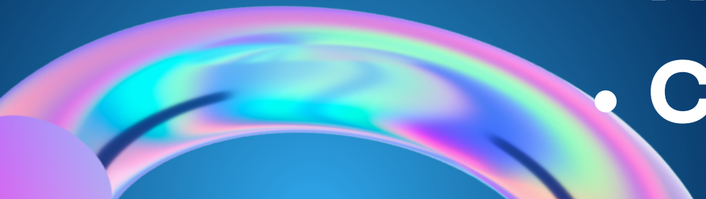
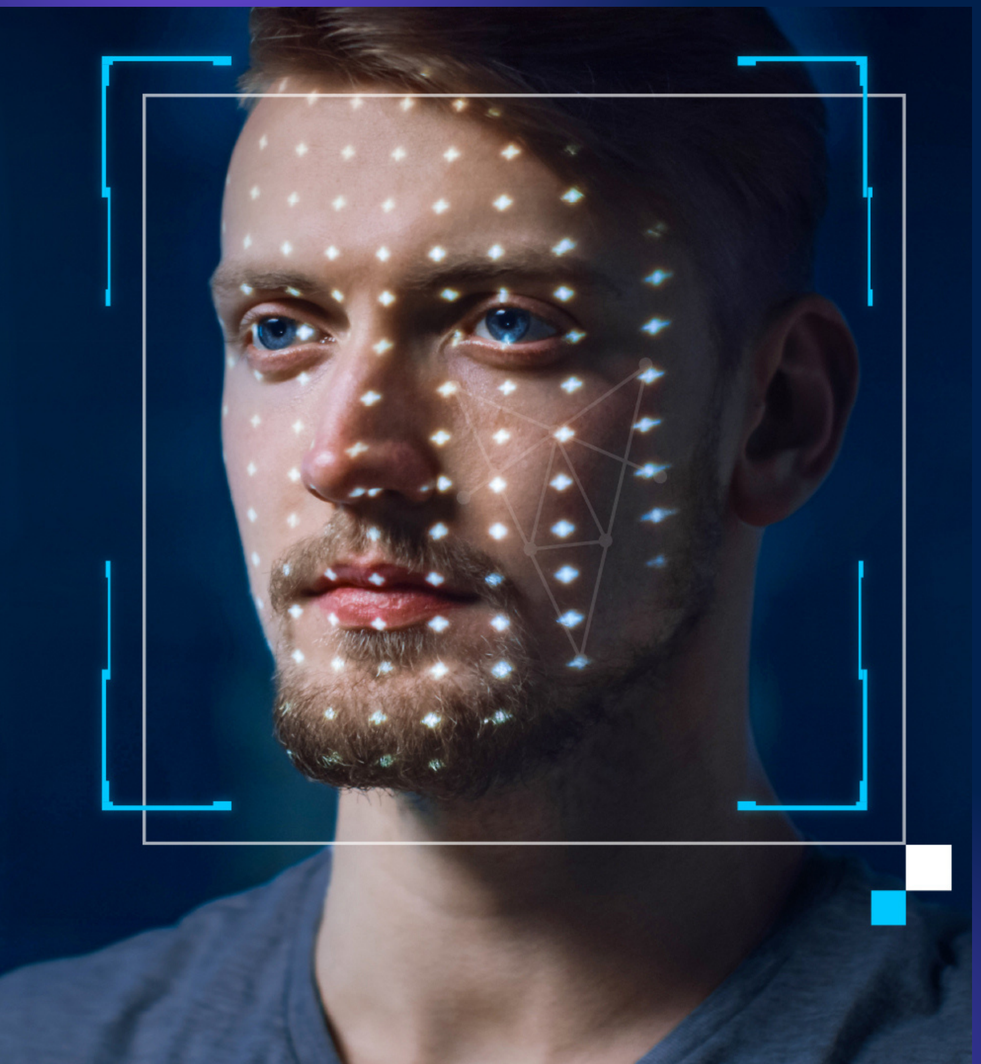
DEEPFAKE DETECTION

CONTENT:

- Abstract
- Objectives
- How it works?
- Modules Used
- Description
- Implementation
- Testing and output
- Importance
- Real-time Applications
- Conclusion



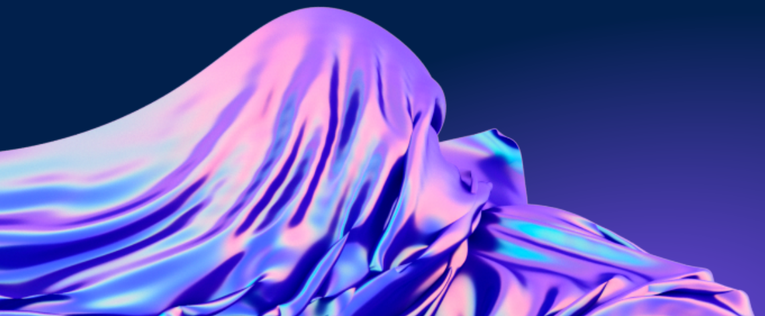
ABSTRACT:



The mini project focuses on detecting deepfake content in videos using a combination of traditional face detection techniques and the DeepFace library. Deepfake technology has become a significant concern due to its potential misuse, making it essential to develop robust detection methods. The project utilizes OpenCV for face detection and the DeepFace library, which leverages deep learning models for face verification.

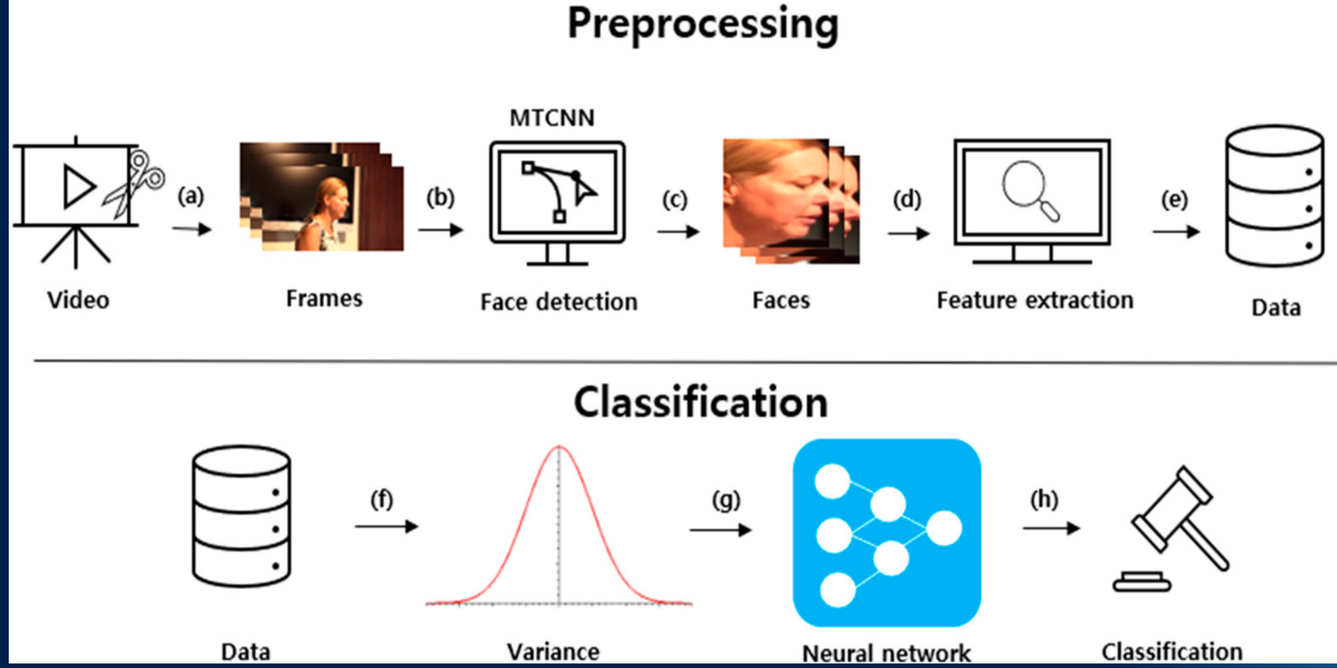
Objectives:

- Our project will distinguish and classify the videos as deepfake or real.
- Our project will reduce the abuses and misleading of the common people on tne world wide web.
- Our project aims to educate and raise awareness among users about the existence and potential dangers of deepfake videos.



How it works?

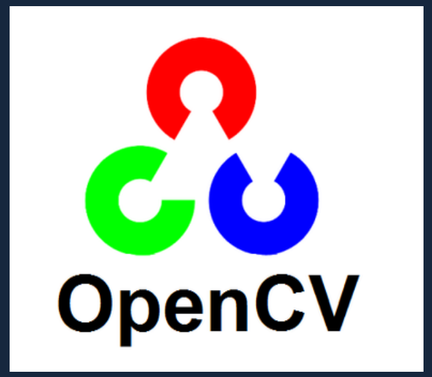
It detects deepfake in a video using OpenCv and Deepface. It analyzes each frame ,identifies faces, and verifies them .If manipulation is detected, it signals a potential deepfake .



Modules Used:

OpenCV

Used for video file manipulation,frame processing,face detection and display



DeepFace

Employed for deepfake verification using the FaceNet model

Description :

Face Detection:

The Haar Cascade classifier is employed for face detection in each frame of the video.The classifier identifies potential face regions based on predefined features and scales.

Deepfake Verification:

For each detected face, a region of interest (ROI) is extracted from the frame.
The extracted face region is passed to the DeepFace.verify function for deepfake verification.The DeepFace.verify function utilizes a pre-trained deep learning model (specifically, the FaceNet model) to compare the input face against a reference face.

Result Handling:

If a deepfake is detected (result["verified"] is True), the deepfake_detected flag is set to True.The loop breaks if a deepfake is detected or if the user presses the 'q' key.

Implementation:

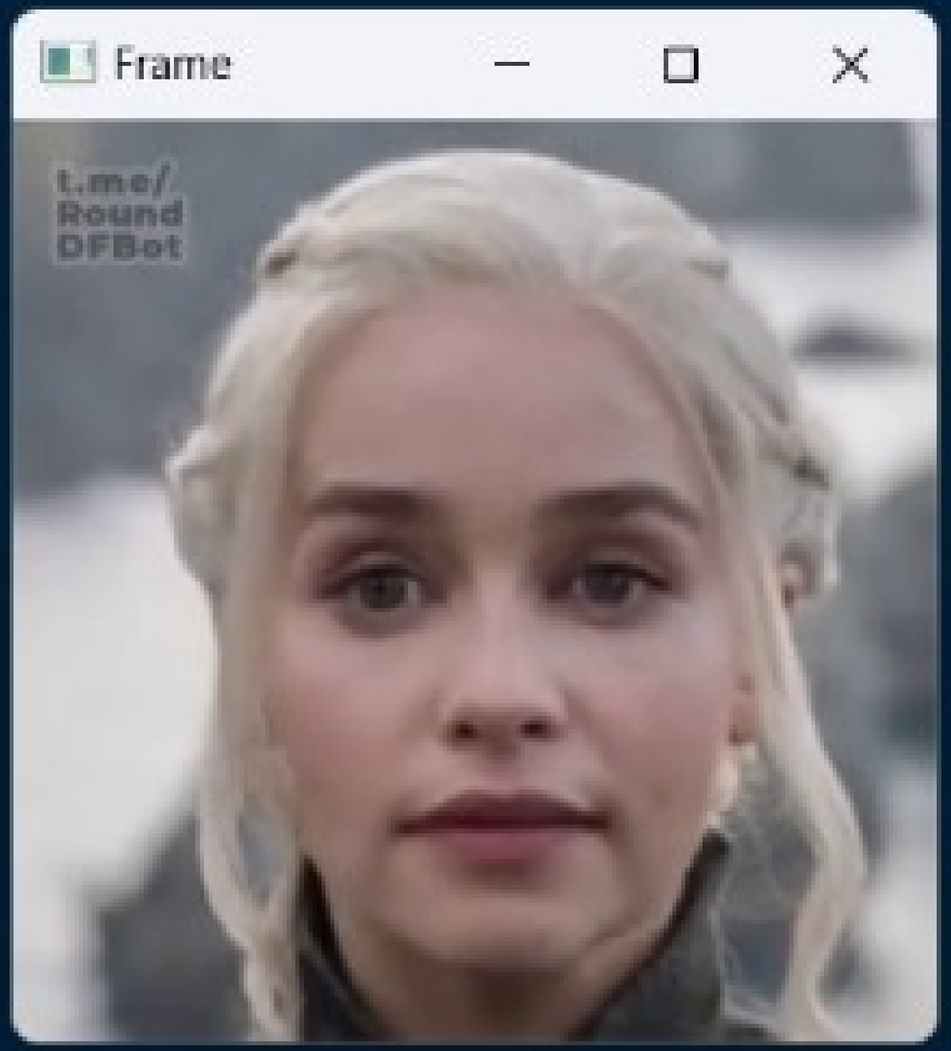
```
import cv2
from deepface import DeepFace
def detect_deepfake(video_path):
    # Load the pre-trained face detection model from OpenCV
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    # Open the video file
    cap = cv2.VideoCapture(video_path)
    while cap.isOpened():
        # Read a frame from the video
        ret, frame = cap.read()
       if not ret:
            break
       # Convert the frame to grayscale for face detection
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
       # Detect faces in the frame
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=
(30, 30))
```

```
# Check if faces are present in the frame
if len(faces) > 0:
        # Crop the detected face(s) and use DeepFace for verification
         for (x, y, w, h) in faces:
                face = frame[y:y+h,x:x+w]
                try:
                        result = DeepFace.verify (frame, "E:\python\referance
            image.jpg",detector_backend='opencv', model_name="Facenet")
                        # The 'verified' field indicates if the content is likely manipulated
                         if result["verified"]:
                                print("This video may contain deepfake“)
                         else:
                                print("No deepfake detected.")
```

```
                except Exception as e:
                        print("Error:”,str(e))
            # Display the frame
            cv2.imshow('Frame', frame)
            # Break the loop if 'q' is pressed
            if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            # Release the video capture object and close all windows
             cap.release
cv2.destroyAllWindows()
# Replace 'path/to/your/video.mp4' with the actual path to your video file
video_path = "E:\python\video_name.mp4"detect_deepfake(video_path)
```

Output

This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.
This video may contain a deepfake.



Importance

- Preserve trust in media
- Prevent AI misuses
- Protect individual reputation
- Maintain data integrity
- Prevent the spread of false content

Real-time application

Social Media Platforms

Actively identify and flag deepfake videos on social media platforms.

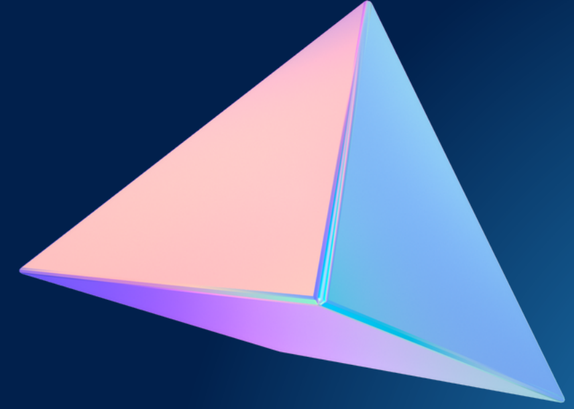
political compaigns

Actively monitor and counter deepfake videos during political compaigns.

Cyber-security Monitoring

Actively monitor network traffic for deepfake threads and providing real-time protection

Conclusion:



The integration of OpenCV and DeepFace empowers robust video analysis. As we harness these tools, it is crucial for individuals to exercise awareness in their social media activities, discerning what to share responsibly. This mindful approach ensures a safer and more trustworthy online environment.

Thank You!!!

presented by

Santhiya S

Abinaya A