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: It involves the following steps:

- Data Collection: Gather video footage with the faces to be recognized.
- Video Preprocessing: Extract frames, resize, standardize, and reduce noise.
- Face Detection: Use a model to identify and extract faces from frames.
- Face Alignment (Optional): Align faces to a standardized pose.
- Data Annotation: Label detected faces with identities.
- Data Augmentation (Optional): Apply data augmentation for diversity.
- Data Splitting: Divide data into training, validation, and test sets.
- Data Storage: Organize and store preprocessed data.
- Train a Model: Use deep learning to train a face recognition model.
- Model Evaluation: Assess the model's performance on a validation set.
- Testing and Deployment: Test and deploy the model.
- Inference on New Videos: Apply the model to recognize faces in new video data, considering privacy and ethical concerns.

Recognizing faces:involves the following key steps:

- Extract frames from the video.
- Detect faces in each frame using algorithms or models.
- Optionally, track faces across frames for continuity.
- Apply face recognition to identify individuals using known face databases.
- Set a similarity threshold for matching faces.
- Annotate and visualize recognized faces.

- Handle privacy and ethical considerations.
- Evaluate system performance.
- Optimize for real-time processing.
- Deploy the system in relevant applications while adhering to privacy and ethical standards.

Preparing the training data: involves these key steps:

- Collect diverse video footage with faces.
- Extract frames, standardize, and reduce noise.
- Annotate frames with face bounding boxes.
- Optionally, apply data augmentation for diversity.
- Split data into training, validation, and test sets.
- Organize and format data for training.
- Ensure balanced positive and negative examples.
- Review and ensure annotation accuracy.
- Normalize pixel values and perform model-specific preprocessing.
- Implement a data loading pipeline for model training.
- Optionally, apply data augmentation during training.
- Train the face detection model using the prepared data.

Loading the data and recognizing faces:

- Load the video and extract frames.
- Apply face detection to locate faces in frames.
- Use a trained face recognition model to identify faces.
- Compare detected faces to a database of known individuals.
- Apply a threshold to decide on face matches.
- Optionally, annotate and visualize recognized faces.
- Address privacy and ethical considerations.

- Monitor and optimize system performance.
- Deploy the system for real-time or specific applications.
- Implement post-processing for complex scenarios.

Code:

```
import cv2
import numpy as np
from google.colab import files
from io import BytesIO
from IPython.display import display, Image
import PIL

# Function to perform face detection on a video
def detect_faces_in_video(video_path):
    # Load the pre-trained face detection model from OpenCV
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

    # Open the video file
    video_capture = cv2.VideoCapture(video_path)

    # Check if the video capture is opened successfully
    if not video_capture.isOpened():
        print("Error: Could not open video capture.")
```

```
    return


while True:
    # Read a frame from the video
    ret, frame = video_capture.read()

    if not ret:
        print("End of video.")
        break

    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Perform face detection
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5, minSize=(30, 30))

    # Draw rectangles around the detected faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame with face detections
    img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    pil_img = PIL.Image.fromarray(img_rgb)
```

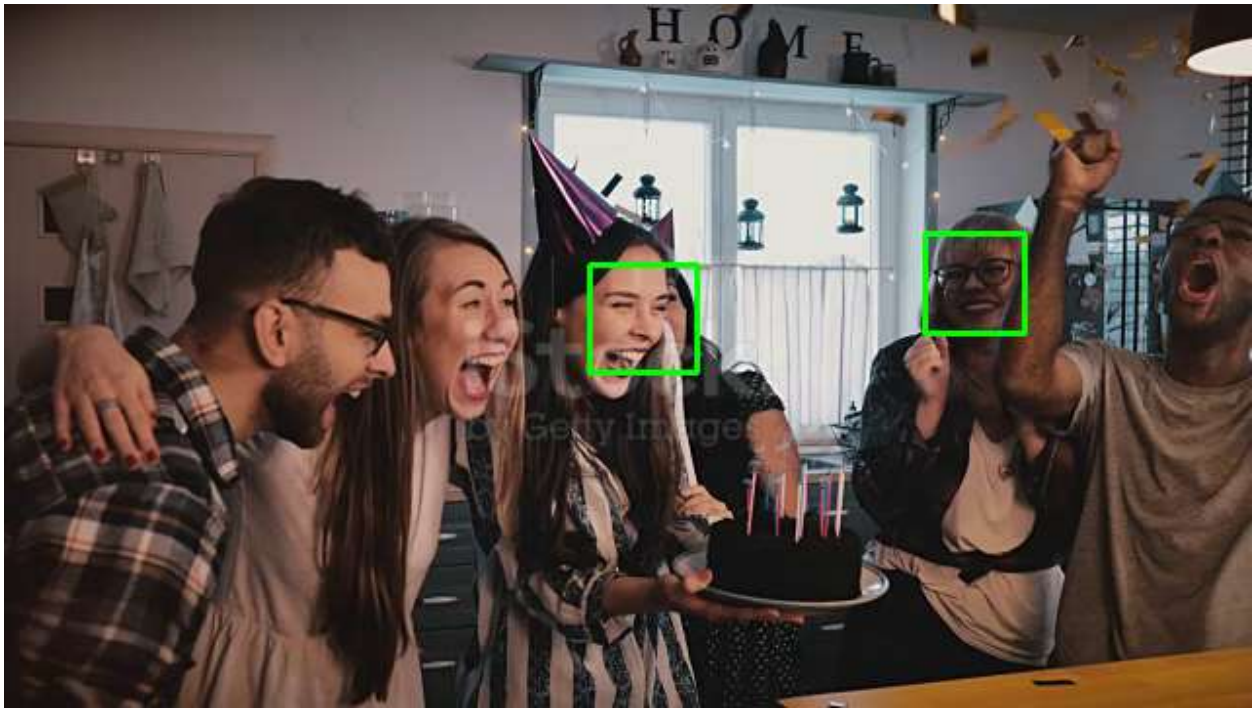
```
        display(pil_img)

    # Release the video capture object
    video_capture.release()

# Upload a video file for face detection
uploaded = files.upload()

# Process the uploaded video
if len(uploaded) > 0:
    video_path = list(uploaded.keys())[0]
    detect_faces_in_video(video_path)
else:
    print("No video file uploaded.")
```

Output:



Conclusion:

Finding and recognising faces within video frames is the process of face detection in a video. This procedure entails frame extraction, the use of face identification models or algorithms, and maybe the use of face recognition for identity verification. It is essential to prepare a broad training dataset because it guarantees the model's accuracy. Important factors to take into account include addressing privacy and ethical issues, monitoring system performance, and optimising for real-time deployment. In the end, face detection in videos is crucial for applications like surveillance and access control, and its capabilities are continuously being improved by deep learning and computer vision techniques.