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 faces

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.

```
%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 while running on local

```
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=947318989803-6bn6qk8qdgf4n4g3pfee6491h

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

Change current working directory to project folder (1 mark)

```
import os

project_path="/content/drive/My Drive/Face-Detection"

os.chdir(project_path)
```

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 person id

```
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"

```
load_image("/content/drive/My Drive/Face-Detection/PINS/pins_Aaron Paul/Aaron Paul0_262.jpg")
```

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, Flatten, Activation

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
```

Load the model (2 marks)

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

```
from tensorflow.keras.models import load_model
model = vgg_face()
model.summary()
#### Add your code here ####
```

```
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
_________________________________________________________________
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
_________________________________________________________________
```

```
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 vector from the pre-trained model.

```
# 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,)
```

```
embedding_vector
```

```
array([ 0.03170307, -0.0150513 , -0.01243402, ...,  0.00043141,
        0.00219081, -0.00908097], dtype=float32)
```

Generate embeddings for all images (5 marks)

- Write code to iterate through metadata and create embeddings for each image using `vgg_face_descriptor.predict()` and store i with name `embeddings`
- If there is any error in reading any image in the dataset, fill the emebdding vector of that image with 2622-zeroes as the final embe from the model is of length 2622.

```
embeddings=[]
for i in metadata:
  image_path=i.image_path()
  img = load_image(image_path)
  img = (img / 255.).astype(np.float32)
  img = cv2.resize(img, dsize = (224,224))
  print(img.shape)
  try:
    embedding_vector = vgg_face_descriptor.predict(np.expand_dims(img, axis=0))[0]
    print(embedding_vector.shape)
    embeddings.append(embedding_vector)
  except:
    embedding_vector=np.zeros(2622*1)
    emeddings.append(embedding_vector)
```

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))
```

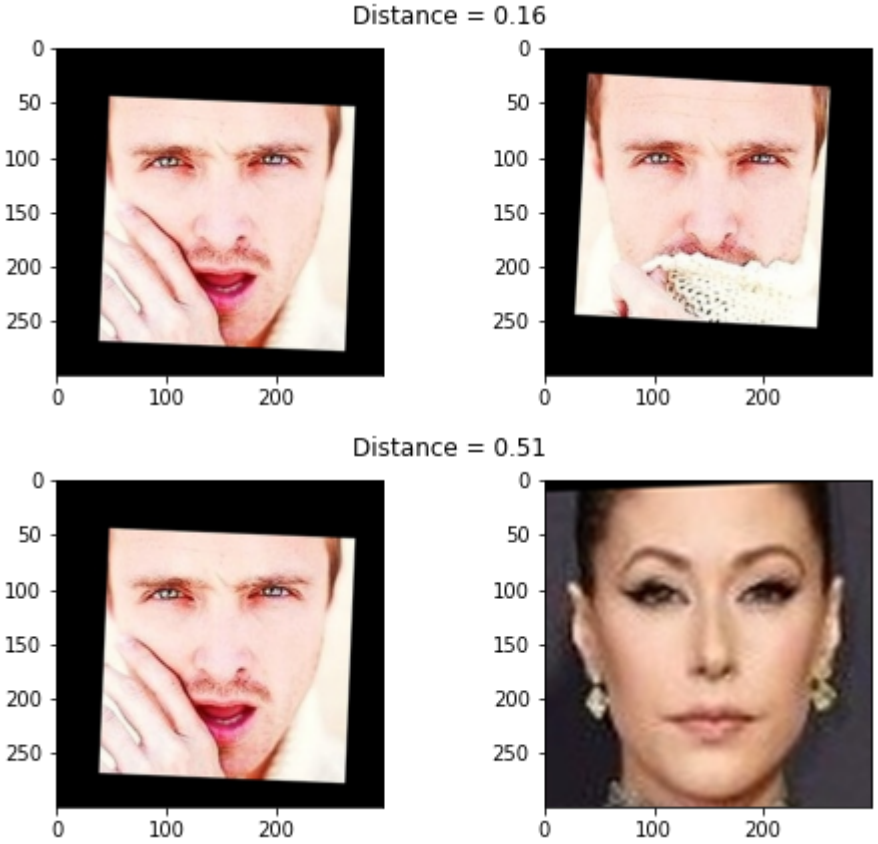
Plot images and get distance between the pairs given below

- 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)
```



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
train_pos=np.where(train_idx==True)
test_pos=np.where(test_idx==True)
```

```
metadata[0].image_path().split("/")[1]
```

```
'pins_Aaron Paul'
```

```
y_train=[]
for i in train_pos:
  y_train.append(metadata[i])
```

```
y_train=np.array(y_train)
y_train=y_train.reshape(2523)
```

```
embed_array=np.array(embeddings)
Xtrain=[]
for i in train_pos:
  Xtrain.append(embed_array[i])
```

```
X_train=np.array(Xtrain)
```

```
X_train.shape
```

```
(1, 2523, 2622)
```

```
embeddings[0]
```

```
array([ 0.03170307, -0.0150513 , -0.01243402, ...,  0.00043141,
        0.00219081, -0.00908097], dtype=float32)
```

```
embed_array.shape
```

```
(2839, 2622)
```

```
X_train=X_train.reshape(2523,2622)
```

```
X_train.shape
```

```
(2523, 2622)
```

```
X_train[0]
```

```
array([ 0.03497704, -0.00105059, -0.01248933, ..., -0.01053091,
        0.0017932 ,  0.02439154], dtype=float32)
```

```
y_test=[]
for i in test_pos:
  y_test.append(metadata[i])
```

```
y_test=np.array(y_test)
y_test=y_test.reshape(316)
```

```
y_test.shape
```

```
(316,)
```

```
Xtest=[]
for i in test_pos:
  Xtest.append(embed_array[i])
```

```
X_test=np.array(Xtest)
```

```
X_test.shape
```

```
(1, 316, 2622)
```

```
X_test=X_test.reshape(316,2622)
```

```
embed_array[9]
```

```
X_test[1]
```

```
array([ 0.02626053,  0.00147696, -0.00927285, ..., -0.00515147,
        0.00603828,  0.01470516], dtype=float32)
```

Encode the Labels (3 marks)

- Encode the targets
- Use LabelEncoder

```
ytrain=[]
for i in y_train:
  ytrain.append(str(i).split("/")[1])


ytest=[]
for i in y_test:
  ytest.append(str(i).split("/")[1])
```

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
y_train=label_encoder.fit_transform(ytrain)
y_test=label_encoder.fit_transform(ytest)
```

Standardize the feature values (3 marks)

- Scale the features using StandardScaler

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

scaler = StandardScaler()
```

```
scaler = StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=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(n_components=3)
X_train=pca.fit_transform(X_train)
X_test=pca.fit_transform(X_test)
```

```
X_train.shape
```

```
(2523, 3)
```

```
y_train.shape
```

```
(2523,)
```

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(kernel='linear')
clf.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
```

```
clf.score(X_train, y_train)
```

```
0.4724534284581847
```

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(example_image)
example_identity = encoder.inverse_transform(example_prediction)[0]

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-315-2a09229850ad> in <module>()
      7 example_image = load_image(metadata[test_idx][example_idx].image_path())
      8 #example_prediction = clf.predict(example_image)
----> 9 example_identity = encoder.inverse_transform(example_prediction)[0]
     10
     11 plt.imshow(example_image)

NameError: name 'encoder' is not defined

SEARCH STACK OVERFLOW
```

```
clf.predict(X_test)[10]
```

1