Advance Visual Recognition Project-1

Jainav Sanghvi (IMT2020098)

Task: Face Verification

Implementation of a face verification system using the face image dataset provided. Three approaches have been tried out:
(i) **PCA** based approach (ii) **AutoEncoder** based approach & (iii) **DNN** based approach

Dataset Preprocessing Pipeline:

Dataset: The dataset encompasses 832 images, distributed across 49 individuals with each individual represented by upto 19 images capturing diverse facial poses.

Image Pre-processing:A standardized preprocessing pipeline is applied to homogenize the images, resizing all to a consistent dimension of 512x512. Furthermore, the images undergo a conversion to grayscale. Employing a cascade classifier, facial detection is performed to isolate the primary facial region, subsequently resizing it to the desired 512x512 dimensions.

Dataset Link: https://www.kaggle.com/datasets/jshark/iiitb-faces

Train-Test Partition: The images of each person are split into the train and test dataset in 80:20 proportion.

Total images: 832
Total **train** images: 647
Total **test** images: 185

Vector Flattening: The images' multi-dimensional structure is transformed into one-dimensional vectors, facilitating their manipulation within subsequent computations.

Normalization: The images are flattened into one-dimensional vectors. This process is instrumental in centering the data and mitigating variance that might impede subsequent analysis.

So Initially,
Train Dataset Shape: (647, 262144)
Test Dataset Shape: (185, 262144)

Here each image feature is represented as a row.

Part a: PCA based approach:

Principal Component Analysis is a linear dimensionality reduction technique. Face recognition using Principal Component Analysis (PCA) is a technique that involves using PCA to reduce the dimensionality of face images and then using the resulting lower-dimensional representations to perform face recognition.

PCA has been implemented in **two** different ways and each implementation has been run in **three** different settings based on number of components taken i.e. 128, 256 and 512.
Two different implementations of PCA have been used:

1. **PCA from sklearn:**

- PCA module is imported from
- the fit method has been used in order to fit the normalized training data
- the transform method has been used on the testing data in order to apply dimensionality reduction

2. **PCA from Scratch**

EigenFace Calculation: The covariance matrix is calculated from the normalized data. For computing the eigenvectors and corresponding eigenvalues of this covariance matrix, the top K eigenvalues are chosen, along with the associated eigenvectors. This selection process is pivotal in capturing the most salient facial features and patterns inherent in the dataset.

- The shape of the eigen vector matrix is (k, 262144)

Image Representation: Each image's representation is redefined as a linear combination of the eigenfaces. This affords a novel basis for describing facial features, effectively transforming the original high-dimensional data into a reduced-dimensional space defined by the eigenfaces.

Weight Computation: The weights for each image in both the test dataset and train dataset are computed, delineating their relationship with the eigenfaces. These weights provide a compact representation of an image's features within the eigenface subspace, constituting the core of subsequent recognition and identification processes.

Train weights shape: (647, k)

Test weights shape: (185, k)

Feature Vector Computation: In order to get a k dimensional feature vector (k is the number of components of PCA), the **test normalized face vector matrix** is multiplied with the **eigen faces matrix** which was computed from the training data.

First 50 Eigen Faces

Part b: AutoEncoder based approach:

The Autoencoder is an unsupervised neural network architecture consisting of an encoder and a decoder. They work by compressing input data into a lower-dimensional representation (encoding) and then attempting to reconstruct the original input from this encoding. This compression and reconstruction process helps in learning useful features or representations of the input data.

Architecture:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 4)	112
max_pooling2d (MaxPooling2D)	(None, 128, 128, 4)	0
conv2d_1 (Conv2D)	(None, 64, 64, 16)	592
conv2d_2 (Conv2D)	(None, 32, 32, 16)	2320
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 16)	0
conv2d_3 (Conv2D)	(None, 8, 8, 32)	4640
conv2d_4 (Conv2D)	(None, 4, 4, 64)	18496
conv2d_5 (Conv2D)	(None, 2, 2, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 128)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 128)	16512

Total params: 116,528
Trainable params: 116,528
Non-trainable params: 0

Encoder Model

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	132096
reshape (Reshape)	(None, 1, 1, 1024)	0
conv2d_transpose (Conv2DTranspose)	(None, 2, 2, 128)	1179776
conv2d_transpose_1 (Conv2DTranspose)	(None, 4, 4, 64)	73792
conv2d_transpose_2 (Conv2DTranspose)	(None, 8, 8, 32)	18464
up_sampling2d (UpSampling2D)	(None, 16, 16, 32)	0
conv2d_transpose_3 (Conv2DTranspose)	(None, 32, 32, 16)	4624
conv2d_transpose_4 (Conv2DTranspose)	(None, 64, 64, 16)	2320
up_sampling2d_1 (UpSampling2D)	(None, 128, 128, 16)	0
conv2d_transpose_5 (Conv2DTranspose)	(None, 256, 256, 4)	580
conv2d_transpose_6 (Conv2DTranspose)	(None, 512, 512, 3)	111

Total params: 1,411,763
Trainable params: 1,411,763
Non-trainable params: 0

Decoder Model

Encoder:

- Takes an input image.
- Contains multiple convolutional layers to extract features
- Uses MaxPooling layers for downsampling.
- Ends with a Flatten layer and a dense layer (bottleneck) to produce an encoded representation (latent space) of the input image, with a dimensionality of 128.

Decoder:

- Takes the encoded representation as input.
- Contains dense and convolutional transpose layers to upsample and reconstruct the original image.
- Ends with a final convolutional layer with a sigmoid activation function to generate pixel values.

Training:
During training, the Autoencoder is optimized to minimize the **Mean Squared Error** (MSE) loss between the input image and the reconstructed image.The training process aimed to capture the most salient facial features in the latent space.

Obtaining the feature vectors of the images:

- After training, the encoder part of the Autoencoder has learned to compress input images into a lower-dimensional representation, capturing essential features.

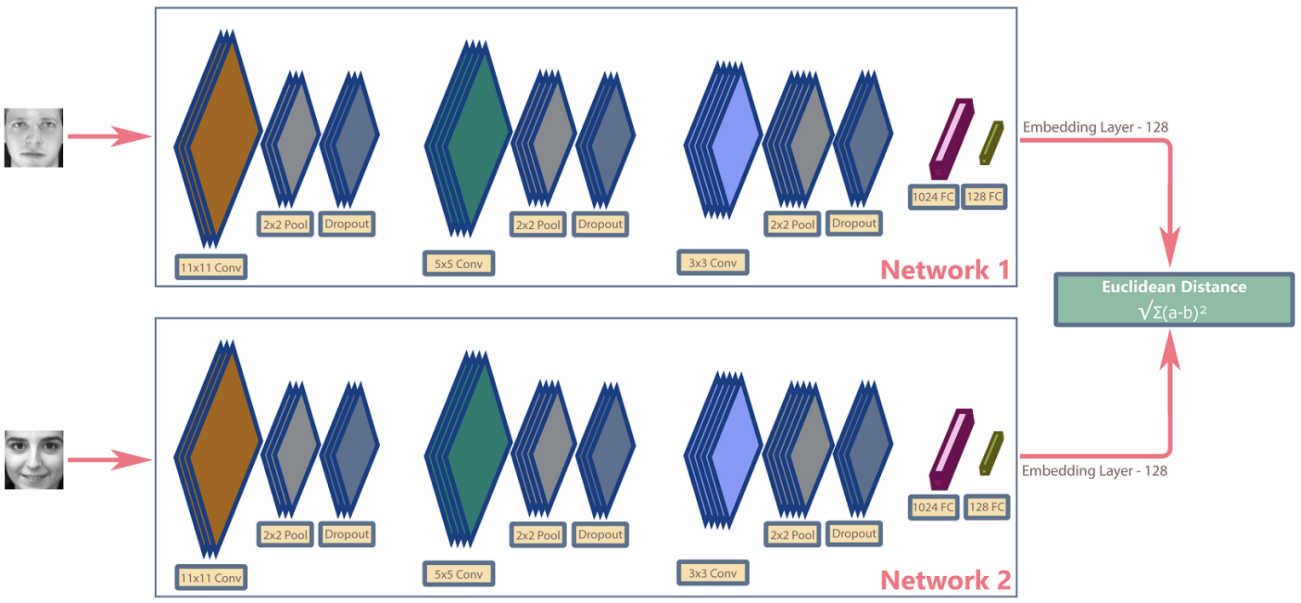
- To convert test images into a lower-dimensional representation, the test images are passed through the trained encoder.

- The encoder takes the test image as input and produces a compact vector, typically with a dimensionality of 128 in this case.

- This lower-dimensional representation serves as an efficient and informative feature vector that can be compared with other images for tasks like face verification.

Part c: DNN based approach:

- **Using a Siamese network trained using Triplet Loss**
  1. **Model description:** In contrast to a typical Convolutional Neural Network (CNN), the Siamese Network doesn't categorize images into specific classes or labels. Instead, its primary task is to determine the dissimilarity or distance between any pair of provided images. It consists of two identical subnetworks (hence the name "Siamese") that share the same architecture and weights.
     When the two images share the same label, the network aims to adjust its parameters, including weights and biases, to minimize the distance between these two images. Conversely, if the images belong to distinct labels, the network should increase the distance between them.
     The role of the Siamese network's encoder is to transform the provided images into their respective feature vectors. To achieve this, we are utilizing a pre-trained Xception model, which is built upon the Inception_V3 model.



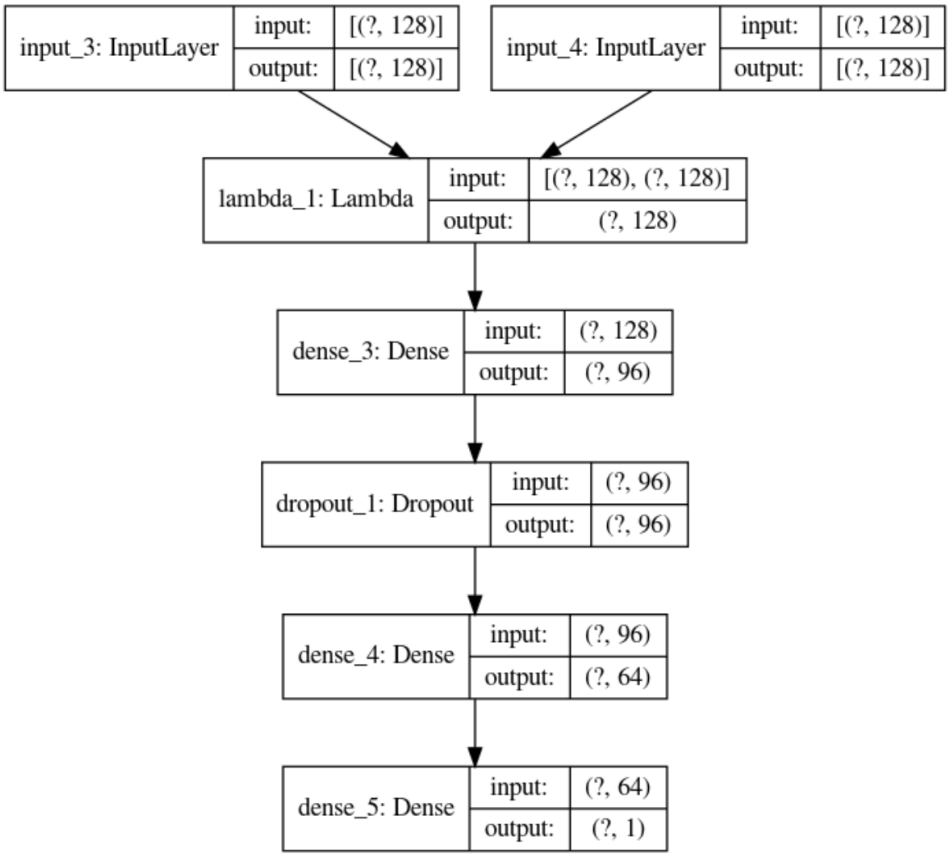
  2. **Creating triplets:** Triplets of (Anchor, Positive & Negative) are created from the training data, for training the Siamese Network using Triplet loss. Positive is the same person and negative is a different person than the anchor.
  3. **Obtaining the feature vectors of the images:** After the training process, the encoder of the Siamese Network has learnt to create embeddings corresponding to the facial images such that similar faces are close to each other in the embedding space and dissimilar faces are farther apart in the embedding space. We pass all the images through this encoder to obtain their corresponding feature vector.

- **Using a pretrained FaceNet model**
  - **Model description:** FaceNet is a start-of-art face recognition, verification and clustering neural network. It is a 22-layers deep neural network that directly trains its output to be a 128-dimensional embedding. FaceNet learns and extracts various facial features. These features are then mapped to a 128-dimensional space, where images belonging to the same person are close to each other and far from images belonging to different subjects.
  - **Obtaining the feature vectors of the images:** A pretrained Facenet model which has been trained on the VGGFace2 dataset consisting of ~3.3M faces and 9000 classes has been used to predict a 128-dimensional feature vector for each image in the dataset.

Performing face verification by feeding the feature vectors of facial images into a simple neural network (classifier):

- **Architecture of the classifier:** We use a simple neural network to classify pairs of images into two classes — “the two images are of the same person” and “the two images belong to different people”. This is the definition of the Face Verification task.
  The classifier computes the euclidean distance between the feature vectors of the two input images, and passes this distance through some simple non-linear computations to perform the classification.

  The architecture of the classifier is as follows:



- **Obtaining pairs of images from the train and test dataset:**

  The feature vectors obtained from the various architectures described above need to be paired in order to be fed into the classifier. The classifier then classifies the pairs of images into two classes. If all possible pairs of feature vectors are paired together, then an extremely large amount of pairs (~ 200,000) would be generated, moreover the number of dissimilar pairs would be much greater than the number of similar pairs. This results in the creation of a biased dataset on which the classifier is trained on, which would lead to the classifier becoming biased towards dissimilar pairs. Hence in order to generate an even and unbiased dataset which contains the same number of similar and dissimilar pairs while also ensuring that every person is adequately represented in this dataset the following strategy is used seperately for training and testing data:

  - **Compute a frequency distribution based on the number of images of each person present in the dataset** which calculates the "contribution" of each person to the dataset.

  - **Iterate over each persons folder** and create all possible pairs with the images from that folder and label each pair with a '0' (representing similar images). This results in the creation of all possible similar images.

  - **Iterate over pairs of two different folders** and create pairs of images after calculating the number of pairs to be made from each folder using the frequency distribution which was computed, label each pair with a '1'(representing dissimilar images). This ensures that all persons are represented in the dataset in the ratio of the number of images that they have. Hence it leads to a more equitable distribution.

  - **If the dissimilar images are not equal to the similar images after the above steps,** randomly choose people and create random pairs until the number of dissimilar images become equal to the number of similar images. The number of random pairs created would be very small compared to the size of the dataset. For example, in the training data, we had to create 214 random pairs and the total number of pairs were 8192. (4096 similar and 4096 dissimilar)

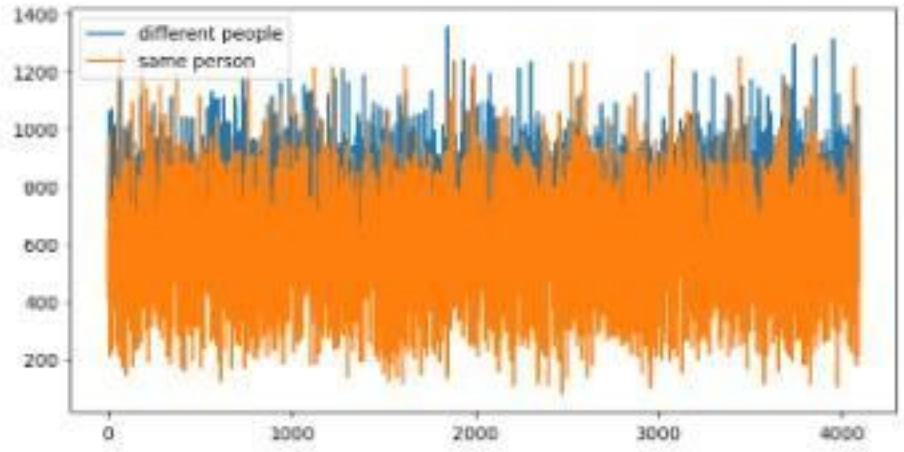
  The feature vector pairs generated in this manner are then fed to the classifier.

- **Results on the test image-pairs (for all the approaches tried out):**

1) Accuracy for PCA based approach:

	Feature-vector dimension = 128	Feature-vector dimension = 256	Feature-vector dimension = 512
Using sklearn	89.96%	88.64%	84.09%
From scratch	80.87%	84.85%	88.64%

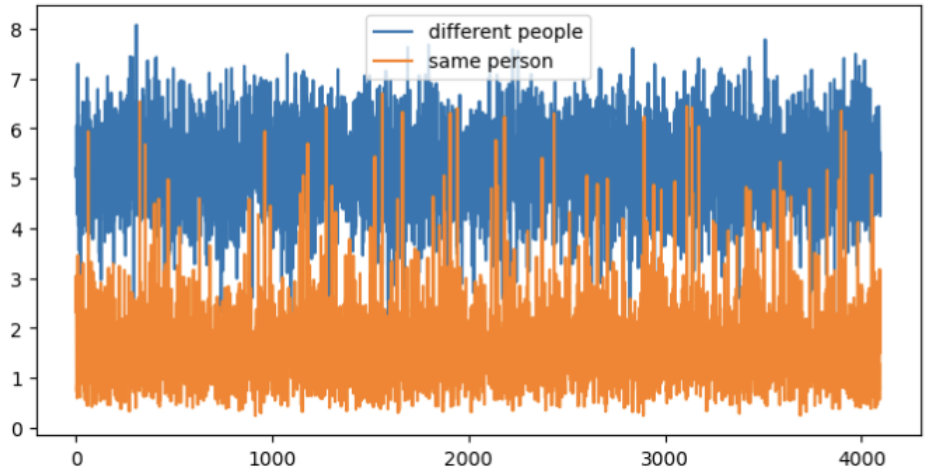
Plot of distances between image-pairs of same person as well as distances between image-pairs of different people: (plot is made for image pairs from training data)



2) Accuracy for AutoEncoder based approach:

	Feature-vector dimension = 128	Feature-vector dimension = 256	Feature-vector dimension = 512
AutoEncoder trained for 150 epochs	95.08%	92.99%	96.02%
AutoEncoder trained for 250 epochs	93.94%	93.94%	94.70%

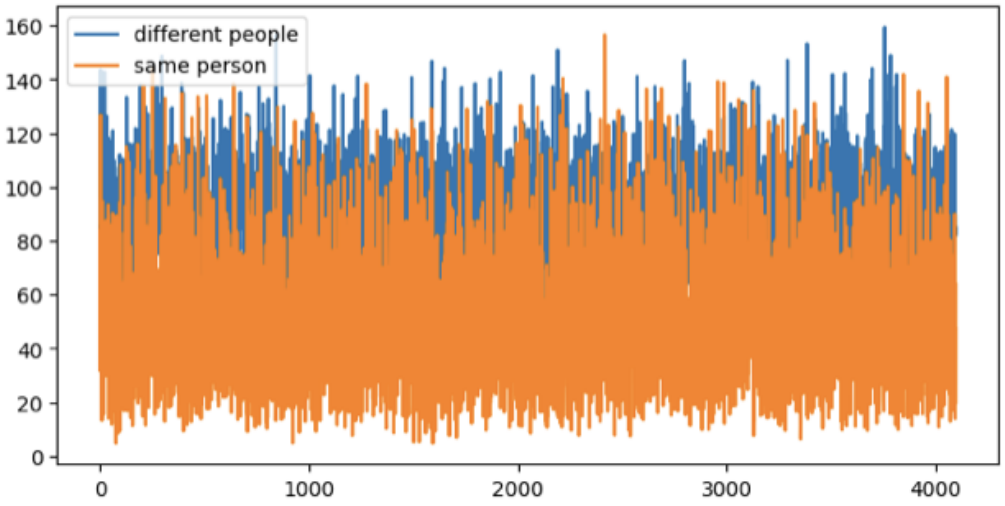
Plot of distances between image-pairs of same person as well as distances between image-pairs of different people: (plot is made for image pairs from training data)



3) Accuracy for Siamese Network based approach:

	Feature-vector dimension = 128	Feature-vector dimension = 256	Feature-vector dimension = 512
Siamese Net trained for 10 epochs	97.35	96.59	98.67%
Siamese Net trained for 30 epochs	98.30%	99.24	98.11%

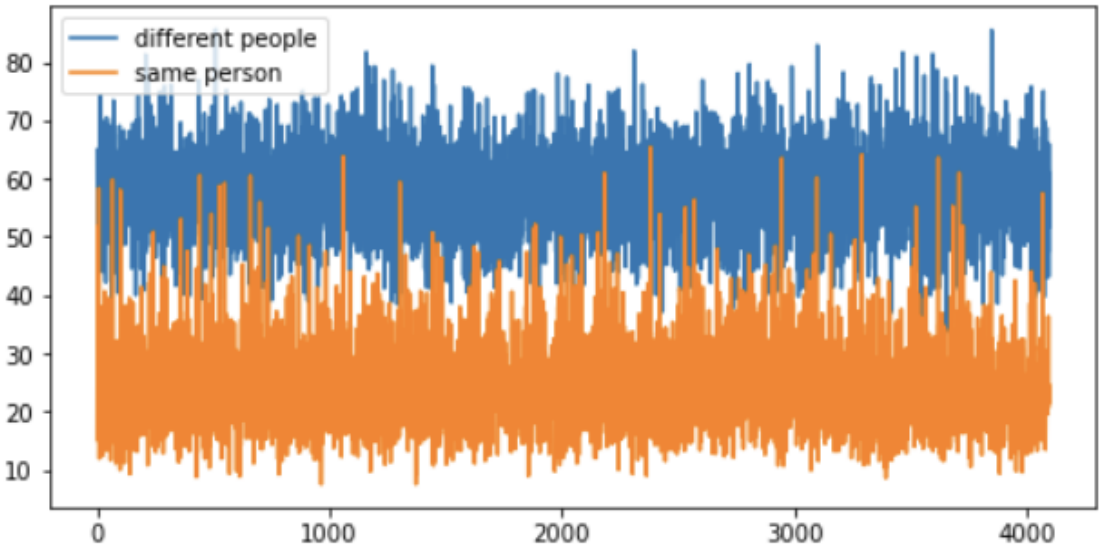
Plot of distances between image-pairs of same person as well as distances between image-pairs of different people: (plot is made for image pairs from training data)



4) Accuracy for FaceNet based approach:

	Feature-vector dimension = 128
Using pretrained model	98.67%

Plot of distances between image-pairs of same person as well as distances between image-pairs of different people: (plot is made for image pairs from training data)



Observations:

- For PCA, on average, the “sklearn" variant performs better than the “from scratch” variant.

- On average, the performance of the different approaches tried out, in ascending order is as follows:
  PCA_approach < AutoEncoder_approach < SiameseNetwork_approach < FaceNet_approach

  Specifically, the best accuracy of 99.24% is given by the Siamese Network approach with a feature vector dimension of 256, when trained for 30 epochs.

- From the plot of distances between image pairs, we can see that the DNN based approaches (i.e. Siamese network and FaceNet) is able to group together images of same people way more nicely as compared to the PCA or AutoEncoder based approaches.

Links to our kaggle notebooks:

- **For PCA based approach:**
  https://www.kaggle.com/code/jshark/faceverification-pca

- **For AutoEncoder based approach:**
  https://www.kaggle.com/code/jshark/task1-part2-autoencoder

- **For Siamese Network based approach:**
  https://www.kaggle.com/code/kotturug/task1-part3-siamese

- **For FaceNet based approach:**
  https://www.kaggle.com/code/riddhich/task1-part3-facenet