Nayan Biramane

Roll no:- 21

Aim: To perform Face detection on Video

Objective: Performing face recognition Generating the data for face recognition Recognizing faces preparing the training data Loading the data and recognizing faces.

Theory:

Generating the data for face recognition:

To generate data for face recognition in a video, you typically need to perform the following steps:
Data Collection: You will need a video containing faces that you want to recognize. This video can be either recorded by you or obtained from another source.
Face Detection: Use a face detection algorithm (like Haar cascades or a deep learning-based model) to detect faces in each frame of the video. OpenCV provides a simple way to do this, as shown in your code.
Face Recognition (Optional): If you want to recognize specific individuals, you'll need a dataset of known faces along with labels (names or IDs). You can use a pre-trained deep learning model for face recognition, like FaceNet or VGGFace, or train your own model using a labeled dataset.
Data Annotation (Optional): If you are building a face recognition dataset, you need to annotate the detected faces with the correct labels. This is important for training a face recognition model.
Data Storage: Save the detected and annotated data in a suitable format. You can store it as images with labels or in a database, depending on your needs.
Recognizing faces:

Recognizing faces in a video involves not only detecting faces but also identifying or recognizing individuals from those detected faces. To do this, you typically need a pre-trained deep learning model for face recognition. The Open cv library from python provides the necessary tools for face detection in a video. It is a pre trained model which helps in recognizing faces in a video.

Preparing the training data:

Preparing training data for face detection in videos involves preparing a dataset that can be used to train a face detection model. The goal is to create a dataset that includes both positive samples (images containing faces) and negative samples (images without faces) to train a machine learning model. Here are the key steps and considerations for processing training data for face detection in videos:

Data Collection:
Gather a diverse and representative set of video clips that contain faces. These videos can be sourced from various sources, such as movies, TV shows, or recorded footage.Capture negative samples by selecting frames from the same videos where no faces are present. This helps the model learn to distinguish between faces and non-faces.

Data Annotation:
For positive samples (frames with faces), annotate the location of each face in the image. This typically involves specifying the coordinates of the bounding box around each face.For negative samples (frames without faces), no annotations are necessary.

Data Preprocessing:

Resize images to a consistent resolution. Most deep learning models expect input images of the same size.Normalize pixel values to a specific range, such as [0, 1] or [-1, 1].Augment the dataset by applying random transformations (e.g., rotation, scaling, brightness adjustments) to increase model robustness.

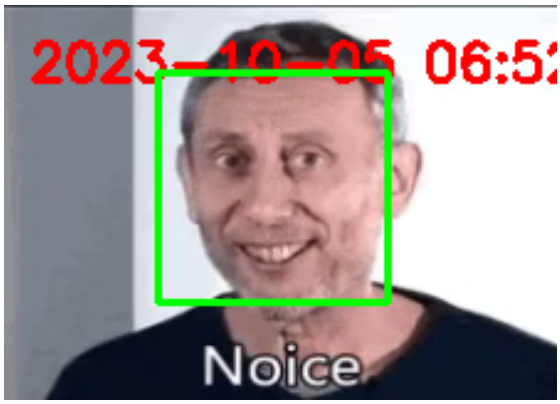
Loading the data and recognizing faces:
Loading data and recognizing faces for face detection in videos involves several steps, including data preparation, model selection, and inference. Load the trained face detection model or a pre-trained model that has been fine-tuned for your specific use case. For each frame in the video, pass it through the face detection model to identify the locations of faces. The model will return bounding box coordinates for detected faces.

Code:-

```
import cv2
import datetime
from google.colab.patches import cv2_imshow

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture('WhatsApp Video 2023-10-05 at 12.11.53 PM.mp4')

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5,
minSize=(30, 30))
```

```
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    cv2.putText(frame, timestamp, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2_imshow(frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Output:-



Conclusion:-

In this experiment we have tried to detect faces in a video. Through this experiment we have learned how to generate the data for face recognition, recognize faces, prepare the training data and load the data and recognize faces. To do this, you typically need a pre-trained deep learning model for face recognition. The Open cv library from python provides the necessary tools for face detection in a video. It is a pre-trained model which helps in recognizing faces in a video. This face detection system can be used for many purposes. In automation systems, it used to detect the face and match it for verification. Mobile phone companies use it for detecting faces using their camera to make autofocus on the people who are taking images