Sistema de Comparação de Imagens de Faces, em Múltiplas Resoluções, Baseado em Redes Neurais Siamesas

Comparison System of Faces Images, in Multiple Resolutions, Based on Siamese Neural Networks

Raphael Brito Alencar [1] orcid.org/0000-0003-4934-478X
Byron Leite Dantas Bezerra [1] orcid.org/0000-0002-8327-9734

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil.

E-mail do autor principal: Raphael Brito Alencar rba2@ecomp.poli.br

Resumo

Neste trabalho é apresentado um sistema de comparação de imagens de faces, capaz de ser utilizado com variadas resoluções, utilizando um método para cálculo de similaridade, que permite ser utilizado, por exemplo, em aplicações de biometria. O método funciona para conjuntos de dados com muitas categorias e poucos exemplos por categoria, além de categorias não visualizadas durante a fase de treino. A ideia principal é aprender uma função que mapeia padrões de entrada em um espaço destino de forma que a norma L1 no espaço de destino aproxima a distância "semântica" no espaço de entrada. O processo de aprendizagem minimiza uma função de perda discriminativa que converge para uma métrica de similaridade, que deve ser pequena quando os pares de faces são da mesma pessoa e grande para pares de faces de pessoas diferentes. Uma arquitetura siamesa de redes neurais convolucionais, robusta a distorções geométricas, foi utilizada para mapear os padrões de entrada no espaço destino.

Palavras-Chave: Faces; Similaridade; Verificação; Convolucionais; Siamesas.

Abstract

In this work a system of face image comparison is presented, capable of being used with various resolutions, using a method for calculating similarity, which allows to be used, for example, in biometrics applications. The method works for data sets with many categories and few examples per category, as well as categories not visualized during the training phase. The main idea is to learn a function that maps input patterns to a destination space so that the L1 norm in the destination space approximates the "semantic" distance in the input space. The learning process minimizes a discriminative loss function that converges to a similarity metric, which should be small when the pairs of faces are of the same person and large for pairs of faces of different people. A Siamese architecture of convolutional neural networks, robust to geometric distortions, was used to map the input patterns in the target space.

Key-words: *Faces; Similarity; Verification; Convolutionals; Siamese.*

DOI: 1025286/repa.v5i1.1192

1 INTRODUÇÃO

Modelos de classificação tradicionais, tais como redes neurais, precisam conhecer previamente, a natureza das classes em estudo, além de exigir exemplos de treino para todas elas. Entretanto, quando o número de classes é muito grande ou o número de exemplos por classe é pequeno, o trabalho se torna árduo e difícil para tais modelos.

Para aplicações de verificação de faces o número de classes pode ser enorme e com poucos exemplos por categoria. Uma abordagem comum para tal aplicação são os chamados *distance-based methods,* que consistem em calcular uma métrica de similaridade entre o padrão a ser verificado e uma biblioteca de protótipos armazenados **[1]**.

O nível de detalhes que uma imagem comporta é descrito pela resolução desta imagem. Dessa forma, quando duas imagens de resoluções diferentes, sendo uma maior que a outra, são comparadas, a quantidade de informação extraída da imagem com maior resolução será completamente diferente da quantidade de informações extraídas da imagem de menor resolução. Essa discrepância na quantidade de informações extraídas gera um grande problema para sistemas de comparação de faces, que recebem como entrada imagens com resoluções variadas. Sabendo que existe uma grande quantidade de dispositivos capazes de realizar captura em diversas qualidades e resoluções, os sistemas tendem a classificar como diferentes, as imagens com variações consideráveis de resolução.

A solução apresentada neste artigo é calcular uma métrica de similaridade a partir dos dados de treinamento (imagens de faces) e utilizá-la para verificar imagens similares de classes não conhecidas anteriormente (faces de pessoas não vistas durante a fase de treino). Para tal, é preciso encontrar uma função que mapeia padrões de entrada em um espaço destino de forma que uma distância simples neste espaço destino se aproxime da distância "semântica" do espaço de entrada.

Dada uma família de funções $G_w(X)$ parametrizadas por W, a solução procura valores para o parâmetro W tais que a métrica de similaridade $E_W(X_1, X_2) = |G_w(X_1) - G_W(X_2)|$ é menor se X_1 e X_2 são da mesma classe e maior caso contrário. O sistema proposto é treinado a partir de pares de padrões vindos de um conjunto de treinamento. A *loss function* é minimizada por pares (X_1, X_2) quando X_1 e X_2 pertencem à mesma categoria e maximizada quando pertencem a categorias distintas. Como a mesma função G_w é usada para processar ambas entradas, a arquitetura é chamada de siamesa.

Para verificar a similaridade entre imagens de faces é preciso que a arquitetura utilizada para extrair características do conjunto de treinamento, seja robusta o suficiente para distorções geométricas da entrada. Sendo assim, utilizaremos redes neurais convolucionais.

2 TRABALHOS ANTERIORES

A ideia de mapear imagens faciais em espaço de baixa dimensão antes da comparação, começou com o método conhecido como Eigenface baseado em PCA, onde é uma projeção linear treinada de forma não discriminativa para maximizar a variância **[2]**. Extensões não-lineares baseadas no Kernel-LDA e no Kernel-PCA também foram discutidas **[3]**. No entanto, o principal problema dessas abordagens é que elas são sensíveis as transformações geométricas das imagens de entrada e a algumas outras variabilidades como, por exemplo, mudanças nas expressões faciais e acessórios.

Existem alguns trabalhos que descrevem métricas de similaridade que são localmente invariantes a um conjunto de transformações, dentre eles estão o *Tangent Distance method* **[4]** e o *elastic matching* **[5]**. Além destes, alguns trabalhos defendem algoritmos de normalização baseados em distorção para reduzir as variações de pose **[6]**.

Alguns trabalhos mais recentes como FaceNet **[7]** e DeepFace **[8]**, conseguem angariar resultados extremamente satisfatórios, mapeando imagens faciais em um espaço euclidiano compacto, chegando a atingir precisão de até 99% em alguns bancos de imagens de faces amplamente utilizados como *Labed Faces in the wild (LFW)* e *YouTube Faces DB.*

3 SISTEMA PROPOSTO

O problema consiste em encontrar uma função $G_w(X)$ que mapeia padrões de entrada de alta dimensão em saídas menores, dadas as relações de vizinhança entre as amostras de entrada. Mais precisamente, dado um conjunto de vetores de entrada $T = \{X_1,\ldots,X_P\}$, onde $X_i \in R^D, \forall i = 1,\ldots,n$ encontre uma função paramétrica $G_w : R^D \rightarrow R^d$ com $d \ll D$, que obedeça as seguintes condições:

1. Medidas de distância simples no espaço de saída (como a distância euclidiana) devem aproximar as relações de vizinhança no espaço de entrada.

2. O mapeamento não deve ser restrito à implementação de medidas de distância simples no espaço de entrada e deve ser capaz de aprender invariantes para transformações complexas.

3. Deve ser fiel mesmo para amostras cujas relações de vizinhança são desconhecidas.

3.1 Função de perda – Contrastive Loss

Considere o conjunto τ de vetores de treinamento de alta dimensionalidade X_i. Assuma que para cada $X_i \in \tau$ exista um conjunto S_{xi} de vetores de treinamento que são similares a X_i. Esse conjunto pode ser obtido por algum conhecimento anterior que não depende de uma distância simples, como exemplo, proximidade temporal. Um mapeamento efetivo de alto para baixo espaço dimensional mapeia vetores de entrada semelhantes em pontos próximos no espaço de saída e vetores diferentes em pontos distantes. Dessa forma, introduz-se um novo tipo de função de perda, cuja minimização pode gerar tal mapeamento. Essa função de perda roda sobre pares de entrada, diferentemente de outras abordagens de aprendizagem. Sendo $X_1, X_2 \in \tau$ um par de vetores de entrada e Y um label binário atribuído a este par. $Y = 0$ se X_1 e X_2 são considerados similares e $Y = 1$ caso contrário. Definir uma função de distância parametrizada a ser aprendida D_w entre X_1 e X_2 como a distância euclidiana entre as saídas de G_w. Isto é:

$$D_W(X_1, X_2) = |G_W(X_1) - G_W(X_2)|_2 \quad (1)$$

utilizando $D_W(X_1, X_2)$ como D_w. A função de perda tem o formato geral de:

$$L(W) = \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i) \quad (2)$$

$$L(W, (Y, X_1, X_2)^i) = (1 - Y) L_S(D_W^i) + Y L_D(D_W^i) \quad (3)$$

onde $(Y, X_1, X_2)^i$ é o i-ésimo par com label Y, Ls é a função de perda parcial para um par de pontos similares, L_D é a função de perda parcial para pontos dissimilares, e P é o número de pares de treinamento, que pode ser tão grande quanto o quadrado do total de amostras.

Ls e L_D devem ser elaboradas de forma a minimizar a função de perda L em relação a W fazendo com que resulte em valores baixos de D_W quando as imagens são do mesmo indivíduo e valores altos de D_W quando contrário.

Sendo assim, a função exata de perda é dada por:

$$L(W, Y, X_1, X_2) = \frac{(1 - Y) * {D_w}^2}{2} + \frac{(Y) * \{Max(0, m - D_w)\}^2}{2} \quad (4)$$

onde $m > 0$ é uma margem que define um raio em torno de $G_W(X)$, tal que um par dissimilar só contribui com a função de perda caso sua distância esteja dentro desse raio (Figura 1). Isso faz sentido, pois só é desejável otimizar o aprendizado da rede com base em pares que são realmente diferentes, mas que a rede julga ser bastante semelhantes.

L_D, termo contrastivo relativo aos pares dissimilares, é essencial para a equação dado que se fosse levado em consideração apenas o termo L_s, para minimizar $D_W(X_1, X_2)$, poderia ser gerada uma solução pobre, já que D_w e L poderiam ser iguais a zero, ou seja, G_w seria uma constante.

3.2 Passo a passo de treinamento

Para o treinamento da rede, é preciso criar o conjunto de treinamento pareando as imagens presentes na base de dados.

1. Para cada amostra X_i:
   a. Encontre o conjunto $S_{Xi} = \{X_j\}^P_{J=1}$ com as amostras similares a X_i.
   b. Pareie X_i com outras amostras de treinamento e atribua labels para os pares de forma que $Y_{ij} = 0$ se $X_j \in S_{Xi}$ e $Y_{ij} = 1$ caso contrário.
   c. Combine os pares gerados formando um conjunto de treinamento rotulado.
   d.
2. Repetir até convergir:
   a. Para cada par (X_i, X_j) no conjunto de treinamento, faça:
      i. Se $Y_{ij} = 0$, atualize W de forma a decrementar $D_w(x_i, x_j) = |G_w(X_i) - G_w(X_j)|_2$
      ii. Se $Y_{ij} = 1$ atualize W de forma a incrementar $D_W(X_i, X_j) = |G_w(X_i) - G_w(X_j)|_2$

a minimização da função de perda é responsável por essa variação na distância euclidiana.

DOI: 1025286/repa.v5i1.1192

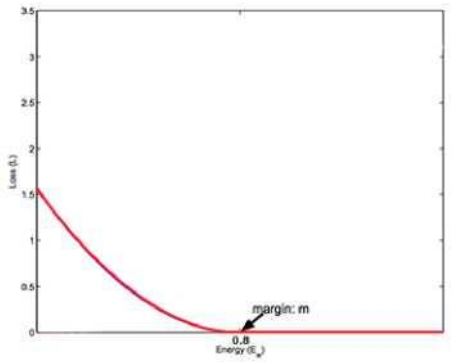

Figura 1: Gráfico da função de perda versus a energia (distancia).

3.3 Redes Neurais Convolucionais

Como estamos trabalhando com imagens e desejamos mapeá-las em pontos num espaço de baixa dimensionalidade, utilizamos duas redes convolucionais idênticas (arquitetura siamesa), que compartilham de um vetor de parâmetros comum, para aplicar uma métrica de similaridade aprendida [x] (ver Figura 2).

As redes neurais convolucionais possuem vários filtros diferentes, consistindo em parâmetros treináveis que podem se encaixar espacialmente em uma determinada imagem para detectar recursos como bordas e formas. Esse grande número de filtros consegue capturar recursos espaciais da imagem com base nos pesos aprendidos por meio do processo de *back-propagation* e camadas empilhadas de filtros que podem ser usadas para detectar formas espaciais em cada nível subsequente. Dessa forma, elas podem sintetizar uma imagem em uma representação altamente abstrata, fácil de prever.

As redes neurais convolucionais são sistemas multicamadas, não-lineares e que são capazes de aprender recursos de baixo e alto nível de forma integrada, operando no nível dos pixels de uma imagem. Mapeiam essas imagens para saídas detectando recursos locais, sendo invariantes a deslocamento (*shift invariant)* ou invariantes a espaço *(space invariant)* e assim conseguindo representações robustas a distorções geométricas das imagens de entrada.

Uma rede neural Siamesa consiste de duas redes idênticas que aceitam um par de imagens e um label como entrada. As saídas das duas redes servem de entrada para um módulo de custo que fornece a energia escalar $E_W(X_1,X_2) = |G_w(X_1) - G_w(X_2)|$. A função de perda combina o label com a energia (distância) gerada para produzir a perda escalar *Ls* ou *LD* a depender do label Y. Através do processo de *back-propagation*, o gradiente da função de perda em relação ao vetor de parâmetros que é compartilhado entre as duas redes, é computado. A cada passo o vetor de parâmetros é atualizado de acordo com um método de gradiente estocástico usando a soma dos gradientes fornecidos pelas duas redes.

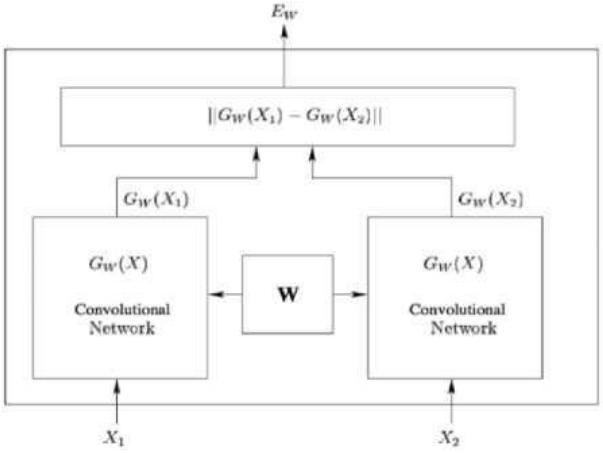

Figura 2: Arquitetura Siamesa

O compartilhamento dos parâmetros (pesos) entre as duas redes, garante que duas imagens semelhantes possam ser mapeadas, por suas respectivas redes, em locais próximos, já que cada uma das redes calcula a mesma função.

Utilizando redes convolucionais, elas são treinadas de ponta a ponta para conseguir mapear as imagens em saídas. Além disso, por conta do compartilhamento dos pesos e das múltiplas camadas, elas podem aprender detectores de características locais com invariantes de deslocamento, mantendo a invariância a distorções geométricas das imagens, como foi relatado na seção 2.3.

Para esse trabalho foram testadas diversas arquiteturas, porém só será apresentada a que forneceu melhores resultados. Para fins de leitura, C_x representa uma camada convolucional, MP_X uma camada max pooling, D_x representa uma camada de dropout e F_x uma camada completamente conectada. A arquitetura final ficou da seguinte forma:

- **C_1:** com 3 filtros de tamanho 3x3 e função de ativação ReLu.
- ***MP_1:*** com janela de convolução de tamanho 3x3.
- ***D_1:*** com 20% de taxa de *drop.*
- ***C_2:*** com 5 filtros de tamanho 3x3 e função de ativação ReLu.
- ***MP_2:*** com janela de convolução de

tamanho 3x3.

- ***D_2:*** com 20% de taxa de *drop.*
- **C_3:** com 7 filtros de tamanho 5x5 e função de ativação ReLu.
- ***MP_3:*** com janela de convolução de tamanho 3x3.
- ***D_3:*** com 20% de taxa de *drop.*
- **C_4**: com 9 filtros de tamanho 7x7 e função de ativação ReLu.
- ***MP_4:*** com janela de convolução de tamanho 3x3.
- **D_4:** com 20% de taxa de *drop.*
- ***C_5:*** com 11 filtros de tamanho 9x9 e função de ativação ReLu.
- **F_1:** com 11 nós e função de ativação sigmoid.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
conv2d_1 (Conv2D)	(None, 224, 224, 3)	84
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 3)	0
dropout_1 (Dropout)	(None, 111, 111, 3)	0
conv2d_2 (Conv2D)	(None, 111, 111, 5)	140
max_pooling2d_2 (MaxPooling2	(None, 55, 55, 5)	0
dropout_2 (Dropout)	(None, 55, 55, 5)	0
conv2d_3 (Conv2D)	(None, 55, 55, 7)	882
max_pooling2d_3 (MaxPooling2	(None, 27, 27, 7)	0
dropout_3 (Dropout)	(None, 27, 27, 7)	0
conv2d_4 (Conv2D)	(None, 27, 27, 9)	3096
max_pooling2d_4 (MaxPooling2	(None, 13, 13, 9)	0
dropout_4 (Dropout)	(None, 13, 13, 9)	0
conv2d_5 (Conv2D)	(None, 13, 13, 11)	8030
flatten_1 (Flatten)	(None, 1859)	0
dense_1 (Dense)	(None, 11)	20460

Total params: 32,692
Trainable params: 32,692
Non-trainable params: 0

Figura 3: Sumário das redes siamesas.

Essa arquitetura proposta gerou um total de 32 mil parâmetros compartilhados entre as redes como mostrado na Figura 3.

Dado que filtros menores conseguem coletar o máximo possível de informações locais, as duas primeiras camadas convolucionais devem ser responsáveis por detectar as bordas nas imagens. As duas últimas camadas convolucionais devem ser responsáveis por identificar as informações de mais alto nível, tais como formas e acessórios. O número de filtros vai aumentando gradativamente para aumentar a profundidade do espaço de recursos, fazendo com que a rede consiga aprender mais níveis de estruturas abstratas globais. Além disso, outra utilidade de tornar o espaço de recursos mais profundo e estreito é reduzir o espaço de recursos para entrada na camada completamente conectada. Como as informações nas bordas das imagens são importantes para o treinamento, optou-se por utilizar o *padding = 'same*'.

4 EXPERIMENTO

Para aplicar o modelo descrito anteriormente foi desenvolvida uma arquitetura siamesa de redes convolucionais, que será demonstrada mais adiante, treinada e testada em uma base de dados de imagens de faces de 1000 pessoas distintas. Essa base de dados foi obtida em https://cyberextruder.com/.

Para o desenvolvimento da arquitetura do sistema, foi utilizada a biblioteca para Deep Learning em Python, Keras. O Keras é uma API de alto nível que roda sobre TensorFlow, CNTK ou Theano.

4.1 Pré-processamento de Dados

O conjunto de dados utilizado, possui subpastas nomeadas de 000001 a 001000, representando cada indivíduo presente na base. Cada subpasta dispõe de uma quantidade diferente de imagens, chegando a ter, em alguns casos, apenas uma amostra de imagem. Isto gerou um problema ao criar o conjunto de pares para treinamento, já que nesses casos, não é possível gerar um par genuíno (Y = 0) para as pessoas correspondentes a subpasta. Para evitar problemas com repetições de imagens no mesmo par, caso só exista uma amostra para um indivíduo, só são gerados pares impostores (Y = 1) com o mesmo.

Todas as imagens têm dimensões 600x600, são dispostas em diversos espaços de cores, tais como RGB e escala de cinza, variações na luminosidade, expressões faciais e poses diversas, acessórios e posicionamentos diversos, além de diversos ruídos, tais como marcas d'água (Ver Figura 4).

DOI: 1025286/repa.v5i1.1192

Figura 4: Imagens do Cyberextruder Ultimate Face Data Set

Foi necessário realizar um pré-processamento nas imagens para alcançar bons resultados. Inicialmente foi feito um alinhamento das faces para que elas fossem dispostas de forma que olhos sempre ficassem na horizontal (ver Figura 5). Para tal, foram utilizadas as bibliotecas OpenCv e dlib. Após identificar a face em cada imagem através das bibliotecas citadas, foi preciso detectar estruturas faciais chaves na região da face. Existe uma grande quantidade de detectores de pontos chave na face, porém todos tentam localizar essencialmente boca, sobrancelhas, olhos e nariz. O detector de pontos de referência incluído na biblioteca dlib começa usando:

1. Um conjunto de treinamento de marcos faciais em uma imagem. Essas imagens são rotuladas manualmente, especificando coordenadas (x,y) das regiões ao redor de cada estrutura facial.
2. A probabilidade entre pares de pixels de entrada

Dado esses dados de treinamento, um conjunto de árvores de regressão é treinado para estimar as posições de cada marco facial diretamente das próprias intensidades de pixel, nenhuma extração de característica é feita.

O resultado final é um detector facial que pode ser usado para detectar características faciais em tempo real e com previsões de alta qualidade **[9]**.

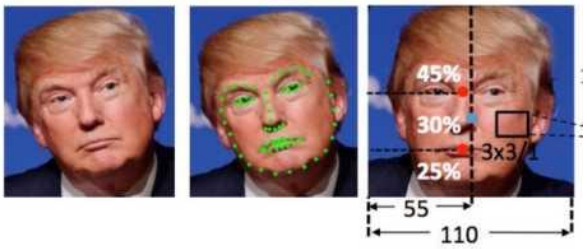

Figura 5: Imagem com marcas corrigidas horizontalmente

Em seguida, a detecção de faces foi utilizada para marcar pontos e fazer um corte na imagem, retornando uma nova imagem apenas com a face. Por fim, o espaço de cores foi modificado para que todas as imagens fossem transformadas para escala de cinza, como mostra a Figura 6.

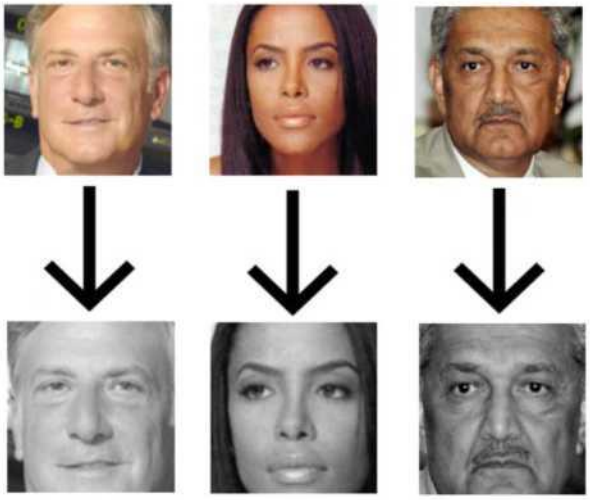

Figura 6: Imagens ajustadas para treinamento.

Com as imagens corretamente ajustadas, o conjunto de dados foi particionado em 70% para treinamento e 30% para testes. Para dividir, foi utilizada a biblioteca *scikit-learn* que possui a função *"train_test_split()"* capaz de varrer um diretório e seus sub diretórios e realizar um *shuffle* para escolher de forma aleatória as imagens utilizadas para o treinamento e para teste, sem repeti-las.

4.2 Treinamento

Para a fase de treinamento, são necessários dois conjuntos de dados:

1. Conjunto de treinamento: conjunto responsável por aprender os pesos da rede.

2. Conjunto de validação: conjunto responsável por testar a performance do sistema.

A avaliação da função de perda de cada conjunto a cada época da fase de treinamento é essencial para evitar *over-fitting.*

A porcentagem de pares impostores aceitos e a porcentagem de pares genuínos rejeitados foi utilizada para calcular a performance do sistema. De forma mais clara, esse cálculo foi realizado medindo a norma da diferença entre as saídas de um par e escolhendo um valor de *threshold* que determina um *trade-off* entre cada um dos dois percentuais.

Para avaliar os resultados do experimento foi utilizada a *Receiver Operation Characteristic* ou curva ROC. A curva ROC consiste da taxa de validação ou *Validation Rate* (VA) e da taxa de falsa aceitação ou *False Acceptance Rate* (FAR). Já que a tarefa do trabalho proposto é determinar quando duas imagens de face são da mesma pessoa, a curva ROC serve como ilustração da performance de classificação binária (sim ou não) **[10]**.

Definindo ***TP(thresh)*** como as amostras de teste, que foram classificadas como pares genuínos e ***FA(thresh)*** como pares impostores, que foram erroneamente classificados como pares genuínos, temos:

$$TP(thresh) = \{(X_1,X_2) \in \wp_{gen}, D_w \leq thresh\} \quad (5)$$
$$FA(thresh) = \{(X_1,X_2) \in \wp_{imp}, D_w \leq thresh\} \quad (6)$$

A taxa de validação e a taxa de falsa aceitação são calculadas como:

$$VR(thresh) = TP(thresh)\ \wp_{gen} \quad (7)$$
$$FAR(thresh) = FA(thresh)\ \wp_{imp} \quad (8)$$

4.3 Resultados

Utilizando um total de 14500 pares, sendo 10000 pares impostores e 4500 genuínos como conjunto de treino e 6300 pares, sendo 5100 impostores e 1200 genuínos como conjunto de testes, foi possível obter bons resultados com o CyberExtruder Ultimate Data Set. O modelo foi testado também no conjunto de dados LFW com um total de 17945 pares, sendo 11945 pares impostores e 6000 pares genuínos como conjunto de treino e 5734 pares, sendo 3583 impostores e 2151 genuínos como conjunto de testes (ver Tabela 1).

O LFW é um conjunto com melhores amostras que o CyberExtruder Ultimate Dataset, sendo uma base de dados extramamente utilizada para verificar a acurácia de outras abordagens e modelos como FaceNet.

Tabela 1: Resultado acurácia de treino e teste em duas bases de dados distintas: CyberExtruder e LFW.

Accuracy	CyberExtruder	LFW
Training Acc	94,34%	98,25%
Test Acc	86,20%	92,13%

Por se tratar de uma base de dados com imagens mais diversificadas e com menos ruídos que a base utilizada para a presente solução, percebesse um melhor desempenho em termos de acurácia.

O *threshold* definido para alcançar essa acurácia foi 0.4. Sendo assim, ao enviar um par de imagens para o sistema, caso a distância mapeada entre as imagens seja menor que 0.4, as imagens devem pertencer a mesma pessoa, caso seja maior, as imagens devem pertencer a pessoas diferentes.

Para avaliar melhor como o sistema se comporta, basta verificar as curvas das funções de perda de cada um dos conjuntos de treinamento e teste, mostradas na Figura 7.

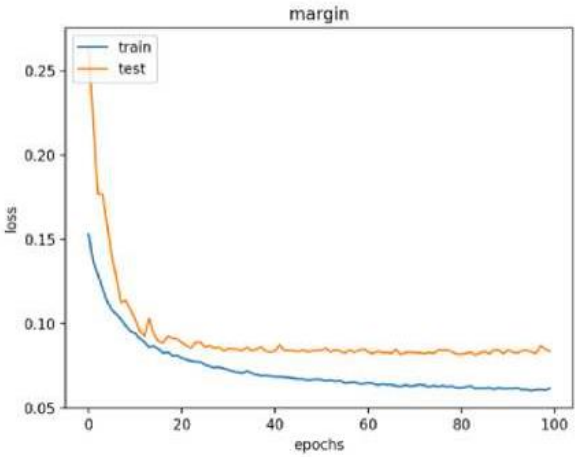

Figura 7: Gráfico das funções de perda para cada conjunto.

A Figura 8, mostra imagens de duas pessoas distintas e as respectivas distâncias:

DOI: 1025286/repa.v5i1.1192

Figura 9: Comparação em imagens de pessoas não vistas durante a fase de treinamento e as distâncias entre elas.

5. CONCLUSÕES

Este trabalho apresentou um sistema de comparação de imagens de faces que utiliza uma métrica complexa de similaridade. O sistema se adequa para os casos de verificação de face em que o número de classes na base de dados é muito grande ou os exemplos por classe são poucos, além de conseguir verificar similaridade entre imagens não vistas durante a fase de treinamento

Para tratar a diferença de resolução entre as imagens, as imagens são convertidas para escala em cinza depois redimensionadas para um tamanho pré-definido em comum. A partir dessa conversão é calculada uma média dos elementos de cada matriz das imagens.

Para otimização e aprendizagem da rede, foi utilizada uma função de perda contrastiva que ao ser minimizada, consegue mapear um vetor de características extraídos de uma imagem, em um espaço de baixa dimensionalidade. Dessa forma, é calculada a distância euclidiana entre os pares de pontos mapeados no espaço de destino e essa distância determina o quão similares ou dissimilares as imagens das faces são.

REFERÊNCIAS

[1] CHOPRA, S.; HADSELL, R.; LECUN, Y. **Learning a Similarity Metric Discriminatively, with Application to Face Verification**. Proceedings of Computer Vision and Pattern Recognition. vol.1. p.539-546, 2005.

[2] TURK, M.; PENTLAND, A. **Eigenfaces for recognition**. Journal of Cognitive Neuroscience, vol. 3(1), 1991.

[3] YANG, M. H.; AHUJA, N.; KRIEGMAN, D. **Face recognition using kernel eigenfaces**. Proceedings of the IEEE International Conference on Image Processing (ICIP), 1:37 40, 2000.

[4] SIMARD, P. Y.; *et al.* **Transformation invariance in pattern recognition tangent distance and tangent propagation**. International Journal of Imaging Systems and Technology, 11(3), 2000.

[5] LADES, M.; *et al*. **Distortion invariant object recognition in the dynamic link architecture**. IEEE Transactions on Computers, 42(3):300-311, 1993.

[6] MARTINEZ, A. M**. Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(6):748-763, 2002.

[7] SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. **FaceNet: A Unified Embedding for Face Recognition and Clustering**. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[8] TAIGMAN, Y. *et al.* **DeepFace: Closing the Gap to Human-Level Performance in Face Verification**. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[9] KAZEMI, V.; SULLIVAN, J. **One millisecond face alignment with an ensemble of regression trees**. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 1867-1874, 2014.

[10] VATSA, M.; SINGH, R.; MAJUMDAR, A. **Deep Learning in Biometrics**. CRC Press, 2018.