Face recognition

Dataset

Aligned Face Dataset from Pinterest

This dataset contains 10.770 images for 100 people. All images are taken from 'Pinterest' and aligned using dlib library.

In [1]:

```
import tensorflow
tensorflow.__version__
```

Out[1]:

```
'2.3.0'
```

Mount Google drive

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Setting the project path

In [3]:

```
project_path = "/content/drive/My Drive/Colab Notebooks/"
```

Extract the zip file

- Extracted Aligned Face Dataset from Pinterest.zip

In [4]:

```
from zipfile import ZipFile
with ZipFile(project_path +'Aligned_Face_Dataset_from_Pinterest.zip', 'r') as z:
  z.extractall()
print("Extraction done!")
```

```
Extraction done!
```

Function to load images

- Defined a function to load the images from the extracted folder and map each image with person id

In [5]:

```
import numpy as np
import os
from tqdm.notebook import tqdm
class IdentityMetadata():
    def __init__(self, base, name, file):
        # print(base, name, file)
        # dataset base directory
        self.base = base
        # identity name
        self.name = name
        # image file name
        self.file = file

    def __repr__(self):
        return self.image_path()

    def image_path(self):
        return os.path.join(self.base, self.name, self.file)

def load_metadata(path):
    metadata = []
    exts = []
    for i in os.listdir(path):
        for f in os.listdir(os.path.join(path, i)):
            # Check file extension. Allow only jpg/jpeg' files.
            ext = os.path.splitext(f)[1]
            if ext == '.jpg' or ext == '.jpeg':
                metadata.append(IdentityMetadata(path, i, f))
                exts.append(ext)
    return np.array(metadata), exts

metadata, exts = load_metadata('PINS')
labels = np.array([meta.name for meta in metadata])
```

Function to load image

- Defined a function to load image from the metadata

In [6]:

```
import cv2
def load_image(path):
    img = cv2.imread(path, 1)
    # OpenCV loads images with color channels
    # in BGR order. So we need to reverse them
    return img[...,::-1]
```

Loading a sample image

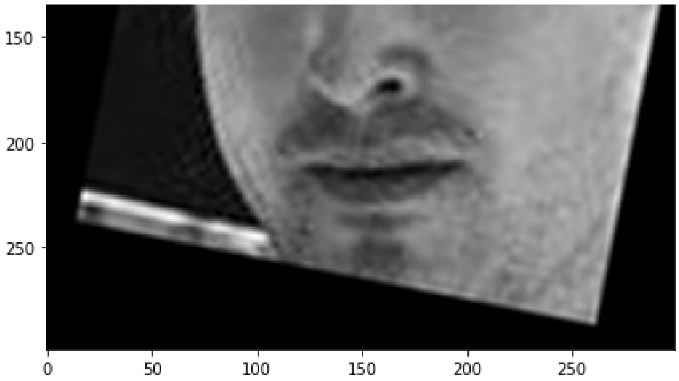
- Loaded one image using the function "load_image"

In [7]:

```
import matplotlib.pyplot as plt
n = 15
img_path = metadata[n].image_path()
img = load_image(img_path)
```

In [8]:

```
fig = plt.figure(figsize = (15, 7.2))
ax = fig.add_subplot(1, 1, 1)
title = labels[n].split('_')[1]
ax.set_title(title, fontsize = 20)
_ = plt.imshow(img)
```



VGG Face model

- Here I am using a predefined model for VGG face

In [9]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import ZeroPadding2D, Convolution2D, MaxPooling2D, Dropout, Flatten, A
ctivation

def vgg_face():
    model = Sequential()
    model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2,2), strides=(2,2)))

    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2,2), strides=(2,2)))

    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(256, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(256, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(256, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2,2), strides=(2,2)))

    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2,2), strides=(2,2)))

    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1,1)))
    model.add(Convolution2D(512, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2,2), strides=(2,2)))

    model.add(Convolution2D(4096, (7, 7), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Convolution2D(4096, (1, 1), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Convolution2D(2622, (1, 1)))
    model.add(Flatten())
    model.add(Activation('softmax'))
    return model
```

Loading the model

- Loaded the model defined above
- Then load the given weight file named "vgg_face_weights.h5"

In [10]:

```
model = vgg_face()
model.load_weights(project_path +'vgg_face_weights.h5')
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
zero_padding2d (ZeroPadding2 (None, 226, 226, 3)       0         
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 64)      1792      
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 226, 226, 64)      0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 224, 224, 64)      36928     
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 64)      0         
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 114, 114, 64)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 128)     73856     
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 114, 114, 128)     0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 128)     147584    
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 128)       0         
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 58, 58, 128)       0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 56, 56, 256)       295168    
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 58, 58, 256)       0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 256)       590080    
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 58, 58, 256)       0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 256)       590080    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 256)       0         
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 30, 30, 256)       0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 512)       1180160   
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 30, 30, 512)       0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 512)       2359808   
_________________________________________________________________
zero_padding2d_9 (ZeroPaddin (None, 30, 30, 512)       0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 512)       2359808   
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 512)       0         
_________________________________________________________________
zero_padding2d_10 (ZeroPaddi (None, 16, 16, 512)       0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 14, 14, 512)       2359808   
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 16, 16, 512)       0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 512)       2359808   
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 16, 16, 512)       0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 512)       2359808   
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 512)         0         
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 1, 1, 4096)        102764544
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 4096)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 1, 1, 4096)        16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 1, 4096)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 1, 1, 2622)        10742334
_________________________________________________________________
flatten (Flatten)            (None, 2622)              0
_________________________________________________________________
activation (Activation)      (None, 2622)              0
=================================================================
Total params: 145,002,878
Trainable params: 145,002,878
Non-trainable params: 0
_________________________________________________________________
None
```

Get vgg_face_descriptor

In [11]:

```
from tensorflow.keras.models import Model
vgg_face_descriptor = Model(inputs=model.layers[0].input, outputs=model.layers[-2].output)
```

Generating embeddings for each image in the dataset

- The below code is an example to load the first image in the metadata and get its embedding vector from the pre-trained model.

In [12]:

```
# Get embedding vector for first image in the metadata using the pre-trained model

img_path = metadata[0].image_path()
img = load_image(img_path)

# Normalising pixel values from [0-255] to [0-1]: scale RGB values to interval [0,1]
img = (img / 255.).astype(np.float32)

img = cv2.resize(img, dsize = (224,224))
print(img.shape)

# Obtain embedding vector for an image
# Get the embedding vector for the above image using vgg_face_descriptor model and print the shape


embedding_vector = vgg_face_descriptor.predict(np.expand_dims(img, axis=0))[0]
print(embedding_vector.shape)
```

```
(224, 224, 3)
(2622,)
```

Generating embeddings for all images

In [13]:

```
embeddings = []
embeddings = np.zeros((metadata.shape[0], 2622))
for i, meta in tqdm(enumerate(metadata)):
  try:
    image = load_image(str(meta))
    image = (image/255.).astype(np.float32)
    image = cv2.resize(image, (224, 224))
    embeddings[i] = vgg_face_descriptor.predict(np.expand_dims(image, axis = 0))[0]
  except:
    embeddings[i] = np.zeros(2622)
```

```
Exception ignored in: <bound method ScopedTFGraph.__del__ of
<tensorflow.python.framework.c_api_util.ScopedTFGraph object at 0x7fcf1245deb8>>
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/c_api_util.py", line 54
, in __del__
    self.deleter(self.graph)
KeyboardInterrupt
```

Function to calculate distance between given 2 pairs of images.

- Considering distance metric as "Squared L2 distance"
- Squared l2 distance between 2 points (x1, y1) and (x2, y2) = (x1-x2)^2 + (y1-y2)^2

In [14]:

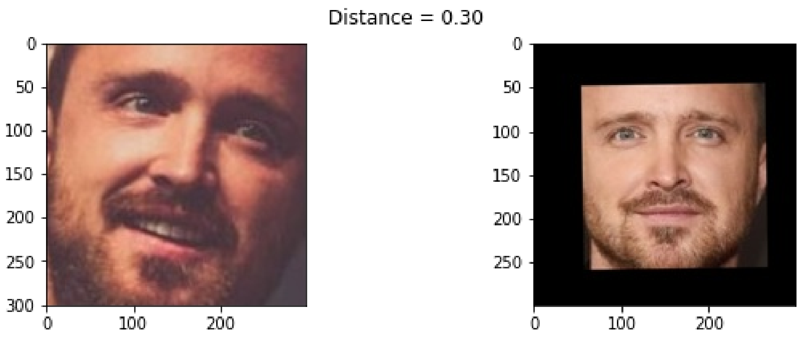
```
def distance_btw_photo(emb1, emb2):
    return np.sum(np.square(emb1 - emb2))
```

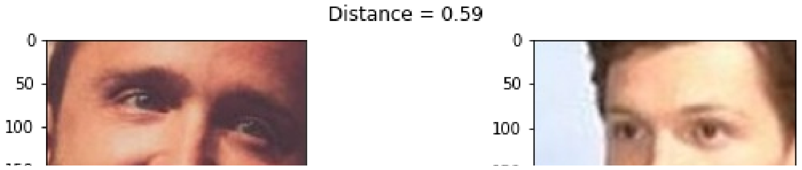
Plotting images and get distance between the pairs given below

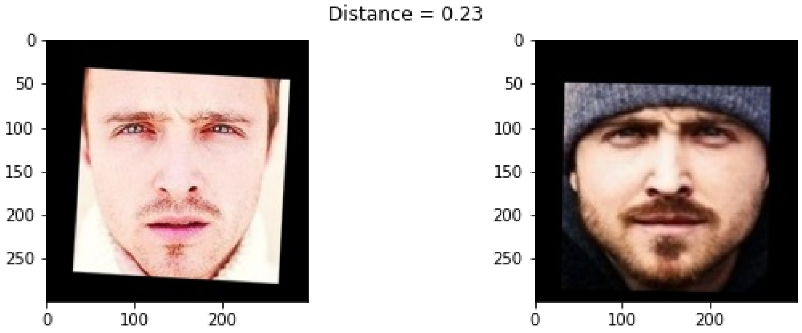
- 2, 3 and 2, 180
- 30, 31 and 30, 100
- 70, 72 and 70, 115

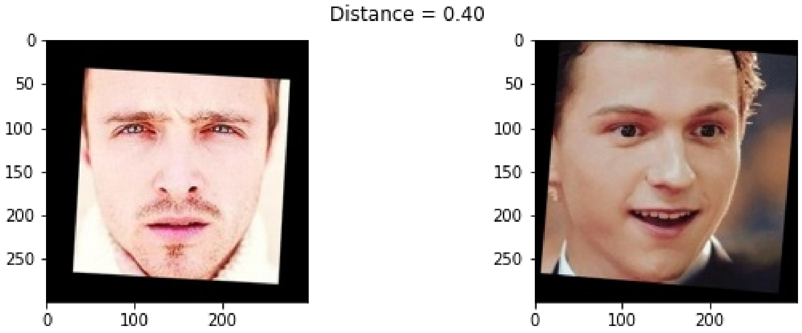
In [15]:

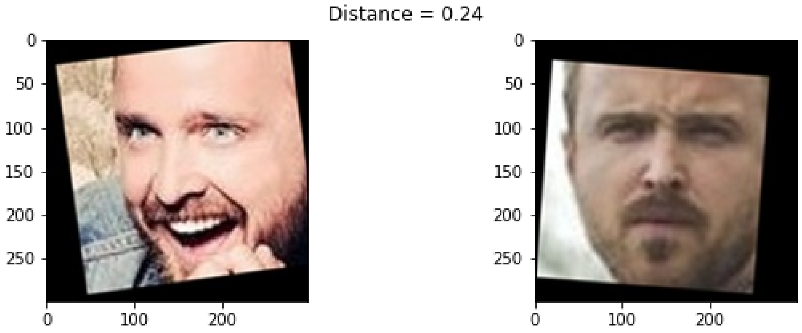
```
import matplotlib.pyplot as plt

def photo_pair(idx1, idx2):
    plt.figure(figsize=(10,3))
    plt.suptitle(f'Distance = {distance_btw_photo(embeddings[idx1], embeddings[idx2]):.2f}')
    plt.subplot(121)
    plt.imshow(load_image(metadata[idx1].image_path()))
    plt.subplot(122)
    plt.imshow(load_image(metadata[idx2].image_path()));

photo_pair(2, 3)
photo_pair(2, 180)

photo_pair(30, 31)
photo_pair(30, 100)

photo_pair(70, 72)
photo_pair(70, 115)
```

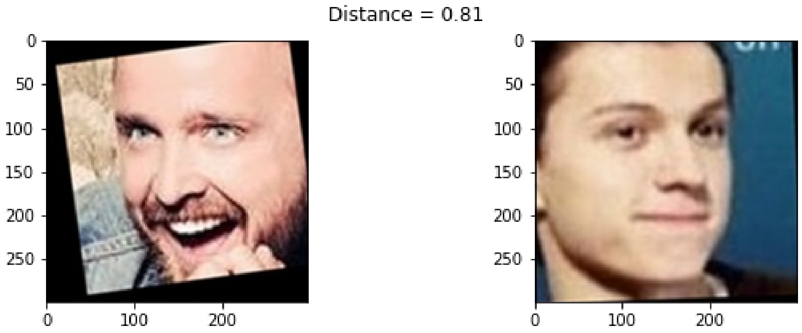












Create train and test sets

- Created X_train, X_test and y_train, y_test
- Used train_idx to seperate out training features and labels
- Used test_idx to seperate out testing features and labels

In [16]:

```
train_idx = np.arange(metadata.shape[0]) % 9 != 0      #every 9th example goes in test data and rest
go in train data
go in train data
test_idx = np.arange(metadata.shape[0]) % 9 == 0


X_train = embeddings[train_idx]

X_test = embeddings[test_idx]

targets = np.array([m.name for m in metadata])

y_train = targets[train_idx]

y_test = targets[test_idx]
display(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(9573, 2622)


(1197, 2622)


(9573,)


(1197,)
```

Encoding the Labels

- Encode the targets
- Using LabelEncoder

In [17]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

Standardizing the feature values

- Scaled the features using StandardScaler

In [18]:

```
# Standarize features
from sklearn.preprocessing import StandardScaler
SS= StandardScaler()
X_train_ss = SS.fit_transform(X_train)
X_test_ss = SS.transform(X_test)
print(X_train_ss)
```

```
[[ 3.19805992  0.52774607 -0.11897772 ...  1.42434394 -0.80854239
  -0.39171251]
 [ 2.18526897 -1.24667989 -1.46033746 ...  1.39736512 -0.34711078
  -1.72953757]
 [ 1.18554871 -1.02450524 -1.11199746 ...  1.77426602 -0.60884155
  -1.0169936 ]
 ...
 [-0.34476561 -2.2002836   0.06508968 ...  0.67130937  0.33887495
   1.40616098]
 [-0.83382842 -0.38144794 -0.22324448 ...  0.39823153  1.03637841
   0.93006594]
 [-1.1736013   0.12612719  1.30961817 ... -1.80802258  0.61677233
   1.38634548]]
```

Reducing dimensions using PCA

- Reduced feature dimensions using Principal Component Analysis
- Set the parameter n_components=128

In [19]:

```
from sklearn.decomposition import PCA

n_components = 128

pca = PCA(n_components=n_components, whiten=True)
pca.fit(X_train_ss)
X_train_pca = pca.transform(X_train_ss)
X_test_pca = pca.transform(X_test_ss)
```

Build a Classifier

- Used SVM Classifier to predict the person in the given image
- Fit the classifier and print the score

In [20]:

```
from sklearn.svm import SVC

model_svc= SVC(kernel='linear')
model_svc.fit(X_train_pca, y_train)
print('Accuracy for svc train set: {0:.6f}'.format(model_svc.score(X_train_pca, y_train)))
```

```
Accuracy for svc train set: 0.999791
```

Test results

- Took 10th image from test set and plot the image
- Reported to which person(folder name in dataset) the image belongs to

In [21]:

```
import warnings
# Suppress LabelEncoder warning
warnings.filterwarnings('ignore')

example_idx = 10

example_image = load_image(metadata[test_idx][example_idx].image_path())
example_prediction = model_svc.predict([X_test_pca[example_idx]])
example_identity = le.inverse_transform(example_prediction)[0]

plt.imshow(example_image)
plt.title(f'Identified as {example_identity}');
```