**Introduction:**

Since the 1960s, face recognition has been a topic of interest. Over the years, technological advancements have enabled the enhancement and integration of this technology into everyday life. Face recognition is increasingly being used by law enforcement and security organizations in security surveillance and crime detection. This technology has been integrated into every aspect of our lives, from phone unlocking mechanisms to prevention or detection of identity theft.

Face recognition involves using an individual’s face in order to recognize and identify them. It has a wide variety of uses, ranging from employee identification in large corporations to student attendance in educational institutions. A face recognizer is made up of two parts. The first part involves a face detection method, which consists of finding faces in a photo or in a video stream, which often poses a challenge during the face detection. The second part involves verifying or identifying the person's identity. While there are numerous methods for achieving the same, this paper proposes a method to implement a Face Recognition system using machine learning which can be used in many day-to-day applications.

In this project, I have used the concept of HOG face detection to find the coordinates of the face in the image. This method of face detection is used to convert the image to grayscale and then draws the arrows that indicate the flow from lighter pixels to darker pixels, these arrows are called gradients. This process is useful to extract the actual features from the image and remove the distinction between dark and bright images of the same person. This helps in capturing the actual feature from the image. I have used the deep learning method called Convolutional Neural Network (CNN) to create face recognition model.

**Dataset**

For the dataset, I have used the face recognition dataset from Kaggle[1]. This dataset consists of 31 folders each of which represents a person who is a famous celebrity. Each of these folders contains 50 to 100 images of that celebrity. This dataset consists of a total amount of 2562 images and has a size of 726 MB. For analysis, I have excluded 2 folders from the data as both folders contain a single image of that celebrity.

**Concept of HOG:**

A feature extraction module named Histogram of Oriented Gradients (HOG) is frequently employed to extract facial features from the input passed to the algorithm. Other uses of this algorithm are in object detection in the field of computer vision. The HOG feature emphasizes an object's structure or shape and only evaluates if a pixel is an edge in the case of edge features. HOG also gives the edge direction. To achieve this, the gradient and orientation of the edges are extracted. Furthermore, "localized" components of these orientations are calculated. As a result, the entire image is divided into smaller patches, and the gradients and orientation are evaluated for each of those regions. In the end, the HOG would create a distinct Histogram for each of these zones. The term "Histogram of Oriented Gradients" refers to the histograms that are produced by the gradients and orientations of the pixel values

**WORKING OF HOG DETECTOR:**

HOG is a simple yet effective feature description. It is frequently used for item detection, including cars, animals, and fruits, in addition to face detection. Because HOG is a reliable objection detection technique that uses the local intensity gradient distribution and edge direction to characterize object shape.

Step1: In HOG, we divide the image into smaller cells.

Step2: Compute histogram for each cell using the gradients and orientations of the pixels.

Step3: Put these histograms together to create the feature vector, which creates a single, distinctive histogram for each face from all the little histograms.

I have tested the code on sample images.

**The input sample image:**

A picture containing text, monitor, indoor, electronics

Description automatically generated

Fig. 1: input sample image

**HOG Output:**

**A picture containing table

Description automatically generated**

Fig. 2: HOG Output image

**Output Image:**

A person's face on a screen

Description automatically generated with medium confidence

Fig. 3: Output Image

**Results**

I have tested the model on the test images, and the sample outputs observed are as follows.

A collage of a person's face

Description automatically generated with medium confidence

Fig. 4: Predictions of test images

The confusion matrix for output data as follows:

Chart

Description automatically generated

Fig. 6: Confusion Matrix

I have used the “accuracy” metric to evaluate the model. Accuracy gives the percentage of images that are correctly classified by the model. It is calculated by dividing the number of correctly classified predictions by the total number of predictions.

**Accuracy** =  (TP+TN)(TP+TN+FP+FN)

where True Positives (TP) is the number of images that are correctly classified. True Negatives (TN) is the number of images if correctly predicted it does not belong to a certain class. False Positives (FP) are the number of images that are incorrectly classified. False Negatives (FN) are incorrectly predicted that it does not belong to a certain class.

**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated**

**Chart

Description automatically generated**

Fig.7: Accuracy plot for different epochs

The accuracy plot shows the accuracy and validation accuracy of the model with respect to epochs on the x-axis and accuracy percentage on the y-axis. We can see that the model was continuously increasing its accuracy for 8 epochs and then stayed the same with few fluctuations from the 8th epochs to the 50th epochs. From the plot, we can conclude that training the model beyond the 9th epoch was overfitting the model since the training accuracy was increasing but the validation accuracy was decreasing. The final model accuracy was 80.3%. I also performed an accuracy test on a completely new set of images and got a model accuracy of 80.4%.