Otimização de algoritmo de Reconhecimento Facial via aplicação de arquitetura FaceNet Residual baseada em *Deep Learning*

M. Veloso Dias, *CEFET-MG*

***Resumo* —Algoritmos de Inteligência Artificial (IA) baseados em *Deep Learning* representam um tópico de grande complexidade cuja compreensão é de suma importância para a ciência e tecnologia do futuro próximo. O presente trabalho desenvolve e otimiza o funcionamento de uma rede neural convolutiva (CNN) capaz de reconhecer faces humanas. Cada rosto em uma foto é detectado e extraído por um algoritmo MTCNN (*Multi-Task Cascaded Convolutional Neural Network*). Cada face detectada atravessa a CNN FaceNet *Inception*-Residual que gera um vetor de 128 valores de saída, utilizado para reconhecer as identidades presentes na imagem. Isto é possível pois a IA foi treinada para fazer com que vetores de imagens de diferentes pessoas possuam uma Norma Euclidiana maior do que os vetores de fotos de uma mesma pessoa. Observou-se que a metodologia aplicada foi capaz de alcançar até 87% de assertividade no reconhecimento.**

***Palavras-Chave* — *Deep Learning*, Inteligência Artificial, Reconhecimento Facial, Visão Computacional**

I. Introdução

CNN, do inglês *Convolutional Neural Network,* é uma rede neural que é estruturada realizando conexões entre suas camadas através de um filtro e uma operação de convolução. Tais arquiteturas representam as mais avançadas Inteligências Artificiais de Visão Computacional (VC) [1, 2, 3, 4]. Uma aplicação da VC é o reconhecimento facial, cuja acurácia é responsável pela segurança de sistemas vitais como destravar telefones celulares e realizar pagamentos.

Apesar de recentes avanços na pesquisa em Visão Computacional no CEFET-MG [5, 6, 7], pouco esforço está sendo direcionado para a compreensão profunda e melhoria do funcionamento de arquiteturas de reconhecimento baseadas em *Deep Learning (DL)*[3].

É evidente a tendência de se aplicar tecnologias baseadas em *Machine Learning* e *Deep Learning* em metodologia de caixa preta [5, 6]. Assim abre-se espaço para a realização de novas pesquisas que tenham como foco a compreensão do funcionamento destes algoritmos.

Especificamente sobre o trabalho desenvolvido em [5], a inteligência aplicada para realização do reconhecimento foi LPBH (*Local Binary Pattern Histograms)* e, para detecção, utilizou-se Haar Cascade**.** De acordo com estudos relevantes em Visão Computacional estas não representam o estado da arte destas tecnologias [1, 2, 3, 4].

Tal fato é verificado ao se comparar os melhores resultados obtidos por uma tecnologia de reconhecimento à base de LBP (*Local Binary Pattern)* [8] àqueles obtidos por inteligências baseadas em CNN's [1, 2, 4] na base de dados LFW (*Labeled Faces in the Wild)* [4], que é tida como parâmetro por notórios membros da comunidade científica de visão computacional[1, 8]. Enquanto a metodologia LBP de Chen *et al.* [8] alcançou 95,17%, Taigman *et al.* [1] alcançou 97,35% e Schroff *et al.* [2] alcançou 99,63%, ambos aplicando CNN's.

Tendo em vista estes fatos é proposto o presente trabalho que se utiliza da exploração a fundo do funcionamento de CNN's realizada na primeira parte do trabalho de conclusão de curso [3] para desenvolver uma inteligência artificial baseada nesta tecnologia. Otimizando-se, assim, a performance de reconhecimento de [5] através da metodologia que representa o estado da arte do reconhecimento facial e da visão computacional.

De antemão, para facilitar a aplicabilidade comercial do sistema, definiu-se que apenas 3 fotos de treinamento seriam utilizadas para cada pessoa. O número de fotos foi definido através de *brainstorming* em busca de um valor que seria considerado razoável para a maioria das pessoas, do ponto de vista de experiência do usuário.

Além disso, determinou-se que tempo de cadastramento deveria ser o mínimo possível. Estes parâmetros foram definidos tendo em mente um sistema de segurança comum que possa se basear em CNN's, visto que se a aplicação for confiável o suficiente para sistemas de segurança, também será aplicável a outros sistemas que possuem exigências menos rígidas. Vale ressaltar que sistemas de segurança comumente utilizam diferentes métodos com resultados não correlacionadas para confirmar uma identidade, podendo ser o presente trabalho um destes.

Os resultados desejados foram obtidos com sucesso, tendo o presente trabalho diminuído em 97% a quantidade de dados necessários para cadastrar cada pessoa, em comparação com [5]. Foi alcançado também um baixo tempo de treinamento por identidade de 1,150 s e uma assertividade no reconhecimento de 87% em determinadas condições.

Trabalho de Conclusão de Curso submetido para análise da banca avaliadora na data __/__/2020.

M. V. Dias, é formando em Engenharia Elétrica no Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, MG, Brasil (e-mail: mattheusveloso2@hotmail.com).

II. Revisão Bibliográfica

A. Visão Computacional e o Reconhecimento facial

VC é a área de pesquisa que permite que computadores exerçam funções primárias que são realizadas pelos olhos humanos. A detecção de objetos e o reconhecimento facial, por exemplo, são subáreas da VC [9].

Reconhecimento facial é responsável por dizer à qual identidade pertence certa imagem. Tal procedimento é feito comparando-se a imagem em questão a um dado banco de dados de pessoas anteriormente cadastradas [10].

É importante diferenciar reconhecimento facial de verificação facial. Verificação é um procedimento mais simples que define se duas imagens pertencem à mesma identidade ou não. A verificação pode também ser utilizada como parte do algoritmo de reconhecimento através de uma comparação com todas as imagens do banco de dados.

A.1) Falsos Positivos e Falsos Negativos

O maior problema que um sistema de segurança pode gerar é conceder acesso a uma pessoa não autorizada pois este acredita que a pessoa em análise da VC seja alguém que está cadastrado em seu banco de dados. Denomina-se tal problema de Falso Positivo. Um sistema de reconhecimento facial é propício a gerar falsos positivos se não conseguir discernir entre duas pessoas com alto grau de certeza.

Falsos negativos são problemas muito menos graves uma vez que se tratam de negar acesso a uma pessoa autorizada. É bastante provável que tal pessoa consiga tentar novamente e ser reconhecida pelo sistema com sucesso, sendo perda de tempo o prejuízo máximo causado.

B. Treinamento de Redes Neurais

Redes neurais são um método de *Machine Learning* que possuem camadas ocultas interconectadas algebricamente, além das camadas de entrada e saída [11]. Uma rede neural com diversas camadas ocultas é denominada de *Deep Learning.*

As camadas ocultas possuem operadores que recebem as saídas (ativações) das camadas anteriores, multiplicam-nos por uma matriz de pesos, somam ao resultado um valor de viés e, então, obtém um resultado final aplicando uma função não linear (função de ativação, normalmente ReLU, Tangente hiperbólica ou Sigmoid [12]). A equação (1) demonstra, algebricamente a ativação (resultado) de uma camada oculta a partir de suas entradas e parâmetros.

$$A^n = f(W^n * A^{n-1} + B^n) \quad (1)$$

Onde A^n é a matriz de ativações da camada atual (n), W^n a matriz pesos desta camada, A^{n-1} a matriz de ativações da camada anterior, B^n a matriz de vieses da camada atual e $f()$ a função de ativação. Se a matriz pesos representar a conexão de todas as ativações da camada anterior à todas as ativações da camada seguinte, tem-se uma camada denominada de completamente conectada, que é o tipo de camada clássico de redes neurais.

É necessário que uma rede neural receba treinamento prévio para que possa aprender a executar uma atividade específica [12]. Este treinamento pode ser organizado de duas formas: na primeira, engenheiros de software definem quais características a rede neural deve procurar em uma determinada entrada e, a partir destas, aprender a solucionar o problema em questão. Um exemplo seria medir a distância entre os olhos e cor da pele para ser capaz de reconhecer à qual pessoa pertence determinada foto. Estas e outras características quantitativamente compreensíveis ao raciocínio humano são consideradas "definidas manualmente" (*hand-crafted)* e foram a base dos algoritmos de *machine learning* até a chegada da *Big Data*.

O segundo método de treinar uma rede neural é não limitar seu funcionamento a características definidas manualmente e permitir que esta aprenda como resolver o problema em questão somente a partir dos dados que se utiliza. Este tipo de tática funciona melhor que o anterior num cenário onde há abundância de dados de treinamento [1, 10], explicitando a importância da *Big Data*.

Os dados de treinamento para algoritmos de classificação são uma base de dados com entradas e saídas corretamente rotuladas [10]. No caso do presente trabalho, por exemplo, a entrada da rede neural são as informações em RGB dos pixels de uma imagem e a saída é a identidade da pessoa à qual pertence a imagem.

Previamente a qualquer aprendizado, a rede neural fará previsões completamente aleatórias para diferentes entradas. Entretanto, definindo-se uma função perda (*Loss*) para todos os resultados da base de dados, é possível calcular a derivada parcial de cada variável (pesos e vieses) na rede neural em relação a esta. Tal cálculo de derivadas é denominado de *Back-Propagation.*

Tais derivadas formam um gradiente multidimensional que aponta para o valor mínimo de perda. A cada iteração de treinamento a rede neural aprende a se aproximar deste valor mínimo global de perda, que representa o valor que pode ser alcançado que apresenta a maior quantidade de acertos na base de dados de treino [12].

Uma rede neural possui também parâmetros não-treináveis que necessitam de ser escolhidos previamente. Tais parâmetros são denominados de hiperparâmetros [12].

C. Redes Neurais Convolucionais

Uma CNN é um tipo específico de rede neural que possui camadas do tipo convolucionais e *pooling* no lugar de camadas completamente conectadas [3]. Os pesos W^n destas camadas, que vão a treinamento, são os valores dentro dos filtros de cada camada. A Figura II.1 traz um exemplo da camada convolucional destacando o procedimento de convolução.

Vale ressaltar os hiperparâmetros que devem ser escolhidos para estas camadas. Estes são as dimensões e quantidades dos filtros (f), o *stride* (s) e o padding (p).

Um valor *6*(f=2)* indica que serão aplicados 6 filtros de dimensões 2x2. Já um valor *s=2* indica que o passo dado pelo filtro nas dimensões de A^{n-1} será de dois em dois pixels, como na Figura II.2. A Figura II.1 é um exemplo de *s=1.* O padding, por sua vez, indica quantos pixeis de valor 0 serão adicionados a cada volume antes da convolução. Ele pode ser "válido" ou *"*semelhante*"*, p=0 (válido) é representado pelas figuras II.2. e II.1, onde não há padding e o volume diminui suas dimensões após a convolução. Já o padding "semelhante" (*same)* adiciona linha e colunas até fazer com que a convolução gere um volume de dimensões semelhantes ao da camada anterior.

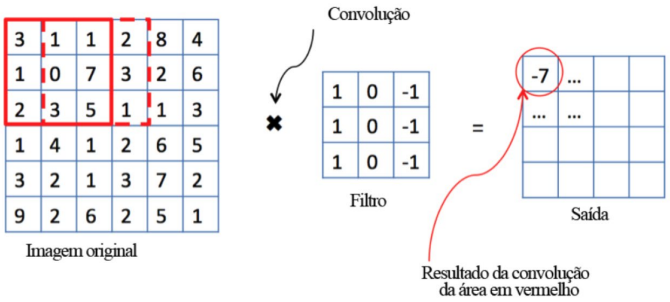

Figura II.1 – Camada Convolucional, com filtro 3x3, padding válido e stride 1, adaptada de [13]

As camadas de *pooling* diferenciam-se das convolucionais pois seu filtro procura qual o valor máximo ou médio dentro de sua área de aplicação (*max pooling* ou *average pooling*, respectivamente). Um exemplo pode ser visto na Figura II.2.

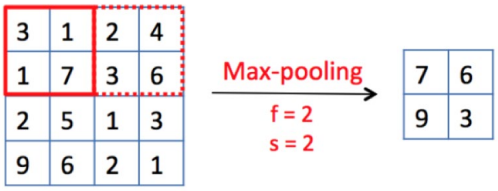

Figura II.2 – Camada Max-pooling com filtro 2x2 e stride 2 obtida de [13]

Um exemplo de rede neural convolucional pode ser encontrado na Figura II.3. CNN's tem, em seu final, a vetorização das ativações de camadas convolucionais na forma de camadas completamente conectadas que reduzem a quantidade de valores nos vetores e uma unidade de *softmax* que calcula a probabilidade de cada candidato à classificação [1].

D.Arquitetura Inception-Residual

Um tipo de camada especial criado por Szegedy *et al.* [14] é a camada do tipo *Inception.* Tal camada aplica, nas ativações das camadas anteriores, filtros de diferentes dimensões e *pooling em paralelo*.

Além disso, Szegedy *et al.* [15] propõe também que se coloque uma conexão identitária (residual) a cada duas camadas. Desta forma permite-se que esta tenha ativações idênticas àquelas presentes duas camadas atrás. Tal fator é importante pois o procedimento de descida do gradiente tem dificuldade em aprender a função identidade em determinada camada [10]. Tal função é útil algebricamente pois garante que nenhuma camada adicionada à rede reduzirá a sua performance, uma vez que ela pode ser simplesmente ignorada. Um exemplo desta arquitetura residual está na Figura II.4.

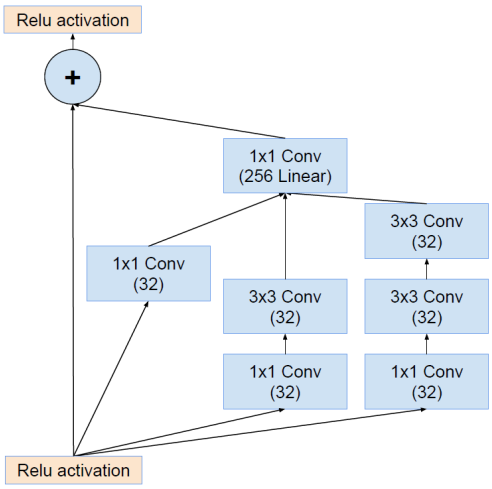

Figura II.4 – Bloco Inception-Residual 35

O algoritmo desenvolvido por [15] propõe uma arquitetura denominada *"inception-ResNet-v1"* que combina as características de *inception* e de redes residuais através de dois tipos de blocos de camadas. Estes são divididos em blocos *inception*-residuais, e blocos de redução. Neste artigo foram inclusos um exemplo de cada (Figura II.4 e Figura II.5), apesar de que no trabalho foram implementados todos os 6 blocos propostos por [15] necessários para criar a *"inception-ResNet-v1"*.

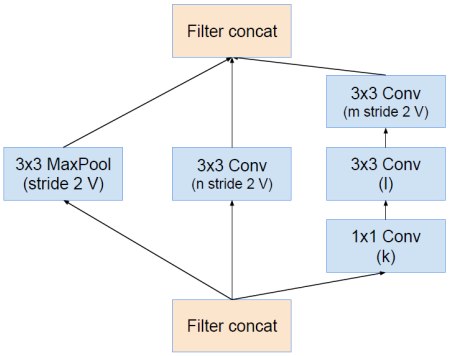

Figura II.5 – Bloco Reduction-A

A função dos blocos *inception*-residuais é aprofundar a rede neural, permitindo que esta aprenda características mais avançadas e com melhores capabilidades de reconhecimento. A função dos blocos de redução é reduzir a largura e altura dos volumes convolucionais, diminuindo a quantidade de dados.

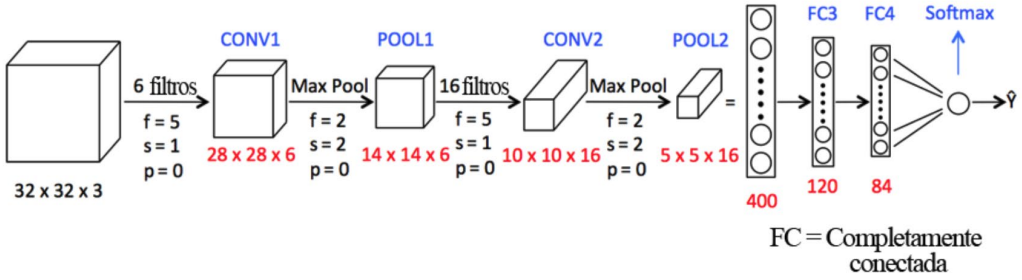

Figura II.3 - Exemplo de CNN adaptado de [13]

Nota-se nas figuras II.4 e II.5 que ambos são blocos de *inception,* mas apenas o bloco 35 é residual, devido à sua conexão direta entre as ativações. No bloco *Reduction-A* é apresentado um filtro de concatenação, que tem a função de unir as saídas das diferentes camadas num formato compreensível para as camadas seguintes.

E. Algoritmo de otimização: Batch Normalization e Dropout

Um problema do treinamento da rede neural é garantir que ela não gerará resultados que satisfaçam apenas a base de dados de treinamento (*overfitting)* como também generalizem para outras bases de dados [16]. Tal problema é medido através da variância dos resultados.

Traz-se a atenção também ao viés dos resultados, que está inversamente relacionado à capacidade da rede neural de ser assertiva para com os seus dados de treinamento. O objetivo de um algoritmo de otimização é conduzir o treinamento para um caminho que resulte em baixa variância e baixo viés [16].

Para este fim, tem-se o algoritmo de otimização, que aplica *batch normalization* e *dropout [16]* para realizar o treinamento mais rápido e assertivo possível [17].

F. Distância L2 (Normalização Euclidiana)

Uma forma de se calcular a distância entre dois vetores P e Q é calcular a Normalização Euclidiana ou distância L2 entre eles. A fórmula de cálculo é representada pela equação (2).

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

Onde p_i representa cada valor individual do vetor P, q_i representa cada valor do vetor Q e n é a quantidade de valores em ambos vetores [18].

G. Detecção via redes neurais

Outra função fundamental da Visão Computacional é detectar um determinado objeto de interesse em uma imagem [9]. O exemplo a ser aplicado no presente trabalho é o de encontrar rostos em uma determinada foto.

Uma rede neural poderia ser treinada para realizar a detecção e reconhecimento ao mesmo tempo, entretanto, algoritmos em *Deep Learning* necessitam de grandes quantidades de dados para serem assertivos. Visto que há uma quantidade muito maior de dados disponíveis para treinamento de uma rede de detecção em separado de uma rede de reconhecimento, faz sentido que se separe a execução destas tarefas em duas etapas distintas em um determinado algoritmo [10].

III. METODOLOGIA

O presente trabalho construiu a aplicação real de uma rede neural para reconhecimento facial, ou seja, não foram utilizados métodos que aumentam a performance do algoritmo que não seriam aplicados em situações reais. A arquitetura geral desenvolvida é resumida pela Figura III.1.

Tais métodos são utilizados em praticamente todos os artigos que alcançam o estado da arte da tecnologia [10], entretanto, em aplicações práticas eles geram muito gasto computacional desnecessário com pouco ganho em performance. Um exemplo é o de aplicar diferentes treinamentos de uma mesma rede neural à uma entrada para obter diferentes resultados e calcular a média entre estes.

Desta forma, os resultados a serem obtidos através deste método são esperados a serem reproduzidos diretamente em aplicação prática.

A. A Rede Neural

O método de escolhido para realizar o reconhecimento foi aplicar uma CNN profunda. Tal escolha se justifica uma vez que esta tecnologia tem sido capaz de alcançar e expandir, seguidas vezes, o estado da arte do reconhecimento facial [1, 2, 4].

B. CNN de Vetorização para reconhecimento: FaceNet Inception-Residual

Foi definido que a metodologia a ser aplicada seria semelhante à FaceNet [2], que é uma arquitetura CNN do tipo *inception*. Além disso, a partir dos avanços obtidos por [15] para facilitar o treinamento com menor poder computacional, definiu que a rede seria também residual.

A justificativa da utilização de camadas *inception* é de que, dado um grande poderio computacional para treinamento da rede neural, esta técnica permite que a o algoritmo de otimização encontre que tipo de camada ele deseja ter em cada etapa, evitando a interferência humana na escolha deste. A presença de conexões residuais, adicionalmente, reduz o custo computacional do treinamento de redes profundas sem que haja perda de desempenho.

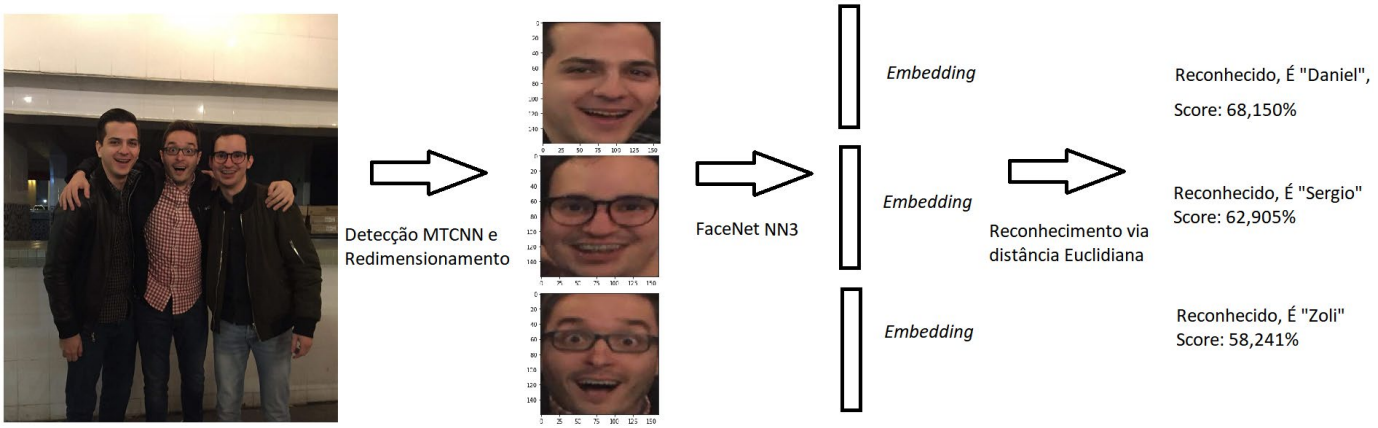

Figura III.6 - Representação gráfica do algoritmo de detecção e reconhecimento facial

Já o algoritmo de otimização utilizado é uma descida do gradiente aplicada em *batch-normalization* com *dropout* de probabilidade de manutenção das ativações igual a 80%. Esta metodologia permite a maximização da performance de uma rede neural sendo também altamente eficiente em evitar o fenômeno de *overfitting* [15]. Garante-se, assim, que a rede neural seja capaz de ter um baixo viés e baixa variância [16].

B.1) Arquitetura FaceNet

A versão da arquitetura FaceNet aplicada foi desenvolvida em Keras [17] com inspiração na versão em TensorFlow disponibilizada por Sandberg [19]. Sandberg realizou o treinamento da rede FaceNet em três milhões de imagens e disponibilizou seus dados de treinamento em licença *Open-Source*, possibilitando importa-los a este projeto e aplicar uma rede neural altamente robusta sem ser necessário treiná-la novamente.

Com isso os dados de treinamento foram importados em forma de pesos das conexões neurais. Na verdade, obter uma base de dados nova de treinamento deste tamanho seria um grande desafio e diversos dias ou meses seriam necessários para retreinar a rede neural com o poderio computacional à disposição da realização desta pesquisa.

Um ponto importante a ser destacado é que a metodologia de vetorização para reconhecimento utilizada neste trabalho pode ser denominada de caixa-cinza, visto que a rede neural FaceNet *Inception*-Residual construída por [17] é aplicada de forma inalterada, apesar de que foi realizada extensa exploração sobre como ela está funcionando internamente. Seria possível alterar a sua arquitetura interna via código, porém uma mudança poderia gerar custos computacionais de treinamento demasiadamente altos, fugindo ao escopo deste trabalho.

Tem-se então que a arquitetura FaceNet é um tipo de CNN sem camada softmax no final cuja saída gera vetores de 128 valores que representam cada face analisada. Estes vetores são compostos pelas características mais importantes de cada rosto para fins de reconhecimento facial definidas via treinamento.

Estas características são distintas das características criadas manualmente e é virtualmente impossível compreender o que elas de fato representam, pois são resultados de milhões de conexões em diferentes camadas de diferentes tipos na rede neural.

Esta é justamente a vantagem de se utilizar uma rede neural em relação aos demais métodos de *machine learning*, visto que a rede neural não está limitada pela escolha manual de características feita por humanos, como mencionado na seção II.B.

Enfim, a opção por esta arquitetura se justifica também pois ela foi capaz de superar os resultados do estado da arte anteriores testados na base de dados LFW em 30% [2]. A aplicação mais complexa da arquitetura FaceNet (NN2) atingiu 99,63% de acerto ultrapassando os resultados de [1], que havia atingido 97,35%.

Na verdade, a arquitetura NN2 se mantém há 5 anos como o estado da arte da tecnologia de verificação facial de acordo com as métricas de comparação do banco de dados LFW [4], quando comparados apenas artigos publicados e revisados por pares (*peer reviewed)*.

B.2) Treinamento da arquitetura FaceNet

Para que a CNN seja capaz de transformar uma imagem RGB em uma representação de 128 características (denominada *embedding*), é necessário que tal instrução esteja explícita como um objetivo durante o treinamento.

Com este fim, o treinamento foi feito a partir da busca do mínimo de uma função perda denominada de perda tríplice (*Triplet Loss)* via descida de gradiente. Tal função calcula a distância algébrica L2 se utilizando de três imagens: uma denominada "Âncora" ou de determinada identidade, outra denominada "Negativa" que se sabe ser de outra identidade e uma denominada "Positiva" que se sabe ser da mesma identidade da âncora em questão [2].

O objetivo do treinamento é aproximar ao máximo a distância L2 entre os *embeddings* de mesma identidade e expandir tal distância entre aqueles de identidade diferente. Algebricamente tem-se tal fato exposto na inequação (3).

$$\|f(x^a) - f(x^p)\|_2^2 + \alpha < \|f(x^a) - f(x^n)\|_2^2 \tag{3}$$

Na inequação (3) tem-se a fórmula para que seja calculada a perda tríplice para um conjunto de 3 imagens. $f(x)$ representa uma *embedding* resultante da aplicação da rede neural em entrada matricial x.

As entradas x^a, x^p e x^n representam as imagens âncora (a), positiva (p) e negativa (n). O operador $\| \ \|_2^2$ representa a normalização euclidiana elevada ao quadrado. O termo α é um hiperparâmetro adicionado a esta equação para garantir que haja uma distância mínima entre os termos de ambos lados da inequação.

Então aplica-se a inequação (3) para todos os N trios x^a, x^p e x^n de uma determinada base de dados, obtendo-se, assim, a função perda a ser minimizada (4).

$$L = \sum_i^N \left[\left\| f(x_i^a) - f\left(x_i^p\right) \right\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{4}$$

Para que haja convergência na descida do gradiente a partir da equação (4), é necessário que os trios sejam escolhidos de forma a serem casos difíceis, ou seja, imagens de pessoas que naturalmente se pareçam e gerem distâncias menores entre a imagem âncora e a negativa [2]. Finalmente, tem-se a otimização realizada a partir da descida do gradiente calculada através de *back-propagation* da função acima.

B.3) Detalhamento da arquitetura NN3 modificada

A arquitetura aplicada da FaceNet escolhida foi a NN3, que aceita entrada imagens recortadas de cada face de tamanho 160x160x3, escolhida em detrimento da NN2 uma vez que ela representa uma versão com uma acurácia próxima à rede neural NN2 mas possui como vantagem uma quantidade exponencialmente menor de parâmetros, acelerando a execução do código [2, 19].

Todas as camadas da NN3 são colocadas em sequência e convergem para uma camada totalmente conectada que gera o vetor de 128 características para cada imagem. A Figura III.2 resume a arquitetura aplicada.

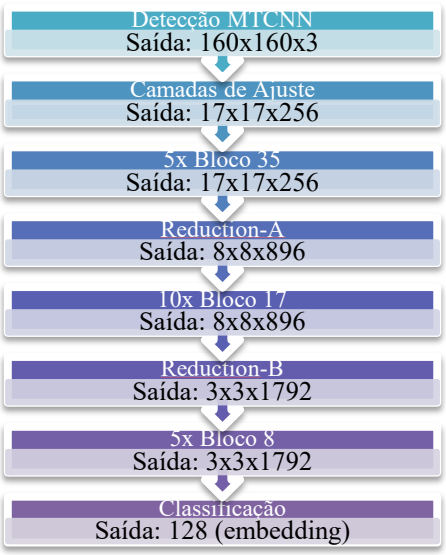

Figura III.1 – Arquitetura FaceNet NN3 Inception-residual completa e resumida.

Como mencionado em II.D, a arquitetura é construída a partir da aplicação conjunta de blocos *Inception* Residuais definida por [15]. Foi necessário o desenvolvimento de camadas de "Camadas de Ajuste" e de "Classificação" (vide Figura III.2 e Figura III.3). Estas etapas fazem o ajuste inicial da imagem na entrada para o formato 17x17x256 antes do primeiro bloco 35 e recebem a saída em formato 3x3x1792 do último bloco 8 e a transformam no vetor desejado (vide Figura III.2).

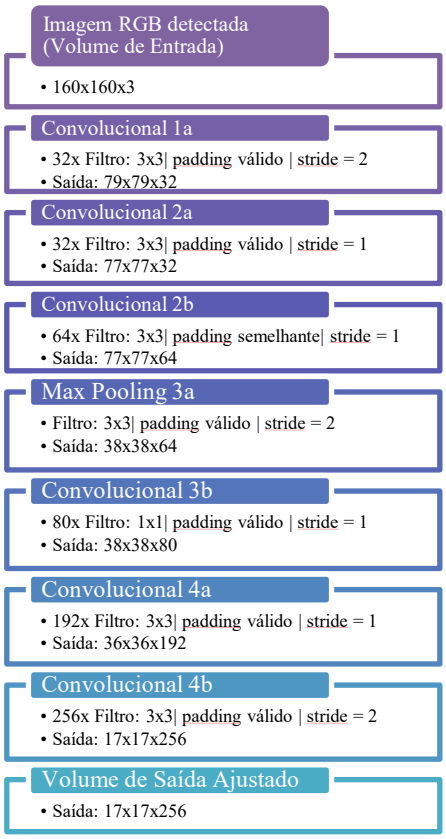

Figura III.2 – Detalhamento das Camadas de Ajuste, em sequência de cima para baixo

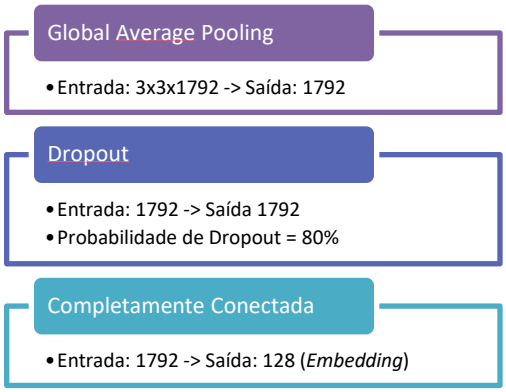

Figura III.3 – Detalhamento das Camadas de Classificação

Esta etapa então é representada na Figura III.1 como a etapa intermediária "FaceNet NN3" e representa a vetorização das imagens em embeddings 128-dimensionais.

C. Arquitetura MTCNN de detecção

É necessário que a entrada da FaceNet NN3 receba imagens RGB de 160x160 pixeis, desta forma utilizou-se de uma outra rede neural denominada de MTCNN para realizar a detecção de imagens em fotos da base de dados, extrair a face e redimensioná-la para um tamanho 160x160x3.

Tal rede neural foi desenvolvida por [20] através dos métodos propostos por *Zhang et al.* [21] que se utiliza de uma CNN em cascata treinada para encontrar 5 pontos na face de uma pessoa em uma foto e, então, encontrar um retângulo na imagem que contenha a face detectada (Figura III.5). Enfim, tal face é extraída e prossegue para a CNN de reconhecimento.

Figura III.4 - Pontos localizados (olhos, nariz e cantos da boca) e retângulo do rosto escolhidos pelo algoritmo. Fonte: Zheng et al [21]

Foi utilizada uma aplicação do algoritmo de [19] adaptada para funcionar em Keras por [20]. Diferentemente da arquitetura CNN de reconhecimento, esta arquitetura MTCNN de detecção e alinhamento não é o foco deste trabalho, por isso, foi aplicada com uma metodologia de caixa-preta com resultados analisados empiricamente.

D. Algoritmo de reconhecimento: Análise algébrica dos vetores em pós processamento

Após uma imagem atravessar as etapas de detecção (MTCNN) e vetorização (FaceNet), tem-se em mãos um vetor de 128-bytes para cada face detectada.

O espaço vetorial em que estes vetores existem é 128-dimensional. Utilizando-se de uma base de dados pré-cadastrada é possível mapear este espaço multidimensional com diferentes identidades. Devido à maneira como a CNN foi treinada identidades parecidas estarão próximas umas das outras e identidades distintas estarão distantes.

Para cada 3 fotos cadastradas na base de dados como vetores de cada pessoa, formam-se pequenos *clusters* de identidades diferentes no espaço 128-dimensional.

Para realizar o reconhecimento de uma face nova cuja identidade é desconhecida (IDesc) foi desenvolvida uma versão modificada do método de *K-Nearest Neighbours* (KNN) com o valor de K igual a 2.

O motivo pelo qual se optou por um valor de K igual a 2 e não igual a 1 é pelo fato de que desejou-se priorizar um software que gerasse a menor quantidade possível de falsos-positivos, tendo em mente um possível sistema de segurança para aplicação deste algoritmo.

Tal método foi aplicado encontrando-se os dois vetores na base de dados com menor distância euclidiana até a *embedding* $f(IDesc)$. Analisam-se então as duas distâncias encontradas, se ambas forem menores do que um valor limite definido em código, considera que o valor é próximo o suficiente para acusar detecção de uma pessoa conhecida pelo sistema. Uma checagem extra é feita para garantir que os dois vetores encontrados próximos de IDesc são pertencentes à mesma identidade. Satisfeitas as condições a foto IDesc é então reconhecida como sendo pertencente àquela identidade, como na Figura III.5.

O valor limite igual a 1.00 foi escolhido empiricamente com o mesmo fim de reduzir falsos positivos, caso determinada aplicação deseje gerar menos falsos negativos, mesmo que gere mais falsos positivos, é possível também diminuir este valor. Um valor de K=3 também não é desejável pois 3 é o número de fotos cadastradas de uma pessoa no sistema. Se alguma foto cadastrada for bastante diferente do que a pessoa é hoje, a distância entre esta foto e a imagem em avaliação será maior que as restantes, diminuindo a efetividade de reconhecimento do sistema.

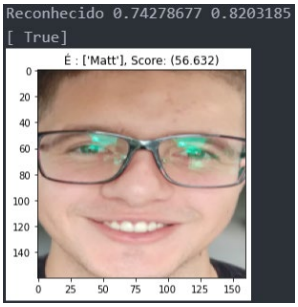

Figura III.5- Exemplo de IDesc para reconhecimento

Na Figura III.5, o algoritmo retornou "Reconhecido" e em seguida as distâncias entre os dois vetores mais próximos (Figura III.8). Em seguida ele comparou se a identidade prevista é a correta, registrando "True", ou seja, correto. Além disso foi calculado um score de certeza de 56.6%.

Este score foi criado para facilitar a compreensão para o usuário final. Tal score funciona da forma apresentada na equação (5).

$$Score = \frac{1}{2^{Dist2}} * 100\% \quad (5)$$

Onde Dist2 é a distância de f(IDesc) para o segundo embedding mais próximo. Para valores de Dist2 menores ou iguais a 1 (o valor limite definido), o Score será maior ou igual a 50% e então o algoritmo acusará o reconhecimento. Caso contrário, se Dist2 for maior que 1, o Score será menor que 50%, fazendo o algoritmo então acusar que não houve o reconhecimento com sucesso.

Acessando a base de dados para compreender com quais imagens o algoritmo tomou a decisão de reconhecimento da Figura III.5, tem-se a Figura III.6.

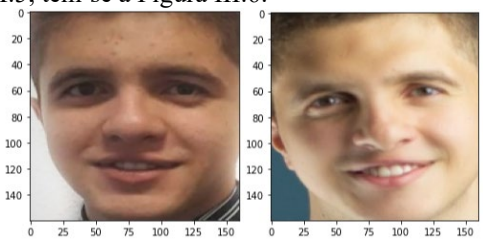

Figura III.6 – Duas imagens do banco de dados cujos vetores geraram distâncias L2 de 0.7428 e 0.8203 em relação a f(IDesc).

De todas as 156 imagens de um banco de dados de 52 identidades, estas foram as imagens mais próximas encontradas. Como elas pertencem à uma mesma identidade, o algoritmo acusa o reconhecimento.

No último caso em que ambas distâncias sejam menores que o valor limite, mas não pertencem à mesma identidade no banco de dados, o algoritmo acusará que não tem certeza referente àquela identidade, evitando assim um falso-positivo e possivelmente gerando um falso-negativo. Nesse caso será gerado uma saída "incerta" (*unsure)*.

E. Python, Keras e GitHub

Para realizar a programação, foi criado um repositório do GitHub TM [22]. Os códigos foram escritos em Python com base em Keras, TensorFlow e Numpy. A inspiração para as aplicações em Keras e TensorFlow das redes neurais foram obtidas de [17, 19, 23].

IV. Resultados e Discussões

Analisou-se a performance da inteligência desenvolvida a partir de testes feitos em 2 bases de dados diferentes: uma "natural" (*nat*), com fotos para reconhecimento em diversas condições de iluminação de 52 pessoas diferentes, e uma base de dados "padrão" (*std*), que se utilizou apenas de fotos claras, retas e sem acessórios no rosto como óculos escuros e boinas destas mesmas pessoas.

Destaca-se que tanto *nat* quanto *std* foram reconhecidas com base na mesma base de dados cadastrada (*cad)* de 3 fotos de cada pessoa, com um total de 156 fotos cadastradas de 52 pessoas.

Justifica-se que ambos contextos fossem utilizados para que se obtivessem resultados, uma vez que em um reconhecimento menos rigoroso, por exemplo em redes sociais, deve-se encontrar mais dados próximos à primeira base. Já num sistema de segurança espera-se que os dados a serem analisados sejam padronizados, permitindo que se avalie o desempenho deste algoritmo em diferentes aplicações.

Na primeira base de dados (*nat)* os resultados obtidos foram:

- 15,7% de resultados incorretos, sendo 5,7% falsos positivos e 10,0% falsos negativos;
- 11,5% de resultados considerados "incertos";
- 72,8% de reconhecimentos corretos.

Na segunda base de dados (*std*) os resultados foram:

- 11,7% de resultados incorretos, sendo 9,1% falsos positivos e 2,6% de falsos negativos;
- 7,8% de resultados considerados "incertos";
- 80,5% de reconhecimentos corretos.

Estes resultados são os mais importantes obtidos neste estudo pois demonstram que a inteligência artificial desenvolvida de fato funciona.

A grande quantidade de falsos positivos em ambas bases de dados demonstra que esta inteligência teria dificuldades ser aplicada em sistemas de segurança. Para resolver este imbróglio pode-se utilizar paralelamente de outros métodos de segurança com baixa correlação à CNN aplicada que possa aumentar o grau de certeza das decisões. Poderia também ser elevado o valor de K, cujos efeitos são explorados na seção IV.E.

A utilização de somente dois métodos não correlacionados com taxa de falsos positivos semelhante à deste trabalho (5,7%) alcançaria uma nova taxa de somente 0,32%. Portanto o presente trabalho pode ser apresentado sob esta ótica no presente estado. Além disso, um desenvolvimento no alinhamento prévio à MTCNN é também uma opção para melhorar o desempenho.

A. Tempo de cadastramento e execução

A fim de realizar o reconhecimento é necessário que *cad* esteja completamente convertida em vetores. Esta conversão foi considerada como o procedimento de cadastramento. Este é simplesmente o processo de detectar todas as faces na base de dados e vetorizá-las.

A execução, por outro lado, trata-se de detectar rostos em uma imagem, vetorizá-los e reconhece-los, como na Figura III.1.

O CPU utilizado para todas as tarefas de teste foi um Intel core i7 3770k. A Tabela I consolida os resultados de tempos de cadastramento e execução, para o algoritmo desenvolvido e para a base de dados que possui 52 identidades.

TABELA I
RESULTADOS DE CADASTRAMENTO E EXECUÇÃO

Grandeza	Resultado
Tempo médio de cadastramento por identidade com 3 imagens no banco de dados	1,150 s
Tempo médio de detecção e vetorização por imagem	0,383 s
Tempo médio de Detecção por imagem	0,301 s
Tempo médio de Execução do KNN por identidade detectada	0,209 s
Tempo total para reconhecer uma pessoa	0,592 s

O tempo total de reconhecimento de uma pessoa demonstra que a execução é feita em tempo hábil. Desta forma este sistema serviria para a maioria dos sistemas de segurança deste ponto de vista.

Uma possível melhoria que pode ocorrer é otimizar a velocidade do processo de detecção, visto que este ocupa mais da metade do tempo que o algoritmo necessita para reconhecer um indivíduo.

Mesmo assim, aplicações como [24] (cujo código é a base de [5]) e [17] se utilizam de HAAR Cascade para a detecção, que consegue realizar detecção em tempo real com uma taxa de 10-25 *frames* por segundo. Tal fato serve para exacerbar a melhoria que seria necessária para que o algoritmo de MTCNN aplicado ao presente trabalho alcançasse a velocidade de processamento do método concorrente.

B. Problema com Identificação de Asiáticos e Crianças

Duas falhas relevantes encontradas nos resultados ficam aparentes ao analisar-se, especificamente, as fotos de pessoas de etnias orientais e de crianças.

Tal amostragem foi obtida num estudo particular sobre o viés étnico e etário do algoritmo. Em busca do motivo da alta quantidade de falsos positivos, encontrou-se que para adultos de etnias europeias, americanas e africanas, a taxa de reconhecimento se mantém próxima da média em 1 desvio padrão.

Entretanto, com asiáticos, os resultados obtidos foram:

- 50,0% de reconhecimento assertivo;
- 10,7% de resultados "incertos";
- 39,3% de falsos positivos.

Tal número altíssimo de falsos positivos indica que este mesmo algoritmo tem um viés étnico para com essas pessoas. É provável que tal viés seja fruto da base de dados de treinamento de [19]. Um novo treinamento poderia resolver tal problema.

Além deste viés, encontrou-se também que o algoritmo é pouco eficaz para diferenciar crianças. Este resultado também se deve, provavelmente, a um viés incluso na base de dados de treinamento. A taxa de acerto do teste etário com crianças foi de apenas 33,3%.

Mais especificamente, viéses etários e étnicos são frutos de uma base de dados quando esta possui poucos exemplos destes para treinamento da rede neural de reconhecimento. Isto é um problema pois o algoritmo não foi treinado para diferenciar essas pessoas e apresentará, como é o caso, uma performance pior para com estas.

C. Acertos Desafiadores

A Figura IV.1 traz a atenção, por outro lado, à resultados difíceis que geraram acertos.

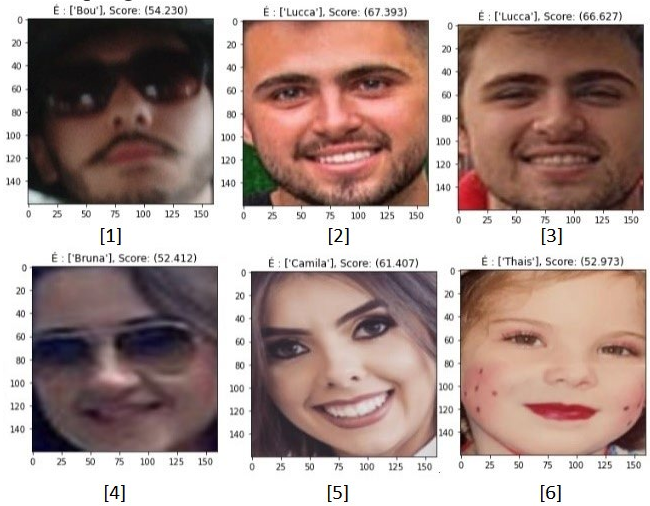

Figura IV.1 - Acertos considerados difíceis realizados pela rede neural

Especificamente, tem-se em IV.1.1 a foto uma pessoa utilizando óculos, chapéu e em condições de baixa iluminação. As duas fotos seguintes (IV.1. 2 e IV.1.3) são de uma pessoa cujo irmão gêmeo univitelino também está cadastrado na base de dados. A foto IV.1.4 testa uma situação com baixa qualidade da imagem de uma pessoa usando óculos, onde a rede neural foi capaz de atravessar o *score* mínimo de 50%. A foto IV.1.5 mostra uma pessoa que foi cadastrada sem maquiagem, mas está bastante diferente na foto devido à presença desta. Finalmente, na sexta foto tem-se o reconhecimento de uma pessoa maquiada em uma foto tirada em sua infância.

Estes resultados demonstram que a rede neural é capaz de identificar pessoas em situações em que até humanos teriam dificuldade. Que é um resultado também obtido por [2], visto que na LFW a arquitetura FaceNet alcançou 99,63% de acertos enquanto o resultado mediano humano é de 97,53% de acertos [1].

D.Erros Esperados

A diferenciação entre gêmeos e o reconhecimento de pessoas em fotos de baixa qualidade são desafios enfrentados por todos os algoritmos de reconhecimento facial [25]. Neste trabalho, obtiveram-se resultados que seguem esta tendência.

Os resultados relacionados à baixa resolução de imagens foram analisados de forma qualitativa e mostram que, à medida que a resolução diminuiu, o score diminui e a quantidade de falsos positivos e negativos aumenta. Um exemplo deste padrão de resultados é representado pela Figura IV.2.

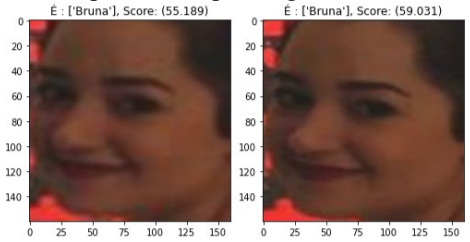

Figura IV.2 - Mesma imagem reconhecida com resoluções diferentes

Esta conclusão é válida visto que a foto da direita alcançou um score 7,0% maior e possui mais alta resolução. Logo, quanto melhor for a câmera utilizada para o reconhecimento, melhor será sua performance.

Os resultados com gêmeos, por outro lado, podem ser encontrados na Tabela II.

TABELA II
TESTE COM IRMÃOS GÊMEOS

Grandeza	Resultado
Acertos	2
Falsos Positivos	1
Unsures	5

Fica claro que este algoritmo tende a gerar *unsures* para gêmeos, o que faz sentido pois até para o olho humano é difícil diferenciá-los.

Além disso, pode-se afirmar que qualquer base de dados que se utilize de uma amostra da população em geral como distribuição estará enviesada contra gêmeos. Isto decorre da regularização dos algoritmos de otimização, que os impediras de alcançar o nível de variância necessária para diferenciar pessoas bastante semelhantes.

Uma forma possível de resolver este problema é cadastrar na base de dados uma *flag* que indique se a pessoa tem uma irmã ou irmão gêmeo univitelino. Desta forma, quando o algoritmo gerar um positivo ou *unsure* cuja distância aponta para identidades cadastradas como gêmeos, ele poderá atravessar estes resultados por uma nova rede neural, que seria treinada especificamente para diferenciar gêmeos. Melhorando, assim, os resultados finais.

E. Comparação entre valores de K

Como mencionado em III.D, a escolha de K se deu como K=2 uma vez que entre os valores possíveis está se mostrou um bom meio termo entre a geração de falsos positivos e o desempenho de assertividade do reconhecimento. Os valores possíveis para K são 1, 2 e 3, uma vez que cada identidade possui 3 fotos cadastradas.

Foi feita então a comparação para os valores de K para a base de dados *std.* A Figura IV.3 traz uma representação gráfica das mais importantes métricas para os valores possíveis de K na base de dados *std.*

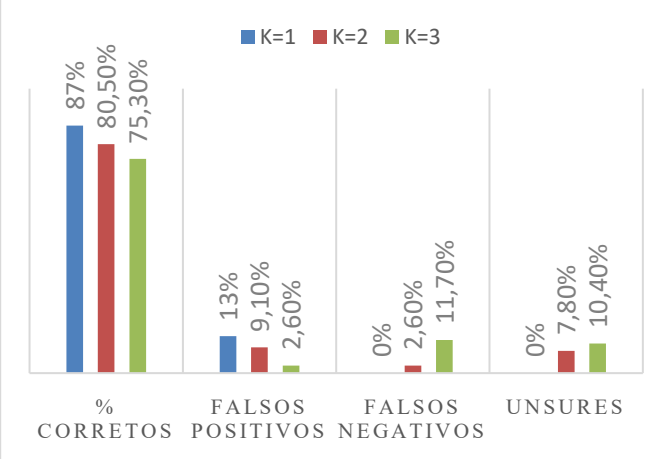

Figura IV.3 - Comparação de diferentes métricas para valores de K

Percebe-se que este resultado confirma a previsão feita durante o processo de desenvolvimento sobre o *tradeoff* entre assertividade e falsos positivos. No entanto, não se observou um aumento significativo na quantidade de resultados incertos, visto que Dist3 é comumente maior que 1.00, então tende a gerar mais negativos.

Esta escolha dependerá do que exige cada aplicação da presente arquitetura, demonstrando a adaptabilidade deste sistema desenvolvido para diferentes situações. Outra possibilidade seria cadastrar mais fotos para cada usuário e, desta forma, aumentar o valor de K sem necessariamente diminuir a performance do sistema.

Vale destacar que foi alcançado um melhor resultado, do ponto de vista de sistemas de segurança com foco em reduzir falsos positivos, com um valor de K=3. Isto decorre da redução dos falsos positivos em 76,9% em relação à arquitetura utilizada (K=2).

Ainda assim, a escolha de K=2 se sustenta, pois, este sistema demonstrou ser uma escolha intermediária entre a geração de

falsos positivos e o desempenho de assertividade do reconhecimento.

V. Conclusões

O estudo foi capaz de reproduzir resultados satisfatórios de reconhecimento facial, alcançando 87% de acertos em uma dada base de dados. Desta forma, provou-se que a rede neural convolucional FaceNet é aplicável em ambientes acadêmicos com limitado poderio computacional.

Além disso, o conhecimento profundo da arquitetura CNN propiciou que discussões fossem feitas acerca de possíveis vieses de treinamento e pontos de melhoria no sistema.

O alto número de falsos positivos, no entanto, representa um obstáculo que este algoritmo deve ultrapassar antes de ser implementado para sistemas de segurança.

VI. Agradecimentos

Agradeço ao CEFET-MG por me proporcionar estes anos de aprendizado. Ao professor orientador Anthony Chiaratti por toda a paciência, apoio e *insights.* Ao professor co-orientador Túlio Carvalho pela vital participação na idealização deste projeto. À minha irmã Maria Fernanda, aos meus pais Francine e Cassiano e aos meus avós Edna e Ronaldo por todo apoio e compreensão que foram necessários na realização deste trabalho.

VII. Referências bibliográficas

[1] Y. Taigman, M. Yang, M. Ranzato & L. Wolf. "*DeepFace: Closing the Gap to Human-Level Performance in Face Verification*". *IEEE Conf. on CVPR, 2014.*

[2] F. Schroff, D. Kalenichenko, J. Philbin, "*FaceNet: A Unified Embedding for Face Recognition and Clustering*". *IEEE Conf. on CVPR, 2015.*

[3] Dias, M. V., "Desenvolvimento de uma Rede Neural de Visão Computacional conectada à Nuvem", *Trabalho de Conclusão de Curso I, CEFET-MG, 2019.*

[4] University of Massachusetts, "*Labeled Faces in The Wild Results"*, [Online]. Disponível: http://vis-www.cs.umass.edu/lfw/results.html#hdlbp, 2018

[5] Brum, A. L., & Oliveira, M. C., "Face Unlock: Uma aplicação de Visão Computacional". *Iniciação científica, 2018, CEFET-MG.*

[6] G. B. de Castro, "Solução para Contagem de Pessoas em Epsaçõs Públicos Usando Visão Computacional", *Trabalho de Conclusão de Curso II*, *CEFET-MG, 2019.*

[7] A. L. Brum, M. C. Oliveira, A. d. Almeida, "Interferência da Maquiagem e Acessórios no Reconhecimento Facial". *Febrar, 2018.*

[8] D. Chen, X. Cao, F. Wen, J. Sun, "*Blessing of Dimensionality: High-dimensional Feature and its Efficient Compression for Face Verification*". *IEEE Conf. on CVPR, 2013.*

[9] R. Szeliski, "*Computer Vision: Algorithms and Applications*". Springer, Disponível: http://szeliski.org/Book/, 2010.

[10] A. Ng. "*Convolutional Neural Networks*", Disponível: https://www.coursera.org/learn/convolutional-neural-networks/home/welcome, 2017.

[11] K. Collins-Thompson, "*Applied Machine Learning in Python"*. Coursera, 2017, Disponível: https://www.coursera.org/learn/python-machine-learning/home/welcome

[12] A. Ng, "*Neural Networks and Deep Learning".* Coursera, 2017, Disponível: https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome

[13] M. Cavaioni, "*DeepLearning series: Convolutional Neural Networks". Medium, 2018,* Disponível: https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Aguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "*Going Deeper With Convolutions*". *IEEE Conf. on CVPR, 2015.*

[15] C. Szegedy, S. Joffe, V. Vanhoucke, A. Alemi, "*Inception v-4, Inception-ResNet and the Impact of Residual Connections on Learning*". *arXiv:1602.07261v2, 2016.*

[16] A. Ng, "*Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization".* Coursera, 2017. Disponível: https://www.coursera.org/learn/deep-neural-network/home/welcome

[17] H. Taniai, "*Facenet implementation by Keras2*", 2018, Repositório GitHub, https://github.com/nyoki-mtl/keras-facenet.

[18] N. Bourbaki, "Chapters 1–5". *Topological vector spaces*. Springer, 1987.

[19] D. Sandberg, "*Face Recognition using TensorFlow",* 2018, Repositório GitHub, https://github.com/davidsandberg/facenet

[20] I. de P. Centeno, "*MTCNN face detection implementation for TensorFlow, as a PIP package*", 2018, Repositório GitHub, https://github.com/ipazc/mtcnn.

[21] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, "*Joint Face Detection and Alignment using Multi-Task Cascaded Convolutional Networks*", *IEEE Signal Processing Letters, Vol.: 23 , no.: 10, Out. 2016.*

[22] M. V. Dias, "*Repository for Face Recognition Thesis*", 2020, Repositório GitHub, https://github.com/Mattveloso/DeepFaceRecThesis

[23] J. Browniee, "How to develop a Face Recognition System Using FaceNet in Keras". 2019, Disponível: https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/

[24] J. Le, "*Use Python and Open CV to recognize multi face and show the name",* 2017, Repositório GitHub, https://github.com/trieutuanvnu/Raspberry-Face-Recognition

[25] A. Afaneh, F. Noroozi, Ö. Toygar, "*Recognition of identical twins using fusion of various facial feature extrators*". *EURASIP Journal on Image and Video processing, 2017.*