FACE RECOGNITION PROJECT

By: Swagata Chakraborty

July 2021

1. Image collection:

In this project, we build a dataset of 150 images each of Barack Obama, Donald Trump, and George W Bush. The dataset is expandable, we can add more folders of 150+ training images of any person.

2. Loading images:

Using the OS library, we access the image folders and load the images and the names of the folder i.e., Barack Obama, Donald Trump, and George W Bush.

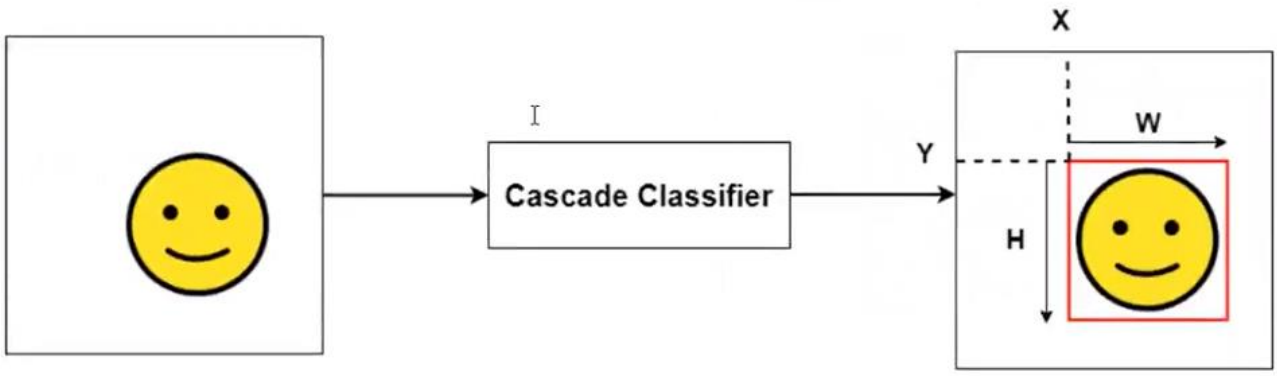
3. Pre-processing images:

For pre-processing images, first we need to identify the **Region of Interest (ROI)**. For a face recognition project, we need to detect the faces at first. And for face detection, the ROI is the face itself. Hence, we need to detect the faces and crop it off the image.

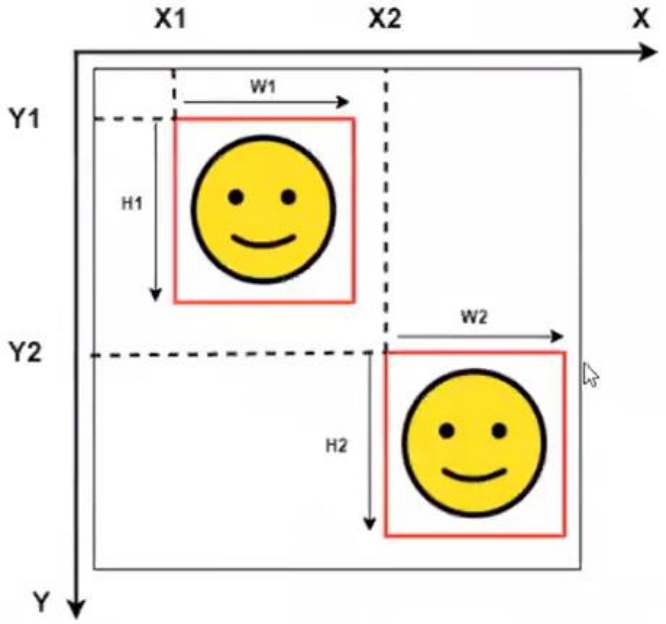
For face detection, we have used **Haarcascacde Frontal Face Classifier**. Several pre-trained haarcascade classifiers are available which are capable of detecting only one specific object at a time. Face classifier detects multiple faces from a given image but sometimes detects some wrong objects as well.

We load the Haarcascacade Classifier in our code using the CascadeClassifier function present in OpenCV library. Then we run a loop for each folder, inside each folder we run a loop for accessing each image present inside the folder. We convert each image into grayscale because grayscale images are single-dimensional whereas RGB images have three dimensions. By reducing dimensions, we reduce the working complexity of the model. Moreover, OpenCV library works on grayscale images only i.e., single dimensional images.

Then pass those grayscale images to the CascadeClassifier function. The function returns an array of one or more parameters/dimensions. It returns 4 parameters for each image – [x,y,w,h] where x and y gives the starting position of the ROI i.e., the face from the left upper corner of the image, w and h gives the height and width of the faces.



To be sure of what is detected by the classifier, we can draw a rectangle surrounding the ROI parameters returned by the function. While doing so we saw that some wrong objects has been detected as ROI. Hence, we need to remove those objects in order to increase the accuracy of our model.




```
Faces=[[X1,Y1,W1,H1],
       [X2,Y2,W2,H2]]
```

So, we loop through each [x,y,w,h], crop the faces, convert the faces into a common dimension (50*50 in this project) and add the faces to the dataset if it is a correctly detected face and reject the wrong detected objects. However, this choosing had to be done manually for each and every image.

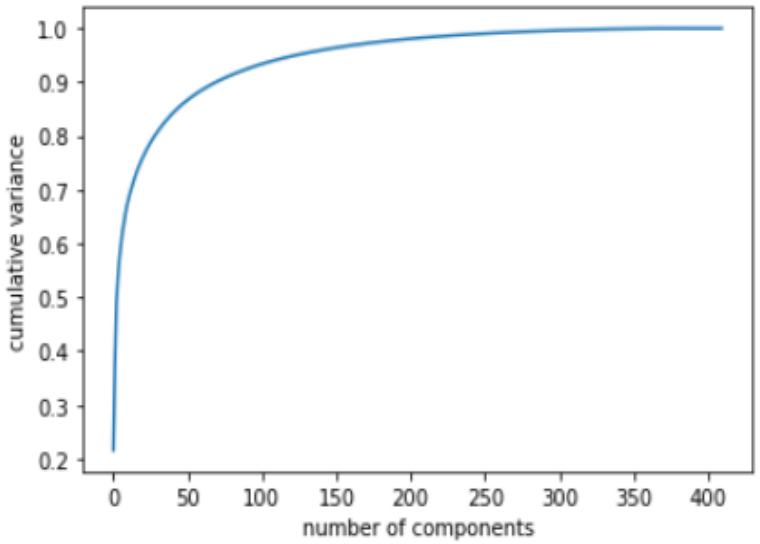
Then we flatten the images and save them in a NumPy array. We flatten the array because of multiple reasons, one possible reason could be 1D arrays take less amount of memory space. Specially, when we are dealing with large number of images, flattening the array brings significant changes like reducing memory usage and faster model training. Moreover, OpenCV libraries cannot

work with multidimensional images. For neural networks we don't need to flatten the images.

4. Model Development:

For model development, we load the pre-processed dataset and do a train-test split at first. On printing the size of the dataset, we get (410,2500) which implies we have 410 images of 2500 features/pixels. Now, a machine learning algorithm cannot handle 2500 features. It will give very low accuracy if such a large number of features are given.

Hence, we need to find a way to reduce these features without loosing important information. This is done using **Principal Component Analysis (PCA)**. PCA analyses the most important features of the images and removes the less important ones. But how many features/components should we removed is determined by plotting a graph. The graph shows the cumulative variance for various number of components.



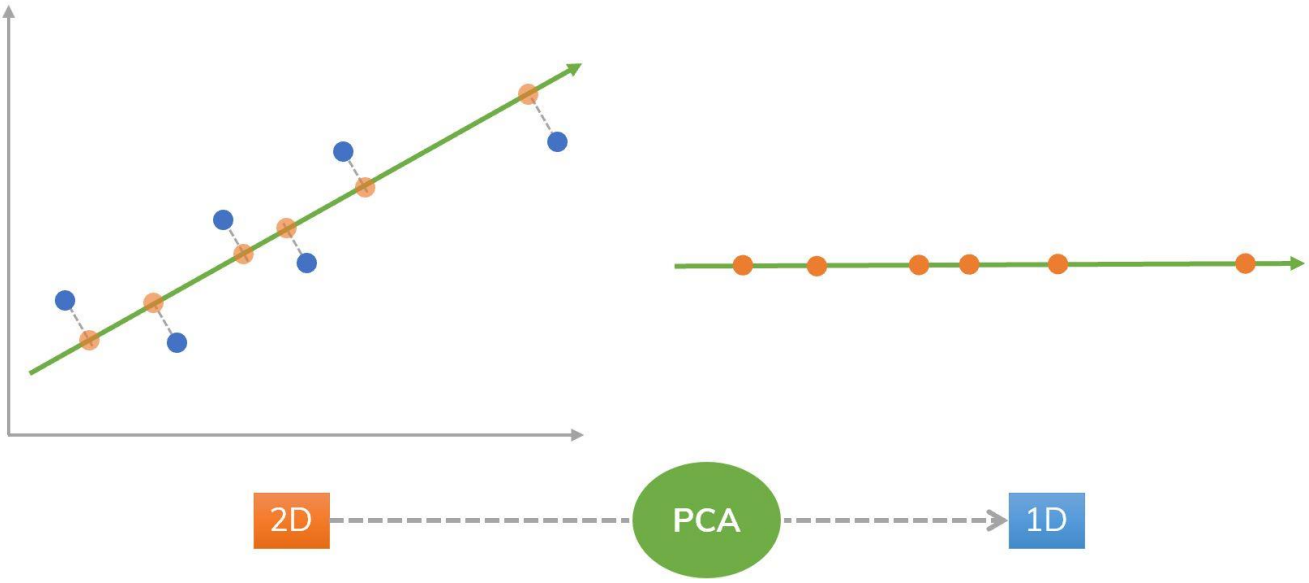
From the graph we understand that cumulative variance is almost constant 150 number on components onwards. Hence, we choose 150 as the optimal number of components.

Next, we create a pipeline where we first pass the image through PCA for dimension reduction and then through **Support Vector Classifier (SVC)** i.e., the classification version of Support Vector Machine (SVM).
We feed the train data and train target to the model and then do the prediction. We measure the accuracy of the model, which in this project came 0.93

In the end, we save the model using Joblib library.

Principal Component Analysis:

It is a dimension reduction technique which is used to reduce large number of features to required number of features without loosing important information. It is also used to reduce overfitting problem. Suppose we want to convert 2D features to 1D i.e., we need to project all the two-dimensional points into a single line.



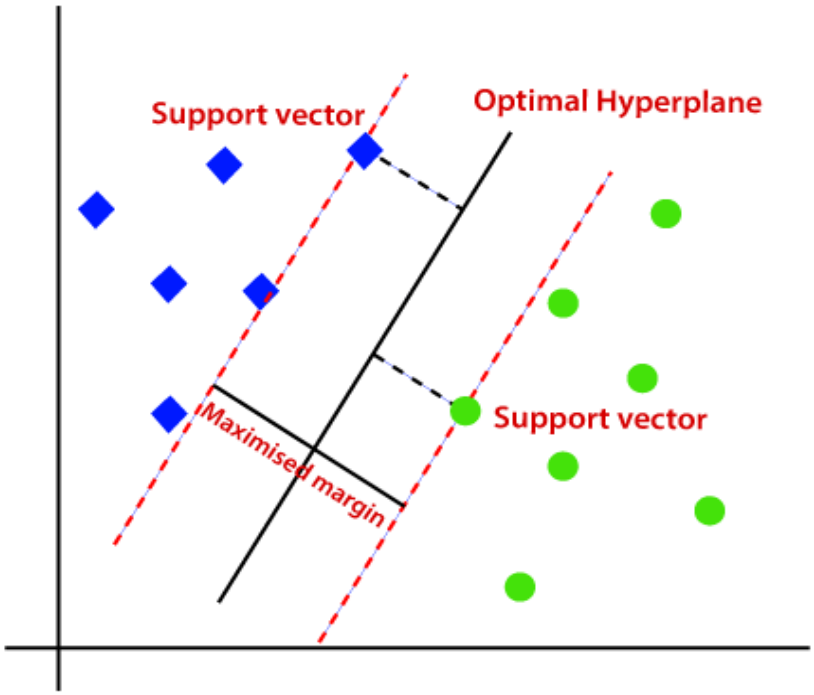
First, we need to draw the best fit hyperplane called Principal Component line and try to project every single point to the line. We may create another perpendicular principal component line and try to project every single point to the line. But it may cost lots of variances and such large variance we lead to lots of loss of information. Hence the optimal line is the one having least variance. Suppose we want to convert 1000 features to 100. PCA will create many principal component lines perpendicular to each other with respect to the dimensions. Finally, the top 100 best line having least variance will be selected.

Support Vector Machine:

It is a supervised machine learning algorithm which can be used to solve both classification and regression problems. For classification problems we use Support Vector Classifier (**SVC**). It creates a hyperplane and two margin lines having some distance from the optimal hyperplane. While creating the hyperplanes it makes sure that the data is linearly separable. The margin planes are parallel to the hyperplane and it must pass through at least one the

nearest points on each side. The nearest point through which the margin plane passes is called the support vector.

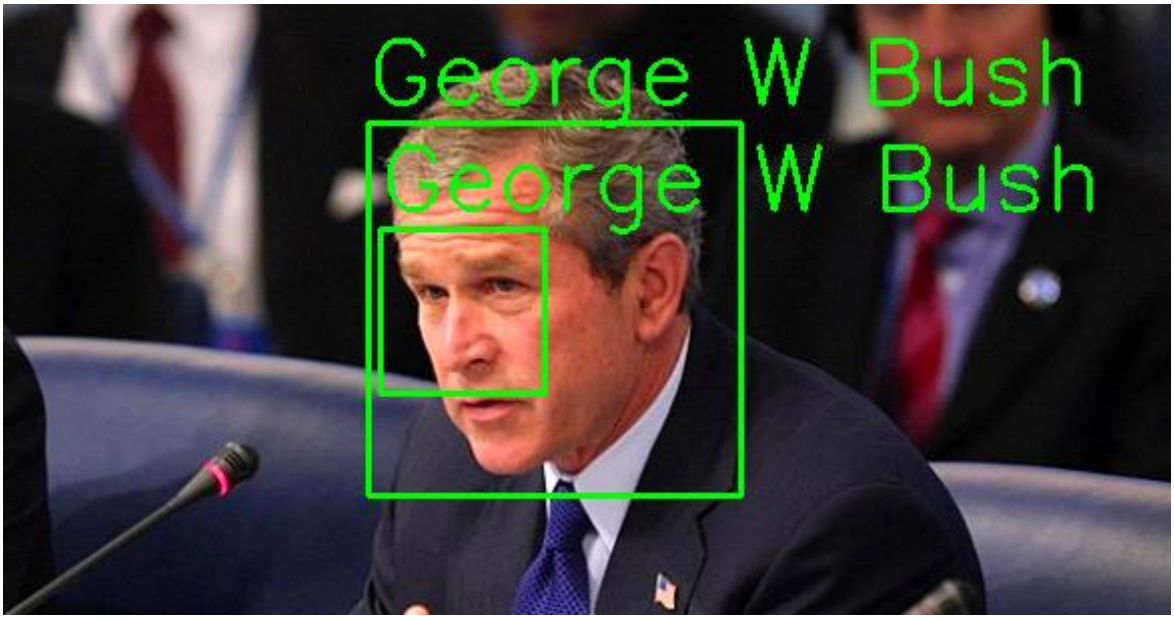
The importance of the margin is that it creates a cushion around the hyperplane and divides the groups in better way. Any test data that falls beyond the margin or in the margin area can be easily classified to some group. The margin distance must be maximum. It creates multiple hyperplanes and checks the margin distance. The optimal hyperplane is chosen when the margin distance is maximum. The higher the marginal distance, more generalized model we have. However, this is possible for linearly separable data.



In case of non-linearly separable data, SVM uses a technique called **SVM Kernels** which transforms low dimension into higher dimensions. By doing this, the data becomes linearly separable by a hyperplane. There are various types of SVM Kernels like, Polynomial Kernel, RBF Kernel, and Sigmoid Kernel. In this project, we have used RBF kernel i.e., Radial Basis Function Kernel. RBF finds support vector classifiers in infinite dimensions. It projects the x1, x2 parameters to x1, x2, x1*x2, x1^2, (x1^2)*x2, x2^2, x1*(x2^2), (x1^2)*(x2^2), x1^3, …… up to infinity. It considers infinite number of interactions between the variables.

5. Testing:

We load the test images, and they undergo the same pre-processing like the train images. Then we predict the name and create a rectangle around the faces of the train images. We add the names 10px above the rectangle and give the font style, font scale, color and thickness. All predictions were correct, just one image of George W Bush was identified twice.



6. Conclusion:

In this project, we have successful build a face recognition algorithm which works almost perfectly. We have used various tools like Haar Cascade Classifier, PCA and SVC in making the project.