Question: Detecting faces from images

Aim: To implement Project on Detecting faces from images.

Theory:

Face detection, also called facial detection, is an artificial intelligence (AI)-based computer technology used to find and identify human faces in digital images and video. Face detection technology is often used for surveillance and tracking of people in real time. It is used in various fields including security, biometrics, law enforcement, entertainment and social media.

Face detection uses machine learning (ML) and artificial neural network (ANN) technology, and plays an important role in face tracking, face analysis and facial recognition. In face analysis, face detection uses facial expressions to identify which parts of an image or video should be focused on to determine age, gender and emotions. In a facial recognition system, face detection data is required to generate a faceprint and match it with other stored faceprints.

How face detection works:

Face detection applications use AI algorithms, ML, statistical analysis and image processing to find human faces within larger images and distinguish them from nonface objects such as landscapes, buildings and other human body parts. Before face detection begins, the analyzed media is preprocessed to improve its quality and remove images that might interfere with detection.

Face detection algorithms typically start by searching for human eyes, one of the easiest features to detect. They then try to detect facial landmarks, such as eyebrows, mouth, nose, nostrils and irises. Once the algorithm concludes that it has found a facial region, it does additional tests to confirm that it has detected a face.

To ensure accuracy, the algorithms are trained on large data sets that incorporate hundreds of thousands of positive and negative images. The training improves the algorithms' ability to determine whether there are faces in an image and where they are.

Steps to Implement Detecting faces from images :

Step 1: Import the OpenCV Package

Now, let's import OpenCV and enter the input image path with the following lines of code:

```
import cv2

imagePath = 'input_image.jpg'
```

Step 2: Read the Image

Then, we need to read the image with OpenCV's imread() function:

```
img = cv2.imread(imagePath)
```

Step 3: Convert the Image to Grayscale

To improve computational efficiency, we first need to convert this image to grayscale before performing face detection on it:

```
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Step 4: Load the Classifier

Let's load the pre-trained Haar Cascade classifier that is built into OpenCV:

```
face_classifier = cv2.CascadeClassifier(

    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"

)
```

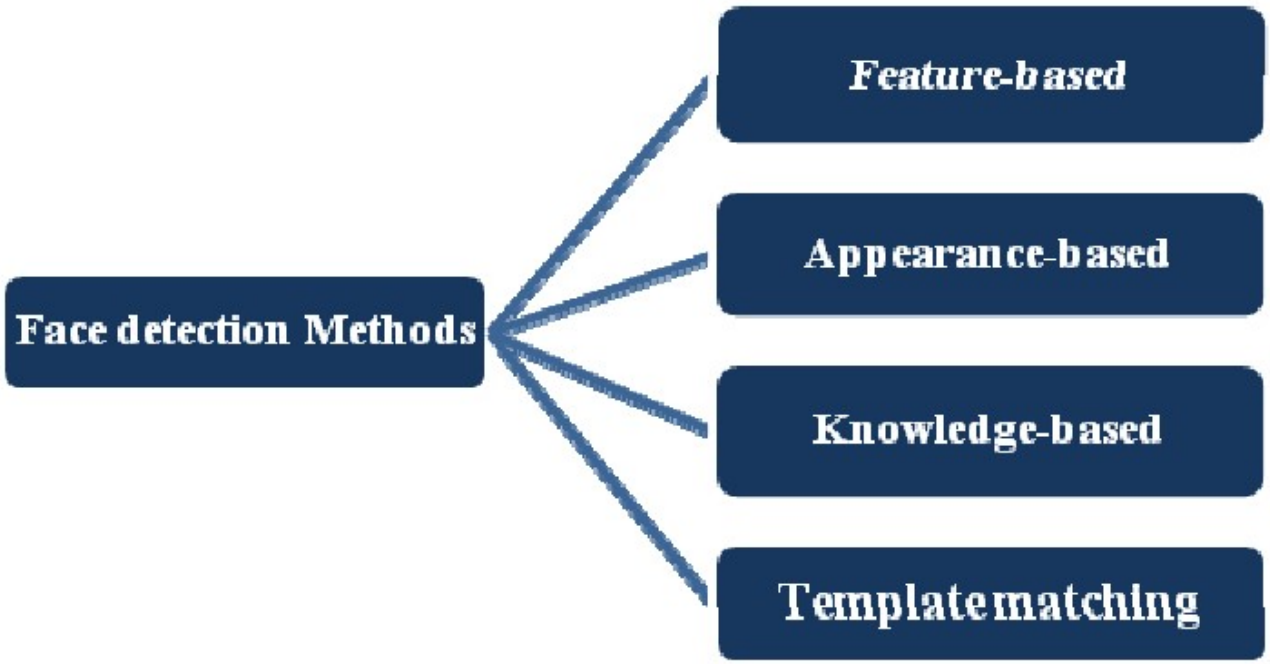
Step 5: Perform the Face Detection

We can now perform face detection on the grayscale image using the classifier we just loaded:

```
face = face_classifier.detectMultiScale(

    gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(40, 40)

)
```

Face Detection Methods:-

Yan, Kriegman, and Ahuja presented a classification for face detection methods. These methods divided into four categories, and the face detection algorithms could belong to two or more groups. These categories are as follows-



1.Knowledge-Based:-

The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex- A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules. There could be many false positive if the rules were too general or too detailed. This approach alone is insufficient and unable to find many faces in multiple images.

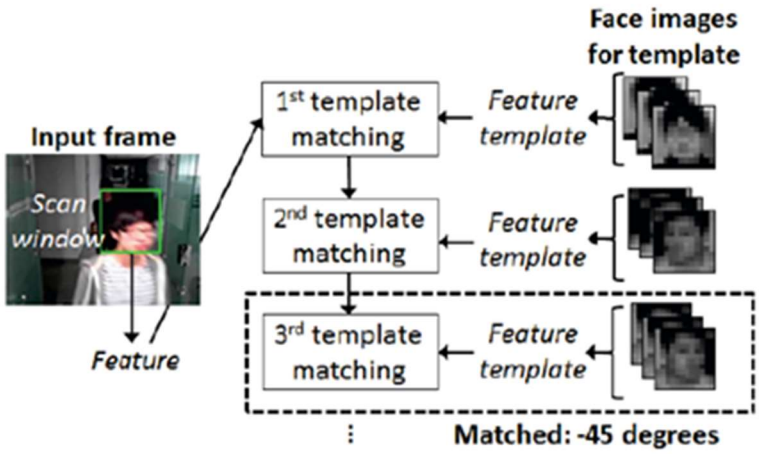
2.Feature-Based:-

The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions. The idea is to overcome the limits of our instinctive knowledge of faces. This approach divided into several steps and even photos with many faces they report a success rate of 94%.

3.Template Matching:-

Template Matching method uses pre-defined or parameterised face templates to locate or detect the faces by the correlation between the templates and input images. Ex- a human face can be divided into eyes, face contour, nose, and mouth. Also, a face model can be built by edges just by using edge detection method. This approach is simple to implement, but it is inadequate for face detection. However, deformable templates have been proposed to deal with these problems.

Template Matching



4.Appearance-Based:-

The appearance-based method depends on a set of delegate training face images to find out face models. The appearance-based approach is better than other ways of performance. In general appearance-based method rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images. This method also used in feature extraction for face recognition.

The appearance-based model further divided into sub-methods for the use of face detection which are as follows-

Eigenface-Based:-

Eigenface based algorithm used for Face Recognition, and it is a method for efficiently representing faces using Principal Component Analysis.

Distribution-Based:-

The algorithms like PCA and Fisher's Discriminant can be used to define the subspace representing facial patterns. There is a trained classifier, which correctly identifies instances of the target pattern class from the background image patterns.

Neural-Networks:-

Many detection problems like object detection, face detection, emotion detection, and face recognition, etc. have been faced successfully by Neural Networks.

Support Vector Machine:-

Support Vector Machines are linear classifiers that maximise the margin between the decision hyperplane and the examples in the training set. Osuna et al. first applied this classifier to face detection.

Sparse Network of Winnows:-

They defined a sparse network of two linear units or target nodes; one represents face patterns and other for the non-face patterns. It is less time consuming and efficient.

Naive Bayes Classifiers:-

They computed the probability of a face to be present in the picture by counting the frequency of occurrence of a series of the pattern over the training images. The classifier captured the joint statistics of local appearance and position of the faces.

Hidden Markov Model:-

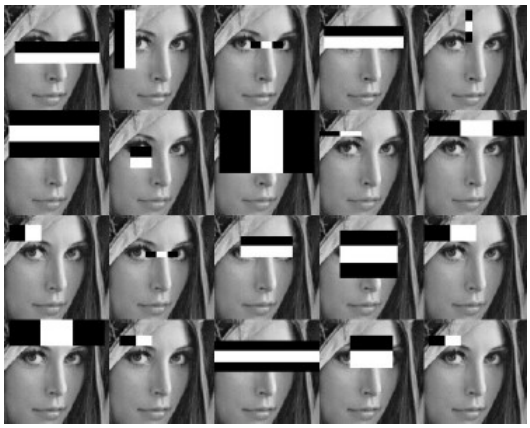
The states of the model would be the facial features, which usually described as strips of pixels. HMM's commonly used along with other methods to build detection algorithms.

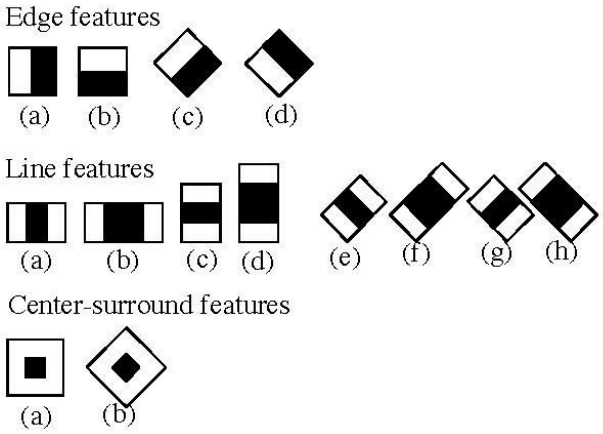
Information Theoretical Approach:-

Markov Random Fields (MRF) can use for face pattern and correlated features. The Markov process maximises the discrimination between classes using Kullback-Leibler divergence. Therefore this method can be used in Face Detection.

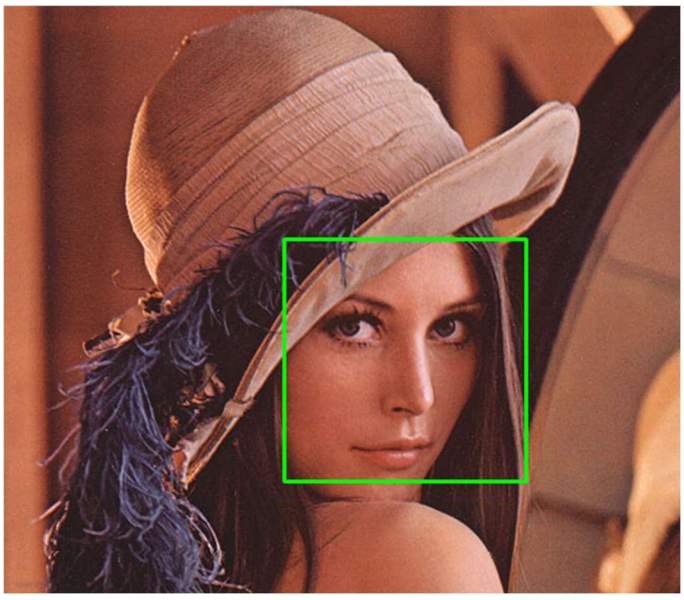
Inductive Learning:-

This approach has been used to detect faces. Algorithms like Quinlan's C4.5 or Mitchell's FIND-S used for this purpose.





The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.



Code:

```
!git clone https://github.com/virajdas/ColabFaceRecognition-OpenCV/
```

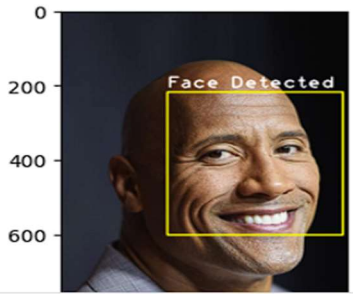
```
import cv2
from matplotlib import pyplot as plt
%matplotlib inline

cascade = cv2.CascadeClassifier("ColabFaceRecognition-
OpenCV/haarcascade_frontalface_default.xml")
img = cv2.imread("/content/WhatsApp Image 2023-07-17 at 20.59.07.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = cascade.detectMultiScale(gray, 2, 3)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)
    cv2.putText(img, "Face Detected", (x, y - 10), 1, 3, (255, 255, 255), 3)

plt.imshow(img)
```

output:

```
!git clone https://github.com/virajdas/ColabFaceRecognition-OpenCV/

Cloning into 'ColabFaceRecognition-OpenCV'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 57 (delta 15), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (57/57), 252.16 KiB | 1.95 MiB/s, done.
```

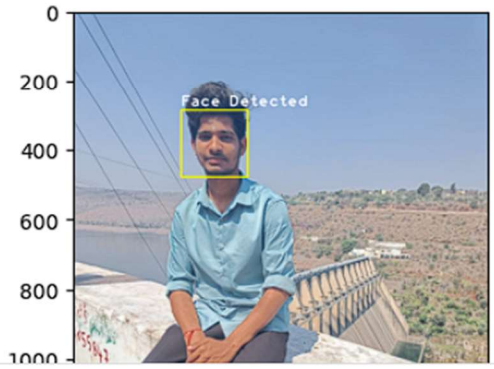
```
import cv2
from matplotlib import pyplot as plt
%matplotlib inline

cascade = cv2.CascadeClassifier("ColabFaceRecognition-OpenCV/haarcascade_frontalface_default.xml")
img = cv2.imread("ColabFaceRecognition-OpenCV/me.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = cascade.detectMultiScale(gray, 2, 3)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)
    cv2.putText(img, "Face Detected", (x, y - 10), 1, 3, (255, 255, 255), 3)

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f19c8645e40>
```




```
import cv2
from matplotlib import pyplot as plt
%matplotlib inline

cascade = cv2.CascadeClassifier("ColabFaceRecognition-OpenCV/haarcascade_frontalface_default.xml")
img = cv2.imread("/content/WhatsApp Image 2023-07-17 at 20.59.07.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = cascade.detectMultiScale(gray, 2, 3)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)
    cv2.putText(img, "Face Detected", (x, y - 10), 1, 3, (255, 255, 255), 3)

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f19c7a5cca0>
```



Advantages:

- Helps find missing people.
- Protects businesses against theft.
- Improves medical treatment.
- Strengthens security measures.
- Makes shopping more efficient.
- Reduces the number of touchpoints.
- Improves photo organization.

Disadvantages

- Threatens privacy.
- Imposes on personal freedom.
- Violates personal rights.
- Data vulnerabilities.
- Misuse causing fraud and other crimes.
- Technology is still new.
- Errors can implicate innocent people.
- Technology can be manipulated.

Applications:

Face detection is used in biometrics, often as a part of (or together with) a facial recognition system. It is also used in video surveillance, human computer interface and image database management.

Photography

Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows that use a pan-and-scale Ken Burns effect.

Modern appliances also use smile detection to take a photograph at an appropriate time.

Marketing

Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific toward the detected race/gender/age.

An example of such a system is *OptimEyes* and is integrated into the Amscreen digital signage system.

Emotional Inference

Face detection can be used as part of a software implementation of emotional inference. Emotional inference can be used to help people with autism understand the feelings of people around them.

Lip Reading

Face detection is essential for the process of language inference from visual cues. Automated lip reading has applications to help computers determine who is speaking which is needed when security is important.

Conclusion:

In conclusion, now the world becomes more and more better because of the advance in science and technology, so face recognition is slowly recognized by people, and we also began to use it in different fields. Face recognition is the use of human facial features to complete identification.