Trabalho final – Reconhecimento de Faces

Bruno Reinoso

Eduarda Machado

Matheus Sales

NOVA FRIBURGO

2019

Sumário

1. Introdução:

Uma das áreas estudadas na matéria de Álgebra Linear Numérica foi a de *machine learning* e análise de dados. Desta forma, neste trabalho, foi posto em prática parte do conhecimento adquirido nesta disciplina. Além disso, somados à estudos feitos fora de sala de aula, utilizamos destes mesmos conhecimentos para criar um reconhecedor de faces. Ou seja, a partir da análise de um banco de dados previamente fornecido pelo professor, foi possível criar um algoritmo capaz de reconhecer as pessoas, com determinada taxa de acerto, por meio de duas formas distintas: análise bruta dos dados e artifício do PCA (análise dos componentes principais).

2. Desenvolvimento Teórico:

2.1 – Machine Learning:

Traduzindo ao pé da letra, “machine learning” significa “aprendizado de máquina”. É um termo utilizado para designar a área da computação que trabalha com o conceito de inteligência artificial. Ou seja, a área que estuda como as máquinas podem aprender a executar tarefas que são feitas por pessoas.

Nesta área, há a presença de códigos que possuem regras previamente estabelecidas e que permitem os computadores (ou as máquinas, como preferir) tomarem decisões com base em testes e dados já feitos e fornecidos, respectivamente. Desta forma, pode-se dizer que o computador “aprende” com suas falhas, o que fica sugestivo já pelo termo em si.

Atualmente, este tipo de tecnologia é utilizado diariamente em nossas vidas. Como principais exemplos práticos temos o Spotify e o Youtube que, a partir de dados fornecidos por nós usuários, conseguem indicar músicas e vídeos que se adequam aos nossos gostos singulares e individuais.

Neste trabalho, o conceito de *machine learning* se aproximou ao utilizado pela rede social Facebook, onde é possível, por meio de fotos antigas, reconhecer determinada pessoa antes mesmo de indicarmos que ela está presente na nova imagem em questão.

2.2 Álgebra Linear Numérica e Machine Learning

Ao longo do curso de ALN, pudemos estabelecer uma ponte entre a utilização de matrizes e a área de *machine learning*. Por meio de processos como o SVD conseguimos as ferramentas necessárias para entender e replicar o processo de aprendizado de máquina. Não somente isso, no trabalho em questão iremos demonstrar artifícios utilizados no algoritmo que tornaram possível o reconhecimento de faces, de acordo com suas estratégias individuais.

2.3 Análise Bruta de Dados X Principal Component Analysis (PCA)

2.3.1 Análise Bruta de Dados

Neste tipo de análise, a ideia é comparar uma imagem com a outra por meio de cada pixel. Ou seja, posta duas imagens “lado a lado”, o computador irá comparar pixel a pixel afim de encontrar semelhanças entre ambas. Por conta desta comparação milimétrica, a demora é consideravelmente grande e também pode não ser tão assertiva quando comparada ao outro método utilizado (PCA).

2.3.2 PCA

Já neste tipo de análise, o computador busca encontrar semelhanças entre as imagens, correlações entre ambas. A partir de uma transformação linear ortogonal, é feita a conversão de um conjunto de observações de variáveis possivelmente relacionadas em um conjunto de valores de variáveis linearmente não correlacionadas, que são chamadas de componentes principais.

Portanto, como é criado um “padrão” para o reconhecimento de determinado objeto, esta análise se torna mais ágil que a citada anteriormente.

3. Desenvolvimento

3.1 Organização das imagens

As imagens foram separadas em pastas que representavam o nível de dificuldade da análise que seria feita posteriormente. Desta forma, as pastas criadas foram: very-easy, easy, médium, hard e extras.

Cada imagem possui dois números, separados por um traço (-). O primeiro número representa a pessoa em questão e o segundo número a imagem em questão. Como exemplo: “15 – 1” representa a pessoa 15 e sua primeira imagem.

3.1.1 Very-easy

Nesta etapa as imagens analisadas eram simples e cada uma das 5 pessoas possuía apenas duas fotos, uma sorrindo e uma sem sorrir.

3.1.2 Easy

Nesta etapa cada uma das 5 pessoas possuía 8 imagens, sendo cada uma delas semelhantes àquelas duas na etapa anterior. A única diferença é que foram aplicados determinados filtros para escurecer ou embaçar um pouco as imagens adicionais.

3.1.3 Medium

Nesta etapa cada uma das 50 pessoas possuía 13 imagens. Foram aplicados determinados filtros para escurecer algumas das imagens e, além disso, nesta etapa as pessoas estavam com os rostos um pouco inclinados.

3.1.4 Hard

Nesta etapa cada uma das 50 pessoas possuía 14 imagens. Os rostos dos fotografados estavam mais inclinados que na fase anterior, chegando a ficarem totalmente inclinados para ambos os lados, esquerdo e direito. Além disso, algumas das imagens estavam bem mais escurecidas que nas fases anteriores.

3.1.5 Extras

Nesta fase as imagens utilizadas foram retiradas do Facebook, ou seja, com as mais diversas diferenças de luz e angulação. Foi separada em duas subfases: na primeira as imagens eram apenas

do rosto, já na segunda era possível ver a cabeça inteira, junto com o cabelo. Ambas as subfases possuem 5 pessoas, cada uma com 9 fotos.

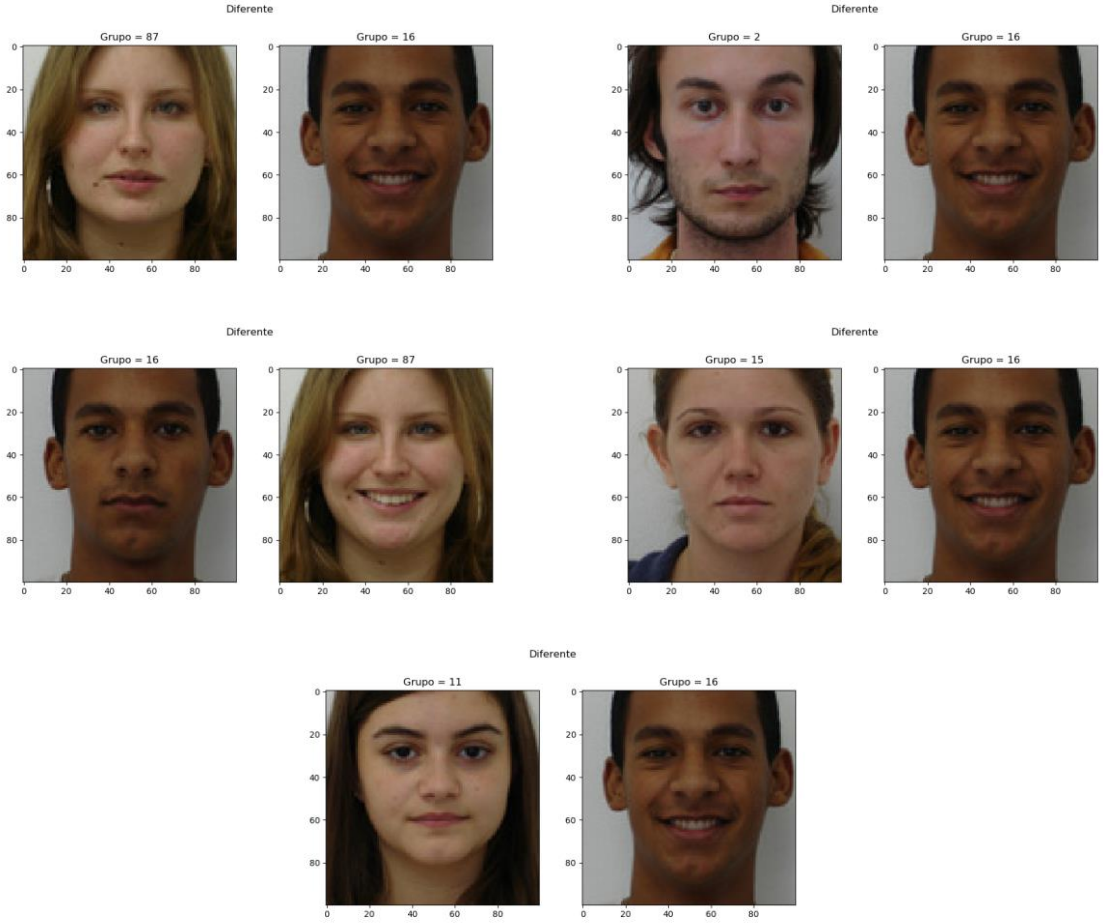
4. Análise de Resultados

A análise dos resultados foi baseada nas taxas de “certo” e “errado” recebidas de acordo com o nível de dificuldade em questão. Cada nível tem sua quantidade de fotos e também a “diferença” nas imagens da mesma pessoa, estando de frente ou de lado e em um ambiente claro ou escuro, por exemplo.

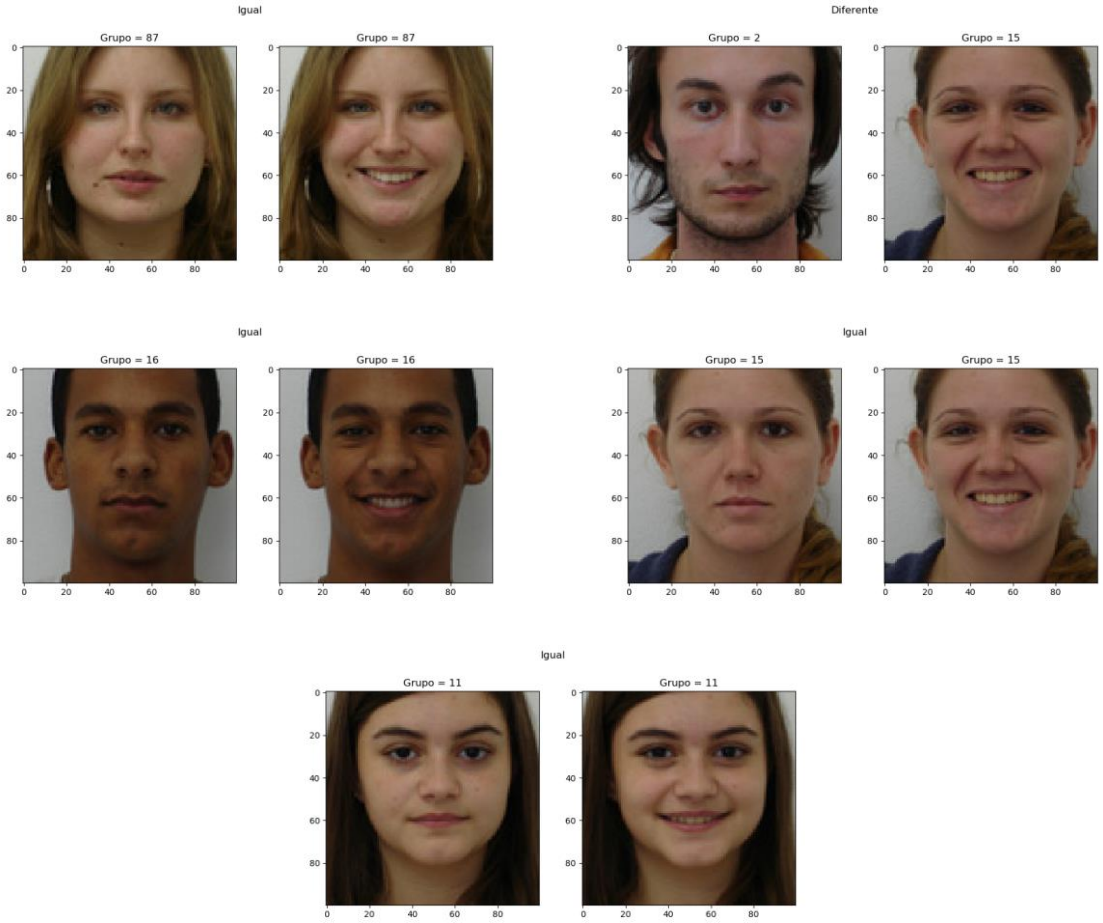
Nesta exposição de dados, levou-se em consideração apenas os níveis very-easy, easy e medium, para demonstração de código e exemplificação. Além disso, como dito anteriormente, as análises se basearam de duas formas: dados brutos e por PCA.

4.1 Very-easy (Rodado apenas 1 vez)

Bruto: 0% de taxa de acerto



PCA: 80% de taxa de acerto



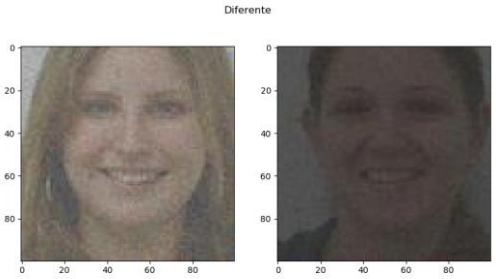
4.2 Easy (Rodado pela segunda vez)

Força Bruta: 100.0%

PCA: 100.0%

Resultados força bruta:

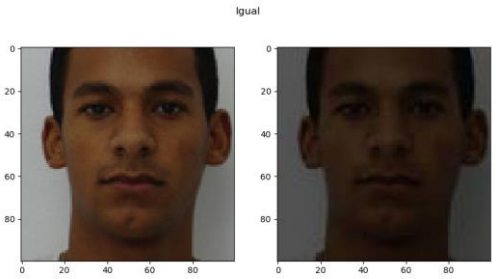
80.0% de acerto



DIFERENTE 87 != 15

min_norm_difference = 9678.646754582998

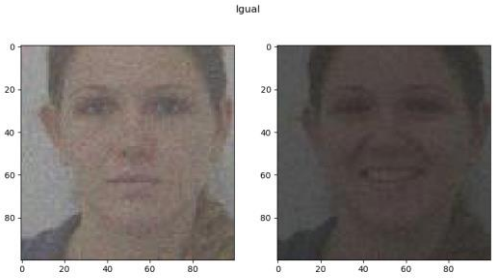
class = 15



IGUAL

min_norm_difference = 8122.45418577415

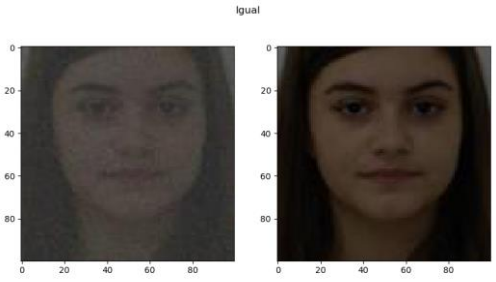
class = 16



IGUAL

min_norm_difference = 9519.894537230966

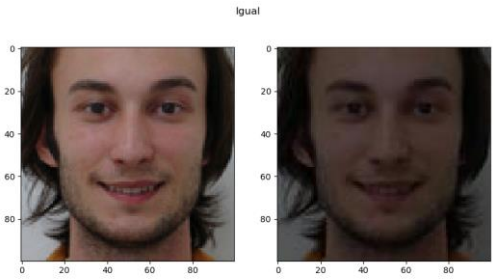
class = 15



IGUAL

min_norm_difference = 7922.699098160929

class = 11



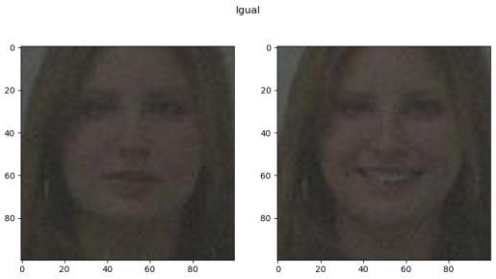
IGUAL

min_norm_difference = 8837.156612847823

class = 2

Resultados PCA:

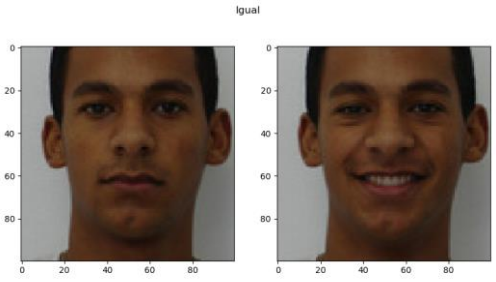
100.0% de acerto



IGUAL

min_norm_difference = 554.1848983498267

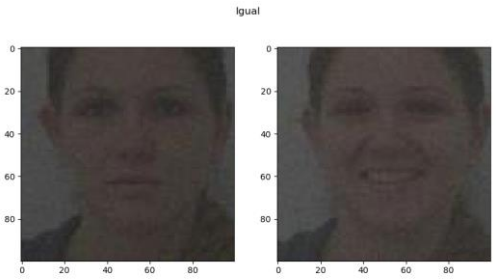
class = 87



IGUAL

min_norm_difference = 1313.0646928920478

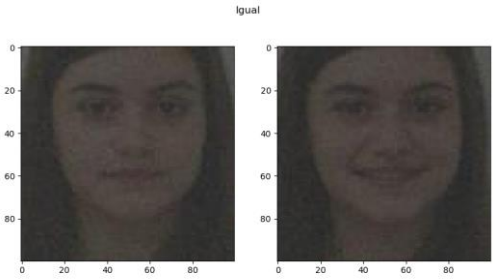
class = 16



IGUAL

min_norm_difference = 388.87692941009954

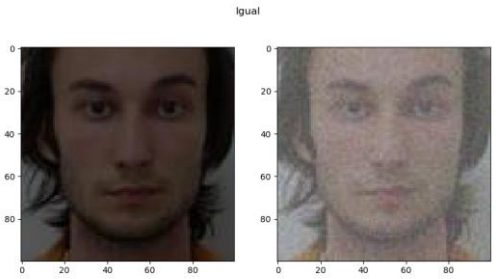
class = 15



IGUAL

min_norm_difference = 372.71635228187523

class = 11



IGUAL

min_norm_difference = 450.34882528074496

class = 2

4.3 Medium (Rodado pela segunda vez)

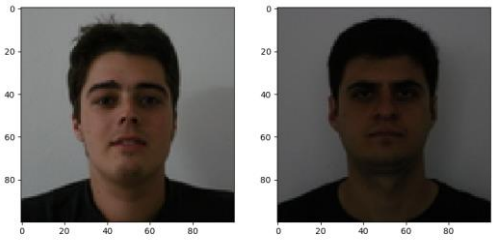
Força Bruta: 38.0% de acerto médio

PCA: 80.0% de acerto médio

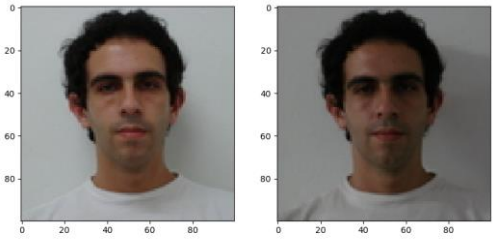
Resultados força bruta:

30.0% de acerto

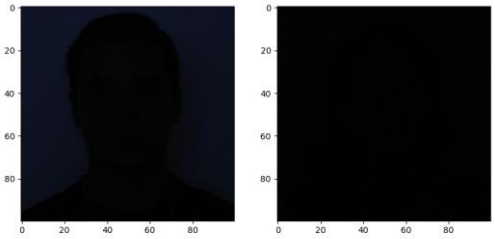
Diferente



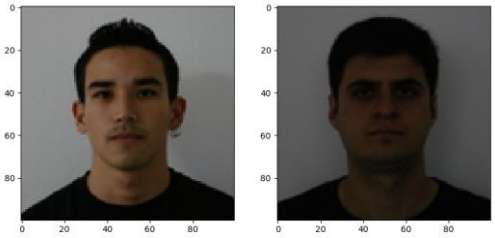
Igual



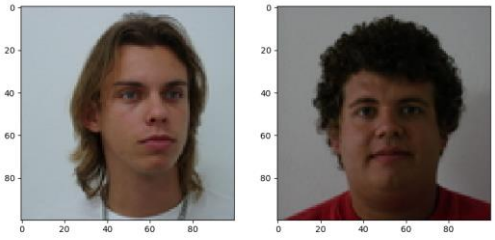
Diferente



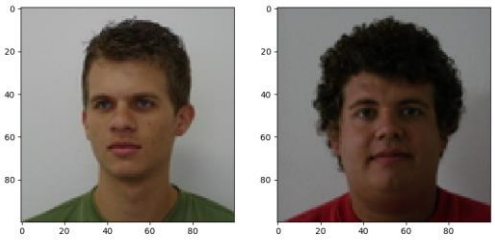
Diferente



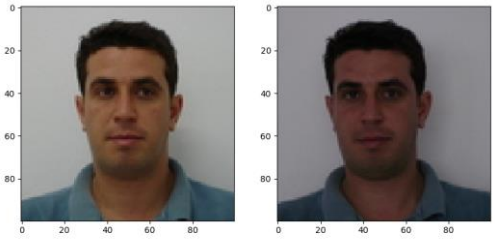
Diferente



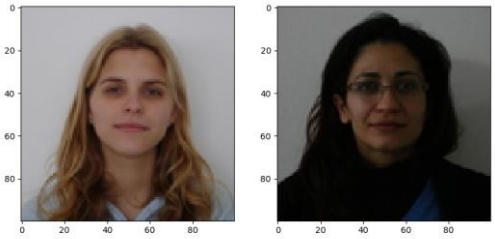
Diferente



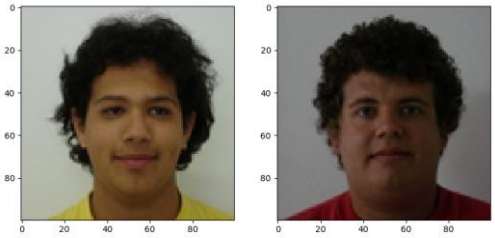
Igual



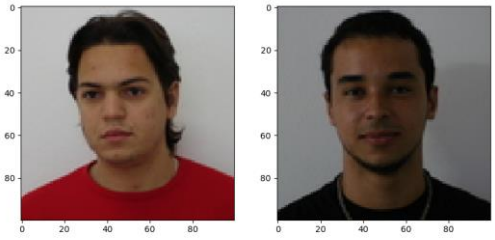
Diferente



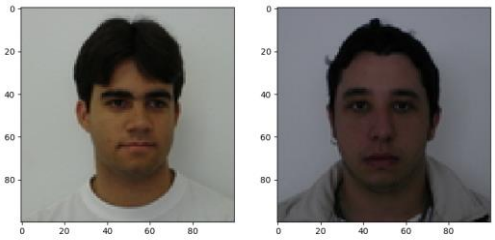
Diferente



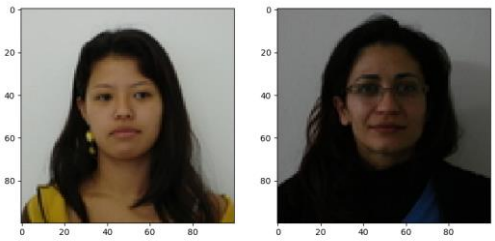
Diferente



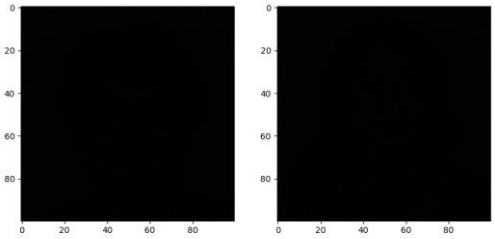
Diferente



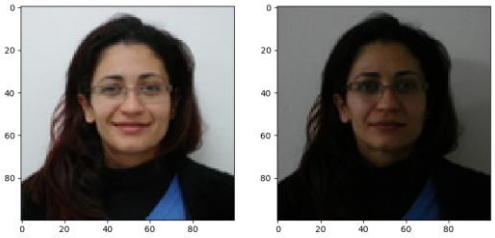
Diferente



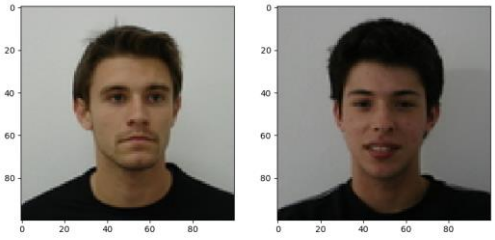
Diferente



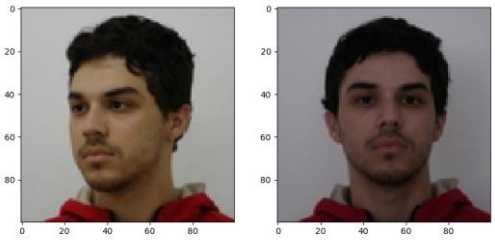
Igual



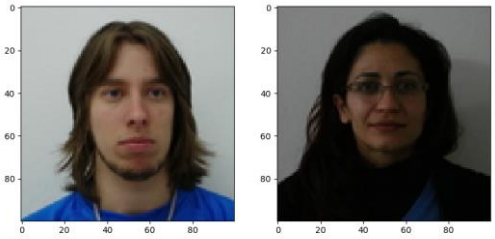
Diferente



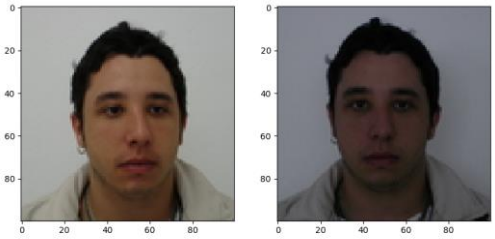
Igual



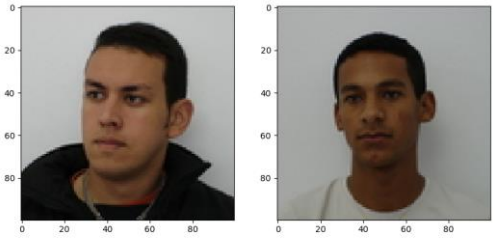
Diferente



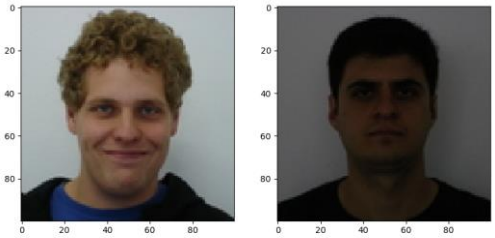
Igual



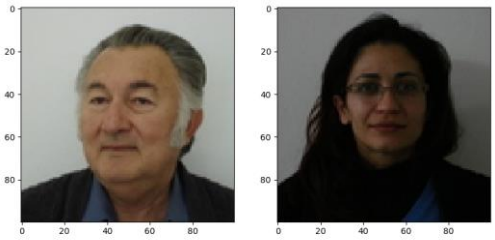
Diferente



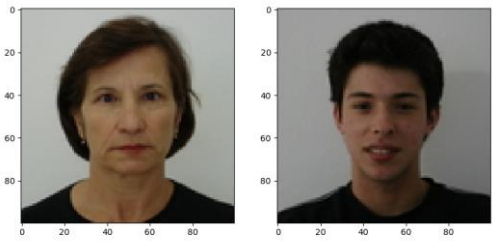
Diferente



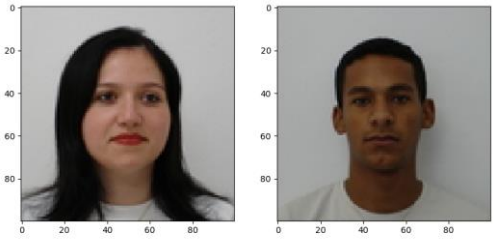
Diferente



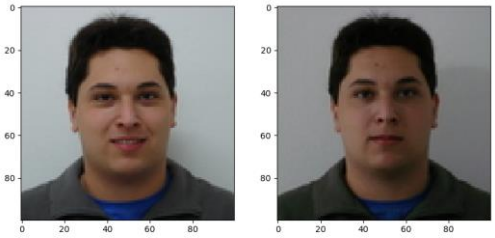
Diferente



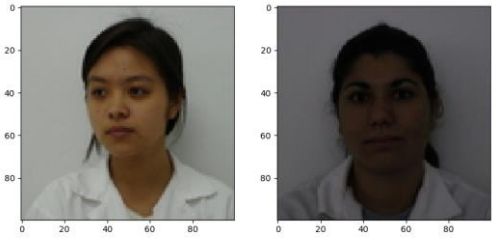
Diferente



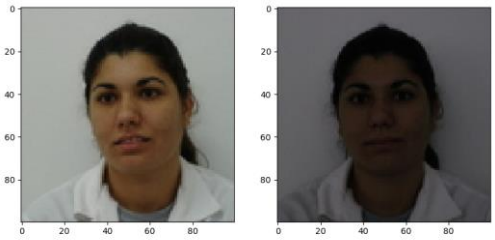
Igual



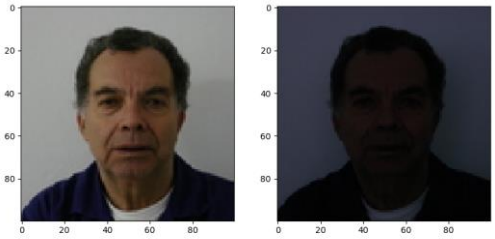
Diferente



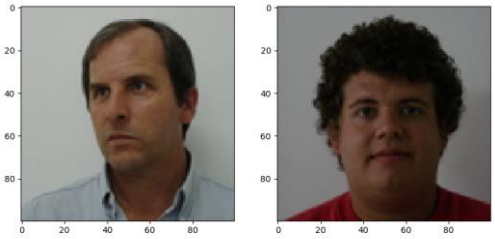
Igual



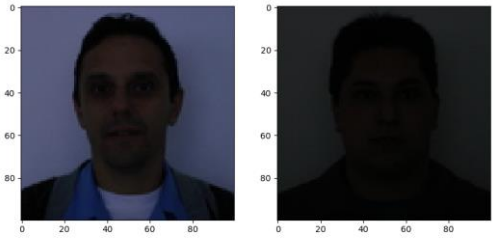
Igual



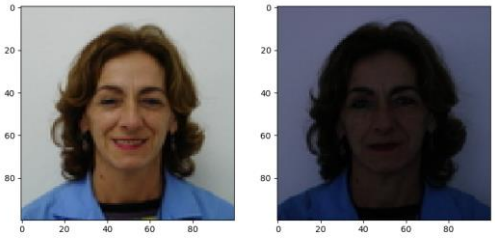
Diferente



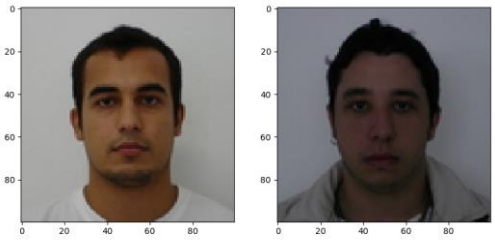
Diferente



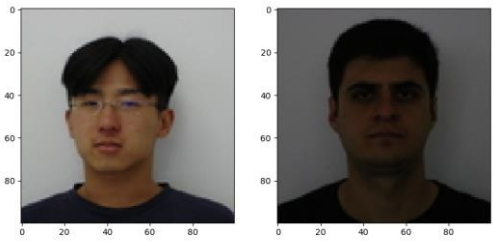
Igual



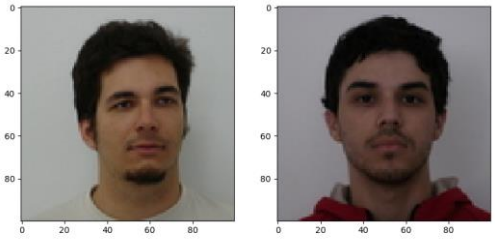
Diferente



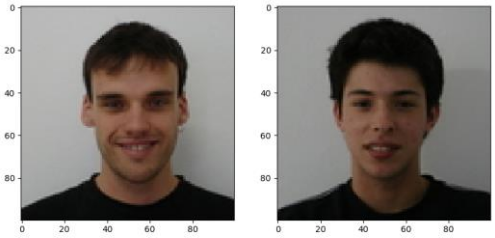
Diferente



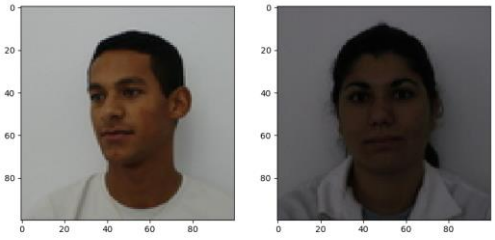
Diferente



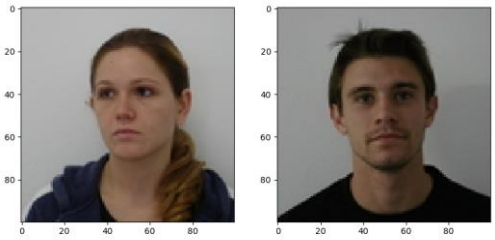
Diferente



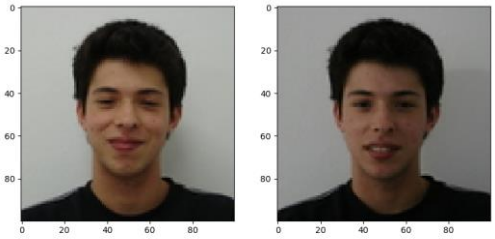
Diferente



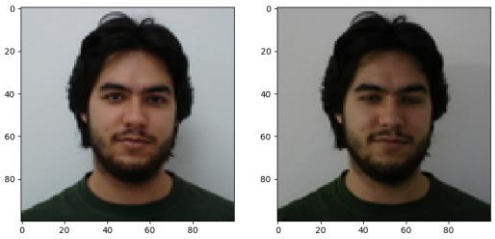
Diferente



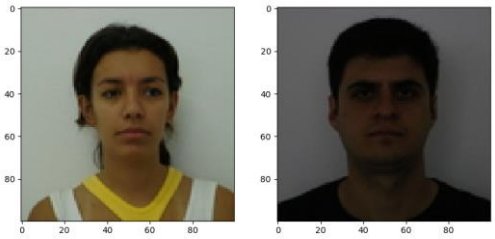
Igual



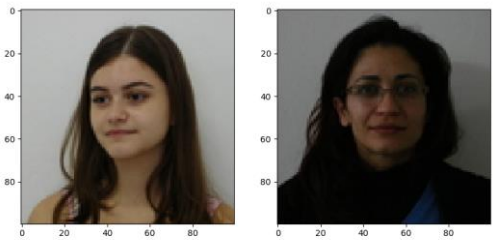
Igual



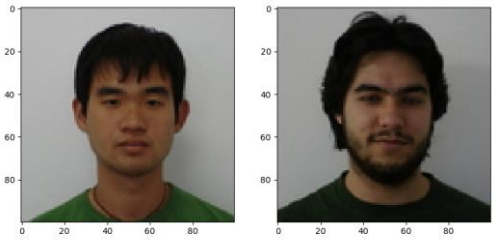
Diferente



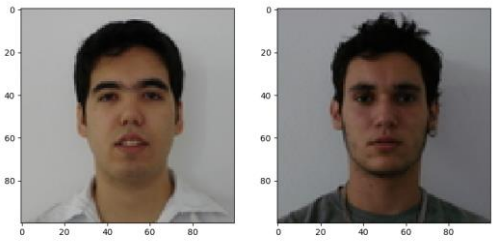
Diferente



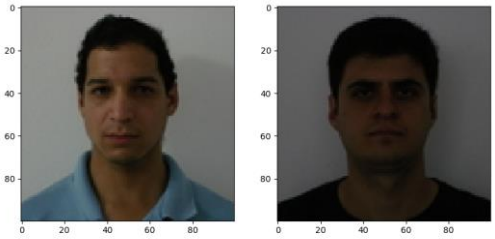
Diferente



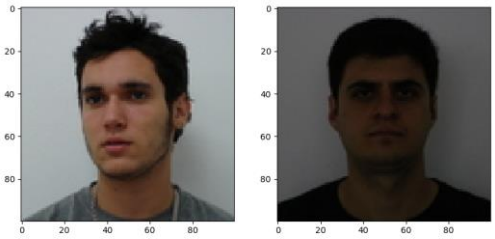
Diferente



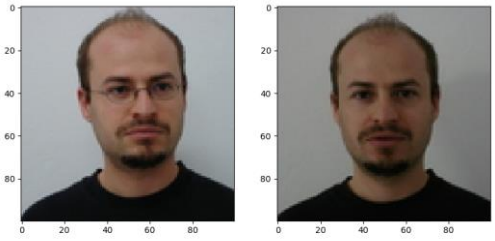
Diferente

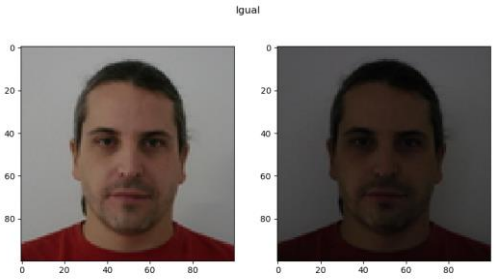


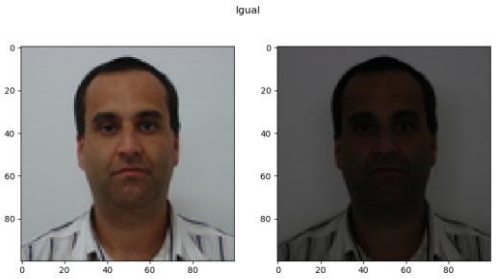
Diferente

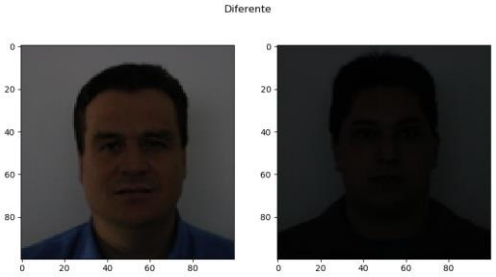


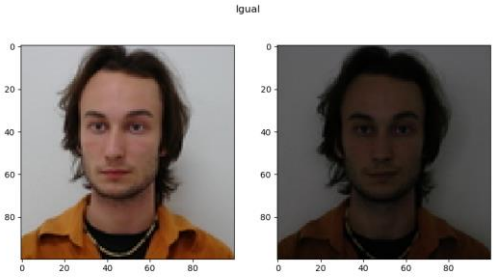
Igual

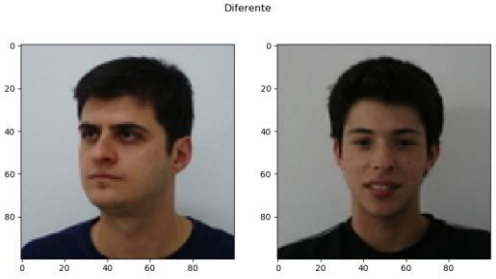



Igual


Igual


Diferente


Igual


Diferente

(Na ordem das imagens)

DIFERENTE 50 != 1
min_norm_difference = 12189.220852868324
class = 1
IGUAL
min_norm_difference = 9958.966663263815
class = 49
DIFERENTE 48 != 37
min_norm_difference = 9122.618428938042
class = 37
DIFERENTE 47 != 1
min_norm_difference = 14896.103282402415
class = 1
DIFERENTE 46 != 38
min_norm_difference = 16942.145702360136
class = 38
DIFERENTE 45 != 38
min_norm_difference = 13055.207428455513
class = 38
IGUAL
min_norm_difference = 12829.4127691021
class = 44
DIFERENTE 43 != 37
min_norm_difference = 15114.578459222737
class = 37
DIFERENTE 42 != 38
min_norm_difference = 17324.108837109052
class = 38
DIFERENTE 41 != 48
min_norm_difference = 16103.036639093882
class = 48
DIFERENTE 40 != 33
min_norm_difference = 15219.589054898954

class = 33
DIFERENTE 39 != 37
min_norm_difference = 17211.032566351154
class = 37
DIFERENTE 38 != 37
min_norm_difference = 23652.02179518698
class = 37
IGUAL
min_norm_difference = 12295.028385489803
class = 37
DIFERENTE 36 != 14
min_norm_difference = 13355.993111708316
class = 14
IGUAL
min_norm_difference = 17275.12338595589
class = 35
DIFERENTE 34 != 37
min_norm_difference = 16212.24469344082
class = 37
IGUAL
min_norm_difference = 13402.554644544449
class = 33
DIFERENTE 32 != 16
min_norm_difference = 16188.445632610934
class = 16
DIFERENTE 31 != 1
min_norm_difference = 16378.138264161773
class = 1
DIFERENTE 30 != 37
min_norm_difference = 15629.9822456713
class = 37
DIFERENTE 29 != 14
min_norm_difference = 14797.414199785042
class = 14

DIFERENTE 28 != 16
min_norm_difference = 16986.42711107901
class = 16
IGUAL
min_norm_difference = 14956.611648364746
class = 27
DIFERENTE 26 != 25
min_norm_difference = 13775.177276536226
class = 25
IGUAL
min_norm_difference = 14713.446707009205
class = 25
IGUAL
min_norm_difference = 12976.509469036733
class = 24
DIFERENTE 23 != 38
min_norm_difference = 14577.209712424392
class = 38
DIFERENTE 22 != 27
min_norm_difference = 11100.478908587684
class = 27
IGUAL
min_norm_difference = 14818.923881307981
class = 21
DIFERENTE 20 != 33
min_norm_difference = 12698.683711314334
class = 33
DIFERENTE 19 != 1
min_norm_difference = 17555.999088630644
class = 1
DIFERENTE 18 != 35
min_norm_difference = 14172.849043152897
class = 35
DIFERENTE 17 != 14

min_norm_difference = 14200.129576873585
class = 14
DIFERENTE 16 != 25
min_norm_difference = 14339.895048430446
class = 25
DIFERENTE 15 != 36
min_norm_difference = 14134.347703378462
class = 36
IGUAL
min_norm_difference = 11700.587848480092
class = 14
IGUAL
min_norm_difference = 11482.027042295276
class = 13
DIFERENTE 12 != 1
min_norm_difference = 14395.893824281979
class = 1
DIFERENTE 11 != 37
min_norm_difference = 16239.926662395985
class = 37
DIFERENTE 10 != 13
min_norm_difference = 15155.45070263501
class = 13
DIFERENTE 9 != 7
min_norm_difference = 12521.503464041369
class = 7
DIFERENTE 8 != 1
min_norm_difference = 11466.956091308626
class = 1
DIFERENTE 7 != 1
min_norm_difference = 15433.1083712906
class = 1
IGUAL
min_norm_difference = 14640.16926131662

class = 6

IGUAL

min_norm_difference = 10941.276616556224

class = 5

IGUAL

min_norm_difference = 13518.475468779754

class = 4

DIFERENTE 3 != 27

min_norm_difference = 8886.60064366572

class = 27

IGUAL

min_norm_difference = 16432.673580400726

class = 2

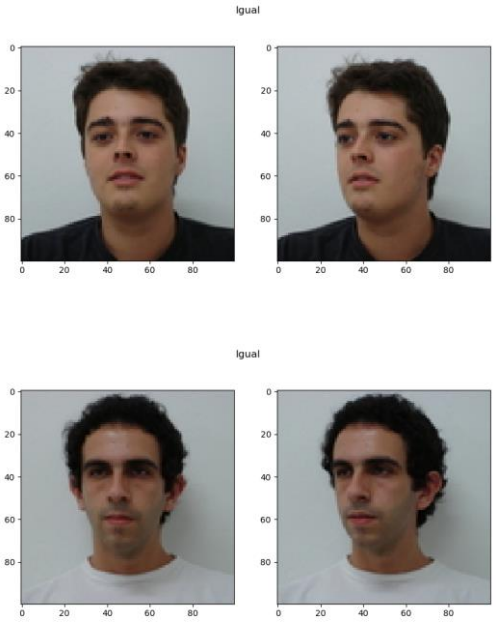
DIFERENTE 1 != 14

min_norm_difference = 15316.092517349194

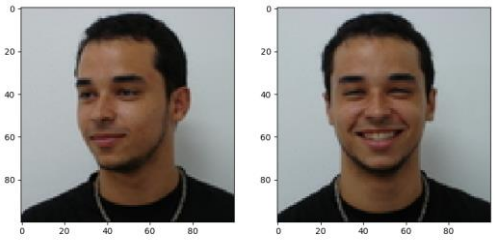
class = 14

Resultados PCA:

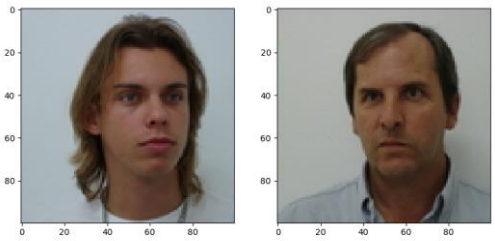
84.0% de acerto



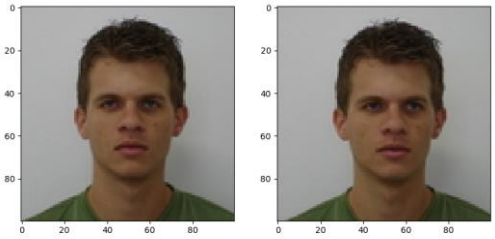
Igual



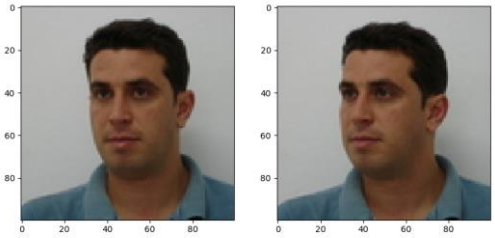
Diferente



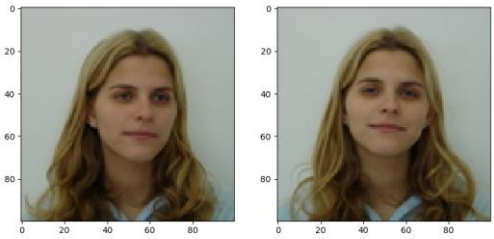
Igual

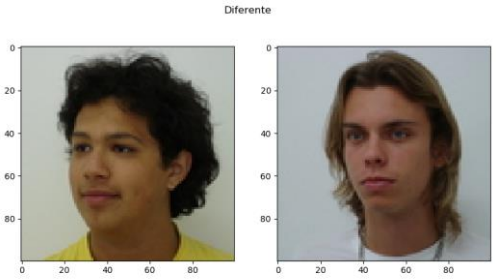


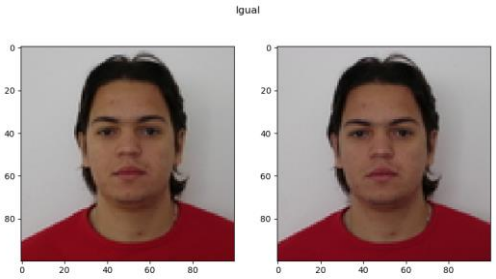
Igual

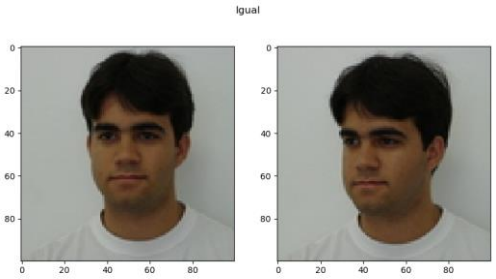


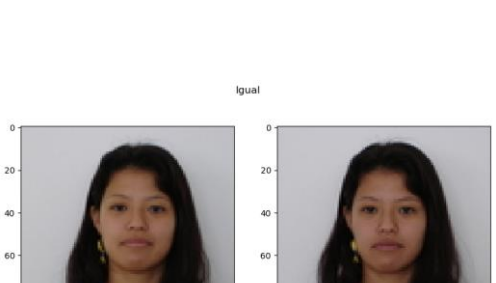
Igual

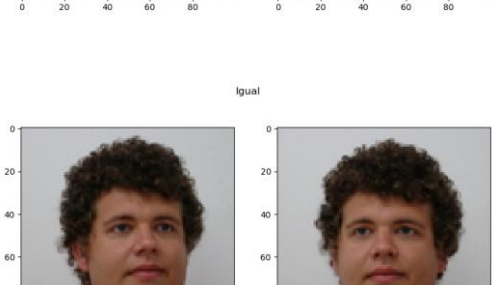


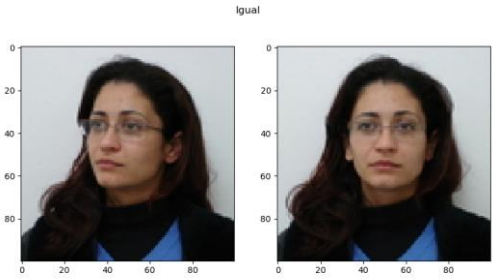

Diferente

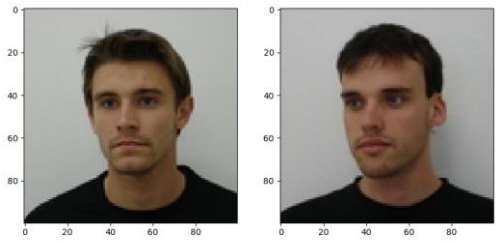

Igual

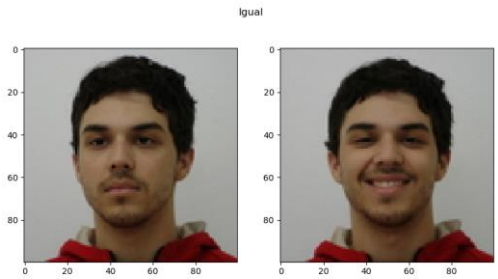

Igual

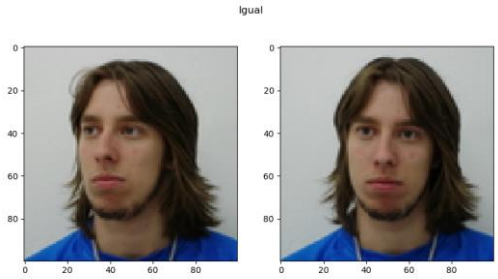

Igual

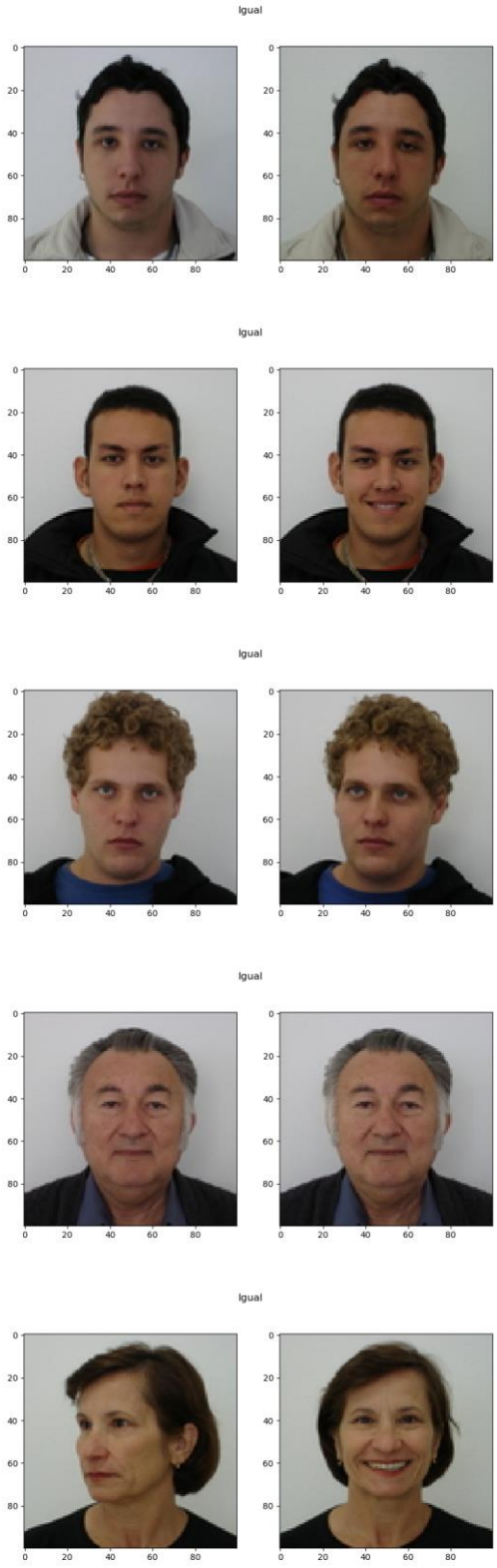

Igual

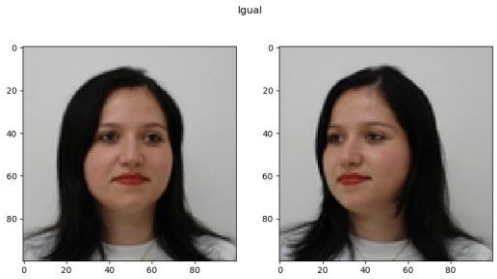

Igual

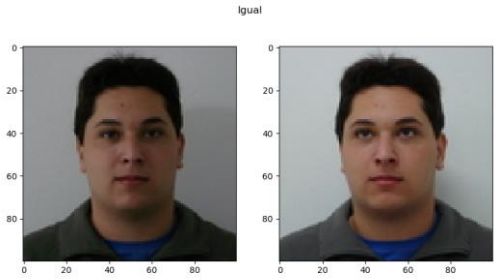

Diferente

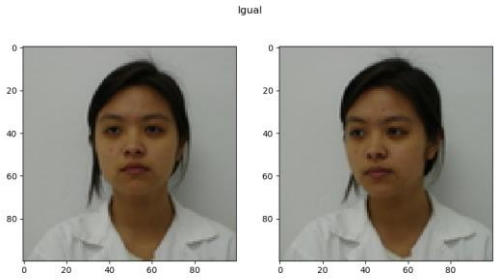

Igual

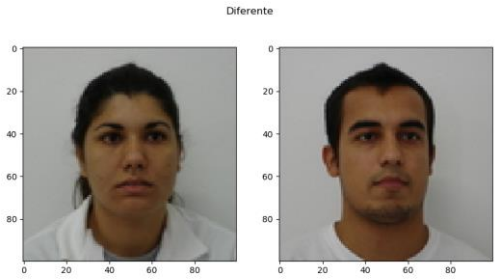

Igual

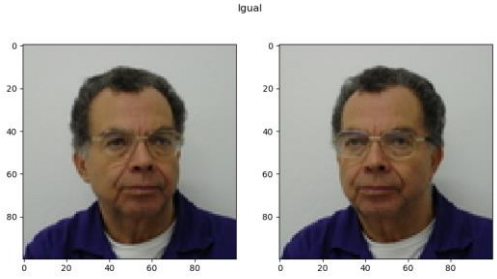

Igual
Igual
Igual
Igual
Igual

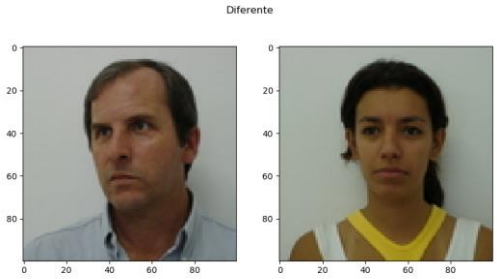

Igual

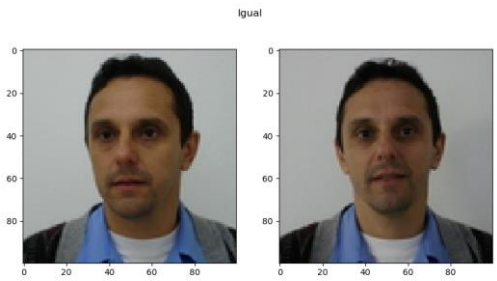

Igual


Igual

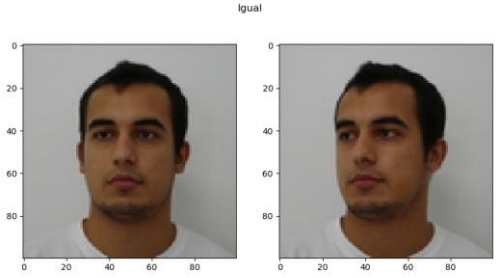

Diferente

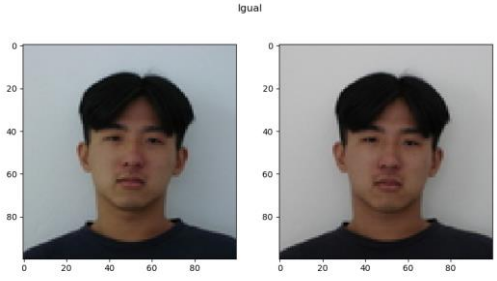

Igual

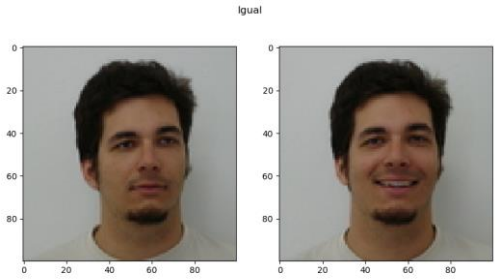

Diferente

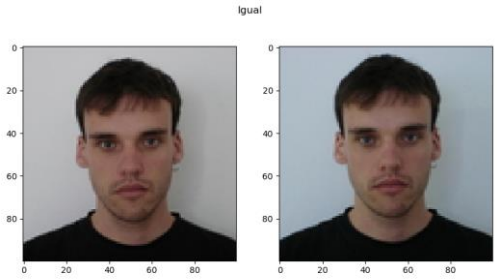

Igual

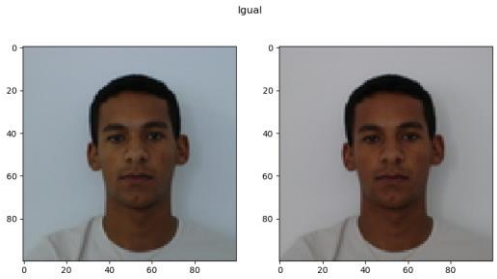

Igual

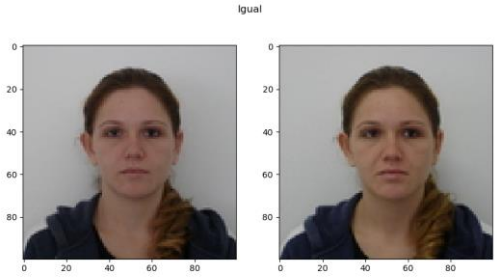

Igual

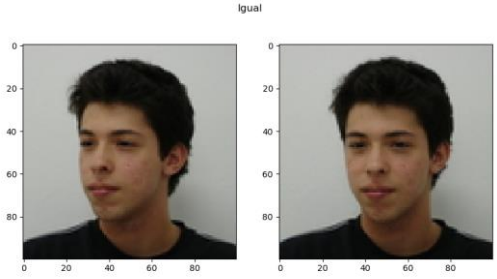

Igual

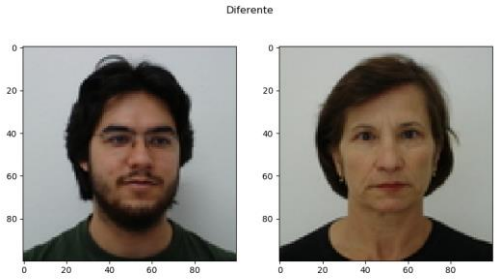

Igual

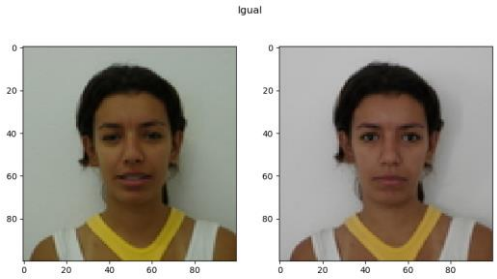

Igual

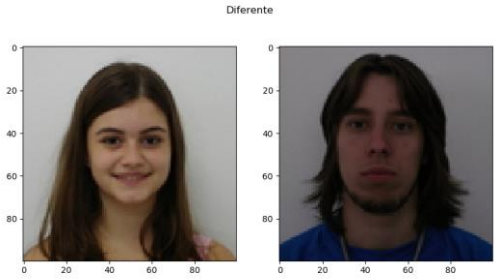

Igual

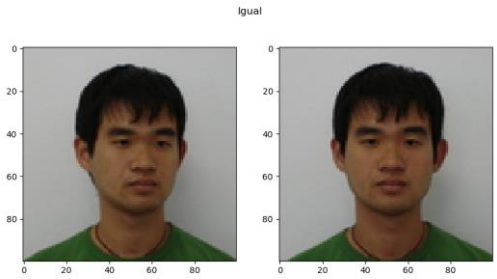

Igual

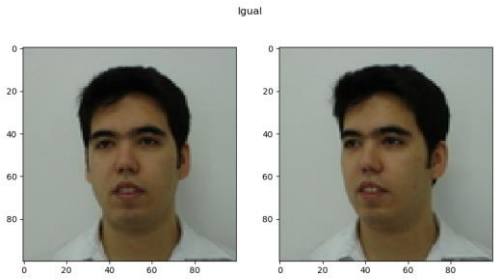

Igual

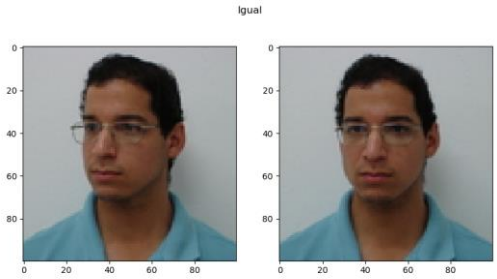

Diferente

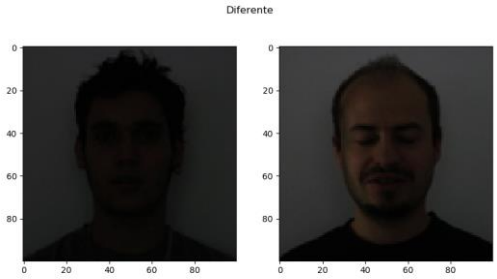

Igual

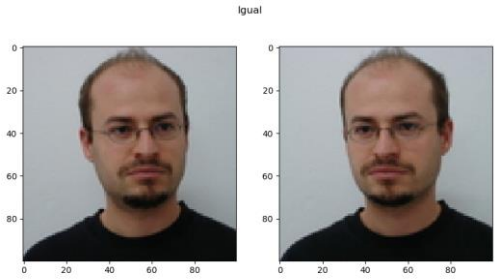

Diferente

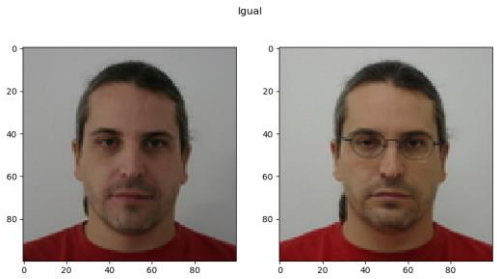

Igual

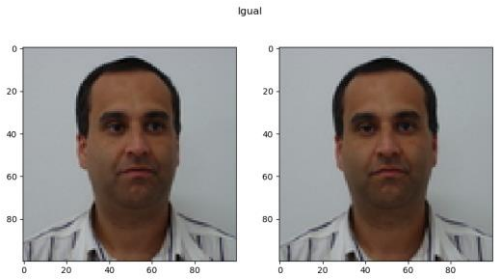

Igual

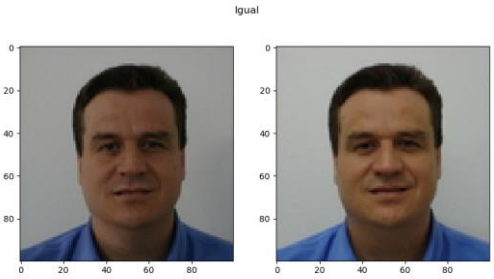

Igual

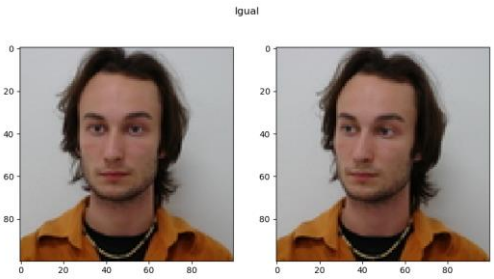

Diferente

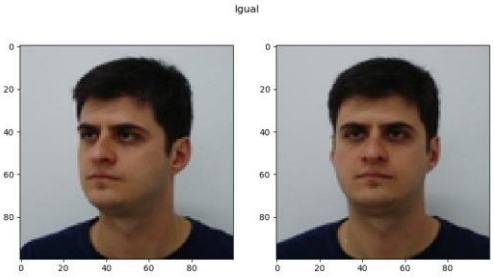

Igual


Igual


Igual







(De acordo com a ordem das imagens)

IGUAL

min_norm_difference = 2080.5505338089415

class = 50

IGUAL

min_norm_difference = 1977.978008087062

class = 49

IGUAL

min_norm_difference = 3434.350158706579

class = 48

IGUAL

min_norm_difference = 1047.84468564811

class = 47

DIFERENTE 46 != 23

min_norm_difference = 4196.522936645652

class = 23

IGUAL

min_norm_difference = 1132.40480823526

class = 45

IGUAL

min_norm_difference = 1723.4309346049386

class = 44

IGUAL

min_norm_difference = 1903.628882038613

class = 43

DIFERENTE 42 != 46

min_norm_difference = 5281.656889347549

class = 46

IGUAL

min_norm_difference = 1551.3736443520404

class = 41

IGUAL

min_norm_difference = 2279.8219867534535

class = 40

IGUAL

min_norm_difference = 2274.072499473703

class = 39

IGUAL

min_norm_difference = 2999.980134468237

class = 38

IGUAL

min_norm_difference = 4445.700609826283

class = 37

DIFERENTE 36 != 17

min_norm_difference = 2841.089361269463

class = 17

IGUAL

min_norm_difference = 1809.793776708662

class = 35

IGUAL

min_norm_difference = 3675.2357341347924

class = 34

IGUAL

min_norm_difference = 2085.728241762099

class = 33

IGUAL

min_norm_difference = 1141.7215810492962

class = 32

IGUAL

min_norm_difference = 2665.721845672829

class = 31

IGUAL

min_norm_difference = 393.91718679816296

class = 30

IGUAL

min_norm_difference = 7031.897235675996

class = 29

IGUAL

min_norm_difference = 8254.802345685777

class = 28

IGUAL

min_norm_difference = 2490.1488447881866

class = 27

IGUAL

min_norm_difference = 1568.501384791121

class = 26

DIFERENTE 25 != 20

min_norm_difference = 2129.7798927892864

class = 20

IGUAL

min_norm_difference = 2106.4523122132628

class = 24

DIFERENTE 23 != 12

min_norm_difference = 3298.972868078089

class = 12

IGUAL

min_norm_difference = 2867.8254347850216

class = 22

IGUAL

min_norm_difference = 1616.5269886351552

class = 21

IGUAL

min_norm_difference = 1515.0543201823937

class = 20

IGUAL

min_norm_difference = 1065.0497583185002

class = 19

IGUAL

min_norm_difference = 1451.7817539505263

class = 18

IGUAL

min_norm_difference = 2370.8656655592576

class = 17

IGUAL

min_norm_difference = 452.24365724966265

class = 16

IGUAL

min_norm_difference = 1526.8404357754443

class = 15

IGUAL

min_norm_difference = 2340.608690593688

class = 14

DIFERENTE 13 != 29

min_norm_difference = 4252.155670181927

class = 29

IGUAL

min_norm_difference = 1604.4781534437955

class = 12

DIFERENTE 11 != 34

min_norm_difference = 8304.892549993592

class = 34

IGUAL

min_norm_difference = 1305.9256812052242

class = 10

IGUAL

min_norm_difference = 1635.2554219376402

class = 9

IGUAL

min_norm_difference = 1563.246590260102

class = 8

DIFERENTE 7 != 6

min_norm_difference = 1773.049952364091

class = 6

IGUAL

min_norm_difference = 2005.8302904103543

class = 6

IGUAL

min_norm_difference = 1468.5567855011277

class = 5

IGUAL

min_norm_difference = 830.2483927421486

class = 4

IGUAL

min_norm_difference = 1502.2072598860102

class = 3

IGUAL

min_norm_difference = 1579.4486439091172

class = 2

IGUAL

min_norm_difference = 2508.227632273123

class = 1

Obs: Imagem 54 está faltando por erro do grupo, não do programa

5. Conclusões

De acordo com os estudos feitos para a realização do trabalho e os dados obtidos pelo mesmo, foi possível perceber claramente que o método de comparação pixel a pixel (bruto) possui um tempo elevado para sua realização quando comparado ao PCA em banco de dados pequenos.

Porém, em banco de dados maiores, a porcentagem de acerto da força bruta é boa, o que é justificável pois se tem mais imagens para comparação, e o tempo entre a mostra de um resultado para o outro é menor do que o PCA, cujo tempo de execução é maior quando se aumenta a base de dados. Ou seja, ambos os algoritmos são muito úteis, mas um será melhor do que o outro dependendo da situação.

6. Experiência dos membros

Eduarda Machado:

O trabalho possibilitou a aplicação de diversos conceitos e técnicas mostradas em sala de aula, de forma a possibilitar o entendimento de como funciona muitas ferramentas que utilizamos nos dias de hoje, como as redes sociais. Além disso, também tornou possível um maior contato com a linguagem de programação Phyton e as bibliotecas especiais para se trabalhar com esse tipo de problema.

Além disso, a experiência de melhorar um reconhecedor, entender cada parte de seu funcionamento e buscar outros meios de torná-lo mais eficiente, como por exemplo outro algoritmo além do PCA, trouxe incentivo para que esse tópico possa ser estudado de modo mais profundo no futuro.

Bruno Reinoso:

Com o trabalho proposto, pude perceber a importância da álgebra linear nos dias atuais da tecnologia. O tema em questão me proporcionou uma interpretação e entendimento sobre o reconhecimento de faces e como ele é utilizado no nosso dia a dia.

Participei não apenas na elaboração do código junto com os demais, mas também participei na confecção do relatório e pesquisas sobre o assunto abordado. Junto com o grupo, debati sobre machine learning e os métodos de análise presentes no trabalho. Desta forma, cada um agregou com seu ponto de vista e na escolha das informações presentes no trabalho.

Além disso tudo, pude aprender uma nova linguagem, que foi o Python. Linguagem a qual eu ainda não havia tido contato. Portanto, este relatório foi de extrema importância para a realização da matéria e também para meu futuro como engenheiro.

Matheus Sales

O trabalho proporcionou a experiência de trabalhar com um software reconhecedor, mais simples do que eu imaginava que seria, entender seu funcionamento e saber questionar que tipo de algoritmo poderia ser melhor para um dado problema. Tudo isto me levou a ideia de que um software que venha a resolver

um dado problema famoso, não necessariamente deve ser um software de difícil construção/entendimento.

Também foi interessante a experiência de ter trabalhado em grupo na criação do código e escrita do relatório. Ter que aprender em grupo sobre as técnicas de análise abordadas e sobre tecnologias nem sempre discutidas no curso (como machine learning, por exemplo) são experiências sempre bem-vindas.

Trouxe capacidade de entender melhor como funcionam determinadas características matematico-computacionais que são utilizadas, hoje em dia, em praticamente toda rede social. Isto só foi possível através da utilização e aplicação da teoria proposta em sala de aula.

A atividade também ajudou a melhorar minhas habilidades com a linguagem de programação Python e através do mesmo voltei a ter contato com Jupyter Notebooks. O contato e descoberta de novas bibliotecas a fim de tentar resolver o problema dado foi interessante.

7.Anexo:

```
import random
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import sklearn.decomposition as decomp
import PillowImage

# Falso infinito, porém suficiente
AUX = 9999999999999

CAMINHO_IMAGENS = ""
IMAGENS_TREINO = {}
IMAGEM_TESTE = {}

#Função para carregar as imagens e separar em duas listas: Imagens para teste e
imagens para treino
def carrega_img(caminho, mapa_classe, n, ext='.jpg'):
    global CAMINHO_IMAGENS, IMAGENS_TREINO, IMAGEM_TESTE
    CAMINHO_IMAGENS = caminho
    IMAGENS_TREINO = mapa_classe
    IMAGEM_TESTE = {'all': []}
    for c in IMAGENS_TREINO:
        a = random.randrange(n)
        for i in range(0,n):
            file_name = c + '-' + str(i+1) + ext
            if i == a:
                IMAGEM_TESTE['all'] = np.ndarray.tolist(np.append(file_name,
IMAGEM_TESTE['all']))
            else:
                IMAGENS_TREINO[c] = np.ndarray.tolist(np.append(file_name,
IMAGENS_TREINO[c]))

#Função para ler as imagens
```

```
def leitor_img(nome_arquivo):
    return mpimg.imread(CAMINHO_IMAGENS + nome_arquivo)


#Função que retorna a classe dado o nome do arquivo da imagem
def define_classe(nome_arquivo):
    return nome_arquivo.split("-")[0] #Separa as imagens em classes(grupos)

#Função que calcula o pca da imagem X, mas antes a redimensiona  de [100][100][3]
para [300][100]
def PCA(X):
    X = X.reshape(300,100)
    pca = decomp.PCA(n_components=2)
    pca.fit(X)
    return pca.transform(X)

#Encontra a classe utilizando as listas e fazendo validação cruzada
def encontra_classe(img, use_pca):
    aux = AUX
    vet_c = 0
    vet_i = []

    for c in IMAGENS_TREINO:
        for i in IMAGENS_TREINO[c]:
            img_compare = leitor_img(i)
            aux_pessoa = aux_pessoa = np.linalg.norm(img[:,:,0:3]-
img_compare[:,:,0:3])
            if (use_pca):
                aux_pessoa = np.linalg.norm(PCA(img[:,:,0:3])-
PCA(img_compare[:,:,0:3]))
            if aux_pessoa < aux:
                aux = aux_pessoa
                vet_i = img_compare
                vet_c = c
    return aux, vet_i, vet_c

#Plota se acertou ou errou e da diferença da norma
def reconhecedor(resultado, use_pca):
    errado = 0
    certo = 0
    for fn in IMAGEM_TESTE['all']:
        img = leitor_img(fn)
        aux, vet_i, vet_c = encontra_classe(img, use_pca)
        if define_classe(fn) == str(vet_c):
            certo = certo + 1
        else:
            errado = errado + 1
        if resultado:
            if define_classe(fn) == str(vet_c):
                print("IGUAL")
            else:
                print("DIFERENTE " + define_classe(fn) + " != " + str(vet_c))
            print('min_norm_difference = ' + str(aux))
            print('class = ' + str(vet_c))
            f, (plt0, plt1) = plt.subplots(1, 2, figsize=(10,5))
            if define_classe(fn) == str(vet_c):
                f.suptitle('Igual')
            else:
                f.suptitle('Diferente')
```

```
            plt0.imshow(img)
            plt1.imshow(vet_i)
            plt.show()
    return (certo*100.)/(certo+errado)


#Função que mostra o percentual de acerto dos algoritmos
def teste(k, use_pca, caminho, mapa_classe, n, ext='.jpg'):
    percentual = 0
    for i in range(k):
        carrega_img(caminho, mapa_classe, n, ext)
        percentual = percentual + reconhecedor(False, use_pca)
    return float(percentual)/k


#Taxas e resultados

#VERY_EASY(FUNCIONANDO)
#print("VERY-EASY")
#print("Força Bruta: " + str(teste(3, False, 'recdev-master/very-easy/', {'2': [],
'11': [], '15': [], '16': [], '87': []}, 2)) + "%")
#print("PCA: " + str(teste(3, True, 'recdev-master/very-easy/', {'2': [], '11':
[], '15': [], '16': [], '87': []}, 2)) + "%")

#FORÇA BRUTA
#carrega_img('recdev-master/very-easy/', {'2': [], '11': [], '15': [], '16': [],
'87': []}, 2)
#percent = reconhecedor(True, False)
#print(str(percent) + "% de acerto")
#PCA
#carrega_img('recdev-master/very-easy/', {'2': [], '11': [], '15': [], '16': [],
'87': []}, 2)
#percent = reconhecedor(True, True)
#print(str(percent) + "% de acerto")

#EASY
print("\nEASY")
print("Força Bruta: " + str(teste(3, False, 'recdev-master/easy/', {'2': [], '11':
[], '15': [], '16': [], '87': []}, 8)) + "%")
print("PCA: " + str(teste(3, True, 'recdev-master/easy/', {'2': [], '11': [],
'15': [], '16': [], '87': []}, 8)) + "%")

#FORÇA BRUTA
carrega_img('recdev-master/easy/', {'2': [], '11': [], '15': [], '16': [], '87':
[]}, 8)
percent = reconhecedor(True, False)
print(str(percent) + "% de acerto")
#PCA
carrega_img('recdev-master/easy/', {'2': [], '11': [], '15': [], '16': [], '87':
[]}, 8)
percent = reconhecedor(True, True)
print(str(percent) + "% de acerto")


#MEDIUM
print("\nMEDIUM")
print("Força Bruta: " + str(teste(1, False, 'recdev-master/medium/', {'1': [],
'2': [], '3': [], '4': [], '5': [], '6': [], '7': [], '8': [], '9': [], '10': [],
                        '11': [], '12': [], '13': [], '14': [], '15': [], '16':
```

```
[], '17': [], '18': [], '19': [], '20': [],
                        '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
                        '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
                        '41': [], '42': [], '43': [], '44': [], '45': [], '46':
[], '47': [], '48': [], '49': [], '50': [],
                                                                        }, 7)) +
"%")
print("PCA: " + str(teste(1, True, 'recdev-master/medium/', {'1': [], '2': [],
'3': [], '4': [], '5': [], '6': [], '7': [], '8': [], '9': [], '10': [],
                        '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
                        '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
                        '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
                                                                '41': [], '42': [],
'43': [], '44': [], '45': [], '46': [], '47': [], '48': [], '49': [], '50': [],
                                                                }, 7)) + "%")

#FORÇA BRUTA
carrega_img('recdev-master/medium/', {'1': [], '2': [], '3': [], '4': [], '5': [],
'6': [], '7': [], '8': [], '9': [], '10': [],
                        '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
                        '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
                        '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
                        '41': [], '42': [], '43': [], '44': [], '45': [], '46':
[], '47': [], '48': [], '49': [], '50': [],
                                        }, 7)
percent = reconhecedor(True, False)
print(str(percent) + "% de acerto")
#PCA
carrega_img('recdev-master/medium/', {'1': [], '2': [], '3': [], '4': [], '5': [],
'6': [], '7': [], '8': [], '9': [], '10': [],
                        '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
                        '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
                        '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
                                        '41': [], '42': [], '43': [], '44': [],
'45': [], '46': [], '47': [], '48': [], '49': [], '50': [],
                                        }, 7)
percent = reconhecedor(True, True)
print(str(percent) + "% de acerto")

#HARD
#print("\nHARD")
#print("Força Bruta: " + str(teste(1, False, 'recdev-master/hard/', {'1': [], '2':
[], '3': [], '4': [], '5': [], '6': [], '7': [], '8': [], '9': [], '10': [],
#                         '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
#                         '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
#                         '31': [], '32': [], '33': [], '34': [], '35': [], '36':
```

```
[], '37': [], '38': [], '39': [], '40': [],
#                         '41': [], '42': [], '43': [], '44': [], '45': [], '46':
[], '47': [], '48': [], '49': [], '50': [],
#                                                                       }, 14)) +
"%")
#
#print("PCA: " + str(teste(1, True, 'recdev-master/hard/', {'1': [], '2': [], '3':
[], '4': [], '5': [], '6': [], '7': [], '8': [], '9': [], '10': [],
#                         '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
#                         '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
#                         '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
#                                                                 '41': [], '42': [],
'43': [], '44': [], '45': [], '46': [], '47': [], '48': [], '49': [], '50': [],
#                                                                 }, 14)) + "%")
#
##FORÇA BRUTA
#carrega_img('recdev-master/hard/', {'1': [], '2': [], '3': [], '4': [], '5': [],
'6': [], '7': [], '8': [], '9': [], '10': [],
#                         '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
#                         '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
#                         '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
#                         '41': [], '42': [], '43': [], '44': [], '45': [], '46':
[], '47': [], '48': [], '49': [], '50': [],
#                                       }, 14)
#percent = reconhecedor(True, False)
#print(str(percent) + "% de acerto")
##PCA
#carrega_img('recdev-master/hard/', {'1': [], '2': [], '3': [], '4': [], '5': [],
'6': [], '7': [], '8': [], '9': [], '10': [],
#                         '11': [], '12': [], '13': [], '14': [], '15': [], '16':
[], '17': [], '18': [], '19': [], '20': [],
#                         '21': [], '22': [], '23': [], '24': [], '25': [], '26':
[], '27': [], '28': [], '29': [], '30': [],
#                         '31': [], '32': [], '33': [], '34': [], '35': [], '36':
[], '37': [], '38': [], '39': [], '40': [],
#                                       '41': [], '42': [], '43': [], '44': [],
'45': [], '46': [], '47': [], '48': [], '49': [], '50': [],
#                                       }, 14)
#percent = reconhecedor(True, True)
#print(str(percent) + "% de acerto")
#
#
#EXTRAS
#FORÇA BRUTA
soma = 0

carrega_img('recdev-master/extras/facebookfaces/crop-inner/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 9)
percent = reconhecedor(True,False)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces/crop-outer/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 9)
```

```
percent = reconhecedor(True,False)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces-2/crop-inner/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 10, ".png")
percent = reconhecedor(True,False)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces-2/crop-outer/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 10, ".png")
percent = reconhecedor(True,False)
soma = soma + percent

print(str(soma / 4.) + "% de acerto\n")

#PCA
soma = 0

carrega_img('recdev-master/extras/facebookfaces/crop-inner/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 9)
percent = reconhecedor(True,True)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces/crop-outer/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 9)
percent = reconhecedor(True,True)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces-2/crop-inner/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 10, ".png")
percent = reconhecedor(True,True)
soma = soma + percent

carrega_img('recdev-master/extras/facebookfaces-2/crop-outer/', {'1': [], '2': [],
'3': [], '4': [], '5': []}, 10, ".png")
percent = reconhecedor(True,True)
soma = soma + percent

print(str(soma / 4.) + "% de acerto\n")
```

8.Referências

- https://www.significados.com.br/machine-learning/
- https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial
- https://pt.wikipedia.org/wiki/An%C3%A1lise_de_componentes_principais
- https://www.youtube.com/watch?v=2MySwcqdi2A
- https://www.codigofluente.com.br/aula-12-scikit-learn-reconhecimento-facial-com-eigenfaces-e-svms/