL EM TEMPO REAL: UMA ABORDAGEM BASEADA EM FACENET

ARACATI
2019

AUGUSTO FRANCO SOARES DE MOURA

SISTEMA PARA RECONHECIMENTO FACIAL EM TEMPO REAL: UMA ABORDAGEM BASEADA EM FACENET

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - Campus Aracati, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Silas Santiago Lopes Pereira

Aracati-CE
2019

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

M929s Moura, Augusto Franco Soares de.
Sistema para reconhecimento facial em tempo real : Uma abordagem baseada em FaceNet / Augusto Franco Soares de Moura. - 2019.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) - Instituto Federal do Ceará, Bacharelado em Ciência da Computação, Campus Aracati, 2019.
Orientação: Prof. Me. Silas Santiago Lopes Pereira.

1. Videomonitoramento. 2. Reconhecimento facial. 3. FaceNet. 4. Labeled faces in the wild. 5. Suport vector machines. I. Titulo.

AUGUSTO FRANCO SOARES DE MOURA

SISTEMA PARA RECONHECIMENTO FACIAL EM TEMPO REAL: UMA ABORDAGEM BASEADA EM FACENET

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - Campus Aracati, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Aprovada em 10 de Abril de 2019

BANCA EXAMINADORA

Prof. Me. Silas Santiago Lopes Pereira (Orientador)
IFCE

Prof. Dr. Reinaldo Bezerra Braga
IFCE

Prof. Dr. Mário Wedney de Lima Moreira
IFCE

DEDICATÓRIA

Aos meus pais, Fernando e Elida pelo apoio e pela dedicação nesta fase desafiadora.
Aos meus irmãos, Fernando Filho e André, pela amizade e pela compreensão das minhas ausências.

AGRADECIMENTOS

Aos meus pais, que me proporcionaram a melhor educação que estava ao alcance.

Aos meus mestres, na pessoa do Prof. Dr. Mauro Oliveira, que me ensinaram não só a grade curricular, mas valores para além da universidade.

Ao meu orientador, Prof. Me. Silas Santiago, pela paciência, insistência e amizade durante a produção deste trabalho.

Aos amigos do curso que ajudaram nos momentos difíceis e disciplinas complicadas.

A todos que, de alguma forma, contribuíram para esta conquista.

RESUMO

As reduções dos custos de instalação e armazenamento aumentaram a demanda por sistemas de segurança, entre eles, o videomonitoramento e a autenticação digital. Esses sistemas de videomonitoramento, quando monitorados por humanos, são passíveis de falha e são de difícil escalabilidade. Sistemas de autenticação podem validar alguém utilizando uma senha ou cartão de outro usuário. Para solucionar esta falha, algoritmos de reconhecimento facial podem ser utilizados, tanto para monitorar o trânsito de indivíduos conhecidos ou intrusos, quanto para autenticação biométrica de um indivíduo. Este trabalho avalia a abordagem FaceNet utilizando o *benchmark* do *dataset* Labeled Faces in the Wild (LFW) , bem como avalia técnica de aprendizado de máquina (*Machine Learning* - ML) *support vector machine* (SVM) para a classificação de *embeddings* gerados utilizando o FaceNet. Também modela um sistema de reconhecimento facial em tempo real combinando FaceNet e SVM que alcança 90% de acurácia utilizando uma *webcam* mediana.

Palavras-chaves: Videomonitoramento. Reconhecimento facial. FaceNet. *Labeled Faces in the Wild*. *Support Vector Machine*.

ABSTRACT

Reductions in installation and storage costs have increased the demand for security systems, including video surveillance and digital authentication. These video surveillance systems, when monitored by humans, are subject to errors and are challenging to scale. Authentication systems can validate someone using a password or a card from another user. To address this fault, facial recognition algorithms can be used to monitor the traffic of known individuals or intruders as well as for biometric authentication of an individual. This work evaluates the FaceNet approach using the benchmark of the Labaled Faces in the Wild (LFW) , as well as evaluates machine learning (ML) technique support vector machine (SVM) for the classification of embeddings generated using FaceNet. It also models a real-time facial recognition system combining FaceNet and SVM that reaches 90% accuracy using a medium webcam.

Keywords: Video Surveillance. Facial Recognition. FaceNet. Labeled Faces in the Wild. Support Vector Machine.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	*Area Under a Curve*
CFTV	Circuito Fechado de TV
CNN	*Convolutional Neural Network*
CVPR	*Conference on Computer Vision and Pattern Recognition*
DVR	*Digital Video Recorder*
KCF	*Kernelized Correltion Filters*
LAR	Laboratório de Redes de Computadores e Sistemas
LDA	*Linear Discriminant Analysis*
LFW	*Labeled Faces in the Wild*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MTCNN	*Multi-Task Convolutional Neural Network*
MTL	*Multi-task Learning*
NMS	*Non-Maximum Supression*
PCA	*Principal Component Analysis*
ReLU	*Rectified Linear Unit*
SGD	*Stochastic Gradient Decendent*
SVM	*Support Vector Machine*
VGG	*Visual Geometry Group*
YFD	*YouTube Faces Database*

SUMÁRIO

1 INTRODUÇÃO

Um sistema de videomonitoramento consiste em uma rede de câmeras interligadas para monitorar áreas públicas ou privadas e é utilizado, principalmente, na prevenção e investigação de crimes e proteção de pessoas, grupos e patrimônio. Esse tipo sistema pode ser simples, funcionando com alguém monitorando determinadas áreas através de monitores de vídeo, ou sofisticado, incluindo sistemas computacionais para identificar pessoas nas imagens (KUMAR; SVENSSON, 2015).

Nos últimos anos, os custos de instalação de sistemas de videomonitoramento, bem como os custos de armazenamento vêm caindo. Com isso, a demanda por sistemas de vigilância em locais públicos e privados tem crescido. Um exemplo prático tem sido a utilização de sistemas de portaria virtual, os quais têm o objetivo de reconhecer e verificar indivíduos suspeitos que possam oferecer alguma ameaça, bem como de autenticar indivíduos que devem possuir acesso a determinado local (YA et al., 2017).

A maioria dos sistemas de videomonitoramento são operados por humanos. Uma ou mais pessoas fazem o trabalho de monitoramento, identificação e rastreio de indivíduos por monitores conectados a dispositivos de Circuito Fechado de TV (CFTV) e *Digital Video Recorder* (DVR), o que permite a exibição das imagens em um monitor para que um usuário possa observar e tomar decisões. O monitoramento por humanos envolve problemas de confiabilidade, já que o humano pode falhar com distrações e não pode monitorar todas as câmeras ao mesmo tempo, podendo deixar passar algo importante; também envolve problemas de escalabilidade, porque para um numero razoável de câmeras é preciso envolver muitos humanos no monitoramento.

O reconhecimento facial pode ser aplicado de várias maneiras na área de automação na segurança, como no controle de acesso a salas, prédios, sistemas computacionais através de identificação de pessoa autorizada ou intruso, também na identificação de pessoa para evitar fraude de registro dupli-

cado com a mesma face, ainda na vigilância por vídeo, monitorando pessoas ou atividades suspeitas, bem como na computação pervasiva, identificando quem é a pessoa que está próxima ou usando determinado dispositivo (SHARIF et al., 2017).

A utilização de uma abordagem inteligente utilizando reconhecimento facial tem o intuito de reduzir ou mesmo eliminar a necessidade de intervenção humana na tarefa monitoramento de imagens das câmeras em tempo real, aquisição de dados e tomada de decisão. O reconhecimento facial também é menos suscetível a erros quando comparado a meios convencionais de autenticação, como uso de senhas ou cartões. Senhas e cartões podem ser roubados e utilizados para autenticar um invasor. A utilização de reconhecimento facial na autenticação elimina esse problema (CHOWDHRY et al., 2013).

Sistemas de reconhecimento facial geralmente utilizam detecção facial para isolar faces em uma imagem. Cada face é então pré-processada para a obtenção de sua representação. O desafio dos sistemas de reconhecimento facial é reconhecer um indivíduo em imagens de diferentes ângulos, com diferentes expressões, além de diferenciar faces de indivíduos diferentes (AMOS; LUDWICZUK; SATYANARAYANAN, 2016).

Abordagens baseadas em *deep learning* têm se mostrado capazes de atingir maiores taxas de acerto e maior velocidade no processamento de imagens. *Deep learning* revela estruturas complexas de dados de alta dimensão, como imagens, resolvendo o problema do aprendizado hierárquico com um único algoritmo ou poucos algoritmos. Por isso, *Convolutional Neural Network* (CNN) é o tipo de algoritmo de *deep learning* mais utilizado para reconhecimento facial (COŞKUN et al., 2017).

Há na literatura um grande número de trabalhos mostrando diferentes abordagens para reconhecimento facial, como em (KIM; JUNG; KIM, 2002), (LI et al., 2009), (AGARWAL et al., 2010), (HU et al., 2015) e (SCHROFF; KALENICHENKO; PHILBIN, 2015). A abordagem proposta em (KIM; JUNG; KIM, 2002) tem um bom desempenho utilizando *kernel principal component analysis* (PCA) para a extração de características e *linear support vector machine*

(SVM) para o reconhecimento. Comparado ao uso de PCA, no trabalho (LI et al., 2009), os autores conseguiram melhorar a taxa de reconhecimento facial combinando PCA e *linear discriminant analysis* (LDA) na extração de características e SVM no reconhecimento facial. Na abordagem (AGARWAL et al., 2010) os autores propõe uma solução utilizando PCA *eigenfaces* para extração de características e redes neurais para classificação, alcançando ótimos resultados. O trabalho (HU et al., 2015) compara algumas arquiteturas de CNN. Mostra que técnicas utilizando CNN são robustas e eficientes para a tarefa de reconhecimento facial. Também demonstra que a combinação de arquiteturas de CNN diferentes conseguem melhorar ainda mais os resultados, pois cada arquitetura extraí características diferentes de partes diferentes da imagem. Os autores do FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015) utilizam *Triplet Loss* como função de erro para treinar uma CNN que tem o objetivo de gerar *embeddings* para as faces detectadas. Esses *embeddings* são pontos em uma hiperesfera. A rede neural é treinada para gerar pontos num hiperespaço que, ao serem normalizados em pontos na superfície de uma hiperesfera, respeitem uma distâncias miníma entre *embeddings* de imagens pertencentes a pessoas diferentes. Os autores conseguem demonstrar que, independente da robustez da CNN utilizada, é possível conseguir taxas altas de acerto. Este trabalho avalia o desempenho do FaceNet em um conjunto de *datasets* distintos, avalia o desempenho da técnica de *machine learning* (ML) SVM para a criação de um classificador para reconhecimento facial e descreve e avalia um sistema de reconhecimento facial em tempo real.

1.1 Objetivos

1.1.1 Objetivo Geral

Construir e avaliar um sistema de reconhecimento facial em tempo real utilizando a abordagem FaceNet nas dependências do Laboratório de Redes de Computadores e Sistemas (LAR) no IFCE Aracati.

1.1.2 Objetivos Específicos

Os objetivos específicos são:

1. Avaliar o desempenho do FaceNet (SANDBERG, 2018) com os datasets *Labeled Faces in the Wild* (LFW) e *YouTube Faces Database* (YFD) (Imagens), e comparar com os resultados apresentados em (SCHROFF; KALENICHENKO; PHILBIN, 2015).

2. Estender a avaliação de desempenho apresentada em (SCHROFF; KALENICHENKO; PHILBIN, 2015) com experimentos adicionais com um *dataset* construído a partir de imagens de integrantes do Laboratório de Redes de computadores e Sistemas (LAR).

3. Desenvolver um sistema de reconhecimento facial em tempo real utilizando a abordagem FaceNet.

4. Avaliar o desempenho do classificador SVM para a tarefa de reconhecimento facial.

5. Avaliar o desempenho do sistema de reconhecimento facial no cenário de avaliação.

1.2 Organização do Trabalho

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica descrevendo os principais conceitos relacionados a *deep learning* e reconhecimento facial. O capítulo 3 apresenta os trabalhos relacionados ao estudo aqui apresentado. O capítulo 4 apresenta a metodologia da pesquisa proposta, caracterizando os *datasets*, modelo e pesos utilizados, bem como o cenário de avaliação e os experimentos realizados. O capítulo 5 apresenta os resultados obtidos nos experimentos descritos na metodologia e o capítulo 6 apresenta as considerações finais do trabalho e ideias para trabalhos futuros na mesma linha de pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos relacionados a *Deep Learning*, redes neurais convolucionais, reconhecimento facial, a técnica de detecção e alinhamento facial MTCNN e *Triplet-Loss*.

2.1 Aprendizado de máquina

Aprendizado de máquina (*Machine Learning* - ML) é a capacidade de um sistema melhorar seu desempenho mudando sua estrutura, programa ou dados a partir de suas entradas ou em resposta a informação externa. Normalmente essas mudanças estão associadas à inteligência artificial (*artificial intelligence - AI*).

Existem dois principais paradigmas de aprendizado de máquina: aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado, os rótulos ou valores que representam cada exemplar do conjunto de treino são conhecidos. Se uma hipótese conseguir prever bem os rótulos do conjunto de treino, então provavelmente conseguirá prever bem para um conjunto maior do mesmo tipo. No aprendizado não supervisionado os dados não possuem rótulos. Geralmente se busca particionar o espaço para agrupar dados semelhantes. Esse tipo de aprendizado é útil quando é preciso transformar dados não classificados em dados com sentido (NILSSON, 1996).

2.1.1 SVM

Support Vector Machine (SVM) é uma técnica de ML supervisionado que mapeia vetores de entrada que etão organizados de forma não-linear para um espaço de características de dimensionalidade muito alta. Nesse espaço de características em alta dimensionalidade é construída uma superfície de decisão linear (CORTES; VAPNIK, 1995). Mapeando vetores de entrada em um

espaço de características de alta dimensionalidade, a separação dos dados se torna mais fácil. SVMs pertencem ao conjunto de métodos supervisionados de ML. SVM tem bom desempenho na solução de problemas com poucos dados (poucos indivíduos por classe), dados organizados em padrão não linear e dados de alta dimensionalidade, porém, pode ser utilizada também em outros problemas de ML como *overfitting* de função (Guo; Chen; Li, 2016).

2.2 Deep Learning Neural Networks

Deep Learning, ou aprendizagem profunda, é um tipo de aprendizado de máquina que possui grande poder e flexibilidade representando o mundo como uma hierarquia de conceitos aninhados, com cada conceito definido em relação a conceitos mais simples, e representações mais abstratas computadas em termos das menos abstratas (GOODFELLOW; BENGIO; COURVILLE, 2016).

Uma definição muito utilizada diz que *deep learning* lida com uma rede neural com mais de duas camadas. O problema dessa definição é que a complexidade das redes profundas aumentou muito na última década. As redes neurais profundas atuais possuem mais neurônios que as redes anteriores, maneiras mais complexas de conexão entre camadas e neurônios, explosão na quantidade de poder computacional disponível para treinamento e extração automática de características. Então uma rede neural profunda pode ser definida como uma rede neural com um grande número de parâmetros e camadas. (PATTERSON; GIBSON, 2017)

Métodos de *deep learning* usam uma cascata de várias camadas de unidades de processamento para a extração de características e transformação. Aprendem vários níveis de representação correspondentes a diferentes níveis de abstração. (WANG; DENG, 2018)

Uma importante fonte de dificuldade em muitas aplicações de inteligência artificial no mundo real é a influência que muitos fatores de variação têm em cada pedaço do dado observado. Pode ser muito difícil extrair características

abstratas de alto nível dos dados não tratados. Algumas características só podem ser identificadas utilizando entendimento sofisticado (próximo ao nível humano) do dado (GOODFELLOW; BENGIO; COURVILLE, 2016).

Deep Learning resolve esse problema introduzindo representações que são expressas em termos de outras representações mais simples. Permite-se que o computador construa conceitos complexos a partir de conceitos mais simples.

O exemplo clássico de modelo de *deep learning* é o *Multilayer Perceptron* (MLP). O MLP é apenas uma função matemática mapeando um dado conjunto de valores de entrada em valores de saída. A função é formada pela composição de várias funções mais simples. Cada aplicação de uma função matemática diferente fornece uma nova representação da entrada.

A ideia de aprender a representação correta para o dado fornece uma perspectiva em *deep learning*. Outra perspectiva é que a profundidade permite que o computador aprenda um programa de várias etapas. Cada camada da representação seria um estado da memória do computador depois de executar outro conjunto de instruções em paralelo. Redes com maior profundidade podem executar mais instruções em sequência. Instruções em sequência possuem grande poder porque instruções posteriores podem se referir aos resultados de instruções anteriores. Nem toda informação em ativações de uma camada necessariamente codifica fatores de variação que explicam a entrada. A representação também armazena informação de estado que ajuda a executar um programa que pode dar sentido à entrada.

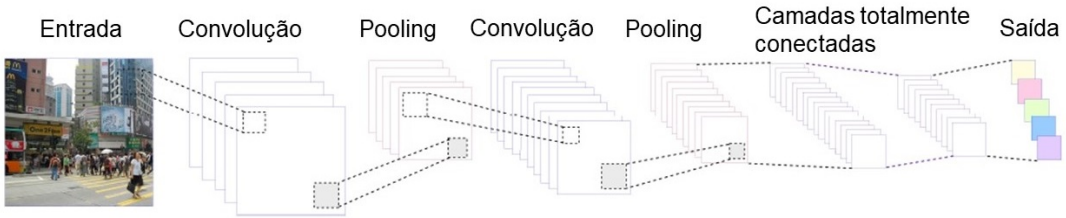
Uma maneira de medir a profundidade de uma rede é contando o número de instruções em sequência que precisam ser executadas para avaliar o modelo. É o maior caminho em um fluxograma que descreve como computar cada saída do modelo dadas suas entradas. Outra maneira considera a profundidade de um modelo como sendo a profundidade do grafo descrevendo como conceitos se relacionam uns com os outros. Nesse caso, a profundidade de um fluxograma dos cálculos necessários para computar a representação de cada conceito pode ser muito mais profunda que o próprio grafo de conceitos.

Como não está claro qual dos dois modos de medir a profundidade é o mais relevante, e porque diferentes pessoas escolhem diferentes conjuntos dos menores elementos para construírem seus grafos, não há um valor único correto para o comprimento da rede neural. Também não existe consenso sobre quão profunda deve ser uma rede para ser considerada "profunda". Todavia, *deep learning* pode ser considerado como o estudo de modelos que envolvem maior quantidade de composições de funções ou conceitos aprendidos que uma *machine learning* tradicional.

2.3 Redes Neurais Convolucionais

Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNN), são projetadas para processar dados formatados como múltiplos vetores (por exemplo, uma imagem colorida composta de três vetores de duas dimensões contendo a intensidade dos pixels nos três canais de cor). As quatro ideias chave por trás de redes neurais convolucionais são: conexões locais, pesos compartilhados, *pooling* e uso de várias camadas. A arquitetura de uma Rede Neural Convolucional está estruturada em uma série de etapas. As primeiras etapas são compostas de camadas convolucionais e camadas de *pooling* (LECUN; BENGIO; HINTON, 2015). A Figura abaixo apresenta um diagrama da arquitetura de uma rede neural convolucional (Figura 1).

Figura 1 – Rede Neural Convolucional para reconhecimento de imagem.



Fonte: adaptada de (HATCHER; YU, 2018)

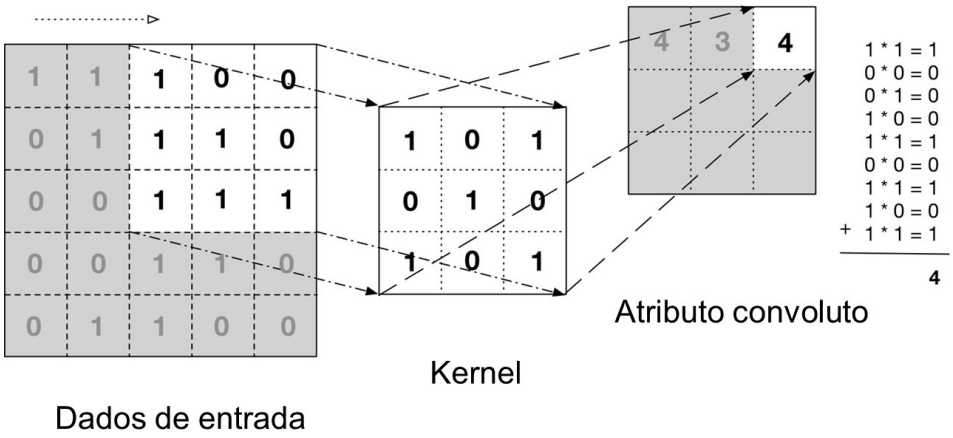
2.3.1 Camadas Convolucionais

O papel da camada convolucional é detectar conjunções locais de características da camada anterior. Nas camadas convolucionais, as unidades são organizadas em mapas de características. Cada unidade está conectada com *patches* locais no mapa de características da camada anterior por um conjunto de pesos chamados de banco de filtros. O resultado dessa soma ponderada local é passada por uma função de ativação não linear, como uma ReLU (*Rectified Linear Unit*). Todas as unidades em um mapa de características compartilham o mesmo banco de filtros. Diferentes mapas de características em cada camada usam bancos de filtros diferentes. A arquitetura é desta forma porque nos dados de um vetor como uma imagem, grupos locais de valores são altamente correlacionados com frequência, e porque estatísticas locais de imagens e outros sinais não variam de acordo com o local. Se um tema pode aparecer em uma parte da imagem, ele pode aparecer em qualquer lugar, por isso, se usa unidades em diferentes locais que compartilham o mesmo conjunto de pesos e detectam o mesmo padrão em diferentes partes do vetor (LECUN; BENGIO; HINTON, 2015). A Figura 2 mostra um diagrama do funcionamento da operação de convolução.

2.3.2 Camadas de Pooling

O papel da camada de *pooling* é combinar características semanticamente semelhantes em uma uma única característica como intuito de diminuir a dimensionalidade do mapa de características. Como as posições relativas das características que formam um tema podem variar um pouco, a detecção confiável do tema pode ser feita grosseiramente na posição de cada característica. Uma unidade típica de *pooling* computa o máximo de um *patch* local de unidades em um ou alguns mapas de características. Unidades de *pooling* vizinhas obtêm a entrada de *patches* que são deslocados por mais de uma linha ou coluna, reduzindo assim a dimensão da representação e criando uma invariância para pequenos deslocamentos e distorções. São empilhadas duas ou três camadas de convolução, não-linearidade e *pooling*, seguidas por mais

Figura 2 – Operação de convolução.



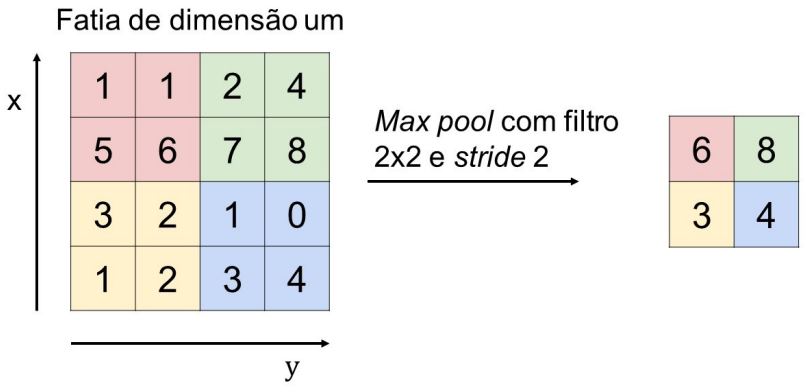
Fonte: adaptada de (PATTERSON; GIBSON, 2017).

convolucionais e camadas totalmente conectadas. Utilizar gradientes de retropropagação em uma rede convolucional é tão simples quanto em uma rede profunda regular, permitindo que todos os pesos em todos os bancos de filtros sejam treinados. (LECUN; BENGIO; HINTON, 2015). A Figura 3 demonstra a operação de *pooling* reduzindo um vetor de dimensão 4x4 para dimensão 2x2 utilizando *max-pooling* com filtro 2x2 e *stride* 2 (saltando de 2 em 2 linhas e/ou colunas).

2.4 Reconhecimento Facial

Reconhecimento facial é dividido em duas categorias: Verificação facial e identificação facial. Em ambos os casos, existe uma galeria de imagens conhecidas e identificadas e uma nova imagem é submetida ao sistema. (WANG; DENG, 2018)

Figura 3 – Demonstração de *max-pooling* com filtro 2x2 e *stride* 2.



Fonte: adaptada de (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

2.4.1 Verificação Facial

Na verificação facial, a nova imagem é comparada com a imagem da galeria (um para um) para verificar se as imagens pertencem à mesma classe. Um exemplo é a autenticação por foto, onde se verifica se a nova foto pertence ao usuário que está tentando autenticar. (WANG; DENG, 2018)

2.4.2 Identificação Facial

Na identificação facial, a nova imagem é comparada com todas as imagens da galeria (um para muitos) para identificar a que classe essa nova imagem pertence. Um exemplo é a identificação de pessoas em sistemas de videomonitoramento, onde se tenta identificar quem é a pessoa passando pela câmera em determinado momento.

Quando as novas imagens sempre pertencem a alguma classe identificada na galeria, temos identificação em conjunto fechado. Quando as novas imagens podem ser de classes que não existem na galeria, temos identificação em conjunto aberto. (WANG; DENG, 2018)

2.5 Detecção e Alinhamento com MTCNN

Multi-task learning (MTL) é um paradigma de aprendizado de máquina que busca melhorar o desempenho de múltiplas tarefas de aprendizado a partir do aproveitamento de informação útil entre essas tarefas. Técnicas de aprendizado de máquina geralmente demandam grande quantidade de dados para ter boa performance, mas nem sempre esses dados estão disponíveis. Em casos onde várias tarefas se relacionam, a utilização de MTL acaba com o problema do volume de dados, visto que é possível melhorar o desempenho do aprendizado compartilhando informação útil entre asa tarefas (ZHANG; YANG, 2017). Como visto na seção 2.3, CNNs têm boa performance quando se trata de aprendizado envolvendo imagens, porém, treinar uma CNN para detectar uma face (ou algum objeto qualquer) pode demandar um *dataset* muito volumoso. Além disso, a rede treinada para ter um bom desempenho pode ser complexa e demandar muito tempo computacional, tornando-a inviável para aplicações em tempo real. Uma tarefa relacionada com o reconhecimento facial é o alinhamento da face, que também pode ser aprendido por uma CNN. A tarefa de detectar uma face pode ser dividida em tarefas menos complexas correlacionadas. Como essas tarefas estão relacionadas, elas podem ser utilizadas em conjunto num sistema de MTL, criando assim uma rede neural convolucional multitarefa (*Multi-Task Cascade Convolutional Network* - MTCNN).

Em (ZHANG et al., 2016), os autores propõem um *framework* baseado em uma cascata de CNNs em três estágios para detectar e alinhar faces, tendo o cuidado de projetar uma arquitetura de CNN leve e rápida para uso em tempo real.

Inicialmente, a imagem é redimensionada em várias escalas para construir uma pirâmide de imagens que será passada como entrada para a cascata de CNNs. Esta piramide de imagens em várias escalas garante que o *kernel* de tamanho fixo possa encontrar faces de diversos tamanhos. Em relação a uma imagem não redimensionada, o *kernel* é pequeno, tornando-o capaz de encontrar faces pequenas numa imagem de maior escala. Da mesma forma, em relação à imagem redimensionada de menor tamanho, o *kernel* é grande,

tornando-o capaz de encontrar faces que ocupem um espaço maior na imagem.

No primeiro estágio é utilizada uma rede totalmente convolucional (*fully convolutional network*) chamada de rede de proposta (*Proposal Network* - P-Net) para coletar as janelas com candidatos a face. A P-Net é treinada para detectar se em um determinado *kernel* há ou não há uma face. As janelas com candidatos são calibrados utilizando vetores de regressão de caixa delimitadora estimadas. Depois é aplicado a técnica *Non-Maximum Supression* (NMS) para juntar candidatos que se sobrepõem. A saída da P-Net é o conjunto das coordenadas das caixas delimitadoras.

No segundo estágio todos os candidatos restantes do primeiro estágio são utilizados para alimentar outra CNN chamada de rede de refinamento (*refine network* - R-Net). A R-Net é treinada para refinar os resultados da P-Net. A R-Net rejeita um grande número de falsos candidatos, realiza a calibragem com a regressão de caixa delimitadora e junta os candidatos sobrepostos com NMS. A saída da R-Net é semelhante à saída da P-Net.

Por fim, no terceiro estágio, os resultados obtidos na R-Net são utilizados como entrada em uma CNN chamada de rede de saída (*output network* - O-Net). A O-Net é treinada para encontrar cinco pontos de referência nas imagens resultantes da R-Net. Os pontos de referência são os dois olhos, as duas extremidades da boca e a ponta do nariz. A O-Net elimina os candidatos que não possuam os pontos de referência e sua saída são os pontos de referência de cada candidato a face da saída da R-Net. Os pontos de referência são usados como referência para alinhar a imagem utilizando rotação, translação e dimensionamento.

Como a tarefa de detecção facial é uma tarefa binária (é face/não é face), esta abordagem utiliza um número reduzido de filtros por camada de convolução e utilizou filtros de menores dimensões para reduzir o custo computacional da detecção facial. Por causa destas medidas é possível conseguir melhores resultados em menor tempo de execução comparado com outras abordagens do estado da arte, tornando esta abordagem rápida o suficiente

para o uso em aplicações de tempo real.

2.6 Triplet-Loss

Uma maneira de fazer uma rede neural aprender os parâmetros necessários para que ela exerça bem seu objetivo calcular o erro da saída através de uma função de erro e utilizar o gradiente descendente para fazer o *back propagation*. A função de erro utilizada no *FaceNet*, função *triplet loss*, mostrou boa performance de aprendizado para reconhecimento facial. No *FaceNet*, uma imagem I é representada por um *embedding* $f(I) \in \mathbb{R}^d$. Isto significa que uma imagem I é representada como a coordenada de um ponto X em um espaço euclidiano de d-dimensões. Esta representação da imagem é transformada em um vetor x que representa uma coordenada na superfície de uma hiperesfera d-dimensional utilizando a normalização L_2. Por exemplo, para uma hiperesfera de raio $R = 1$, X tem que ser transformado em x tal que a normal L_2 de x seja igual a 1.

A Figura 4 mostra o exemplo de uma representação de imagem (*embedding*) em um vetor de *floats* representando um ponto no hiperespaço de 128 dimensões.

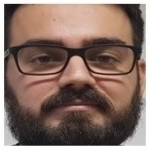
A normalização L_2 calcula a distância euclidiana da origem até um determinado ponto, ou seja, a magnitude ou tamanho do vetor de coordenadas que representa um determinado ponto partindo da origem. É calculada como a raiz quadrada da soma dos quadrados das coordenadas do ponto. No *FaceNet*, tendo X como resultado de $f(I)$, temos a normal L_2 de X na forma da Equação 2.1, onde X representa o vetor resultante de $f(I)$ e d é o número de dimensões do espaço $\mathbb{R}^d$ e X_k é o valor da coordenada k de X:

$$\| X \|_2 = \sqrt{\sum_{k=1}^{n} |X_k|^2} \tag{2.1}$$

Para transformar o vetor resultante de $f(x)$ em um ponto na superfície da hiperesfera, considerando que a hiperesfera possua raio $R = 1$, é preciso

Figura 4 – Exemplo de representação da imagem (*embedding*)

Imagem de entrada



Parte do *embedding* que representa a imagem

```
 1.41722942e-02, -6.42060814e-03,  4.05887179e-02,  4.17513736e-02,
-2.65589766e-02,  5.91065623e-02, -6.09441027e-02,  2.91535407e-02,
 5.02477922e-02, -2.00160090e-02, -7.76700452e-02,  3.20286565e-02,
-3.37881483e-02, -7.94792250e-02, -1.25036966e-02,  1.09269448e-01,
-1.00718863e-01, -8.19067806e-02, -5.35521172e-02,  4.24893089e-02,
 6.34125248e-02,  8.30427259e-02, -4.04413939e-02,  7.96954334e-02,
 1.79372367e-03,  7.85182938e-02,  1.40109088e-03, -4.42891149e-03,
-3.30535732e-02, -2.33513284e-02, -1.96714289e-02, -5.49518410e-03,
-1.60175283e-02, -1.56520947e-03, -2.02690214e-02,  3.69240008e-02,
-1.67795327e-02, -6.61924183e-02, -4.00307700e-02, -5.03334105e-02,
-3.38889170e-03, -7.06048235e-02,  6.83814660e-02, -7.76286647e-02,
-3.09104510e-02,  3.22362594e-02, -7.57363141e-02, -1.68921147e-02,
-3.85526381e-02,  1.46701345e-02, -6.69745356e-03,  3.88275762e-03,
-4.89593968e-02,  5.04415929e-02,  2.52170786e-02, -2.25067288e-02,
-3.93949561e-02,  1.68011747e-02,  3.87323238e-02,  2.80436482e-02,
 3.18065658e-02, -3.68200913e-02,  3.18920128e-02,  3.01541667e-02,
-5.17764548e-03, -2.02433057e-02, -1.23658441e-01,  8.36541352e-04,
 8.01060162e-03, -2.78691147e-02,  2.39250629e-04,  2.57451693e-03,
 4.12108935e-02,  2.94201616e-02,  3.85373309e-02,  5.97260445e-02,
```

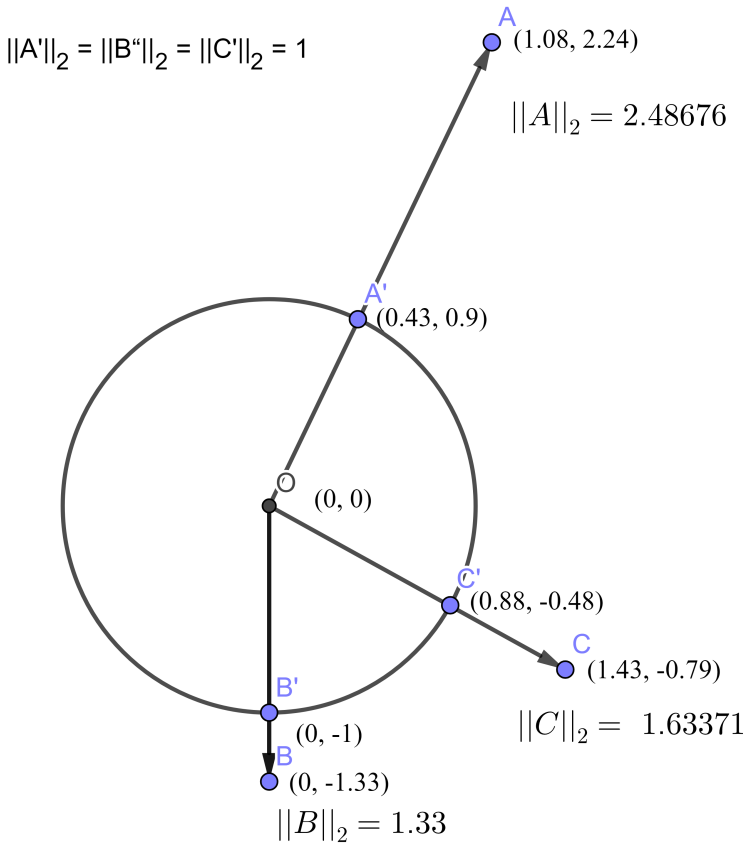
Fonte: elaborada pelo autor

dividir os elementos do vetor resultante de $f(x)$ por sua normal L_2.

A Figura 5 mostra os pontos iniciais A, B e C, de normais L_2 iguais a $2,48676$, $1,33$ e $1,63371$ respectivamente, e os respectivos pontos transformados A', B' e C' de normais iguais a 1, obtidos através da divisão das coordenadas dos pontos iniciais pelas suas respectivas normais L_2.

O objetivo do *triplet loss* é assegurar que o *embedding* da imagem de uma determinada pessoa esteja mais próxima de todos os outros *embeddings* das imagens da mesma pessoa que dos *embeddings* das imagens de outras pessoas (Figura 6. O termo *triplet* indica o uso dos *embeddings* de três imagens. O *triplet* é formado pelos *embeddings* de uma imagem âncora, uma imagem positiva e uma imagem negativa. Tomando o *embedding* da imagem de uma determinada pessoa como x_i^a (imagem âncora), as imagens da mesma pessoa como x_i^p (imagens positivas), as imagens de pessoas diferentes como x_i^n (imagens negativas) e $\mathcal{T}$ como o conjunto de todos os possíveis *triplets* (x_i^a, x_i^p, x_i^n) no conjunto de treinamento, temos que o objetivo to *triplet*

Figura 5 – Pontos A, B e C transformados em A', B' e C de normal L_2 1 no espaço $\mathbb{R}^2$.



Fonte: elaborada pelo autor.

loss é:

$$\underbrace{\parallel x_i^a - x_i^p \parallel}_{d(a,p)} < \underbrace{\parallel x_i^a - x_i^n \parallel}_{d(a,n)} \quad \forall (x_i^a, x_i^p, x_i^n) \in \mathcal{T} \tag{2.2}$$

Sabendo que $\parallel x_i^a - x_i^p \parallel^2$ corresponde à distância $d(a,p)$ entre x_i^a (âncora) e x_i^p (positiva), e que $\parallel x_i^a - x_i^n \parallel$ é a distância $d(a,n)$ entre x_i^a (âncora) e

Figura 6 – Aprendizado com *Triplet Loss*

Fonte: elaborada pelo autor.

x_i^n (negativa), é possível verificar que, se houver uma diferença muito pequena entre $d(a,p)$ e $d(a,n)$, contanto que $d(a,n)$ seja maior, a Equação 2.2 é satisfeita. Porém, esta condição não é interessante para o treino da CNN, já que o aprendizado pode convergir de tal forma que imagens de pessoas diferentes acabem recebendo *embeddings* muito próximos. Para evitar que isso ocorra é acrescentado um fator α à distância $d(a,p)$. O fator α funciona como um limite mínimo, ou margem, para a distância $d(a,n)$, fazendo com que imagens de pessoas diferentes possuam uma separação mínima. Também são elevadas ao quadrado as duas distâncias para que tenham maior impacto na equação. A representação do que se deseja do *triplet loss* fica da seguinte forma:

$$\| x_i^a - x_i^p \|^2 + \alpha < \| x_i^a - x_i^n \|^2 \; \forall (x_i^a, x_i^p, x_i^n) \in \boldsymbol{T} \tag{2.3}$$

Uma vez que se passe um *triplet* pela CNN, o erro (ou *loss*) *l* é calculado baseado na distância $d(a,p)$, que se quer minimizar, na distância $d(a,n)$, que se quer maximizar (por isso terá sinal oposto) e no fator limitador α. Assim temos que o erro de um *triplet* é:

$$l = \| x_i^a - x_i^p \|^2 - \| x_i^a - x_i^n \|^2 + \alpha \tag{2.4}$$

O conjunto $\boldsymbol{T}$ de *triplets* existentes tem cardinalidade N. O erro total que se deseja minimizar para o aprendizado da rede neural é um somatório dos erros dos *triplets*. Como o erro deve ser minimizado, erros negativos são

ignorados, somando apeas os erros positivos. Sendo assim, o erro ou *loss* L que precisa ser calculado e minimizado é:

$$L = \sum_{i}^{N} [\| x_i^a - x_i^p \|^2 - \| x_i^a - x_i^n \|^2 + \alpha]_+ \tag{2.5}$$

Se todos os *triplets* possíveis forem considerados, muitos deles já satisfazem as condições da Inequação 2.3. Esses *triplets*, chamados de *triplets* fáceis, não contribuem para o aprendizado da rede neural e tornam a convergência lenta. Portanto se faz necessário o uso de uma estratégia para selecionar *triplets* difíceis, que vão contribuir para a rápida convergência e para que a rede neural tenha um bom aprendizado. *Triplets* difíceis podem ser encontrados procurando pelos *triplets* que possuam distância $d(a, p)$ maior possível (já que espera a menor distância possível) e que possuam distância $d(a, n)$ menor possível (já que se espera a maior distância possível). Procurar *triplets* difíceis em todo o *dataset* é computacionalmente custoso. Para minimizar o custo computacional, o FaceNet utiliza duas estratégias. A primeira estratégia é dividir o *dataset* em blocos com tamanho na ordem de alguns milhares de exemplares. Assim, ao invés de se calcular *triplets* difíceis em todo o *dataset* (tarefa custosa), calcula-se os *triplets* difíceis por bloco (tarefa menos custosa). A segunda estratégia é focar apenas nos *triplets* com distância $d(a, n)$ difícil. Como a tarefa de encontrar *triplets* nos blocos é mais simples, todos os pares de imagem âncora e imagem positiva são utilizados, enquanto apenas as imagens negativas com maior distância em relação à imagem âncora são selecionadas. Essa abordagem é mais estável e converge mais rápido no início do treino. Ainda, selecionar as imagens negativas mais difíceis pode levar a um mínimo local. Isto pode ser evitado selecionando *triplets* com menor distância $d(a, n)$ mas que possuam uma distância $d(a, p)$ ainda menor satisfazendo a Equação 2.6. Este tipo de *triplet* é conhecido como *triplet* semi-difícil.

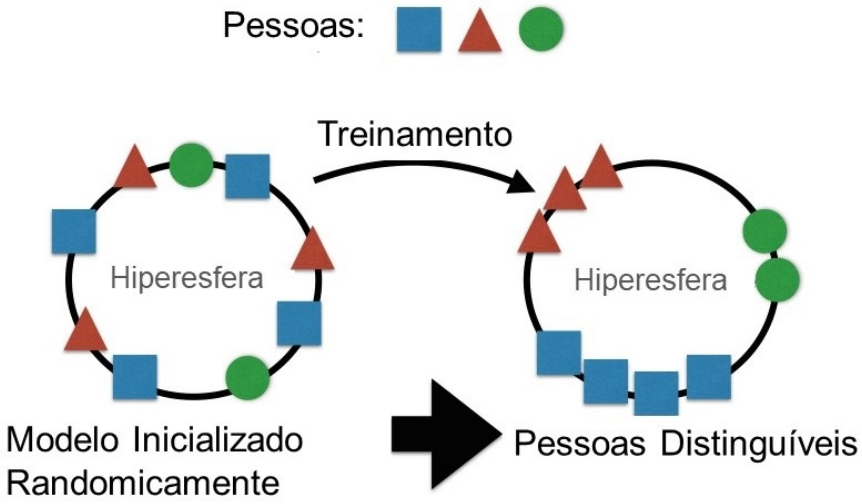
$$\| x_i^a - x_i^p \| < \| x_i^a - x_i^n \| \tag{2.6}$$

Para um aprendizado eficiente da rede neural utilizando *triplet loss*, é

crucial escolher os melhores *triplets* para que se consiga uma rápida convergência utilizando blocos pequenos de imagens (SCHROFF; KALENICHENKO; PHILBIN, 2015).

A Figura 7 ilustra o que acontece quando se utiliza *triplet loss* para o aprendizado da rede neural.

Figura 7 – Ilustração do procedimento *Triplet Loss*



Fonte: adaptada de (AMOS; LUDWICZUK; SATYANARAYANAN, 2016)

3 TRABALHOS RELACIONADOS

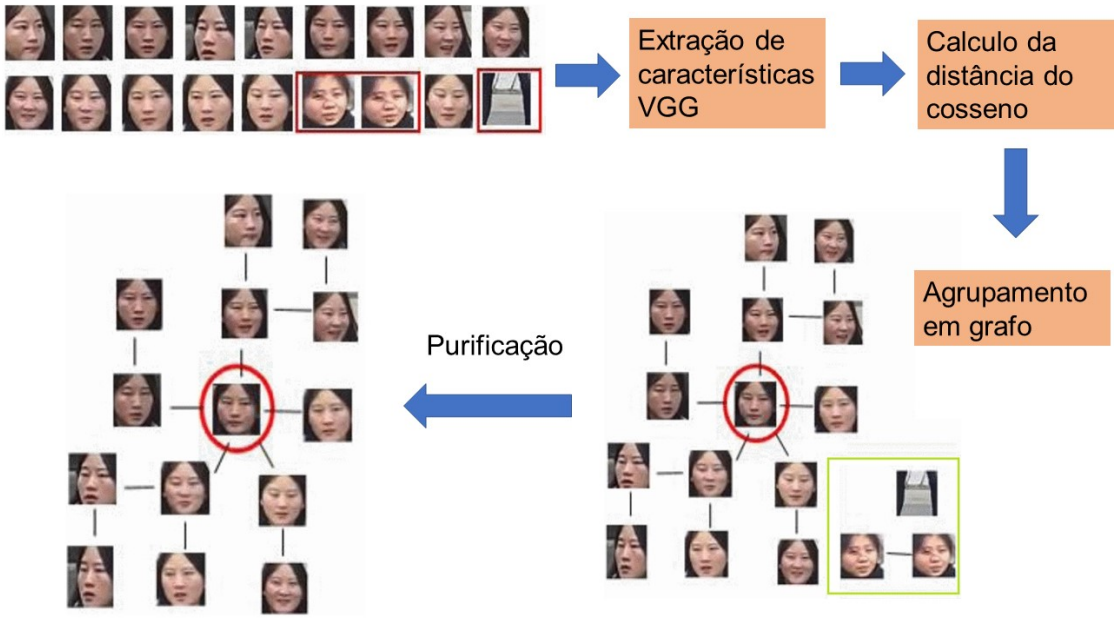
Este capítulo apresenta os principais trabalhos relacionados à pesquisa em reconhecimento facial.

3.1 Videomonitoramento com *Deep Learning*

(YA et al., 2017) propõem um método para reconhecimento facial em vídeos de vigilância no mundo real usando *deep learning*. Este método consiste em duas partes. Na primeira parte, um *dataset* é construído coletando e etiquetando automaticamente as imagens a partir de vídeos de vigilância do mundo real. Essa coleta acontece em quatro fases. Na primeira fase, utilizando detecção facial *haar* e o método de rastreamento *Kernelized Correlation Filters* (KCF), que encontra uma face no vídeo, cria uma classe de identidade e rastreia quadro a quadro coletando amostras dessa face para a classe criada. Na segunda fase é feita uma purificação dentro de cada classe. Nesta fase o modelo facial *Visual Geometry Group* (VGG) extrai características das imagens e depois é calculada a distância do cosseno entre cada imagem. Cada par de imagem que possua uma distância de cosseno abaixo de um determinado limite é ligada por uma aresta. A imagem com mais arestas é escolhida como imagem central e, as imagens que tiverem ligação direta ou indireta com a imagem central pertencem à classe. As outras são eliminadas. Na terceira fase é feita uma purificação entre as classes. É medida a similaridade entre as imagens centrais para descobrir e eliminar classes duplicadas. Na quarta fase é feita uma filtragem baseada em quantidade para eliminar classes que possuam poucas imagens (menos de 100). Na segunda parte é feita um ajuste fino dos pesos do modelo pré-treinado de reconhecimento VGG através de retropropagação (*back-propagation*) utilizando o *dataset* construído e um modelo pré-treinado. Este modelo compreende oito camadas convolucionais e três camadas totalmente conectadas. Cada uma é seguida por uma ou mais não linearidades como *ReLU* ou *max-pooling*. Apenas as camadas totalmente co-

nectadas recebem ajuste fino com o novo *dataset* para fazer com que o modelo fique mais adequado ao domínio do objetivo. Este modo de ajuste fino é amplamente aceito porque as características extraídas pelas primeiras camadas de uma rede neural profunda são genéricas e devem ser úteis para a maioria das tarefas, enquanto as últimas camadas tendem a extrair detalhes mais específicos do domínio do objetivo. A efetividade do experimento deles é verificada de duas maneiras. Primeiro, provando que um *dataset* preciso à partir de vídeos de vigilância do mundo real pode ser construído pelo método proposto com pouca participação de humanos. Depois que, com o novo *dataset*, pode ser feito o ajuste fino do modelo facial VGG e a performance do mesmo no domínio do objetivo é melhorada. O modelo VGG com ajuste fino ao *dataset* coletado automaticamente conseguiu uma taxa de reconhecimento de 92,1%, que é superior à taxa de reconhecimento do modelo VGG sem o ajuste fino, que foi de 83,6%. A Figura 8 descreve o processo de criação automática de uma classe no *dataset*.

Figura 8 – Criação automática de *dataset*.



Fonte: adaptada de (YA et al., 2017).

3.2 OpenFace

Em (AMOS; LUDWICZUK; SATYANARAYANAN, 2016), Os autores apresentam e descrevem a biblioteca *OpenFace* para reconhecimento facial. O *OpenFace* tem por objetivo ser um sistema de alta acurácia com pouco tempo de treino e predição. A estrutura do projeto independe da arquitetura da rede neural. A arquitetura do FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015) foi utilizada. O detector facial *Dlib* é utilizado por ter maior acurácia que o detector do *OpenCV*. Na fase de detecção facial, as faces retornadas podem estar em várias poses e condições de iluminação. Para utilizar um *dataset* reduzido para o treino, a imagem é normalizada de tal forma que os olhos, nariz e boca apareçam sempre em locais semelhantes utilizando transformações afins 2D. Utilizando o detector de pontos de referência do *Dlib*, 68 pontos de referência são detectados e as transformações afins levam os olhos e o nariz para próximo da localização média. A transformação também redimensiona e corta a imagem no limite dos pontos de referência para que a imagem de entrada da rede neural tenha uma dimensão de 96x96 pixels. O treinamento é feito com 500 mil imagens, resultado da combinação dos dois maiores *datasets* de reconhecimento facial rotulados para pesquisa: o CASIA-WebFace e o FaceScrub. Essa quantidade é pequena se comparada com as 200 milhões de imagens do treino do FaceNet e com as 4.4 milhões do treino do DeepFace. A rede neural mapeia a imagem em um *embedding* de baixa dimensionalidade. Utiliza-se uma versão modificada da rede FaceNet denominada *nn4*, a fim de diminuir o numero de parâmetros para o *dataset* reduzido. Utiliza-se também o *triplet loss* do FaceNet a fim de que a rede neural represente faces da mesma pessoa dentro de um limite de distância euclidiana no espaço n-dimensional, e imagens diferentes acima desse limite de distância. Então a rede provê um *embedding* na hiperesfera unitária e a distância euclidiana representa similaridade. Os autores avaliaram o *OpenFace* utilizando o *dataset* LFW (*Labeled Faces in the Wild*) de duas maneiras: Verificação e Classificação. Na verificação, dado um par de imagens, deve-se verificar se as duas imagens pertencem à mesma pessoa. Os dados são separados em dez conjuntos de mesmo tamanho. Nove conjuntos são usados para treino e o último conjunto é usado para

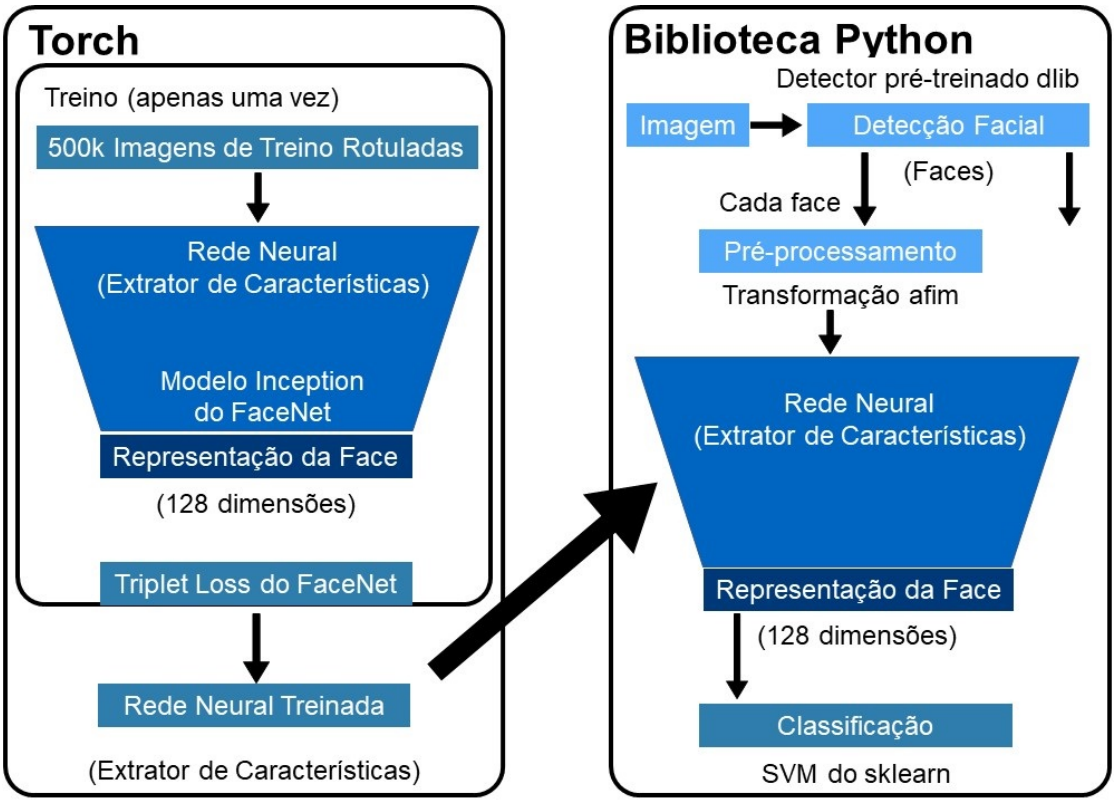
o teste. Os resultados mostraram que a acurácia do *OpenFace* está próxima à acurácia das técnicas do estado da arte. No teste de classificação, é preciso dizer quem é a pessoa na foto. São dadas imagens de várias pessoas, sendo que são no máximo 20 imagens por pessoa. Os dados são então separados randomicamente, colocando 90% das imagens no conjunto de treino e 10% no conjunto de teste. É muito importante para dispositivos móveis que o reconhecimento facial ocorra de maneira rápida, principalmente no caso de óculos inteligentes. Como o tempo de detecção facial entre as técnicas é sempre o mesmo, este foi desconsiderado do cálculo do tempo. Foram considerados os tempos de processamento e predição. Os experimentos mostraram que outras técnicas (*Eigenfaces*, *Fisherfaces* e *Local Binary Paterns Histograms*) têm uma queda maior na acurácia quando comparadas ao *OpenFace*, em relação à quantidade de pessoas a serem classificadas. O tempo de treino do *OpenFace* também é menor. O tempo de predição das outras técnicas aumenta quando se aumenta o número de pessoas, mas o tempo de predição do *OpenFace* se mantém constante. Por fim, os autores verificaram que o *OpenFace* demonstrou possuir acurácia competitiva com o estado da arte, mesmo utilizando um *dataset* com duas ordens de magnitude menor. A Figura 9 descreve a estrutura do projeto OpenFace.

3.3 FaceNet

Em (SCHROFF; KALENICHENKO; PHILBIN, 2015), os autores propõem um modelo que consiste em uma camada de entrada em lotes (*batches*) e uma CNN profunda, seguidas por uma camada de normalização L_2, que resulta em um *embedding* da face, seguida do *triplet loss* durante o treinamento. A Figura 10 mostra a estrutura desse modelo.

Para a camada da CNN, foram exploradas duas arquiteturas: o modelo *Zeiler&Fergus* e o *Inception*. A parte mais importante da abordagem é o aprendizado de ponta a ponta de todo o sistema, independente do modelo escolhido. Para este fim, os autores usam a camada de *triplet loss* que impacta diretamente na verificação, reconhecimento e agrupamento facial. *Triplet loss* é

Figura 9 – Estrutura do projeto OpenFace

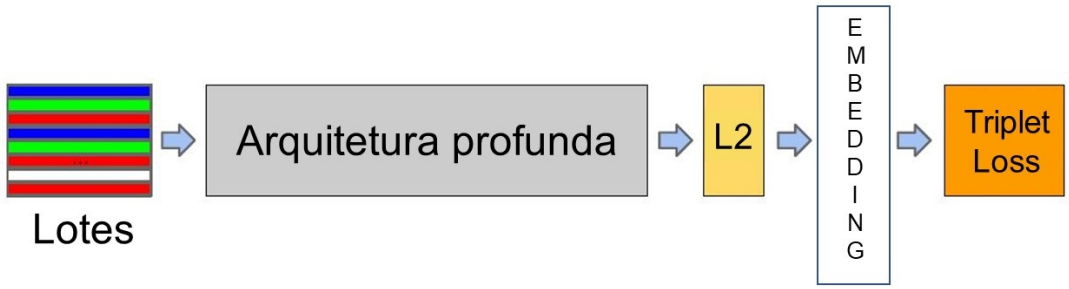


Fonte: adaptada de (AMOS; LUDWICZUK; SATYANARAYANAN, 2016).

inspirada na técnica do vizinho mais próximo. As imagens são acomodadas em um espaço euclidiano de d dimensões. São gerados todos os subconjuntos de três imagens (*triplets*), onde uma imagem é a âncora e as outras duas são uma positiva, que pertence à classe da âncora, e outra negativa, que não pertence à classe da âncora, respectivamente. *Triplet loss* busca aproximar a imagem positiva e distanciar a imagem negativa da imagem âncora, dada a regra em que a distância entre âncora e positiva somada a um determinado limite seja menor que a distância entre âncora e negativa. A seleção de *triplets* que não obedeçam essa regra levam a uma convergência rápida. A Figura 11 ilustra o aprendizado através de *triplet loss*.

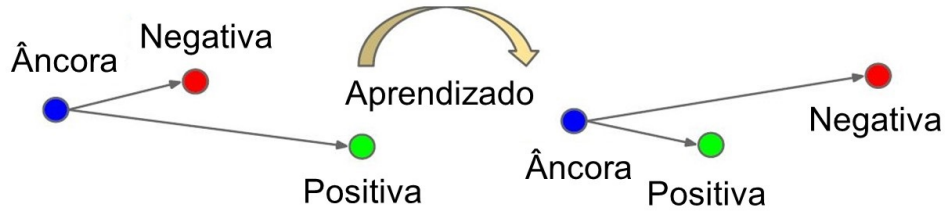
Para a geração e seleção de *triplets*, os autores usaram mini-lotes que podem melhorar a convergência durante a descida do gradiente estocástico

Figura 10 – Estrutura do modelo FaceNet.



Fonte: adaptada de (SCHROFF; KALENICHENKO; PHILBIN, 2015).

Figura 11 – Aprendizado através de *Triplet loss*.



Fonte: adaptada de (SCHROFF; KALENICHENKO; PHILBIN, 2015).

(*Stochastic Gradient Descendent* - SGD). Por outro lado, detalhes da implementação podem deixar grandes lotes mais eficientes. Os autores treinaram a CNN usando SGD via algoritmo *backpropagation* padrão e método de otimização *AdaGrad*. Observou-se que o aumento da acurácia desacelera após algumas horas de treino, porém continuar o treino pôde ainda melhorar a performance significativamente. Para a avaliação da abordagem proposta, os autores usaram entre 100 e 200 milhões de faces de 8 milhões de identidades diferentes. Um detector facial gera uma borda ao redor da face, salva essa imagem, que depois é redimensionada para o tamanho da entrada. Uma das discussões feitas é a relação entre os requisitos computacionais do modelo e a acurácia alcançada (modelos mais pesados acertam mais). Também foi observado se há relação entre a acurácia e o número de parâmetros do modelo, porém não foi encontrada correlação entre estas variáveis. O trabalho também compara o desempenho de quatro modelos selecionados, tendo por um lado a arquitetura

tradicional *Zeiler&Fergus*, e por outro lado os modelos reduzidos baseados no Inception. Enquanto o modelo mais complexo possui maior acurácia, o mais simples é leve o suficiente para rodar rápido em um celular e acerta o suficiente para fazer agrupamento de imagens. Observa-se também quão sensível o sistema é em relação à qualidade da imagem, mostrando que, mesmo que a rede tenha sido treinada para imagens de 220x220 pixels, ela consegue lidar bem com miniaturas de 80x80 pixels. A dimensionalidade usada no trabalho foi 128x128, porém foi possível usar dimensionalidade menor tendo pouca perda na acurácia. Em relação à quantidade de dados para treino, usar mais dados melhora a acurácia, porém à medida que o conjunto de dados fica muito grande, o ganho de acurácia diminui cada vez mais. Utilizando o *dataset* LFW, dois modos foram usados para testar o modelo. Um com colheita de centro fixo da miniatura provida pelo LFW, e outra utilizando alinhamento, chegando a $98,87\% \pm 0,15$ no primeiro método e $99,63\% \pm 0,15$ no segundo. Mesmo arquiteturas menores baseadas na arquitetura *Inception* conseguiram desempenho semelhante. Para o *dataset Youtube Faces Database* (YFD), usando similaridade média de todos os pares dos primeiros cem quadros que o detector facial detectou em cada vídeo, a acurácia chegou a $95,12\% \pm 0,39$. Usando os primeiros mil quadros o resultado foi $95,18\%$. O agrupamento facial feito pela abordagem do trabalho se mostrou incrivelmente robusto quanto a variações em oclusão, iluminação, pose e envelhecimento.

4 METODOLOGIA

Este capítulo apresenta a descrição do desenvolvimento e avaliação de um sistema de reconhecimento facial em tempo real, os experimentos propostos para a validação do FaceNet, os *datasets* utilizados para essa validação e os experimentos com a técnica de *Machine Learning* (ML) *Suport Vector Machine* (SVM) para a criação de um classificador.

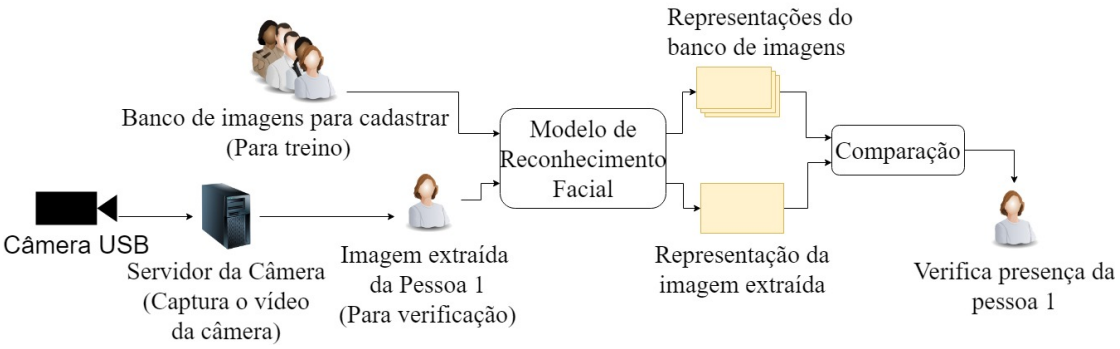
4.1 Cenário de avaliação

Para validar a abordagem proposta, um sistema foi desenvolvido e avaliado nas dependências do Laboratório de Redes (LAR[1]) no IFCE campus Aracati. Foi construído um banco com imagens (*dataset*) dos frequentadores do laboratório para posterior identificação. A coleta de imagens foi autorizada pelos participantes via termo de consentimento. Foram coletadas imagens de 28 pessoas (classes), sendo 11 fotos para a menor classe e 34 fotos para a maior classe, com média de $21,32$ imagens por classe. As imagens foram coletadas utilizando uma câmera de 12 megapixels de um *smartphone Samsung Galaxy S7*, de forma a obter vários ângulos do rosto, com boa iluminação no local e utilizando uma parede branca como fundo. O sistema mostra em tempo real a quantidade de pessoas detectadas em um instante de tempo, bem como a identificação dessas pessoas. Para o *streaming* em tempo real foi utilizada uma webcam Logitech C270 que permite *streaming* de vídeo em HD (720p). A Figura 12 ilustra o cenário descrito.

O sistema utiliza o alinhamento de imagem com MTCNN e a implementação do FaceNet disponível em (SANDBERG, 2018). Algumas modificações foram feitas para ganho de desempenho e adaptação ao fim do protótipo. Para ganho de desempenho, o modelo pré-treinado é carregado apenas uma vez quando o sistema é iniciado, evitando que precise ser carregado cada vez que

[1] http://lar.ifce.edu.br

Figura 12 – Modelagem do Cenário de Avaliação



Fonte: elaborada pelo autor.

se precisa gerar *embeddings*. Foram feitas alterações no método que mostra as identificações em tempo real na imagem da câmera para que também mostre a quantidade de faces detectadas em um dado instante de tempo.

O sistema roda no terminal e possui um menu que separa o sistema nos seguintes módulos:

- Alinhamento das imagens do *dataset* de pessoas cadastradas.
- Geração de *embeddings* das imagens alinhadas.
- Treinamento do classificador SVM.
- Reconhecimento facial em tempo real.

O alinhamento utiliza MTCNN nas imagens originais do *dataset*. Como o processo de alinhamento é custoso e as imagens originais podem ser de boa qualidade e ocupar muito espaço, após alinhar as imagens e guardar o corte com a face com dimensionalidade $160x160$ em uma pasta paralela, as imagens originais são deletadas. Caso novas imagens sejam colocadas na pasta do *dataset* original e o alinhamento seja executado, as novas imagens são alinhadas e salvas na pasta do *dataset* alinhado (não é preciso alinhar todo o *dataset* novamente). A geração de *embeddings* passa as imagens alinhadas pelo FaceNet e armazena os *embeddings* e suas respectivas classes (cada

pessoa cadastrada é uma classe). O treinamento do classificador utiliza os *embeddings* e classes para treinar uma SVM que será utilizada para calcular a probabilidade de uma nova imagem pertencer a cada classe do *dataset*. O reconhecimento em tempo real cria uma janela com o *streaming* da câmera, executa detecção e alinhamento facial com MTCNN para selecionar e alinhar as faces em cada *frame* do *streaming*, gera os *embeddings* das faces alinhadas e utiliza os *embeddings* no classificador para calcular a classe de maior probabilidade para cada face. O nome da classe (pessoa) é mostrado junto a um retângulo desenhado ao redor da face. Também são mostrados o número de faces detectadas no momento e a taxa de quadros por segundo do *streaming*.

4.2 Datasets

Para a seleção de *datasets* para a realização de um conjunto de experimentos, leva-se em consideração o volume de dados, a disponibilidade gratuita e a utilização nos principais testes de *benchmark* da literatura na área de reconhecimento facial. As subseções abaixo detalham os *datasets* utilizados neste trabalho.

4.2.1 Labeled Faces in the Wild (LFW)

O LFW (HUANG et al., 2007) é o *benchmark* padrão na pesquisa em reconhecimento facial, conforme (AMOS; LUDWICZUK; SATYANARAYANAN, 2016). O mesmo possui um total de 13233 imagens de 5749 pessoas diferentes. Este *dataset* está distribuído como mostra a Tabela 1.

4.2.2 YouTube Faces Database (YFD)

O YFD (WOLF; HASSNER; MAOZ, 2011) é um *dataset* composto de vídeos do *YouTube* e baseado no *dataset* LFW. O *dataset* contem trechos de vídeo de pessoas contidas no LFW. O mesmo foi criado para a realização de *benchmarks* de reconhecimento facial em vídeos "*in the wild*", ou seja, sem

Tabela 1 – Distribuição das imagens no *dataset* LFW.

numero de imagens\pessoas	nº de pessoas (% pessoas)	nº de imagens (% imagens)
1	4069 (70,8%)	4096 (30,7%)
2 - 5	1269 (23,8%)	3739 (28,3%)
6 - 10	168 (2,92%)	1251 (9,45%)
11 - 20	86 (1,50%)	1251 (9,45%)
21 - 30	25 (0,43%)	613 (4,63%)
31 - 80	27 (0,47%)	1170 (8,84%)
> 81	5 (0,09%)	1140 (8,61%)

Fonte: adaptada de (HUANG et al., 2007)

restrições, no intuito de avaliar a robustez dos sistemas em relação a variações de poses e mudanças de iluminação. Possui um total de 3425 vídeos de 1595 pessoas. O *dataset* está distribuído como mostra a Tabela 2.

Tabela 2 – Distribuição do *YouTube Faces Database*.

nº de vídeos	nº de pessoas
1	591
2	471
3	307
4	167
5	51
6	8

Fonte: adaptada de (WOLF; HASSNER; MAOZ, 2011)

O YFD também disponibiliza os *frames* dos vídeos para que o *dataset* seja utilizado como *dataset* de imagens. O menor vídeo possui 48 quadros, o maior vídeo possui 6070 quadros e a média de quadros por vídeo é de $181,3$ quadros. Como os outros *datasets* são formados por imagens, para padronizar, foram utilizadas as imagens do YFD, e não os vídeos.

4.2.3 *Dataset criado no Laboratório de Redes de Computadores e Sistemas (LAR)*

Para a realização dos experimentos utilizando o cenário de avaliação, foi necessário a criação de um *dataset* com imagens de pessoas que frequentam um ambiente disponível para para a experimentação. O ambiente selecionado para os testes no cenário de avaliação foi o Laboratório de Redes de Computadores e Sistemas (LAR) localizado no IFCE de Aracati. Para a criação do *dataset* do LAR foram coletadas imagens de bolsistas e professores que frequentam o laboratório, totalizando 28 participantes. A distribuição do dataset pode ser vista na Tabela 3.

Tabela 3 – Distribuição do *dataset* LAR.

nº de imagens	nº de pessoas
11 - 14	2
15 - 18	7
19 - 22	9
23 - 26	5
27 - 30	3
31 - 34	2

Fonte: elaborada pelo autor.

4.3 Modelos e Pesos utilizados

As redes neurais aprendem a extrair características através do ajuste dos pesos. Uma rede bem treinada consegue criar e ter como saída as melhores representações para os dados de entrada. (PRIHASTO et al., 2016) mostra a acurácia de várias abordagens de *deep learning* para reconhecimento facial utilizando o *dataset Labeled Faces in the Wild* (LFW). A abordagem FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015) conseguiu $99,63\%$ de acurácia na Conferência sobre Visão Computacional e Reconhecimento de Padrões (*Conference on Computer Vision and Pattern Recognition* - CVPR) de 2015.

Apesar de ser indiferente quanto à arquitetura utilizada, FaceNet está originalmente implementada utilizando a arquitetura *Inception-ResNet V1* (SZEGEDY; IOFFE; VANHOUCKE, 2016). Em (SANDBERG, 2018) estão disponíveis para *download* os modelos treinados (ou seja, os valores de pesos para a arquitetura de rede utilizada) em dois *datasets*: *CASIA-WebFace*, com mais de 450 mil imagens de mais de 10 mil pessoas; e *VGGFace2*, com mais de 3.3 milhões de imagens de mais de 9 mil pessoas. Os pesos da rede treinada com o VGGFace2, em conjunto com a rede *Inception-ResNet V1*, por conseguirem maior acurácia, foram utilizados para realização dos experimentos descritos nesta proposta.

4.4 Métricas

As métricas utilizadas nos experimentos para gerar resultados são acurácia, área sob a curva ROC (AUC) e tempo de execução. A acurácia é uma taxa de classificação, ou seja, a o número de classificações corretas em relação com o número total de classificações. É a porcentagem de acertos. Esta métrica de *performance* é a mais utilizada (GARCÍA et al., 2009). A métrica AUC representa a probabilidade de que um exemplar negativo (que não pertence à classe) escolhido ao acaso terá menor probabilidade estimada de ser classificado como positivo do que um exemplar positivo (que pertence à classe) escolhido ao acaso (Ling, 2005).

4.5 Experimentos Realizados

Inicialmente, foi realizado um conjunto de experimentos de validação com o FaceNet, a partir da utilização dos *datasets* LFW, *YouTube Faces Database* e o *dataset* criado no LAR, anteriormente descritos, para comparação com os resultados de acurácia obtidos em (SCHROFF; KALENICHENKO; PHILBIN, 2015), visto que os autores avaliam a acurácia do FaceNet para os *datasets* LFW e YFD. (SANDBERG, 2018) possui a implementação se um *script* que realiza testes no padrão estabelecido em (HUANG et al., 2007). O

script utiliza um arquivo de pares (*pairs.txt*), disponível em (LEARNED-MILLER GARY B. HUANG; HUA, 2018), que contém os pares positivos e negativos pré-estabelecidos para padronizar os testes com o *dataset* LFW. O arquivo de pares organiza o *dataset* da seguinte maneira: 10 blocos, cada um contendo 300 pares positivos de imagens (mesma pessoa) e 300 pares negativos (pessoas diferentes), totalizando 6000 pares a serem testados, 3000 positivos e 3000 negativos. O *script* pode comparar a distância euclidiana ou similaridade de cosseno. A distância euclidiana calcula a distância em linha entre os dois *embeddings*, enquanto a similaridade de cosseno calcula a distância angular entre dois *embeddings* tendo como pivô a origem (ponto onde as coordenadas em todos os eixos são iguais a 0). Por padrão, o *script* utiliza distância do cosseno. Os blocos são divididos em 9 blocos de treino e 1 bloco de teste, repetindo o processo 10 vezes, de forma que cada rodada contemple um dos blocos como bloco de teste sem repetir. Em cada rodada é realizado um *cross validation* para encontrar o melhor *threshold* de distância (limiar de distância entre imagens para classificar como iguais ou diferentes) da seguinte forma: Em cada *holdout* são calculados os melhores *thresholds* dos 9 blocos de treino. São testados *thresholds* de 0 a 4 em intervalos de $0,01$ para retornar o melhor *threshold* de cada bloco. O melhor *threshold* de cada bloco de treino é utilizado no bloco de teste para extração das métricas. A média das métricas é retornada e o conjunto de métricas retornadas de todos os *holdouts* do *cross validation* é usado para gerar a acurácia média e a AUC mostrados na subseção 5.1.1.

Os testes preliminares envolveram a utilização do FaceNet utilizando a rede Inception-ResNet V1 e um dos dois conjuntos de pesos (o conjunto treinado com o dataset VGGFace2) disponíveis para esta rede. Este modelo pré-treinados foi utilizado para análise de desempenho. As métricas utilizadas foram tempo de execução, acurácia e área sob a curva (*Area Under a Curve* - AUC). Este experimento consistiu em comparar os pares de imagens gerados anteriormente avaliar o desempenho da tarefa de reconhecimento facial, que envolve a geração de *embeddings*, e o treinamento e avaliação com o modelo preditivo SVM.

Em seguida, foram realizados testes de desempenho com a técnica

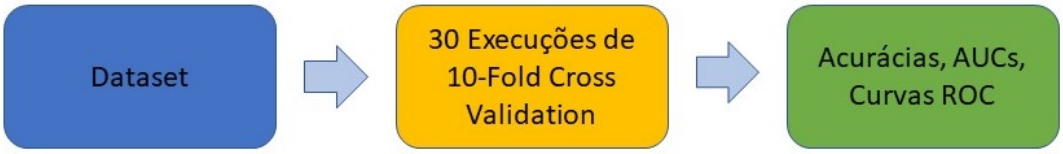
SVM, no intuito de obter um classificador a ser utilizado pelo sistema de reconhecimento facial. Dentre diferentes abordagens supervisionadas de aprendizagem de máquina, a técnica SVM foi escolhida pelas seguintes razões:

- Os autores do FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015) citam (SUN; WANG; TANG, 2014), (TAIGMAN et al., 2014) e (ZHU et al., 2014) que utilizam SVM na tarefa de classificação.
- Em (SANDBERG, 2018) o classificador implementado utiliza SVM.
- No trabalho (KREMIC; SUBASI, 2016) os autores demonstram que SVM com *kernel* linear é mais acertivo que SVM com *kernel* puk e que *random forest* na tarefa de reconhecimento facial.
- Os resultados deste trabalho mostram que SVM é um classificador robusto para os *embeddings* gerados nos *datasets* utilizados.

A avaliação do classificador SVM consistiu em dividir os *datasets* em 10 blocos para realizar uma validação cruzada (*cross validation*). O *cross validation* é repetido 30 vezes de maneira totalmente aleatória para a obtenção de resultados mais confiáveis (JAIN, 1991). A partir dos dados gerados por esse experimento, são gerados a acurácia e a curva ROC da SVM para cada *dataset*, bem como a AUC. A Figura 13 mostra um diagrama com a sequência de passos do experimento.

O último experimento consistiu na avaliação de desempenho do sistema de reconhecimento facial em tempo real. Foram selecionadas aleatoriamente um conjunto de pessoas previamente cadastradas. As pessoas selecionadas passaram na frente de uma *webcam* que serviu como *streaming* para o sistema. A *webcam* utilizada é apresentada com mais detalhes na seção 4.1. Ademais, algumas posições e condições de iluminação foram considerados. Foram registrados os nomes das pessoas que estavam na frente da câmera e os nomes que o sistema atribuiu às imagens capturadas no *streaming*. A avaliação consistiu em comparar o número de atribuições corretas com o número de atribuições incorretas.

Figura 13 – Diagrama do experimento com SVM



Fonte: elaborada pelo autor.

Após a etapa de realização de experimentos, os dados coletados foram utilizados para a criação do sistema descrito na seção 4.1.

4.6 Treinamento e Identificação

O reconhecimento facial em videomonitoramento consiste em duas fases principais: treinamento, que consiste em treinar um classificador a partir do *dataset* com imagens das pessoas que se quer reconhecer; e a identificação, que consiste em identificar quem é a pessoa capturada pela câmera de videomonitoramento.

A fase de treinamento envolve a realização de três etapas: (1) a criação de um banco de imagens identificadas por pessoa; (2) o fornecimento das imagens processadas para um modelo pré-treinado para gerar os *embeddings*, e (3) a geração de um classificador a partir dos *embeddings* criados no passo anterior. Uma vez que uma pessoa possua *embeddings* inserida no treinamento do classificador, ela poderá ser identificada.

A fase de verificação consiste em uma sequência de passos: Inicialmente, deve-se adquirir a imagem da câmera de vigilância. Em seguida, as imagens são extraídas em tempo real diretamente da câmera. A biblioteca *OpenCV*[2] possuí soluções para essa extração. O segundo passo é o pré-processamento das imagens. O mesmo consiste em extrair e alinhar a(s) face(s) do frame, utilizando o MTCNN, para melhor desempenho da rede neu-

[2] https://opencv.org

ral e redimensionamento da imagem, para se adequar à camada de entrada da rede. O terceiro passo envolve a criação de *embeddings* das imagens, passando as fotos pré-processadas como entradas do FaceNet. Após a criação destes *embeddings*, o classificador treinado irá indicar a que classe (pessoa) estas imagens pertencem.

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados dos experimentos realizados no capítulo 4. São apresentados os resultados da avaliação de desempenho do FaceNet, em termos de acurácia, área sob a curva e tempo de execução na geração dos *embeddings*. Em seguida, a técnica SVM é avaliada em um experimento próprio. O capítulo está organizado em seções para as métricas e para cada experimento.

5.1 Avaliação de Desempenho do FaceNet

A avaliação de desempenho do FaceNet é dividida em duas partes. A primeira parte verifica a capacidade da rede treinada utilizando a abordagem FaceNet gerar *embeddings* condizentes com (SCHROFF; KALENICHENKO; PHILBIN, 2015) a partir das imagens dos *datasets* utilizando as métricas acurácia e AUC. A segunda parte avalia o tempo de processamento na tarefa de geração dos *embeddings* a partir das imagens.

5.1.1 Avaliação de acurácia e AUC do FaceNet

Nesse experimento são avaliados a acurácia e AUC da abordagem FaceNet a partir do padrão de testes presente em (LEARNED-MILLER GARY B. HUANG; HUA, 2018). A Tabela 4 mostra os resultados dos experimentos iniciais descritos na seção 4.5 com os *datasets* utilizando o FaceNet.

A Tabela 5 compara o desempenho descrito em (SCHROFF; KALENICHENKO; PHILBIN, 2015) com o desempenho obtido nos testes com a implementação disponível em (SANDBERG, 2018). Os resultados sugerem que, mesmo com um *dataset* de baixa qualidade, como o YFD, o FaceNet consegue criar bons *embeddings* na hiperesfera, separando imagens de pessoas diferentes e unindo imagens da mesma pessoa. Pode ser observado que, quanto

Tabela 4 – Resultados do FaceNet

Dataset	Acurácia	AUC
LAR	$99,68 \pm 0,22$	$0,998$
LFW	$99,55 \pm 0,34$	$1,000$
YFD	$97,23 \pm 0,18$	$0,986$

Fonte: elaborada pelo autor.

melhor o *dataset* (o *dataset* LAR é o de melhor qualidade), maior a acurácia do FaceNet. Em comparação com (SCHROFF; KALENICHENKO; PHILBIN, 2015), a implementação que se encontra em (SANDBERG, 2018) tem resultados semelhantes, demonstrando que a implementação representa bem o que está desctito no artigo.

Tabela 5 – Avaliação de Desempenho do FaceNet em (SCHROFF; KALENICHENKO; PHILBIN, 2015) e (SANDBERG, 2018)

Dataset	FaceNet *(SCHROFF et al., 2015)*	FaceNet *(SANDBERG, 2018)*
LFW	$99,63\% \pm 0,09$	$99,55\% \pm 0,34$
YFD	$95,12\% \pm 0,39$	$97,23\% \pm 0,18$

Fonte: elaborada pelo autor.

5.1.2 Mensuração do tempo de execução do FaceNet

O teste de tempo de execução verifica quanto tempo leva para gerar o *embedding* a partir de uma imagem de face. A Tabela 6 mostra os resultados desse teste, bem como o tamanho dos *datasets*.

Os resultados sugerem que, pela pouca diferença entre os tempos para gerar um único *embedding* e considerando que as imagens alinhadas possuem todas o mesmo tamanho (160x160), a qualidade da imagem não impacta no desempenho, já que os *pixels* são representados da mesma maneira independente da qualidade. O volume de dados de uma imagem de baixa qualidade e de uma de alta qualidade com a mesma resolução vai ser o mesmo na entrada

Tabela 6 – Resultados de tempo de execução do FaceNet

Dataset	Tamanho do Dataset (N. de Imagens)	Tamanho do Dataset (N. de Classes)	Tempo médio para geração de um embedding (em segundos)	Tempo total para geração dos embeddings (em segundos)
LAR	598	28	$0,0477101$	$28,53066$
LFW	13233	5794	$0,0490809$	$649,48783$
YFD	415861	1594	$0,0500012$	$20793,61332$

Fonte: elaborada pelo autor.

da rede neural. A quantidade de imagens no *dataset* mostrou leve impacto no tempo médio de geração por imagem.

5.2 Avaliação de Desempenho do Classificador SVM

A avaliação de desempenho da técnica SVM verifica a capacidade da mesma separar em *clusters* (um para cada classe) os *embeddings* gerados a partir dos *datasets*. A Tabela 7 apresenta acurácia e AUC da SVM com os *datasets* testados.

Tabela 7 – Resultados da SVM

Dataset	Acurácia	AUC
LAR	$100,0\%$	$0,999$
LFW	$99,95\% \pm 0,00299$	$0,999$
YFD	$99,99\% \pm 0,00048$	$0,999$

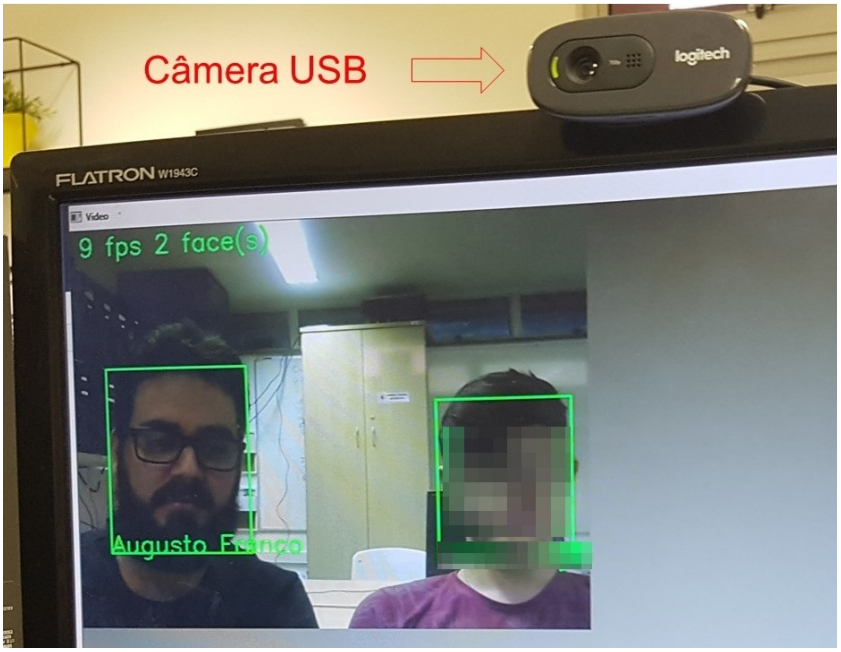
Fonte: elaborada pelo autor.

Os resultados mostram que a técnica SVM treinada a partir dos *embeddings* gerados pelo FaceNet tem uma excelente acurácia, sendo viável à utilização em reconhecimento facial. A SVM treinada é capaz de prever com precisão a que classe uma nova imagem pertence.

5.3 Avaliação de desempenho do sistema proposto

Este experimento verifica a capacidade do sistema de reconhecimento facial em tempo real reconhecer pessoas cadastradas a partir do *streaming* da webcam. Foram selecionadas 15 pessoas para a avaliação e feitas um total de 40 passagens pela câmera (média de $2,66$ passagens por pessoa), computando as classificações corretas e incorretas. O sistema conseguiu classificar corretamente em 36 passagens (90% de acerto). A Figura 14 ilustra o sistema em execução detectando e reconhecendo corretamente 2 faces.

Figura 14 – Sistema em execução



Fonte: elaborada pelo autor.

6 CONCLUSÃO E TRABALHOS FUTUROS

As quedas nos custos e popularização de sistemas de videomonitoramento vêm fazendo com que os mesmos sejam cada vez mais utilizados para os mais diversos fins. Com o aumento da utilização desses sistemas, fica cada vez mais difícil e dispendioso utilizar pessoas para o monitoramento das imagens geradas. Além disso, humanos não lidam bem com grande volume de informação. Para isso, sistemas de reconhecimento facial (em tempo real ou não) podem ser utilizados para automatizar diversos processos envolvendo videomonitoramento. Neste sentido, este trabalho analisou a aplicação de *deep learning* para reconhecimento facial em tempo real.

Neste trabalho, foi modelado um sistema de reconhecimento facial em tempo real utilizando FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015) com a arquitetura de rede *Inception-ResNet V1* combinada com os pesos treinados e disponibilizados para o *dataset* de treinamento *VGGFace2*. Para a tarefa de classificação supervisionada, o classificador SVM foi utilizado. Para analisar a viabilidade dessas técnicas, um conjunto de testes foram realizados. O FaceNet foi avaliado quanto às métricas de desempenho acurácia e AUC utilizando os *datasets* previamente descritos LFW, YFD e LAR, de acordo com o teste de *benchmark* descrito em (LEARNED-MILLER GARY B. HUANG; HUA, 2018). Em seguida, o classificador SVM foi analisado quanto à acurácia e AUC para utilizando os mesmos *datasets*. Por fim, um sistema de reconhecimento facial em tempo real foi modelado e construído. O sistema se divide em quatro partes: (1) o alinhamento, que detecta, recorta e alinha as faces da imagem; (2) o gerador de *embeddings*, que passa as imagens faciais pela rede FaceNet a fim de obter as representações vetoriais (embeddings) dessas imagens; (3) o treinamento do classificador, que treina o algoritmo SVM utilizando os *embeddings* obtidos a fim de classificar novas imagens; e (4) o reconhecimento em tempo real, que abre uma janela com o *streaming* de uma câmera USB conectada ao computador, envia esse *streaming* para alinhamento, realiza a geração de *embeddings* e obtém a predição do classificador para determinar

a que pessoa cadastrada esta imagem pertence (qualquer imagem nova será predita como pertencendo a uma pessoa já cadastrada).

Os resultados mostram que a combinação de FaceNet com SVM tem ótimos resultados. A avaliação da abordagem FaceNet utilizada e disponível em (SANDBERG, 2018) obteve acurácia de $99,55 \pm 0,34$ com o *dataset* LFW, o que representa um desempenho próximo aos $99,63\% \pm 0,09$ descritos em (SCHROFF; KALENICHENKO; PHILBIN, 2015), e obteve uma acurácia de $99,68\% \pm 0,22$ com o *dataset* LAR, gerando um *embedding* a cada $0,0477101$ segundos nesse *dataset*. A técnica SVM conseguiu classificar com acurácias de $100,0\%$, $99,95\% \pm 0,00299$, $99,99\% \pm 0,00048$ *embeddings* gerados a partir dos *datasets* LAR, LFW e YFD. A avaliação do sistema de reconhecimento facial em tempo real, a qual ocorreu nas dependências do Laboratório de Redes de Computadores e Sistemas do IFCE campus Aracati, demonstrou que a solução descrita foi capaz de reconhecer corretamente em 90% das passagens na frente da câmera, mesmo utilizando uma *webcam* de qualidade mediana. A combinação de FaceNet com SVM se mostrou eficaz na tarefa de reconhecimento facial em tempo real.

Trabalhos futuros na nesta linha de pesquisa incluem a avaliação de outras arquiteturas de CNN, além da *Inception-ResNet V1*, e outras abordagens além do FaceNet. Além disso, pretende-se avaliar outras técnicas de *machine learning*. Após execução e análise de experimentos preliminares com outras arquiteturas e outras técnicas de ML, testes mais robustos envolvendo *datasets* completos e maiores poderão ser realizados para estudar estratégias para superar possíveis dificuldades ao se trabalhar com grande volume de dados.

REFERÊNCIAS

AGARWAL, M.; AGRAWAL, H.; JAIN, N.; KUMAR, M. Face recognition using principle component analysis, eigenface and neural network. In: *2010 International Conference on Signal Acquisition and Processing*. [S.l.: s.n.], 2010. p. 310–314. Citado 2 vezes nas páginas 15 e 16.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. *2017 International Conference on Engineering and Technology (ICET)*. [S.l.], 2017. p. 1–6. Citado na página 25.

AMOS, B.; LUDWICZUK, B.; SATYANARAYANAN, M. *OpenFace: A general-purpose face recognition library with mobile applications*. [S.l.], 2016. Citado 5 vezes nas páginas 15, 33, 36, 38 e 43.

CHOWDHRY, D. A.; HUSSAIN, A.; REHMAN, M. Z. U.; AHMAD, F.; AHMAD, A.; PERVAIZ, M. Smart security system for sensitive area using face recognition. In: *2013 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (CSUDET)*. [S.l.: s.n.], 2013. p. 11–14. Citado na página 15.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 19.

COŞKUN, M.; UÇAR, A.; YILDIRIM, ; DEMIR, Y. Face recognition based on convolutional neural network. In: *2017 International Conference on Modern Electrical and Energy Systems (MEES)*. [S.l.: s.n.], 2017. p. 376–379. Citado na página 15.

GARCÍA, S.; FERNÁNDEZ, A.; LUENGO, J.; HERRERA, F. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, Springer, v. 13, n. 10, p. 959, 2009. Citado na página 46.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org. Citado 2 vezes nas páginas 20 e 21.

Guo, S.; Chen, S.; Li, Y. Face recognition based on convolutional neural network and support vector machine. In: *2016 IEEE International Conference on Information and Automation (ICIA)*. [S.l.: s.n.], 2016. p. 1787–1792. Citado na página 20.

HATCHER, W. G.; YU, W. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, IEEE, v. 6, p. 24411–24432, 2018. Citado na página 22.

HU, G.; YANG, Y.; YI, D.; KITTLER, J.; CHRISTMAS, W.; LI, S. Z.; HOSPEDALES, T. When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition. In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. [S.l.: s.n.], 2015. p. 384–392. Citado 2 vezes nas páginas 15 e 16.

HUANG, G. B.; RAMESH, M.; BERG, T.; LEARNED-MILLER, E. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. [S.l.], 2007. Citado 3 vezes nas páginas 43, 44 e 46.

JAIN, R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* [S.l.]: Wiley, 1991. I-XXVII, 1-685 p. (Wiley professional computing). ISBN 978-0-471-50336-1. Citado na página 48.

KIM, K. I.; JUNG, K.; KIM, H. J. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, v. 9, n. 2, p. 40–42, Feb 2002. ISSN 1070-9908. Citado na página 15.

KREMIC, E.; SUBASI, A. Performance of random forest and svm in face recognition. *Int. Arab J. Inf. Technol.*, v. 13, n. 2, p. 287–293, 2016. Citado na página 48.

KUMAR, V.; SVENSSON, J. *Promoting Social Change and Democracy Through Information Technology*. IGI Global, 2015. (Advances in electronic government, digital divide, and regional development). ISBN 9781466685031. Disponível em: <https://books.google.com.br/books?id=jkdLCgAAQBAJ>. Citado na página 14.

LEARNED-MILLER GARY B. HUANG, A. R. H. L. E.; HUA, G. *Labeled Faces in the Wild*. 2018. http://vis-www.cs.umass.edu/lfw/. Acessado em 30-03-2019. Citado 3 vezes nas páginas 47, 51 e 55.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015. Citado 3 vezes nas páginas 22, 23 e 24.

LI, J.; ZHAO, B.; ZHANG, H.; JIAO, J. Face recognition system using svm classifier and feature extraction by pca and lda combination. In: *2009 International Conference on Computational Intelligence and Software Engineering*. [S.l.: s.n.], 2009. p. 1–4. Citado 2 vezes nas páginas 15 e 16.

Ling and C. X. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 3, p. 299–310, March 2005. ISSN 1041-4347. Citado na página 46.

NILSSON, N. J. *Introduction to machine learning: An early draft of a proposed textbook*. [S.l.]: USA; Stanford University, 1996. Citado na página 19.

PATTERSON, J.; GIBSON, A. *Deep Learning: A Practitioner's Approach*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado 2 vezes nas páginas 20 e 24.

PRIHASTO, B.; CHOIRUNNISA, S.; NURDIANSYAH, M. I.; MATHULAPRANGSAN, S.; CHU, V. C.; CHEN, S.; WANG, J. A survey of deep face recognition in the wild. In: *2016 International Conference on Orange Technologies (ICOT)*. [S.l.: s.n.], 2016. p. 76–79. Citado na página 45.

SANDBERG, D. *Face recognition using Tensorflow - GitHub Repository*. 2018. https://github.com/davidsandberg/facenet. Acessado em 26-03-2019. Citado 8 vezes nas páginas 10, 17, 41, 46, 48, 51, 52 e 56.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 815–823. ISSN 1063-6919. Citado 15 vezes nas páginas 10, 15, 16, 17, 33, 36, 37, 39, 45, 46, 48, 51, 52, 55 e 56.

SHARIF, M.; NAZ, F.; YASMIN, M.; SHAHID, M. A.; REHMAN, A. Face recognition: A survey. In: *2017 Journal of Engineering Science and Technology Review (JESTR)*. [S.l.: s.n.], 2017. p. 166–177. Citado na página 15.

SUN, Y.; WANG, X.; TANG, X. Deeply learned face representations are sparse, selective, and robust. *CoRR*, abs/1412.1265, 2014. Disponível em: <http://arxiv.org/abs/1412.1265>. Citado na página 48.

SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. Disponível em: <http://arxiv.org/abs/1602.07261>. Citado na página 46.

TAIGMAN, Y.; YANG, M.; RANZATO, M.; WOLF, L. Deepface: Closing the gap to human-level performance in face verification. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1701–1708. ISSN 1063-6919. Citado na página 48.

WANG, M.; DENG, W. Deep face recognition: A survey. *CoRR*, abs/1804.06655, 2018. Disponível em: <http://arxiv.org/abs/1804.06655>. Citado 3 vezes nas páginas 20, 24 e 25.

WOLF, L.; HASSNER, T.; MAOZ, I. Face recognition in unconstrained videos with matched background similarity. In: *CVPR 2011*. [S.l.: s.n.], 2011. p. 529–534. ISSN 1063-6919. Citado 2 vezes nas páginas 43 e 44.

YA, W.; TIANLONG, B.; CHUNHUI, D.; MING, Z. Face recognition in real-world surveillance videos with deep learning method. In: *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. [S.l.: s.n.], 2017. p. 239–243. Citado 3 vezes nas páginas 14, 34 e 35.

ZHANG, K.; ZHANG, Z.; LI, Z.; QIAO, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, v. 23, n. 10, p. 1499–1503, Oct 2016. ISSN 1070-9908. Citado na página 26.

ZHANG, Y.; YANG, Q. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017. Citado na página 26.

ZHU, Z.; LUO, P.; WANG, X.; TANG, X. Recover canonical-view faces in the wild with deep neural networks. *CoRR*, abs/1404.3543, 2014. Disponível em: <http://arxiv.org/abs/1404.3543>. Citado na página 48.