Instructions

- Some parts of the code are already done for you
- You need to execute all the cells
- You need to add the code where ever you see `"#### Add your code here ####"`
- Marks are mentioned along with the cells

Face recognition

Task is to recognize a face

Dataset

Aligned Face Dataset from Pinterest

This dataset contains 10.770 images for 100 people. All images are taken from 'Pinterest' and alig

```
%tensorflow_version 2.x
```

```
TensorFlow 2.x selected.
```

```
import tensorflow
tensorflow.__version__
```

```
'2.1.0'
```

Mount Google drive if you are using google colab

- We recommend using Google Colab as you can face memory issues and longer runtimes wh

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

Enter your authorization code:
··········
Mounted at /content/drive
```

Change current working directory to project folder (1 mark)

```
import os
os.chdir("/content/drive/My Drive/22march/")
```

Extract the zip file (2 marks)

- Extract Aligned Face Dataset from Pinterest.zip

```
from zipfile import ZipFile
with ZipFile('Aligned Face Dataset.zip', 'r') as z:
  z.extractall()
```

Function to load images

- Define a function to load the images from the extracted folder and map each image with pers

```
import numpy as np
import os

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 = []
    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))
    return np.array(metadata)

# metadata = load_metadata('images')
metadata = load_metadata('PINS')
```

Define function to load image

- Define a function to load image from the metadata

```
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]
```

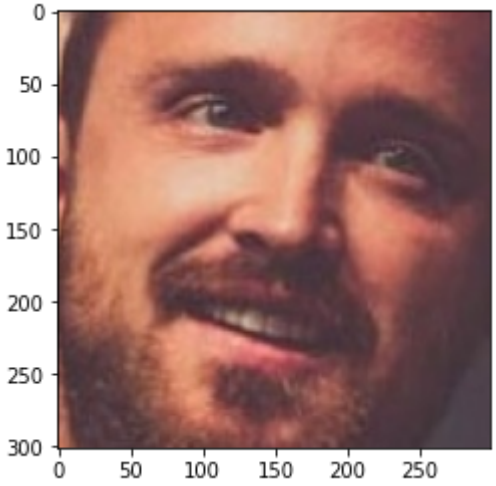
Load a sample image (2 marks)

- Load one image using the function "load_image"

```
import matplotlib.pyplot as plt
%matplotlib inline

image1 = str(metadata[0])
plt.imshow(load_image(image1))
```

```
<matplotlib.image.AxesImage at 0x7f2e68520390>
```



VGG Face model

- Here we are giving you the predefined model for VGG face

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import ZeroPadding2D, Convolution2D, MaxPooling2D, Dropout, F

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(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
```

Load the model (2 marks)

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

```
model = vgg_face()
model.load_weights('vgg_face_weights.h5')
```

Get vgg_face_descriptor

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

Generate embeddings for each image in the dataset

- Given below is an example to load the first image in the metadata and get its embedding vec

```
# 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 t

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

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

Generate embeddings for all images (5 marks)

- Write code to iterate through metadata and create embeddings for each image using `vgg_fa` with name `embeddings`
- If there is any error in reading any image in the dataset, fill the emebdding vector of that imag from the model is of length 2622.

```
# Get embedding vector for first image in the metadata using the pre-trained model
def get_embed(em):
  img_path = em.image_path()
  img = load_image(img_path)
  img = (img / 255.).astype(np.float32) # Normalising pixel values from [0-255] to [0-1]:
  img = cv2.resize(img, dsize = (224,224))
  embedding_vector = vgg_face_descriptor.predict(np.expand_dims(img, axis=0))[0]
  return embedding_vector


# embeddings list
embeddings = np.zeros((metadata.shape[0], 2622))

for i, em in enumerate(metadata):
  embeddings[i] = get_embed(em)


print('Size of each embedding: ', len(embeddings[0]))
```

```
Size of each embedding:  2622
```

Function to calculate distance between given 2 pairs of images.

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

```
def distance(emb1, emb2):
```

```
    return np.sum(np.square(emb1 - emb2))
```

Plot images and get distance between the pairs given below

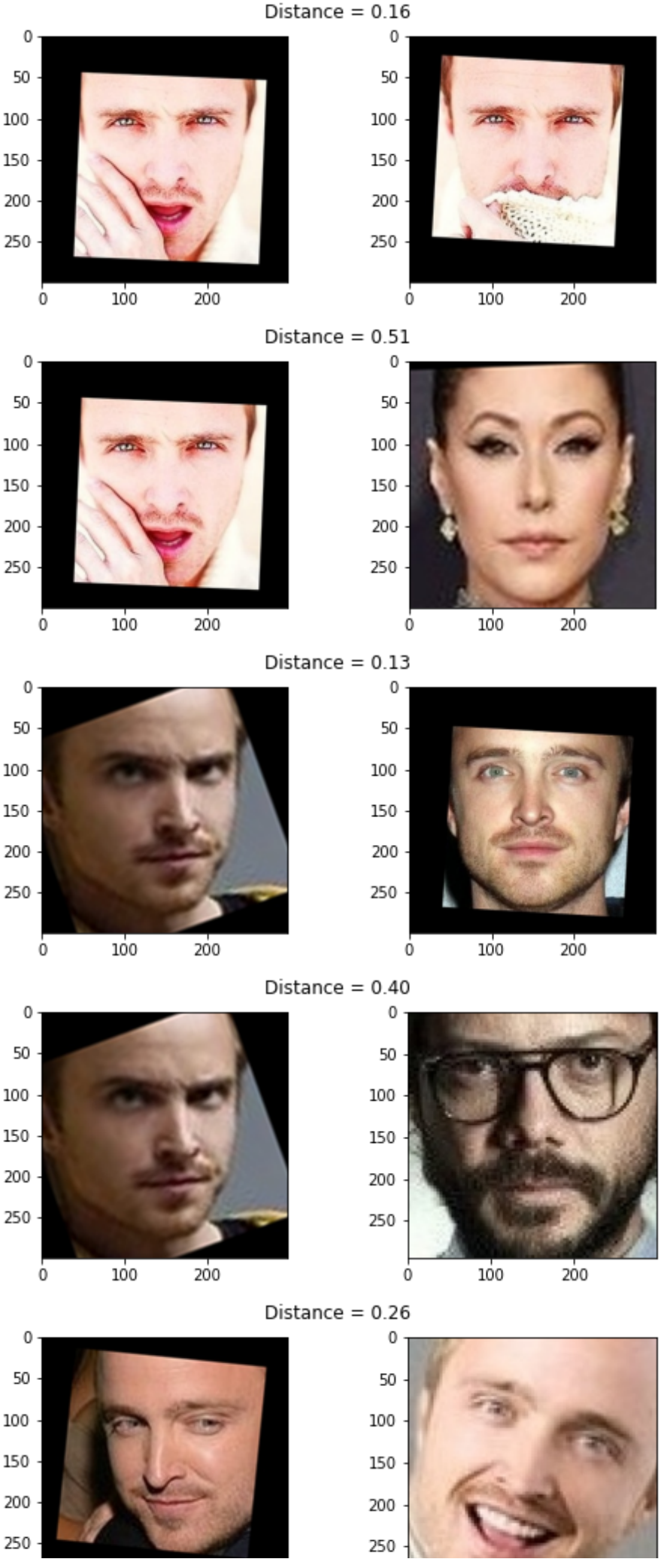
- 2, 3 and 2, 180
- 30, 31 and 30, 100
- 70, 72 and 70, 115

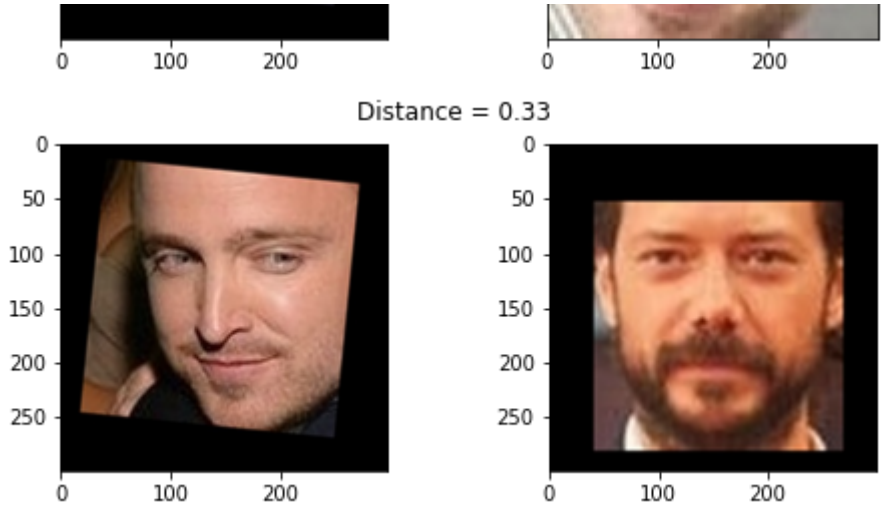
```
import matplotlib.pyplot as plt

def show_pair(idx1, idx2):
    plt.figure(figsize=(8,3))
    plt.suptitle(f'Distance = {distance(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()));

show_pair(2, 3)
show_pair(2, 180)

# Other image pairs:
show_pair(30, 31)
show_pair(30, 100)
show_pair(70, 72)
show_pair(70, 115)
```


Distance = 0.16
Distance = 0.51
Distance = 0.13
Distance = 0.40
Distance = 0.26



Create train and test sets (5 marks)

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

```
train_idx = np.arange(metadata.shape[0]) % 9 != 0
test_idx = np.arange(metadata.shape[0]) % 9 == 0

X_train = embeddings[train_idx]
X_test = embeddings[test_idx]

y = np.array([m.name for m in metadata])
y_train = y[train_idx]
y_test = y[test_idx]
```

Encode the Labels (3 marks)

- Encode the targets
- Use LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)
```

Standardize the feature values (3 marks)

- Scale the features using StandardScaler

```
# Standarize features
from sklearn.preprocessing import StandardScaler
```

```
standard_scaler = StandardScaler()
X_train_scaled = standard_scaler.fit_transform(X_train)
X_test_scaled = standard_scaler.fit_transform(X_test)
```

Reduce dimensions using PCA (3 marks)

- Reduce feature dimensions using Principal Component Analysis

```
from sklearn.decomposition import PCA

pca = PCA(.96)
pca.fit(X_train_scaled)
X_train_pca = pca.transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)


# Checking total components
pca.n_components_
```

```
412
```

Build a Classifier (3 marks)

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

```
from sklearn.svm import SVC

clf = SVC()
clf.fit(X_train_pca, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
# Accuracy of training data
clf.score(X_train_pca, y_train)
```

```
0.997388488457119
```

```
# Accuracy of test data
clf.score(X_test_pca, y_test)
```

```
0.9598997493734336
```

Test results (1 mark)

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

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

example_idx = 10
example_image = load_image(metadata[test_idx][example_idx].image_path())
example_prediction = clf.predict([X_test_pca[example_idx]])
example_identity = label_encoder.inverse_transform(example_prediction)[0]

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