UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
CAMPUS AVANÇADO DE NATAL
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LIELLISON JOSEPH SILVESTRE DE MENEZES

UTILIZAÇÃO DA REDE NEURAL CONVOLUCIONAL FACENET PARA RECONHECIMENTO FACIAL HUMANO POR MEIO DE SMARTPHONES IOS

NATAL
2019

LIELLISON JOSEPH SILVESTRE DE MENEZES

UTILIZAÇÃO DA REDE NEURAL CONVOLUCIONAL FACENET PARA RECONHECIMENTO FACIAL HUMANO POR MEIO DE SMARTPHONES IOS

Monografia apresentada à Universidade do Estado do Rio Grande do Norte - UERN - Departamento de Computação como requisito obrigatório para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Dra. Adriana Takahashi.

NATAL

2019

M543u Menezes, Liellison Joseph Silvestre de
UTILIZAÇÃO DA REDE NEURAL CONVOLUCIONAL FACENET PARA RECONHECIMENTO FACIAL HUMANO POR MEIO DE SMARTPHONES IOS. / Liellison Joseph Silvestre de Menezes. - Natal, 2019.
55p.

Orientador(a): Profa. Dra. Adriana Takahashi.
Monografia (Graduação em Ciência de Computação). Universidade do Estado do Rio Grande do Norte.

1. Machine Learning. 2. CNN. 3. CoreML. 4. iOS. 5. FaceNet. I. Takahashi, Adriana. II. Universidade do Estado do Rio Grande do Norte. III. Título.

LIELLISON JOSEPH SILVESTRE DE MENEZES

UTILIZAÇÃO DA REDE NEURAL CONVOLUCIONAL FACENET PARA RECONHECIMENTO FACIAL HUMANO POR MEIO DE SMARTPHONES IOS

Monografia apresentada à Universidade do Estado do Rio Grande do Norte - UERN - Departamento de Computação como requisito obrigatório para obtenção do título de Bacharel em Ciência da Computação.

Aprovado em ____/____/____.

BANCA EXAMINADORA

_____

Dra. Adriana Takahashi
Universidade do Estado do Rio Grande do Norte

_____

Dr. Anderson Abner de Santana Souza
Universidade do Estado do Rio Grande do Norte

_____

Dr. Wilfredo Blanco Figuerola
Universidade do Estado do Rio Grande do Norte

Dedico este trabalho ao Lampião da Esquina e a comunidade LGBTPQIA+, suas lutas foram o meu sustento e me deram coragem para persistir nesta jornada.

AGRADECIMENTOS

Ao meu cachorro por sempre esta ao meu lado durante todos os dias da minha graduação e nunca ter me abandonado. Agradeço grandemente a todos meus professores que me ensinaram tudo que sei hoje, e sem isso este projeto não existiria. E também sou grato aos meus parentes por mesmo sem entender o que eu estudo, me deram apoio para continuar estudando.

I have a dream that my four little children will one day live in a nation where they will not be judged by the color of their skin but by the content of their character.

Martin Luther King Jr (1963)

RESUMO

Com o avanço da tecnologia, vemos cada vez mais o uso dos smartphones presentes na vida do humano moderno. Tal ferramenta tem uma enorme gama de possibilidades. Uma possibilidade que vem sendo explorada, é o reconhecimento facial. Com a melhoria das câmeras e do processamento dos smartphones, temos este método como possível para uso. Para ser possível o reconhecimento facial por meio dos smartphones, pode-se usar uma série de sensores, entre câmeras, infra-vermelho, projetor de pontos, sensor de ambiente e iluminação por LED, todos estes sensores são necessários para realização do reconhecimento de forma precisa. Porém, uma outra possibilidade é a aplicação de uma rede neural artificial para fim de reconhecer o rosto humano. Neste projeto foi empregada uma rede neural chamada FaceNet de modo a realizar o reconhecimento facial humano por meio da câmera do smartphone com o sistema operacional iOS, com integração entre a rede neural e o smartphone feita pelo CoreML.

Palavras-chave: CNN. CoreML. iOS. FaceNet. Machine Learning.

RESUMEN

Con el avance de la tecnología, vemos cada vez más el uso de teléfonos inteligentes en la vida del ser humano moderno. Tal herramienta tiene una gran variedad de posibilidades. Una de las posibilidad que ha sido explorada es el reconocimiento facial. Con la mejora del procesamiento de cámaras y teléfonos inteligentes, tenemos este método como sea posible para usar. Para permitir el reconocimiento facial a través de los teléfonos inteligentes, se utiliza una variedad de sensores, que incluyen cámaras, infrarrojos, proyectores puntuales, sensores ambientales e iluminación LED, todos estos sensores son necesarios para un reconocimiento preciso. Por mas que, otra métrica es el uso de una red neuronal artificial para reconocer el rostro humano. En este proyecto, se utilizó una red neuronal llamada FaceNet para realizar el reconocimiento facial humano a través de la cámara del teléfono inteligente con el sistema operativo iOS, con integración entre la red neuronal y el teléfono inteligente hecha por CoreML.

Palabras-clave: CNN. CoreML. iOS. FaceNet. Machine Learning.

ABSTRACT

With the advancement of technology, we increasingly see the use of smartphones being present in the life of the modern human. Such a tool has a huge range of possibilities. One possibility that has been explored is facial recognition. With the improvement of cameras and processing of smartphones, we have this method as possible for use. Face recognition is used without the user feeling that it is being used because it is discreet and fast. To enable or recognize facial through smartphones, use a range of sensors, including cameras, infrared, spot projector, ambient sensor and LED illumination, all these sensors are designed to perform recognition accurately. However, a metric is the use of an artificial neural network to recognize the human face. In this project a neural network called FaceNet was used to perform human facial recognition through the smartphone camera with the iOS operating system, with integration between the neural network and the smartphone made by CoreML.

Keywords: CNN. CoreML. iOS. FaceNet. Machine Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
LMNN	*Large Margin Nearest Neighbor*
ML	*Machine Learning*
SVM	*Support Vector Machines*
KNN	*K-Nearest Neighbors*
SGD	*Stochastic Gradient Descent*
SDK	*Software Developer Kit*

SUMÁRIO

1 INTRODUÇÃO

O rosto humano é um item único de cada indivíduo. Existem vários formatos de rostos, desde largos a mais finos, passando por longos e outros nem tanto. Mesmo indivíduos gêmeos monozigóticos, eles se divergem pelos rostos, pois o rosto é único em cada humano. Quando somos novos, normalmente o primeiro rosto que aprendemos são os rostos dos nossos pais, reconhecendo estes como sendo rostos confiáveis para nós.

Tendo em vista o quão único são os rostos humanos, nos dias atuais podemos encontrar facilmente modelos de autenticação que se utilizam do rosto como, alguns smartphones atuais, nos quais reconhecem o rosto do usuário para realizar o desbloqueio e a autenticação de compras internas ou mesmo em sites externos de compra. A singularidade de cada rosto é tão grande que, há muito tempo é usada por meio de dispositivos eletrônicos para o reconhecimento facial. A Sony já trabalhava com o reconhecimento da face humana nas suas câmeras desde 2007, quando a empresa ganhou um certo reconhecimento por lançar câmeras e software de processamento com capacidade de reconhecer o sorriso e retirar em automático uma foto somente com o sorriso do usuário, conforme descreve Sampaio (2007).

Este sistema de reconhecimento de expressão facial foi utilizado por outras fabricantes de máquinas fotográficas em vários modelos de câmeras que eles produziram. Atualmente, com a evolução das câmeras, a tecnologia de reconhecimento facial vem sendo utilizada para criar a perspectiva de profundidade nas fotos. Para isso é, utilizado um par de lentes, e cria-se este efeito de profundidade nas fotos, deixando assim muito mais real as fotos, facilitando o reconhecimento do rosto humano e por sua vez do corpo, e criando um fundo desfocado por trás, tentando se assemelhar com as imagens capturadas pelo olho humano (CARVALHO, 2019).

Vivemos em um momento no qual os smartphones já estão em um nível de evolução muito alta, como bem nos informa Santana (2019), chegando em um patamar de usabilidade satisfatória para a maioria dos usuários. Nos últimos anos observamos lançamentos cada vez mais sofisticados em tecnologia. E mesmo assim, ainda vemos as câmeras dos smartphones sendo usados somente para o registro de fotos de forma tradicional. Todavia com todo o poder de processamento disponibilizado para os smartphones hoje, e com o avanço das câmeras, nós podemos fazer

muito mais com estas ferramentas.

A proposta nesta pesquisa é desenvolver uma ferramenta móvel para reconhecimento facial, utilizando a FaceNet, desenvolvida por Schroff (2015). A FaceNet é uma rede neural convolucional que utiliza a distância euclidiana para análise de pontos que caracterizam a face e posteriormente classificam a face de um indivíduo, sendo mais simples de utilização pelos usuários por necessitar de menos passos de configuração para eles.

Esta rede neural foi escolhida para ser aplicada neste trabalho pelo fato de ter seu desenvolvimento voltado para smartphones e também ter uma melhor integração com o sistema operacional móvel.

1.1 JUSTIFICATIVA

Este trabalho tem como motivação a criação de uma aplicação para smartphones, com o sistema operacional iOS, que utiliza a rede neural convolucional FaceNet para realizar o reconhecimento facial de humanos por meio da câmera do celular, tirando um maior proveito desta tecnologia que temos em mãos atualmente.

Um outro ponto que motiva a criação deste projeto é a falta de um modelo de reconhecimento que seja simples para aquele usuário que não dispõe de muito conhecimento de tecnologia, trazendo uma interface que proporcione uma boa experiência de usuário e que possa ser manuseada por uma maior variedade de pessoas, pois, os modelos atuais não possuem uma interface amigável e isso afasta certos grupos de utilizadores, como, por exemplo o Face ID do iPhone (SUPORTE, 2019) onde a configuração é confusa, no modelo que proposto neste projeto é necessário somente uma única foto do usuário, não sendo necessário uma configuração com várias passos. Assim, criar um dispositivo com uma interface focando na experiência de usuário para que seja de fácil manuseio é a grande motivação para a criação deste projeto.

Os modelos atuais de reconhecimento facial são visto em uma variedade de equipamentos, onde, eles têm uma interface complexa e sem uma forma simples de reconhecimento por meio de quem está utilizando a ferramenta. Os usuários precisam ter certo conhecimento de uso para poder utilizar as aplicações atuais. Este projeto visa mudar este cenário, com uma aplicação de interface simples e de fácil uso para usuários que não disponham de um conhecimento profundo acerca de inte-

ligência artificial, mas que querem desfrutar das opções que a tecnologia pode proporcionar.

1.2 OBJETIVOS

Este projeto tem como objetivo geral a elaboração de um sistema voltado para smartphones capaz de realizar o reconhecimento de faces humanas, bem como identificar se a face é de um determinado indivíduo, tendo o propósito de com isso realizar a autenticação para acesso o diretório de álbuns de fotos do smartphone.

1.2.1 Objetivos Específicos

Além dos objetivos gerais, temos alguns outros pontos que foram necessários para o desenvolvimento deste projeto.

- Pesquisa bibliográfica sobre: redes neurais, CNN, técnicas de reconhecimento de faces.
- Desenvolvimento de um aplicativo para smartphones;
- Utilização da rede FaceNet para reconhecer faces;
- Análise de traços do rostos humanos;
- Instalação dos requisitos necessários para o aplicativo reconhecer faces.

1.3 METODOLOGIA

Este projeto é baseado no tipo de pesquisa exploratória, pelo fato de ser desenvolvido um protótipo do sistema proposto.

Na criação do aplicativo para smartphones, o foco foi para dispositivos com o sistema operacional iOS, e ele foi desenvolvido de forma nativa utilizando a linguagem de programação Swift (SWIFT, 2019), em momento que se utiliza o FaceNet no CoreML. A escolha da linguagem de programação Swift se deu pelo fato de ser uma linguagem proprietária da Apple, sendo uma linguagem totalmente criada com o foco no sistema operacional iOS, desse modo ela tem uma melhor integração com o sistema, e com isso apresenta um melhor desempenho de processamento, que é um

item que o sistema necessita para poder funcionar corretamente. A linguagem de programação Python (PYTHON, 2019) foi utilizada em paralelo com o Swift para a criação da rede neural FaceNet.

A IDE utilizada foi o XCode, desenvolvida pela própria Apple, pois ela dá um suporte maior quanto ao desenvolvimento nativo das aplicações para o sistema operacional iOS. Foi usado um notebook com sistema operacional macOs Mojave, para o desenvolvimento e testes da ferramenta, pela limitação da possibilidade de desenvolvimento para iOS, onde somente podemos criar aplicações utilizando o sistema operacional macOs.

O desenvolvimento deste projeto se deu inicialmente com a utilização da rede neural FaceNet que é o núcleo do sistema, onde nela é realizado o reconhecimento em si dos rostos humanos. A rede já está desenvolvida, mas para utilização, foi feita uma migração do código em Python para o *Core ML* da Apple, utilizando ferramenta própria para este procedimento, no caso a *tf-coreml*. Paralelo a isso, temos a criação da aplicação móvel, que é basicamente o mesmo sistema, porém a parte do aplicativo móvel em si, que é a parte vista pelo usuário.

Para o treinamento da rede foi utilizada a base de dados criada por CHEN, Sheng; LIU, Yang; GAO, Xiang; et al (2018) em seu projeto, por ter uma grande variedade de rostos e de ângulos dos rostos humanos, fazendo com isso que, a rede entenda bem o rosto humano no seu treinamento prévio. E para re-treinamento, foi utilizado uma base própria.

O desenvolvimento deste trabalho é dividido em etapas, para que seja mais claro os pontos a serem feitos e se tenham uma melhor organização do trabalho, as fases do desenvolvimento foram:

Fase 1: O primeiro passo no desenvolvimento deste trabalho é a criação de uma aplicação para smartphones que tenha capacidade de acessar a câmera do dispositivo, que foi desenvolvido em Swift. Após a aquisição e captura de fotos, temos o segundo passo, onde a aplicação é capaz de reconhecer (ou não) o individuo da foto, com ajuda da rede neural.

Fase 2: Para o desenvolvimento do segundo passo, temos o treinamento da rede com as bases de dados, e a migração da rede neural FaceNet. Rede que foi desenvolvida em Python e foi migrada para *Core ML*. Foram usadas as diretivas indicadas por Schoff (2015), para tal desenvolvimento.

Fase 3: Integração da rede neural junto com a aplicação para smartphones, onde teremos a finalização do desenvolvimento. Sendo possível a união das duas parte realizar o reconhecimento facial. Tenso assim a aplicação que realiza a captura das imagens e a rede neural que processa o conteúdo da imagem e a identificação da face humana.

1.4 ESTRUTURA

Além deste capítulo introdutório, este trabalho tem em sua composição mais 5 capítulos assim descritos:

No capítulo 2 se encontra a fundamentação teórica: são apresentados os históricos e as informações da tecnologia adotada. Informações importantes são passadas sobre os aspectos da tecnologia e sua forma de funcionamento e utilização.

No capítulo 3 é descrito um embasamento sobre a rede neural FaceNet, e os projetos relacionados, onde neste capítulo também são apresentado os projetos relacionados que contribuíram como referência para o desenvolvimento deste trabalho.

No capítulo 4 é apresentado a aplicação desenvolvida neste projeto.

No capítulo 5 é explicado o experimento realizado para o reconhecimento do rosto humano neste projeto.

No capítulo 6 temos a conclusão e as perspectivas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para um melhor entendimento deste projeto são apresentados alguns conceitos necessários pela existência de alguns temas que foram utilizados na elaboração deste projeto e são exclusivos de um nicho de conhecimento em Ciência da Computação.

2.1 REDE NEURAL ARTIFICIAL

Russell e Norvig (2013), define uma rede neural artificial sendo um modelo matemático do cérebro humano, onde tenta representar a nível computacional do cérebro humano para que seja possível a representação. Inicialmente, foi criado um único neurônio para que seja possível hoje com a evolução do conhecimento, temos modelos muito complexos que é viável representar muitas habilidades humanos, por meio do computadores. Russell e Norvil, fala sobre isso da seguinte forma:

> Desde 1943, têm sido desenvolvidos modelos muito mais detalhados e realistas, tanto de neurônios como de sistemas maiores no cérebro, levando ao campo moderno da neurociência computacional. Por outro lado, os pesquisadores de IA e os estatísticos tornaram-se interessados nas propriedades mais abstratas das redes neurais, tais como sua capacidade de realizar computação distribuída, de tolerar entradas ruidosas e aprender.

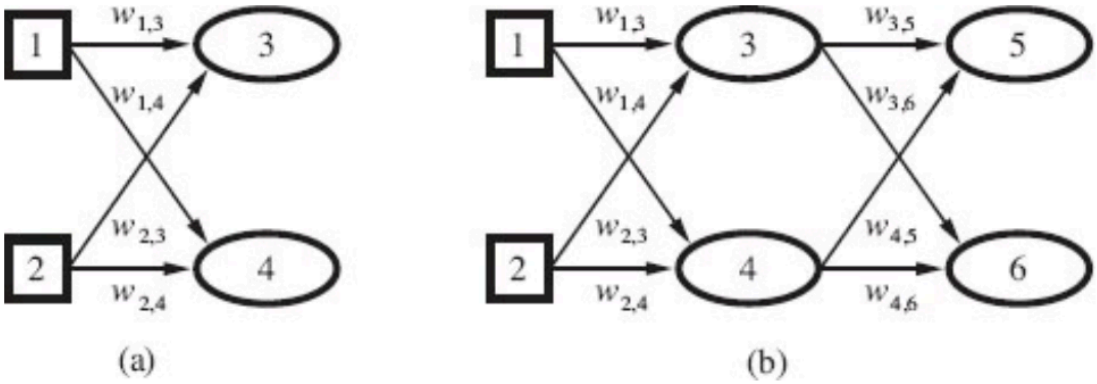
2.1.1 Perceptron

Como citado por Russell e Norvig (2013), no inicio, foi desenvolvido um único neurônio e com isso foi possível o avanço para os modelos que temos hoje. Com este desenvolvimento, foi criado o perceptron, um neurônio artificial de uma rede com todas as entradas conectadas diretamente com as saídas, também conhecido como rede neural de camada única. Para cada entrada na rede temos uma respectiva saída, assim, se na rede temos 4 entradas, temos respectivamente 4 saídas com resultados.

Uma limitação deste tipo de rede, segundo Russell e Norvig (2013), é a impossibilidade de trabalhar com problemas que não seja linearmente separados, sendo impossível aplicar problemas clássicos como o XOR em uma rede de camada

única, para solucionar este problemas foi desenvolvido um novo modelo de perceptron, agora sendo um perceptron de multi-camadas, com novas camadas na rede, com outros modelos matemáticos e com isso sendo possível trabalhar com problemas que sejam linearmente separados, quantos os que não sejam linearmente separados. Na Figura 1, é mostrado a estrutura.

Figura 1: (a) Representação do perceptron. (b) Representação do MLP



Fonte: Russell e Norvig (2013)

2.1.2 SVM

Lorena e Carvalho (2007) descrevem a Máquinas de Vetores de Suporte (*Support Vector Machines* - SVM) como uma técnica de aprendizado de máquina, onde, os resultados obtidos nas aplicações que utilizam esta técnica é comparado como em certos modelos sendo superior aos obtidos por outros algoritmos de aprendizado, sendo sistemas de aprendizagem de máquina treinados com um algoritmo de otimização matemática e que implementam um limite derivado da teoria de aprendizagem estatística. Ela é uma rede neural com duas camadas sendo a primeira uma camada escondida de unidades radiais e, uma segunda com um neurônio de saída. Uma das aplicações da SVM é em soluções de problemas de regressão e no agrupamento de dados, no aprendizado não supervisionado, essa técnica pode ser aplicada também ao reconhecimento de padrões, regressão e extração de características.

Com as SVMs é possível trabalhar com dados de grande dimensão, como descreve Lorena e Carvalho (2007), as outras técnicas de aprendizado de máquina normalmente obtêm classificadores super ou sub ajustados pelos dados de grande dimensões. Um outro fator de destaque é a convexidade do problema de otimização utilizado no treinamento de uma SVM, que define um único mínimo global como existente. O fato de ter um mínimo local na função objetivo minimizada é uma vantagem das SVMs em relação as MLP.

2.2 *DEEP LEARNING*

A Aprendizagem Profunda (*deep learning*) é descrita por Chollet (2018) como sendo uma revolução no campo da inteligência artificial, tendo chegado em um nível bom para resolução de problemas de visão e audição computacional. Um grande diferencial da *deep learning* é que os dados utilizados não são organizados para executar a equações predefinidas, ele utiliza reconhecimentos de padrões para com isso, dar a possibilidade das máquinas de aprender sozinhas com este reconhecimento em várias camadas de processamento, resolvendo problemas que teoricamente só podem ser feitos por humanos, onde agora é possível ser utilizado em máquinas. Ainda de acordo com Chollet, o *deep learning*, vem para suprir alguns pontos que a inteligência artificial ainda não é bem aplicada, como os seguintes pontos:

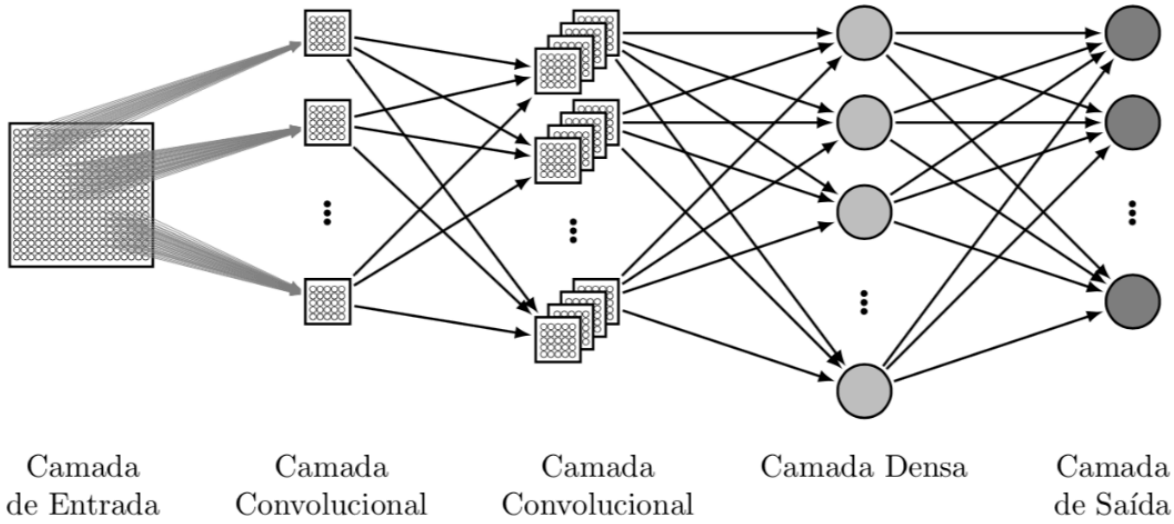
- Classificação de imagem.
- Reconhecimento de fala.
- Transcrição de caligrafia.
- Tradução automática aprimorada.
- Conversão aprimorada de texto em fala.
- Assistentes digitais como virtuais.
- Condução autônoma.
- Segmentação de anúncios aprimorada, usada pelos motores de buscas onlines.
- Resultados de pesquisa aprimorados na web.
- Capacidade de responder a perguntas em linguagem natural.

Menezes (2016) utilizou o *Deep Learning* para o reconhecimento de sinais de trânsito, onde, dividiu em duas etapas, a primeira etapa foi a aquisição de base de dados dos sinais, e a segunda foi a geração de um modelo de treinamento para realizar a classificação destes sinais de trânsito. Esta forma de utilizar o *Deep Learning*, dividindo em etapas, é uma boa prática para o desenvolvimento, segundo Menezes, pelo fato de classificarmos o problema.

2.2.1 Rede Neural Convolucional

Uma rede neural convolucional, também conhecida como *Convolutional Neural Network* (CNN), mostrado na Figura 2, está incluso no estudo de aprendizagem de máquinas, e tem como ponto de destaque a resolução de problemas com estrutura de representação de dados em diferentes maneiras de entrada para a rede, como descreve Deng e Dong (2004).

Figura 2: Estrutura do CCN.



Fonte: Sakurai (2017)

A Figura 2 mostra na camada de entrada, uma imagem, posteriormente nas camadas convolucionais são extraídos características para alimentar o classificador. Uma rede neural convolucional utiliza, segundo Graham (2015), dois tipos de camadas, sendo elas, as camadas de convolução e as camadas de pooling (minerar). Nestas camadas são extraídos características, as informações, que entram na rede,

esta mineração é profunda. Existem algumas técnicas de pooling, como o max-pooling, que é um procedimento onde é utilizado uma matriz de entrada com NxN e retorna como saída desta camada uma matriz ainda NxN, porém, menor. Isso é possível por ser dividido a entrada em quadradas de regiões do pool.

Moacir e Gabriel (2017) descrevem uma técnica chamada de *pooling*, como sendo a forma de reduzir o tamanho da dimensão espacial ao longo das camadas de um rede neural. Este método tem dois propósitos definidos, sendo a melhoria no poder computacional, todavia com a diminuição da dimensão espacial, é mais simples para o computador poder calcular esta técnica, um outro ponto de destaque é a redução do tamanho das imagens, com isso é possível obter um tipo de composição de banco de filtros que é possível ser processado imagens em diferentes espaços-escala.

A próxima camada é a de agrupamento, ou *pooling*, nela é feito a redução de características da imagem, compactando os dados, assim, melhorando o custo computacional aplicado na técnica. Santos (2017) demonstra em seu artigo que, o *pooling* usado na condensação de dados, sendo uma operação que simplifica a informação da saída da camada de convolução, esta informação pode ser modificada por meio de média dos valores de um map ou pelo valor máximo do mapeamento da entrada.

A CNN tem varias arquiteturas, entre elas podemos cita cinco que se destacam em nível de popularidade, sendo elas:

- Recurrent neural networks (RNNs),
- long short-term memory (LSTM)/gated recurrent unit (GRU),
- convolutional neural networks (CNNs),
- deep belief networks (DBN) e
- deep stacking networks (DSNs)

A CNN não é um único método, mais sim, um conjunto de abordagens de classes de algoritmos e topologias que podem ser aplicadas a um grande conjunto de problemáticas, por isso tem varias arquitetutas e dentro de cada arquitetura varias abordagem para resolução de problemas dos mais diversos modos existentes no mundo.

Existem inúmeras arquiteturas de redes neurais convolucionais desenvolvidas atualmente, com vários propósitos e complexidades, porém, a rede desenvolvida por Schroff (2015) tem um ponto de características sobre a maioria das redes deste tipo, que é o fato de ser adaptada para smartphones. Neste caso, ela é independente do sistema operacional, pois está adaptada para os principais do mercado: o Android e IOS, tendo uma versão para cada sistema operacional. Schroff desenvolveu uma rede neural convolucional chamada de FaceNet, se destacando por ter versão otimizada tanto para o sistema operacional Android quanto ao iOS.

O artigo publicado por Schroff (2015) foi adotado como a principal base para o desenvolvimento do aplicativo deste projeto, com ressalva de uso para smartphones, tendo em vista que a FaceNet, mesmo sendo adaptada para smartphones, também pode ser utilizada em computadores, como ideal de uso simples e prático, para reconhecimento facial por meio do smartphone. Como o sistema desenvolvido é voltado para smartphones, pelo fato de serem mais simples de utilizar e mais práticos.

2.3 MACHINE LEARNING

Computadores são capazes de aprender, segundo Donald (1968), a área de machine learning (aprendizagem de maquina), sendo um subcampo da engenharia e da ciência da computação, é a responsável pela capacidade dos computadores aprenderem. Existem várias formas do computador aprender, como, é possível realizar aprendizagem supervisionada ou não, por reforço de ou acertos e erros. Existem inúmeras técnicas para dar a capacidade de aprender aos computadores. É útil que os computadores tenham ela capacidade, para assim ter um ganho maior da usabilidade.

2.4 TENSORFLOW

Para ABADI, Martín; BARHAM, Paul; CHEN, Jianmin; et al (2016), o TensorFlow é uma biblioteca de código aberto de *machine learning* que opera em larga es-

cala, sendo possível trabalhar com um maior numero de tarefas, sendo muito utilizada por este motivo. Ele usa um único gráfico de fluxo de dados para representar todo o cálculo e estado de um algoritmo de aprendizado de máquina, incluindo operações matemáticas individuais, os parâmetros e suas regras de atualização e o pré-processamento de entrada. O gráfico de fluxo de dados expressa a comunicação entre subcomputações explicitamente, facilitando a execução de cálculos independentes em paralelo e a particionamento de cálculos em vários dispositivos.

O TensorFlow (ABADI; BARHAM; CHEN; ET AL, 2016) difere dos sistemas de fluxo de dados em lote em dois aspectos: O modelo suporta várias execuções simultâneas em subgráficos sobrepostos do gráfico geral, e vértices individuais podem ter um estado mutável que pode ser compartilhado entre diferentes execuções do gráfico. A principal observação na arquitetura do servidor de parâmetros é que o estado mutável é crucial ao treinar modelos muito grandes, porque torna possível fazer atualizações locais para parâmetros muito grandes e propagar essas atualizações para etapas de treinamento paralelas o mais rápido possível. O fluxo de dados com estado mutável permite que o TensorFlow imite a funcionalidade de um servidor de parâmetros, mas com flexibilidade adicional, porque torna-se possível executar subgráficos de fluxo de dados arbitrários nas máquinas que hospedam os parâmetros do modelo compartilhado. Como resultado, nossos usuários puderam experimentar diferentes algoritmos de otimização, esquemas de consistência e estratégias de paralelização.

Ainda segundo ABADI, Martín; BARHAM, Paul; CHEN, Jianmin; et al (2016) a principal observação na arquitetura do servidor de parâmetros é que o estado mutável é crucial ao treinar modelos muito grandes, porque torna possível fazer atualizações locais para parâmetros muito grandes e propagar essas atualizações para etapas de treinamento paralelas o mais rápido possível. O fluxo de dados com estado mutável permite que o TensorFlow imite a funcionalidade de um servidor de parâmetros, mas com flexibilidade adicional, porque torna-se possível executar subgráficos de fluxo de dados arbitrários nas máquinas que hospedam os parâmetros do modelo compartilhado. Como resultado, nossos usuários puderam experimentar diferentes algoritmos de otimização, esquemas de consistência e estratégias de paralelização.

O TensorFlow usa um gráfico de fluxo de dados unificado para representar a computação em um algoritmo e o estado em que o algoritmo opera. Inspiramo-nos

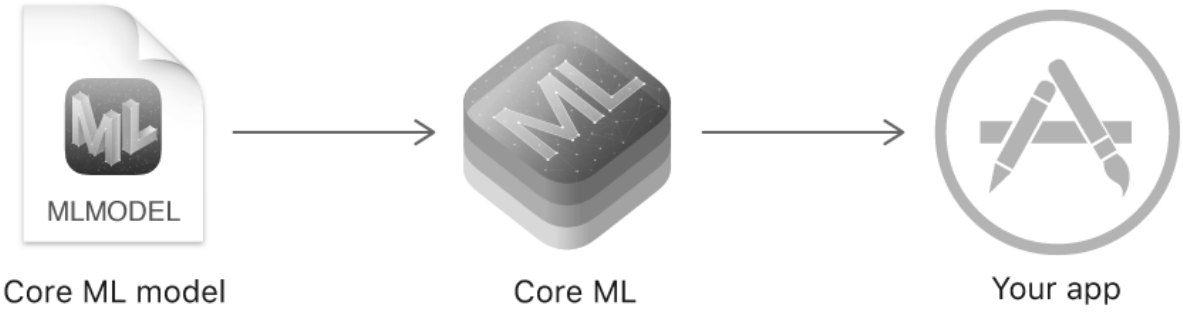
nos modelos de programação de alto nível dos sistemas de fluxo de dados e na eficiência de baixo nível dos servidores de parâmetros (ABADI; BARHAM; CHEN; ET AL, 2016). Diferentemente dos sistemas tradicionais de fluxo de dados, nos quais os vértices do gráfico representam computação funcional em dados imutáveis, o TensorFlow permite que os vértices representem os cálculos que possuem ou atualizam o estado mutável. As arestas transportam tensores (matrizes multidimensionais) entre os nós e o TensorFlow insere de forma transparente a comunicação apropriada entre subcomputações distribuídas. Ao unificar o gerenciamento de computação e estado em um único modelo de programação, o TensorFlow permite que os programadores experimentem diferentes esquemas de paralelização que, por exemplo, descarregam a computação nos servidores que mantêm o estado compartilhado para reduzir a quantidade de tráfego de rede. Também construímos vários protocolos de coordenação e alcançamos resultados encorajadores com replicação síncrona, ecoando resultados recentes que contradizem a crença comum de que a replicação assíncrona é necessária para o aprendizado escalável.

2.5 *CORE ML*

O *Core ML* é uma solução para os desenvolvedores poderem trabalhar com Machine Learning diretamente no celular. Esta solução foi desenvolvida pela Apple (2017) para ser utilizada nos dispositivos com o sistema operacional iOS. Lançado em 2017 junto com o iOS 11, até a criação deste projeto, o *Core ML* tem três versões. A versão 2 foi lançada em 2018 e em 2019 foi lançado a terceira versão desta ferramenta.

Dennis Hills (2018) explica que a estrutura do *Core ML* é dividida em três partes, onde inicialmente temos o modelo do *Core ML*, no qual se encontra a codificação da problemática que ele irá tratar, seguindo pelo núcleo, que gerencia a codificação e o aplicativo que agrupa o modelo e executa o que foi proposto pelo desenvolvedor. Como mostrado na Figura 3.

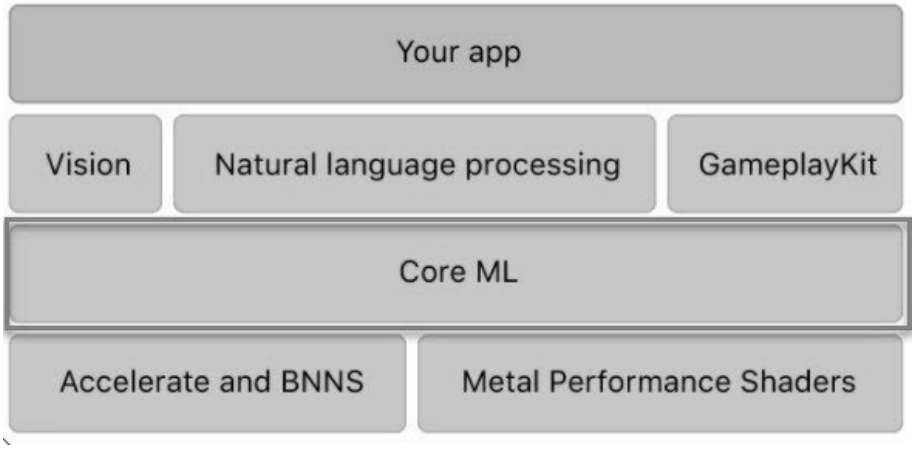
Figura 3: Estrutura do Core ML.



Fonte: Apple (2017)

Ela funciona como uma formar de integração com o *Vision Framework*, que é um Framework voltado para o reconhecimento facial utilizando smartphones. Como veremos na Figura 4 o *Core ML* e o *Vision* interagem diretamente com a aplicação, sem a necessidade de compilar em um servidor web separado do celular, com isso facilita a comunicação, pois, tudo está no mesmo dispositivo.

Figura 4: Estrutura do *Core ML* com *Frameworks* do smartphone.



Fonte: Dennis Hills (2019)

2.6 FERRAMENTAS PARA DESENVOLVIMENTO

O desenvolvimento deste projeto foi possível por meio de algumas ferramentas computacionais que facilitaram a finalização deste projeto. São ferramentas chaves para a criação de softwares para dispositivos eletrônicos modernos.

2.5.1 *Tf-coreml*

WADHWA, Aseem; LIN, Allen; LING, Yu-Cheng; et al (2019) desenvolveram o *tf-coreml*, uma ferramenta capaz de converter TensorFlow para CoreML. O projeto foi iniciado em 2017, junto com o lançamento do CoreML (APPLE, 2017), pela necessidade de mercado de ter uma ferramenta que converte TensoFlow, já disponível no mercado e com aplicações com bons resultados comprovados, em CoreML.

O *tf-coreml* foi desenvolvido na linguagem de programação Python, necessitando somente de uma rede neural já criada e pré treinada, para criar um *script* para conversão. Atualmente, o *tfcoreml* está na versão 3, acompanhando o desenvolvimento do CoreML.

Para converter um rede em TensoFlow para CoreML é necessário um script que recebe os seguintes parâmetro:

- *tf_model_path*: local onde o modelo da rede neural se encontra.
- *mlmodel_path*: caminho do local e nome de onde será salvo o modelo em CoreML.
- *input_name_shape_dict*: item que informa os elementos de entrada da rede neural.
- *output_feature_names*: semelhante ao *input_name_shape_dict*, porém define os elementos de saída da rede.
- *use_coreml_3*: para que seja reconhecida a versão 3 do CoreML, sem isso, só será aplicada até a versão 2.

2.5.2 *Xcode*

O Xcode é uma plataforma de desenvolvimento para produtos da Apple. Nele, é possível desenvolver soluções para iOS, iPadOS, macOS e também para aplicações de realidade aumentada e jogos digitais (JACOBS, 2019). O seu uso é exclusivo em dispositivos com o sistema operacional macOS, sendo impossível utilização em qualquer outro sistema operacional existente atualmente, pelo fato que o Kit de Desenvolvimento de Software (Software Developer Kit - SDK) é proprietário da Apple e incompatível com outros sistemas.

3 FACENET

A FaceNet é uma rede neural convolucional criada pela equipe de desenvolvimento do google (SCHROFF, 2015). Esta rede utiliza técnicas de *deep learning* na sua estrutura, e manuseia abstrações de alto nível de dados para modelar e com um grafo profundo, várias camadas de processamento se aplica os dados na qual está trabalhando, dando a ela um precisão grande com relação as outros redes.

Um item peculiar na FaceNet é a estrutura desta rede com relação a outras redes. A estrutura desta rede é dividida segundo Karunakaran (2019) da seguinte forma: primeiro temos a camada de entrada do lote (*batch*), segundo a entrada de dados temos na camada de CNN ou *Deep Architecture*, a próxima camada é a de *pooling* L2, que resulta na camada de *face embedding* da rede, e por fim, temos um dos maiores diferencias desta rede, que é a *Triplet Loss*, ou Perda Tripa. A estrutura da rede está contida na Figura 5.

Figura 5: Estrutura da FaceNet

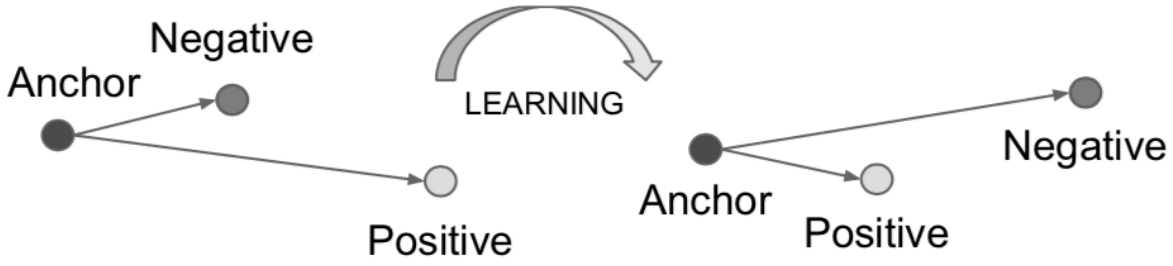
Fonte: Schroff (2015)

A parte mais importante para esta abordagem está na aprendizagem ponta-a-ponta de todo o sistema. Para que isso seja possível ele utiliza a *Triple Loss* que, segundo Karunakaran (2018), reflete diretamente no que se quer alcançar com a rede, sendo a verificação de face, reconhecimento e agrupamento, sendo os principais ponto de funcionamento da *FaceNet*. Sobre a *Triple Loss*, esta técnica é aplicada para que a FaceNet aprenda como é realizado o mapeamento, ela é uma adaptação de classificação de *Kilian Weinberger's Large Margin Nearest Neighbor* (LMNN) desenvolvida pelo Weinberger (2006), ela reúne repetidamente imagens da

mesma pessoa e simultaneamente afasta imagens de qualquer pessoa diferente, e esta lógica foi aplicada na *FaceNet*.

A *Triple loss* funciona da seguinte forma, ele minimiza a distância entre uma âncora para um ponto positivo e maximiza a distancia entre a ancora e o ponto negativo. Queremos fazer a distância entre vetores igual. Para que fotos da mesma pessoa estejam próximas e fotos de outras pessoas estejam distantes. Isso pode ser expresso como perda (*loss*). Imagem tem o ponto positivo que são dados da mesma pessoa, e o ponto negativo, que é a imagem de outras pessoas. Chama-se *Triplet Loss* porque usa um total de três fotos (SCHOFF, 2015). Matematicamente, a rede se esforça para incorporar o *f(x)* de uma imagem qualquer, em um espaço R^d, de modo que a distancia do tamanho do quadrado agrupado, independente da condição da imagem, da mesma identidade é pequeno, e a distancia do quadrado de agrupamento entre um par de imagens faciais de diferentes identidades é maior. Na Figura 6 temos a representação da *Triple Loss*.

Figura 6: Representação da *Triple Boss*



Fonte: Schroff (2015)

O *embedding* é representado por $f(x) \in R^d$, contendo uma imagem X em um espaço euclidiano de d dimensões. Para garantir que uma imagem x_i^a de uma pessoa esta próxima de todos as imagem x_i^p da mesma pessoa na base de treinamento, do que da imagem x_i^n de outra pessoas, se utiliza a seguinte formula:

$$\|f(x_i^a) - f(x_i^p)\|^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|^2 \,, \forall (f(x^a), f(x^p), f(x^n)) \in T \,.$$

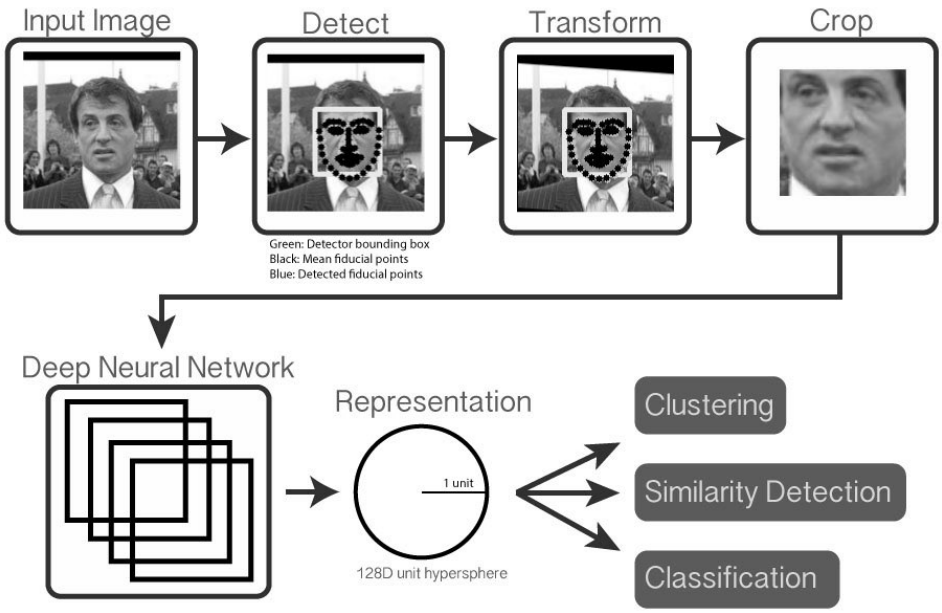
Onde o valor de α é uma margem que é aplicada entre pares positivos e negativos, e T é o conjunto de todos os *triplets* possíveis no conjunto de treinamento.

A seleção do *Triplet* é de extrema importância para o reconhecimento bem sucedido da rede. Schroff (2015) informam em seu artigo que ainda não é possível computar os pontos positivos e negativos de todos os rostos em todo o conjunto de treinamento, e isso pode levar a um treinamento mal sucedido já que não é possível identificar em sua totalidade a face. E existem duas formas de amenizar este problema:

- Fazer a geração dos *Triplet* off-line a cada n etapas, usando o ponto de verificação de rede mais recente e calculando o *argmin* e *argmax* em um subconjunto dos dados
- Ou gerar os *Triplet* online, e isto pode ser feito selecionando os exemplos positivos e negativos e os separando dentro de um mini-lote para facilitar o processamento.

Para utilização da rede no projeto foi aplicado uma rede já pronta. O projeto escolhido foi desenvolvido pelo David Sandberg (2019). Ele desenvolveu utilizando como inspiração a OpenFace, que é outra rede neural com fins muito próximos, onde, ela também utiliza uma rede neural profunda, como demonstrado na Figura 7.

Figura 7: Representação da OpenFace.

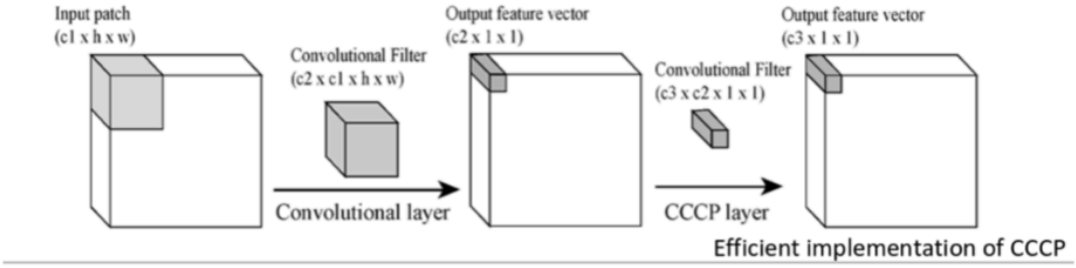


Fonte: Amos (2019)

Para utilização desta rede já desenvolvida, foi usado o *Core ML* da Apple no smartphone a fim de facilitar a comunicação com o dispositivo. Como a FaceNet foi desenvolvida pelo David Sandberg em *Python,* foi utilizado um conversor para converter a FaceNet em Python em um tipo que o dispositivo tenha facilidade de comunicação. Para conversão foi utilizado o *tfcoreml*, que é uma ferramenta de código aberto, mantida por uma comunidade de desenvolvedores.

O arquitetura CNN utilizado na FaceNet. Recebe como entrada uma imagem de tamanho 240x240 que é processado na camada convolucional, passando por inúmeras camadas, como mostrado na Figura 8.

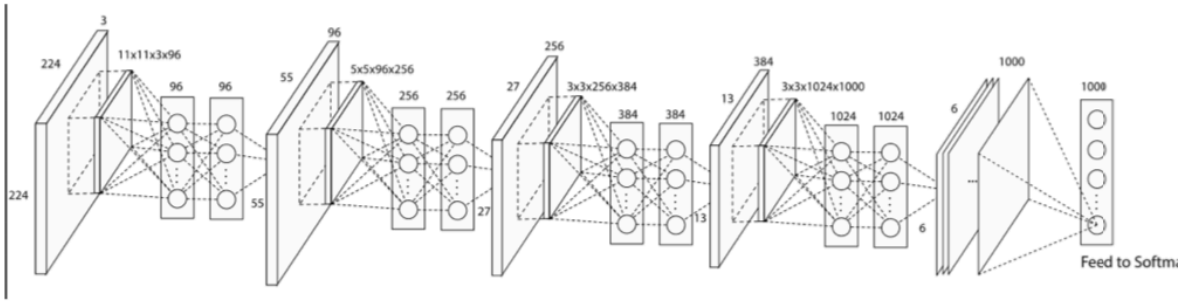
Figura 8: Representação de entradas e saídas da CNN



FONTE: Bombardelli, 2015

Para cada camada de convolução da rede, é processado a imagem de entrada em cada camada e este é o diferencial desta rede, sendo processado inúmeras vezes por ter um numero muito grande de camadas convolucionais por ser de aprendizagem profunda, como mostrada na Figura 9.

Figura 9: Modelo da CNN da FaceNet



Fonte: Bombardelli, 2015

A FaceNet utiliza o algoritmo de classificação Stochastic Gradient Descent (SGD) no processo de treinamento da rede. Para Léon Bottou (2010) a SGD é uma simplificação drástica com relação aos outras CNNs do mesmo tipo. A SGD em vez de calcular exatamente o gradiente em En(fw), em cada iteração estima esse gradiente com base em um único exemplo zt escolhido aleatoriamente, pela formula:

$$wt+1 = wt - \gamma t \nabla w\, Q(zt, wt)$$

Em seus testes o Schroff também utilizou, a nível de comparação, a classificação da rede por LMNN, que é um método de estático de ML. Huyen (2012) define LMNN como um algoritmo muito popular para ML. Ele aprende em utilizando de pseudométricas projetadas para classificação de vizinhança pelos k-vizinhos mais próximos. Para Killian (2005) LMNN é um framework que não requer modificações para problemas na classificação de múltiplas vias, em comparação a *Support Vector Machines* (SVM), que para problemas de várias classes, geralmente envolvem a combinação dos resultados de muitos classificadores binários, ou requerem máquinas adicionais que sejam não triviais. O LMNN pode ser visto como a contrapartida lógica das SVMs nos quais a classificação K-nearest neighbor, ou KNN, substitui a classificação linear.

No seu trabalho, Schroff, Kelenichenko e Philbin (2015) descrevem a utilização do *pooling* como sendo as duas principais diferenças são o uso do *pooling* L2 em vez do pool máximo (m), onde especificado, com relação a outros trabalhos. O *pooling* é sempre 3 × 3, além do *pooling* médio final, e em paralelo aos módulos convolucionais dentro de cada módulo de Iniciação. Se houver uma redução de dimensionalidade após o agrupamento, é indicado com p. Os agrupamentos 1 × 1, 3 × 3 e 5 × 5 são concatenados para obter a saída final. Eles utilizando duas arquiteturas no projeto, para nível de comparação entre elas, sendo a primeira arquitetura consistindo em várias camadas intercaladas de convoluções, ativações não lineares, normalizações de resposta local e camadas máximas de pool. Adicionaram adicionalmente várias camadas de convolução 1 × 1 × d. A segunda arquitetura é baseada no modelo Inception que foi usado recentemente como a abordagem vencedora para o ImageNet, e utilizada no desenvolvimento deste projeto.

3.1 PROJETOS RELACIONADOS

Araujo (2010), desenvolveu em seu mestrado um algoritmo de reconhecimento facial que utiliza rede neural convolucional, não sendo utilizada a FaceNet, mas um propósito muito parecido. Tendo como diferencial o uso de filtros de correlação para se basear no reconhecimento. Ele utiliza os pontos salientes do rosto, como os cantos do olho, boca, nariz, para realizar o reconhecimento, se valendo deles como pontos de controle. O sistema pro ele proposto é dividido em cinco etapas: Segmentação, Correção de iluminação, Classificação, Detecção e Agrupamento. Constituindo-se um algoritmo muito completo e robusto, pecando somente na usabilidade. Sendo necessário para o entendimento de como são os pontos salientes do rosto para poder utilizar na FaceNet, que foi desenvolvida posteriormente.

Silva e Cintra (2015), fizeram um estudo sobre reconhecimento de padrões faciais. Eles declaram que o reconhecimento é utilizado para identificações e autenticações de pessoas, tendo como um dos principais mercados a segurança. Eles dividem o reconhecimento facial em três abordagens, sendo elas Método Holísticos, Geométricos e Híbridos. Para o desenvolvimento do projeto aqui proposto, será utilizada uma abordagem Holísticos, que busca entender de forma integral o problema proposto, fazendo uso de uma rede neural artificial profunda no seu desenvolvimento.

Moacir e Gabriel (2017) em seu artigo explica que os métodos que são utilizados na *deep learning* buscando definir uma série de regras e parâmetros para com isso utilizá-los com um conjunto de dados, e um método para guiar o aprendizado do modelo tendo este conjunto de dados. Ele explica que a d*eep learning* é um técnica bem aplicada para por exemplo em classificação de imagens, detecção de anomalias em sinais de fala, isso de dar pelo fato do modelo que a Depp Learning foi projetada, sendo que ela utiliza um conjunto de dados, no caso um conjunto de exemplos, e um método para definir como será feito a aprendizagem a partir destes exemplos de entrada.

Charles F Jekel e Raphael T Haftka (2018), utilizaram a FaceNet para classificação de encontros online no Tinder. A proposta é um modelo de classificação personalizada com base no histórico do usuário da ferramenta Tinder. Com base no histórico do usuário e junto com a FaceNet é criado um modelo feito por cálculos pro-

babilísticos onde, define se uma próxima pessoa, numa lista de possíveis pessoas para um encontro, é adequada ou não para o usuário da ferramenta. Num exemplo citado no projeto, foram revisados 8.545 perfis de usuários do aplicativo de relacionamento Tinder. Onde para cada perfil revisado, um conjunto de recursos foi construído a partir das imagens do perfil no Tinder, e duas abordagens são apresentadas para ir do conjunto de recursos de cada face a um conjunto de recursos de perfil, sendo eles, primeiro abordagem é a classificação do rosto feita para identificar o rosto, e a segunda abordem é a utilização da FaceNet quanto modelo de classificação destes rostos. O modelo de reconhecimento fácil é o bastante amplo, e pode ter várias utilizações. Baseado na pesquisa desenvolvida por Jekel e Haftka este estudo tem uma finalidade com base parecidas, onde é realizado o reconhecimento facial humano, porém sem aplicação do reconhecimento para um fim de relacionamento romântico e sim para demonstração da possibilidade do reconhecimento.

4 APLICAÇÃO PARA DISPOSITIVOS iOS

No desenvolvimento deste projeto foi utilizada a FaceNet implementada por Ning Lu (2019). Foi utilizado um modelo da Facenet para dispositivos móveis para reconhecimento facial. Para isso, foi utilizada a *tf-coreml*, que é uma ferramenta de conversão recomendada pela Apple na documentação do CoreML. Na utilização desta ferramenta, foi gerado um script para ser utilizada. Como mostrado na Figura 10.

Figura 10: Script de conversão da rede em CoreML

```
import tfcoreml as tf_converter

tf_converter.convert(tf_model_path='my_model.pb',
                     mlmodel_path='my_model.mlmodel',
                     output_feature_names=['softmax'],
                     input_name_shape_dict={'input': [1, 227, 227, 3]},
                     use_coreml_3=True)
```

Fonte: Autoria do Autor

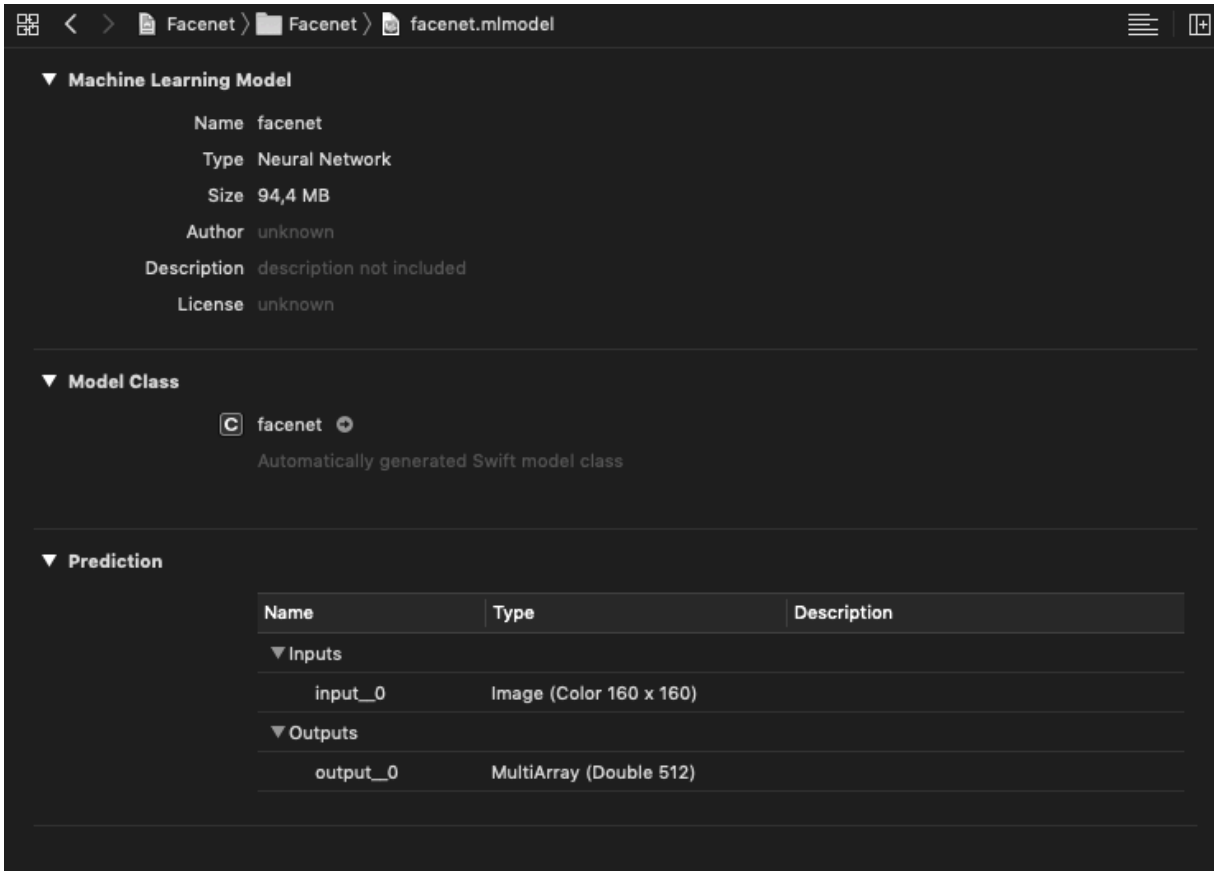
A Figura 10 mostra primeiro a importação da ferramenta *tf-coreml* para posteriormente realizar a conversão do *script*. O script se dá pelo método *convert* que está implementado na *tf-coreml.* Este método recebe como parâmetros básicos dois caminhos, sendo eles o *tf_model_path* e *mlmodel_path*. O *tf_model_path* é o local onde o modelo da rede a ser convertida está, como ele está no mesmo nível de pasta no exemplo da Figura 10, foi colocado somente o nome. Já o *mlmodel_path* é o local e nome de onde será gerado o modelo CoreML que será utilizada pela aplicação implementada para iOS.

Um método também utilizado é o *use_coreml_3*, neste caso ele é utilizado para que a rede gerada tenha a possibilidade de ser utilizada pelo *CoreML 3*, sendo a versão lançada em 2019 pela Apple, trazendo uma série de melhorias para o desenvolvedor, como exemplo, no *CoreML 3* existe a possibilidade de criar uma rede sem a necessidade de conversão.

Outros dois parâmetros utilizados são o *output_feature_names* e o *input_name_shape_dict* que são utilizados para definir as saídas e entradas da rede quando

convertida para *CoreML*, este item é mostrado no *CoreML* pelo *XCode* quando utilizado pelo aplicativo que foi desenvolvido, como mostrado na Figura 11.

Figura 11: CoreML depois de ser importado para o XCode



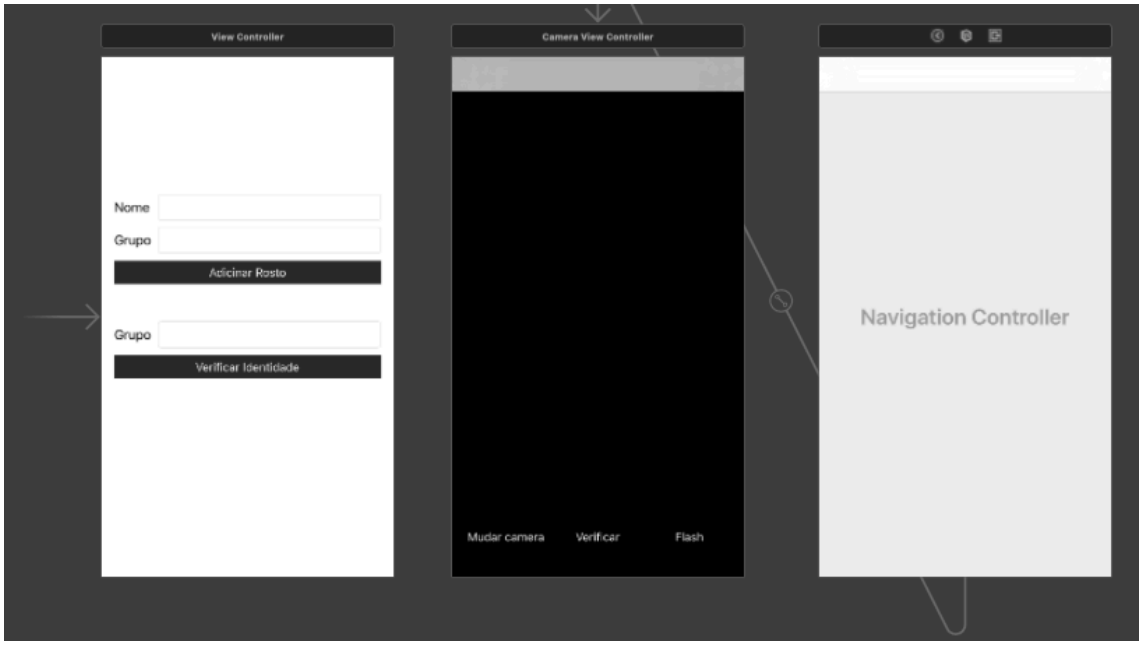
Fonte: Autoria do Autor

Antes de realizar a conversão da rede para o modelo utilizado no CoreML é necessário realizar um treinamento na rede que será utilizada, pois, como o CoreML compila diretamente no celular, que normalmente tem um poder computacional mais baixo que o de um computador, a rede tem que ser previamente treinada para que possa ser utilizada no smartphone. O treinamento prévio da rede que é utilizada é feito para diminuir a demanda de processamento do celular, pois o treinamento é um item que necessita de um maior poder computacional, poder este que o smartphone ainda não tem, e para evitar que o celular não consiga processar o treinamento, o Core ML por padrão só aceita uma rede já treinada, como descreve Apple (2019) na documentação.

4.1 Desenvolvimento do Aplicativo para iOS

O desenvolvimento do aplicativo foi feito no Xcode, utilizando o Swift para programação do aplicativo. No Xcode o aplicativo é dividido em dois pontos, os códigos e as telas. O algoritmo foi feito em Swift e as telas são feitas em *.storyboard*, que é o fluxo das telas que o aplicativo vai ter.

Figura 12: StoryBoard da aplicação



Fonte: Autoria do Autor

Para este projeto foram criados três telas, mostradas na Figura 12, onde, a primeira é a tela inicial para adicionar um rosto utilizado para verificação, a segunda para verificação por meio da câmera, e a terceira, caso sucesso na autenticação do rosto é mostrado os álbuns de fotos do usuário.

A segunda parte no desenvolvimento para iOS, é a construção dos algoritmo para o controle do aplicativo. O fluxo de criação foi padrão do Swift, com a declaração dos botões e caixas de textos presentes nas telas. Para a utilização da foto por meio da rede, é feito a conversão da foto para Base64, mostrado na Figura 13, para assim ser feito o reconhecimento do rosto.

Figura 13: Conversão da base da imagem

```
func enroll(imageBase64: String) {
    let params : NSMutableDictionary? = [
        "image" : imageBase64,
        "subject_id" : idEnrollTextfield.text!
    ]

    let data = try! JSONSerialization.data(withJSONObject: params!, options:
        JSONSerialization.WritingOptions.prettyPrinted)

    let json = NSString(data: data, encoding: String.Encoding.utf8.rawValue)
    request.httpBody = json!.data(using: String.Encoding.utf8.rawValue);
}
```

Fonte: Autoria do Autor

Para o reconhecimento, é feito o controle da câmera e é feito o reconhecimento sem ter a necessidade de que a foto para reconhecer seja salva pelo aplicativo, é pega a foto do rosto e processado pelo CoreML o qual retorna se teve o reconhecimento ou não do rosto cadastrado com o rosto para verificação. Mostrado na Figura 14.

Figura 14: Controle da câmera para reconhecimento

```
@IBAction func recognizeButtonAction(_ sender: Any) {
    let storyBoard : UIStoryboard = UIStoryboard(name: "Main", bundle:nil)
    let cameraController = storyBoard.instantiateViewController(withIdentifier: "camera_view") as! CameraViewController
    cameraController.receivedGalleryName = galleryRecognizeTextField.text!
    self.present(cameraController, animated:true, completion:nil)
}
```

Fonte: Autoria do Autor

Com a câmera pronto para o reconhecimento, o próximo ponto é a entrada na rede, a entrada da rede é a imagem do rosto para verificação, mostrado na Figura 15, com isso é processado pelo CoreML a foto e retorna um bool com a informação de reconhecimento feito com sucesso ou não e no caso de reconhecimento, também é retornado o nome do rosto reconhecido.

Figura 15: Entrada da rede

```
let params : NSMutableDictionary? = [
    "image" : imageBase64,
    ]

let data = try! JSONSerialization.data(withJSONObject: params!, options:
    JSONSerialization.WritingOptions.prettyPrinted)

let json = NSString(data: data, encoding: String.Encoding.utf8.rawValue)
request.httpBody = json!.data(using: String.Encoding.utf8.rawValue);
```

Fonte: Autoria do Autor

5 RESULTADO E DISCUSSÃO

Para o experimento, foram catalogadas fotos de rostos humanos de pessoas em boa condição de iluminação, onde, a foto não tinha muito brilho, muito menos estava muito escura. A foto do rosto para catalogação foi feita de modo que, o rosto esteja centralizado e sem obstáculos ou qualquer outro item na foto. As fotos foram tiradas em um iPhone 7 com o aplicativo instalado.

Ao abrir a aplicação, o usuário é levado a galeria de fotos do smartphone onde pode escolher a foto do rosto em questão para adicionar. Após adicionar um rosto com o nome no aplicativo, a tela é alterado para a opção de reconhecimento, Figura 16.

Figura 16: Tela com botão para verificar rosto

Fonte: Autoria do Autor

O botão Verificar Identidade, mostrado na Figura 16, leva o usuário para a câmera onde realiza o reconhecimento do rosto ao clicar em Verificar, como mostra

na Figura 17. Neste momento, ao clicar em Verificar, a rede neural FaceNet é ativado pelo CoreML e é verificado ser o rosto já foi catalogado e o identifica se já tiver catalogado.

Figura 17: Tela de verificação

Fonte: Autoria do Autor

O aplicativo foi testado com 5 pessoas, entre elas homens cis, mulheres cis, e teve o re-treinamento com 90 rostos. Somente um não obteve sucesso no reconhecimento, pelo fato da foto de amostragem para o aplicativo ter sido tirada em uma condição ruim de reconhecimento, onde não era possível ver o rosto.

Figura 18: Imagem de rosto sem identificação

Fonte: Autoria do Autor

A foto, representada na Figura 18, foi a foto na qual não foi possível reconhecer o rosto, pois não está claro o rosto para o processamento da rede neural identificar o mesmo.

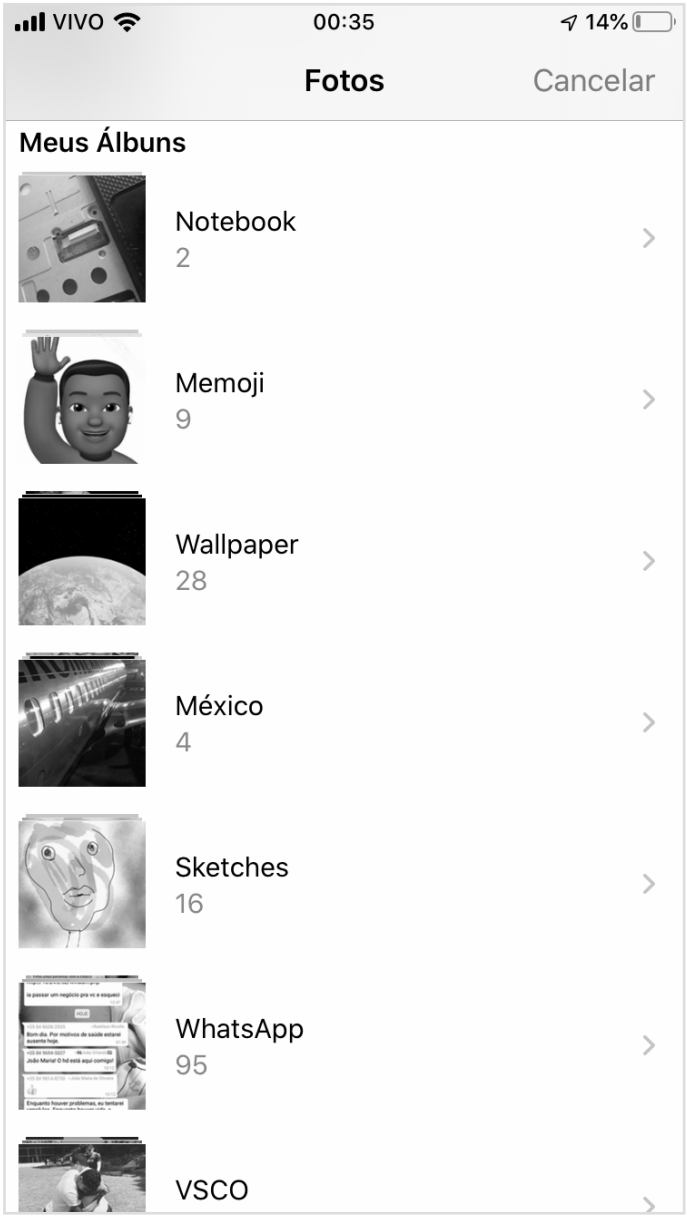
A verificação do rosto define quem é a pessoa que estava olhando para a câmera no momento da verificação. Se o rosto for reconhecido, a autenticação é realizada para o acesso a fotos. Com isso, é necessário realizar o cadastro do rosto de somente da pessoa que venha a ter o acesso das fotos do celular, pois, todos os rostos cadastrados darão tal acesso.

Seguindo a tela de verificação, mostrada na Figura 17. Temos finalmente o acesso as fotos armazenadas no celular do usuário. A autenticação foi necessária para que seja disponível esta nova tela. Se o rosto verificado não estiver cadastrado,

o acesso não será concedido, até que uma nova consulta de um rosto cadastrado seja feito e reconhecido.

Na Figura 19, é mostrada a tela escondida pela autenticação, sendo a tela de álbuns de fotos do usuário.

Figura 19: Tela com álbuns de fotos



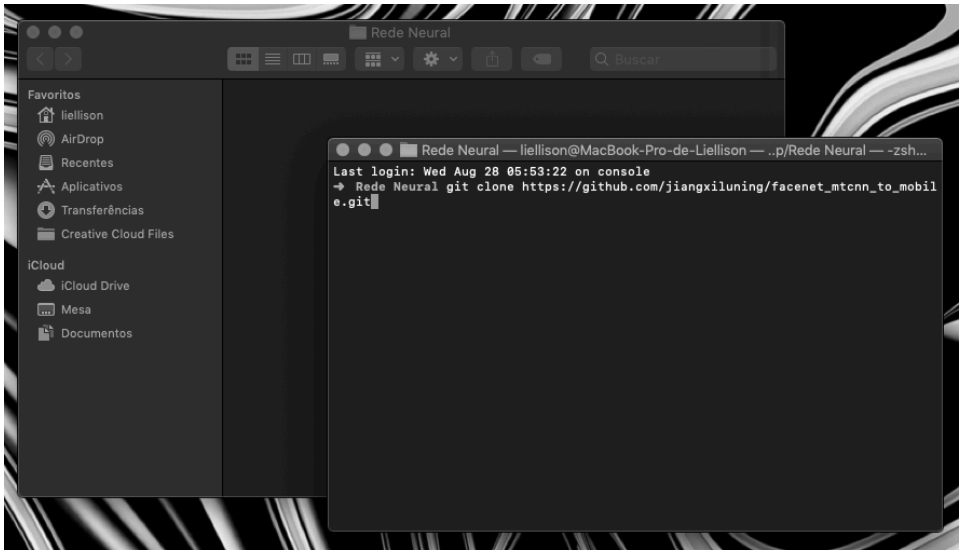
Fonte: Autoria do Autor

5.1 Treinamento da rede neural

O treinamento prévio da CNN é necessário para o CoreML pode agilizar o processamento que será utilizado no smartphone e também pelo fato de que um dos parâmetro usado na conversão é o *tf_model_path* que é um arquivo em *.pb* que é gerado somente depois de ter feito um trenamento na rede.

Para realizar o treinamento da rede que será utilizada, é necessário primeiro que a rede esteja no computador localmente. A CNN que foi utilizada neste projeto, foi desenvolvida por Ning Lu (2019), a FaceNet está disponível em um repositório online, para baixar tal repositório é necessário criar uma pasta no computador para o armazenamento e ter, neste caso, a ferramenta de gerenciamento do repositório para facilitar o armazenamento no computador, o utilizado foi o *Git*. Os procedimentos foram realizados em uma máquina com o sistema operacional MacOS, como informado na metodologia. Na Figura 20, temos a pasta vazia para utilização do repositório onde está a CNN e o demonstrativo do comando no terminal para baixar o repositório onde se localiza a CNN que foi utilizada.

Figura 20: Pasta com terminal para baixar o repositório



Fonte: Autoria do Autor

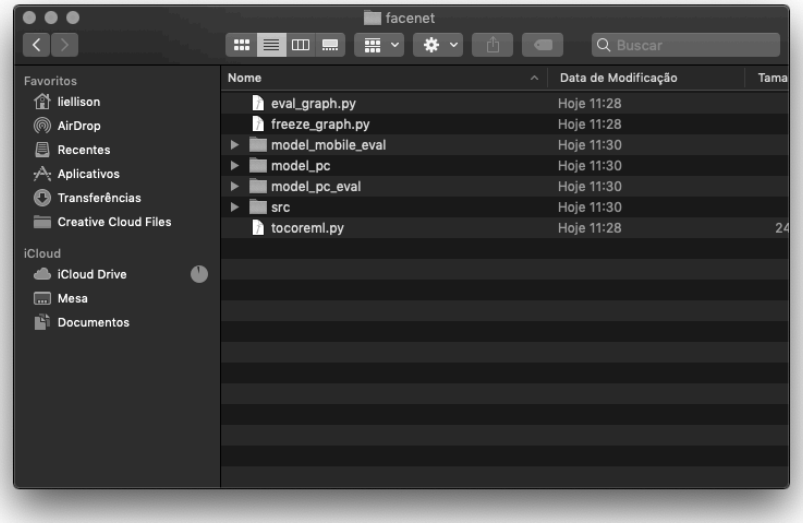
O segundo item a ser realizado depois de ter baixado o repositório para o computador, é entrar na pasta criada com o nome de *facenet_mtcnn_to_mobile* e abrir o terminal para serem executados os comandos para conversão da rede em

TensoFlow para um modelo em CoreML. No mesmo diretório *facenet_mtcnn_to_-mobile* tem o projeto criado da CNN para utilização, que contém os seguintes itens:

- Pasta com o titulo de *facenet*, que contém os algoritmos necessários para a execução da rede.
- Um arquivo *LICENSE.md* que demonstra a licença de uso da rede.
- Pasta com o titulo de *mtcnn*, que é outra CNN diferente da Facenet, mas que neste repositório, tem a possibilidade de utilizar também.
- Os itens *Pipfile* e *Pipfile.lock* que são responsáveis pelo gerenciamento de dependências da rede.
- E por ultimo o *README.md*, que é uma descrição do repositório criada pelo autor.

Como a CNN escolhida foi a FaceNet, é necessário ir até o diretório *facenet*, que contém alguns arquivos em *Python* que são os algoritmos desenvolvidos para a CNN, e treinar a rede. Porém, como é utilizado somente para a conversão para *CoreML*, é necessário realizar uma sequência de comandos no terminal para realizar o treinamento da rede. A lista de arquivos e pastas que tem disponível na pasta de *facenet* pode ser visto na Figura 21.

Figura 21: Pasta referente aos arquivos da CNN FaceNet



Fonte: Autoria do Autor

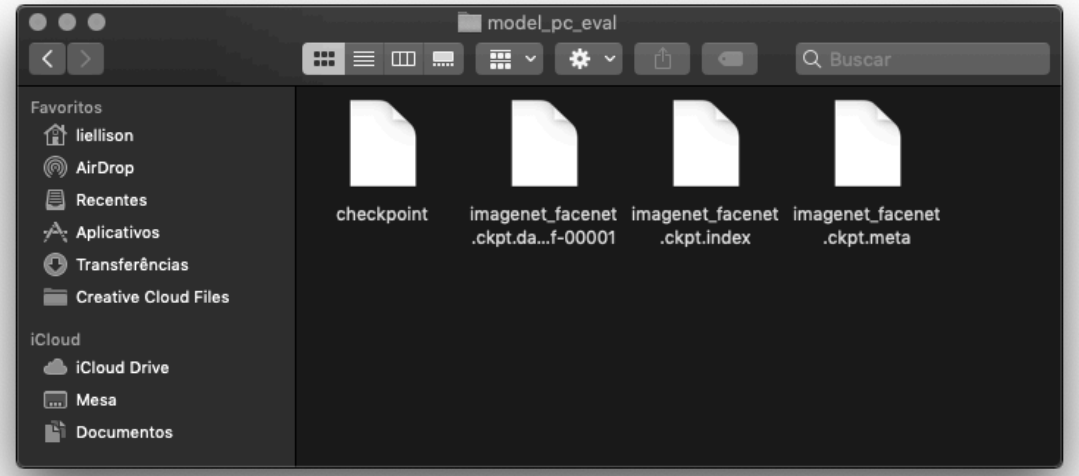
Para o treinamento da rede foi utilizada uma base de imagens criada pelo CASIA-WebFace (Sheng; et al, 2018), que fez uma coleta de rostos autorizados de mais de 10 mil rostos de humanos, coletados em várias ângulos diferentes. A base

de treinamento tem que ser colocada na pasta com o nome de *model_pc* para que possa ser utilizada pelo sistema. Este banco de dados de imagens tem a seguinte estrutura (Shi e Raoul, 2019), sendo cada foto com o nome do rosto, para assim treinar a rede a reconhecer as pessoas:

…
Aaron_Eckhart
Aaron_Eckhart_0001.jpg
Aaron_Guiel
Aaron_Guiel_0001.jpg
Aaron_Patterson
Aaron_Patterson_0001.jpg
Aaron_Peirsol
Aaron_Peirsol_0001.jpg
Aaron_Peirsol_0002.jpg
Aaron_Peirsol_0003.jpg
Aaron_Peirsol_0004.jpg
...

O primeiro algoritmo que é utilizado para o treinamento é o *eval_graph.py*, ele vai ler a base de dados e colocará na pasta *model_pc_eval*. Para executar este procedimento é necessário ir no terminal e digitar o comando *python eval_graph.py model_pc model_pc_eval*. Com isso a base de dados que estava na pasta *model_pc* foi utilizada pelo *eval_graph.py* e o resultado foi colocado na pasta *model_pc_eval*. Com a finalização desse procedimento a pasta *model_pc_eval* deve estar como mostrado na Figura 22.

Figura 22: Pasta *model_pc_eval* atualizada.

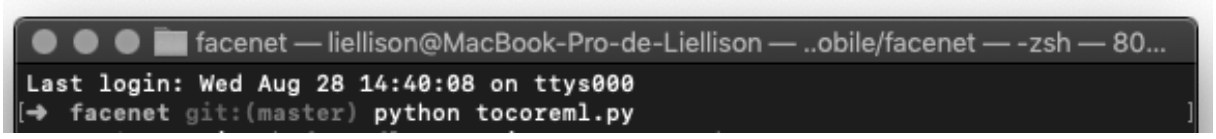


Fonte: Autoria do Autor

Com a pasta *model_pc_eval* atualizada, o próximo item a ser realizado ainda no terminal, é o comando *python freeze_graph.py model_pc_eval facenet.pb*. Nele é criado o arquivo *facenet.pb*, que será utilizado pelo *TensorFlow* para fazer a conversão para que ele possa ser utilizado pelo *TensorFlow Lite* e posteriormente pelo *CoreML*. Para isso é realizado o comando *tflite_convert --output_file model_mobile_eval/facenet.tflite --graph_def_file facenet.pb --input_arrays "input" --input_shapes "1,160,160,3" --output_arrays output --output_format TFLITE* que é responsável por converter a rede já treinada em um modelo *TensorFlow Lite*.

O último passo é a geração de um modelo que o *CoreML* possa utilizar, sendo um arquivo com extensão *.ml*. Para isso é utilizado o comando *python tocoreml.py* no terminal. Na Figura 23 a seguir é exemplificado este procedimento. O Arquivo *tocoreml.py* contém o script necessário para a conversão para o modelo *CoreML*

Figura 23: Comando para conversão da rede em modelo CoreML



Fonte: Autoria do Autor

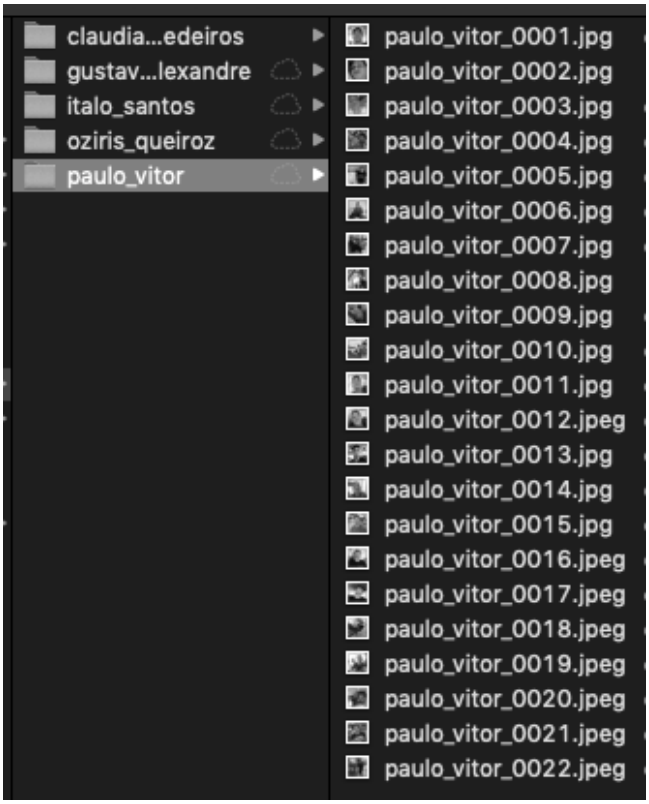
Com a finalização destes procedimentos, um arquivo *facenet.mlmodel* foi gerado. Este arquivo já está em um modelo que o iOS pode utilizar. Para usar deste

arquivo no projeto em iOS é necessário somente importar o arquivo *.mlmodel* na ferramenta *Xcode* para assim poder utilizar o CoreML no projeto em iOS.

5.1.1 Banco de Imagens

Seguindo o treinamento utilizando a base de dados da Casia, foi realizado também um treinamento da rede utilizando uma base de imagens própria. Esta base de imagem consta 5 classes com 90 imagens com tamanho de 240x240 e com resolução de 400px. Nos dois treinamento foi obtido um resultado da precisão do treinamento, utilizando a base de dados da Casia, a precisão foi de 1.098, já com a base de imagens criada para este projeto a precisão ficou em 0.976. Na Figura 24 é mostrado a estrutura da base de imagens utilizada para o treinamento da rede.

Figura 24: Base de imagens para treinamento



Fonte: Autoria do Autor

Com o treinamento realizado foi gerado dois arquivos em Core ML e testado no smartphone o reconhecimento das imagens. Foi testado 20 vezes o reconhecimento com cada base de treinamento, na primeira base de treinamento o resultado foi de 18 acertos contra 2 momentos em que a foto mostrada para o aplicativo era certa e ele não reconheceu, com a segunda base de treinamento foram 20 acertos no reconhecimento, os testes foram feitos em varias condições de ângulos e ambiente, e com 3 classes em cada base de treinamento.

6 CONCLUSÃO E PERSPECTIVAS DE TRABALHOS FUTUROS

A utilização de uma CNN para reconhecimento de faces humanas é uma área dentro de inteligência artificial que é bastante promissora. Existem vários modelos no qual aplicam tal tecnologia. Como por exemplo, a utilização de reconhecimento facial para segurança, onde se tem uma série grande de opções que podem ser aplicadas. A polícia de diversos locais vem utilizando esta tecnologia para ajudar no combate ao crime urbano.

FaceNet é um rede neural proposta com muito potencial, pois tem a capacidade de aprendizagem muito alta. Onde, seu grande diferencial em relação às outras redes, é a capacidade de reconhecer o humano em todas as fases da idade, onde reconhece uma criança e de acordo com seu desenvolvimento a FaceNet entende as mudanças da face da pessoa e sempre a reconhece.

Uma aplicação da rede que pode ser utilizada é o reconhecimento de grande escala, utilizado pela segurança de um evento e fazendo o reconhecimento de todos os rosto presentes, assim tendo um controle maior sobre os participantes e ajudando a promover uma maior segurança no local.

Também pode ser utilizado para o pagamento em transportes públicos, onde com o reconhecimento facial e uma conta cadastrada, pode realizar o pagamento da passagem somente com a face humana para isso, agilizando assim o método de pagamento.

A tecnologia esta disponível para quem tiver interesse em utilizá-la e neste projeto foi detalhado como usar a rede FaceNet para conhecimento. Sobre a privacidade ou qualquer outro tipo de problema que venha a causar com este tipo de técnica, somente a pessoa que está utilizando a tecnologia pode definir até onde pode ir com tal conhecimento. Levando sempre em conta as leis vigentes em um determinado local e respeitando a privacidade e o individualismo do outro.

A utilização da FaceNet no reconhecimento facial humano para autenticação no smartphone é um modelo de autenticação simples para o usuário, não sendo necessário a utilização de um senha alfanumérica para a autenticação, faz com que o processo seja mais rápido, tendo em vista o tempo para digitar o código e o tempo para o reconhecimento, que é rápido por conta da CoreML que já esta dentro da aplicação. O aplicativo criado neste projeto é uma pratica simplificada de autentica-

ção e de segurança, já que alguém por perto não irá ver a senha digitada e tentara repetir a mesma, pois agora a senha é o rosto.

Como trabalho futuro, é possível realizar o treinamento da Facenet para um conjunto de classes (pessoas) e criar uma aplicação de controle de acesso de um determinado grupo pré definido de pessoas para acesso, sendo estes grupos formados por pai e mãe e um outro grupo com os filhos, onde somente os pai e a mãe tem acesso a um determinado tipo de mídia e os filhos a outro tipo, é possível com o treinamento de todos os membros da família e assim definir qual grupo tem acesso a um tipo de conteúdo do smartphone

REFERÊNCIAS

ABADI, Martín; BARHAM, Paul; CHEN, Jianmin; CHEN, Zhifeng; DAVIS, Andy; DEAN, Jeffrey; DAVIN, Matthieu; GHEMAWAT, Sanjay; IRVING, Geoffrey; ISARD, Michael; KUDLUR, Manjunath; LEVENBERG, Josh; MONGA, Rajat; MOORE, Sherry; MURRAY, Derek G. ; STEINER, Benoit; TUCKER, Paul; VASUDEVAN, Vijay; WARDEN, Pete; WICKE, Martin; YU, Yuan; ZHENG, Xiaoqiang. TensorFlow: A System for Large-Scale Machine Learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016. **USENIX**. Savannah, 2016

AMOS, Brandon. **OpenFace**. Disponível em: <http://cmusatyalab.github.io/openface/>. Acesso em 25 jun. 2019.

APPLE. ***Core ML Integrate machine learning models into your app***. Disponível em:<https://developer.apple.com/documentation/coreml>. Acesso em 01 jun. 2019.

APPLE. ***Introducing Core ML***. Disponível em:<https://developer.apple.com/videos/play/wwdc2017/703/>. Acesso em 08 set. 2019.

ARAUJO, Gabriel Matos. **Algoritmo para reconhecimento de características faciais baseado em filtros de correlação**. Rio de Janeiro, 2010.

BEN-DAVID, Shai; SHALEV-SHWARTZ, Shai. **Understanding Machine Learning: fromtheory to algorithms**. Cambridge, 2014.

BLEWITT, Alex. **Kotlin, o "Java Melhorado" da JetBrains, agora é open source**. Disponível em: <https://www.infoq.com/br/news/2012/03/kotlin-opensourced/>. Acesso em 25 abr. 2019.

BOTTOU, Léon. **Large-Scale Machine Learning with Stochastic Gradient Descent**. Princeton, 2010.

BOMBARDELLI, Felipe. FaceNet: A Unified Embedding for Face Recognition and Clustering. Porto Alegre, 2015.

CARVALHO, Diogo. **Galaxy S9 aposta numa câmera que enxerga muito além do olho humano** Disponível em: <https://www.diariodepernambuco.com.br/noticia/tecnologia/2018/03/galaxy-s9-aposta-numa-camera-que-enxerga-muito-alem-do-olho-humano.html>. Acesso em 29 ago. 2019.

CHEN, Sheng; LIU, Yang; GAO, Xiang; HAN, Zhen. **MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices**. Beijing, 2018.

CHOLLET, François. **Deep Learning with Python**. Shelter Island, 2018.

DENG, Li; DONG, Yu. **Deep Learning: Methods and application**. Washington, 2004

DO, Huyen; KALOUSIS, Alexandros; WANG, Jun; WOZNICA, Adam. **A metric learning perspective of SVM: On the relation of LMNN and SVM**. Geneva, 2012.

FERREIRA, Gabriel Sousa; RODRIGUES, Marco Túlio Alves. **Mapeando Atividades Físicas por meio de redes Sociais**. Itaúna, 2018.

GRAHAM, Benjamin. **Fractional Max-Pooling**. Warwick, 2015.

HILLS, Dennis. **Understand Core ML on iOS in 5 Minutes.** Disponível em: <https://medium.com/@dmennis/understand-core-ml-on-ios-in-5-minutes-bc8ba5411a2d> . Acesso em 23 jun. 2019.

JACOBS, Bart. **iOS do princípio com Swift: Configurando o ambiente de desenvolvimento**. Disponível em:<https://code.tutsplus.com/pt/tutorials/ios-from-scratch-with-swift-setting-up-the-development-environment--cms-25128>. Acesso em 12 set. 2019

JEKEL, Charles F; HAFTKA, Raphael T. **Classifying Online Dating Profiles on Tinder using FaceNet Facial Embeddings**. Gainesville, 2018.

K. Zhang; Z. Zhang; Z. Li; Y. Qiao. **Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks** in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, oct. 2016.

K.Q. Weinberger, J.Blitzer,and L.K.Saul. **Distance metric learning for large margin nearest neighbor classification**. In NIPS. MIT Press, 2006. 2, 3.

KARUNAKARAN, Dhanoop. **One shot learning explained using FaceNet**. Disponível em: <https://medium.com/intro-to-artificial-intelligence/one-shot-learning-explained-using-facenet-dff5ad52bd38>. Acesso em 25 jun. 2019.

LORENA, Ana Carolina; CARVALHO, André C. P. L. F.. **Uma Introdução às Support Vector Machines**. São Carlos, 2007.

LU, Ning. **FaceNet 和 MTCNN 转 TFLITE 和 CoreML**. Disponível em: <https://github.com/jiangxiluning/facenet_mtcnn_to_mobile>. Acesso em 17 mar. 2019.

MENEZES, Matheus Chaves; NASCIMENTO, Chrystian Gustavo Martins; BRAZ, Geraldo Junior. **Reconhecimento de sinais de trânsito utilizando deep learning**. In: VI Jornada de Informática do Maranhão. Sao Luís: UFMA, 2016. p. 1.

MICHIE, Donald. **“Memo" Functions and machine learning**. Edinburgh, 1968

NASCIMENTO, Elias. **Veja programa de reconhecimento facial open source em funcionamento [vídeo]**. Disponível em:<https://www.tecmundo.com.br/reconhecimento-facial/88079-veja-programa-reconhecimento-facial-open-source-funcionamento-video.htm>. Acesso em 12 set. 2019.

PONTI, Moacir A; COSTA, Gabriel B. Paranhos. Como funciona o Deep Learning. In: Moacir A. COSTA. **Tópicos em Gerenciamento de Dados e Informações**. São Carlos, 2017. p 63-93.

PYTHON. **Python 3.7.4 documentation**. Disponível em:<https://docs.python.org/3/>. Acesso em 16 set. 2019.

RUSSELL, Stuart; NORVIG, Peter. Inteligencia Artificial. Terceira Edição. Elsevier Editora Ltda, 2013

SAKURAI, Rafael. **Implementando a estrutura de uma Rede Neural Convolucional utilizando o MapReduce do Spark**. São Paulo, 2017.

SAMPAIO, Luciano. **Como funciona o Smile Shutter**. Disponível em: <https://www.tecmundo.com.br/hardware/2526-como-funciona-o-smile-shutter.htm/>. Acesso em 16 mar. 2019.

SANTANA, Bruno. **Por falta de inovação, consumidores estão mantendo seus iPhones por ainda mais tempo.** Disponível em: <https://macmagazine.uol.com.br/post/2019/08/23/por-falta-de-inovacao-consumidores-estao-mantendo-seus-iphones-por-ainda-mais-tempo/> . Acesso em 23 ago. 2019.

SANDBERG, David. **Face recognition using Tensorflow**. Disponível em: <https://github.com/davidsandberg/facenet> Acesso em 28 jun. 2019

SANTOS, Igor Pedro Pinto. **Análise de Sentimento Usando Redes Neurais de Convolução**. Rio de Janeiro, 2017.

SARRAF, S.; TOFIGHI, G. . **Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks**. São Francisco, 2016.

SCHROFF, Florian. **FaceNet: A Unified Embedding for Face Recognition and Clustering.** São Francisco, 2015.

SHI, Yichun; RAOUL, Nicolas. ***DocFace*: Matching ID Document Photos to Selfies**. Disponível em: <https://github.com/seasonSH/DocFace>. Acesso em 23 set. 2019.

SILVA, Alex Lima; CINTRA, Marcos Evandro. **Reconhecimento de padrões faciais: Um estudo**. Mossoró, 2015.

SWIFT. **Documentation**. Disponível em: <https://swift.org/documentation/>. Acesso em 16 set. 2019.

SUPORTE, Apple. **Usar o Face ID no iPhone ou iPad Pro**. Disponível em: <https://support.apple.com/pt-br/HT208109#setup>. Acesso em 28 set. 2019.

WADHWA, Aseem; LIN, Allen; LING, Yu-Cheng; G, Dawer; WU, Yuduo; WESTLAKE, Nicholas; SRIDHAR, Krishna; ROSTER, Vladimi; DINH, Tien; COOK, Sean; POKIDYSHEV, Nikita; SNAPE, Patrick; CASTRO, Santiago; KOONCE, Brett; KHARE,

Arjun; SHKARLINSKA, Anna-Mariia; ROSEMAN, Toby. **tfcoreml**. Disponível em:<https://github.com/tf-coreml/tf-coreml>. Acesso em 20 mar. 2019.

WEINBERGER, Killian Q.; BLITZER, John; SAUL, Lawrence K.. **Distance Metric Learning for Large Margin Nearest Neighbor Classification**. Philadelphia, 2005.

YEGULALP, Serdar. **What is TensorFlow? The machine learning library explained**. Disponível em:<https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>. Acesso em 12 set. 2019.

ZHANG, Lei; GAO, Jianfeng. **MSR Image Recognition Challenge (IRC) @ACM Multimedia 2016**. Disponível em:<https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/>. Acesso em 16 set. 2019.

ZURIARRAIN, José Mendiola. **Android já é o sistema operacional mais usado do mundo**. Disponível em: <https://brasil.elpais.com/brasil/2017/04/04/tecnologia/1491296467_396232.html/>. Acesso em 25 abr. 2019.