Net Guard

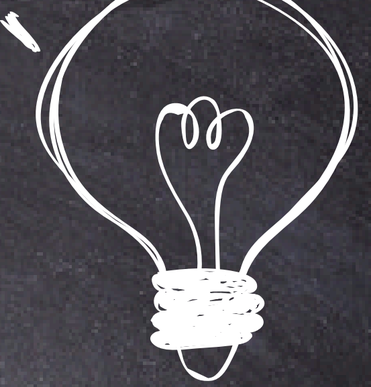
PROJECT TEAM NO : 11

Real-time Intrusion Detection with Virtual Recognition for Network Security

DEEPANSHU DHAMA (88)
SHASHWAT CHAUHAN (80)
SUNNY DAHIYA (63)

TABLE OF CONTENTS

INTRODUCTION



NetGuard is an advanced network security project that integrates real-time intrusion detection withvirtual recognition technology. This project aims to enhance network security by identifying and responding to suspicious activities in real time while providing virtual overlays for visualizing andinterpreting network traffic.

Our goal is to develop a robust system that detects and mitigates security threats while offering an intuitive visual representation of network data. This can be invaluable for network administrators and cybersecurity experts. Net Guard aspires to safeguard networks and enhance threat analysis through the fusion of intrusion detection and virtual recognition.

PROBLEM STATEMENT

Learning how to detect the Face before giving access to it . Facial Recognition systems can be impacted by poor lighting or low image quality. The data may not match up with the person’s nodal points because of camera angles being obscured; this creates an error when matching faceprints cannot be verified in the database. . Matching of face images capture in near infrared spectrum (NIR) to face images of the visible spectrum (VIS) is a very challenging task.

GOALS

- To do face recognition in real time.
- Enhance the Speed i.e. frames/sec.
- Do recognition on high Camera resolution

LITERATURE REVIEW

TITLE	PROS	CONS	Future Works:	OBJECTIVE
Face Detection using Machine Learning	Human-Computer Interaction: Enhance user experiences through facial recognition.	Privacy Concerns: Face detection can raise privacy concerns, and there's a need to address ethical considerations.	Developing more natural and intuitive human-computer interaction.	TO DETECT THE FACE IN HIGHER FRAME

RESEARCH OBJECTIVES

1

To enhance the Frame/sec for Face Recognition System, such that Recognition is done in Real Time.

2

Face Recognition with Real Time Database

3

Presently, work on 30frames/sec Our motto is to achieve higher frames/sec or high-Resolution frames/sec.

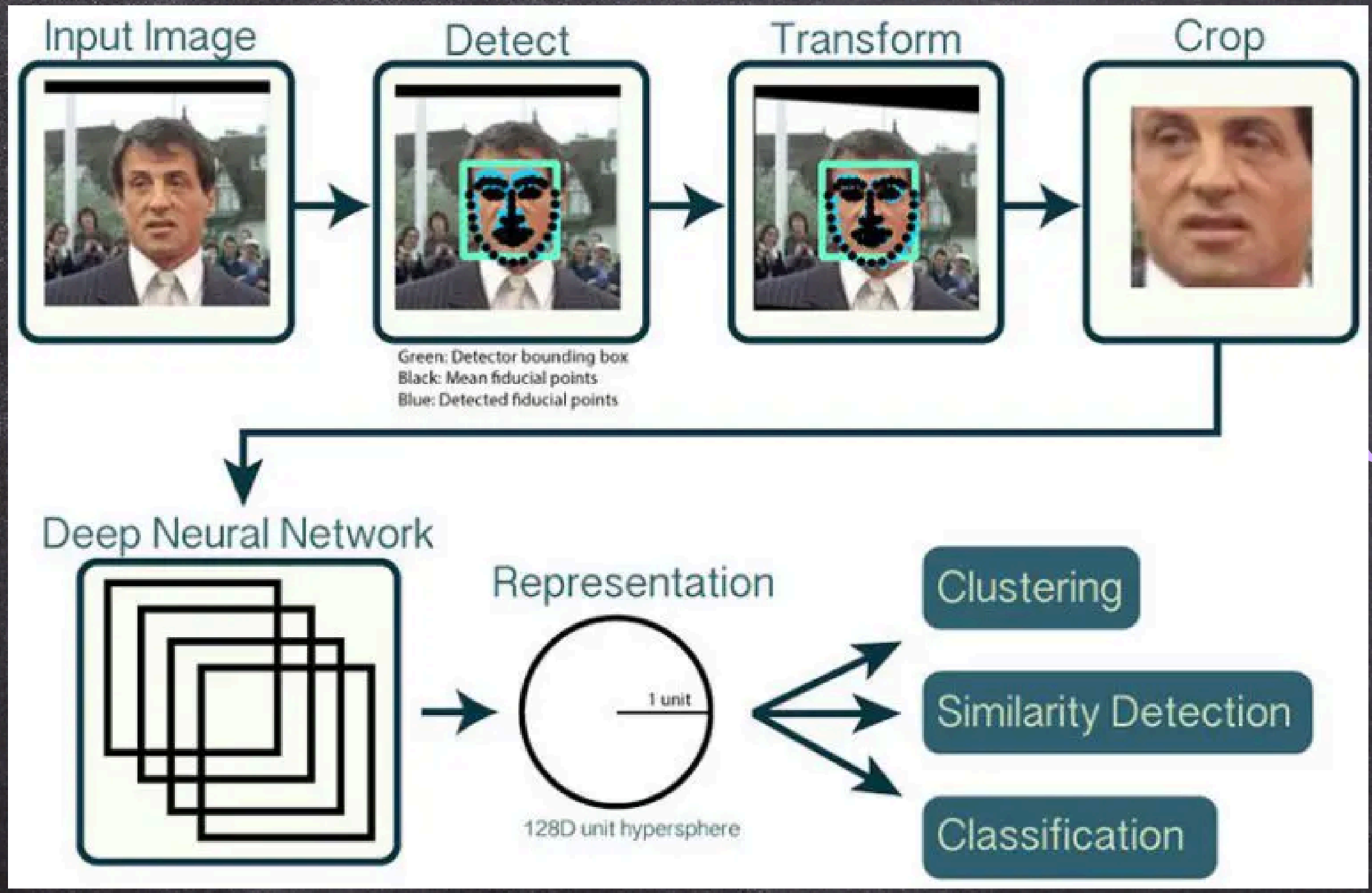
4

Ensure network security and privacy in the detection process

5

extracting those meaningful features from an image, putting them into a useful representation and performing some kind of classification on them.

MODEL ARCHITECTURE

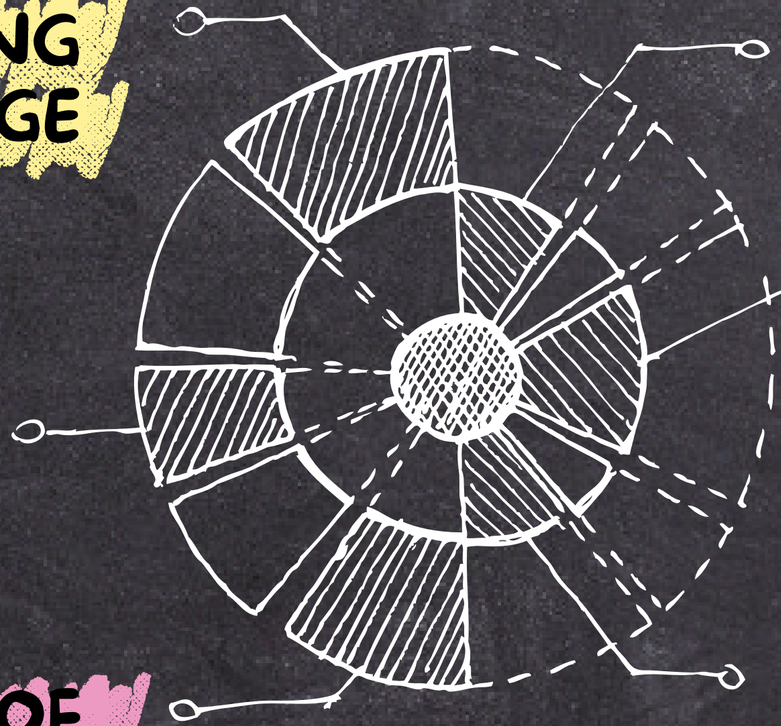

Input Image
Detect
Transform
Crop
Green: Detector bounding box
Black: Mean fiducial points
Blue: Detected fiducial points
Deep Neural Network
Representation
1 unit
128D unit hypersphere
Clustering
Similarity Detection
Classification

APPROACH USED

FACE ENCODING FIRST IMAGE

FACE ENCODIND SECOND IMAGE

COMPARISON OF IMAGES



REAL-TIME ON A WEBCAM

DATASET

ENCODE ALL FACE

SOURCE CODE

Face encoding first image

With the usual OpenCV procedure, we extract the image, in this case, **Messi1.webp,** and convert it into RGB color format. Then we do the “face encoding” with the functions of the Face recognition library.

```
4.  img = cv2.imread("Messi1.webp")
5.  rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
6.  img_encoding = face_recognition.face_encodings(rgb_img)[0]
```

Face encoding second image

Same procedure for the second image, we only change the name of the variables and obviously the path of the second image, in this case: **images/Messi.webp**.

```
8.  img2 = cv2.imread("images/Messi.webp")
9.  rgb_img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
10. img_encoding2 = face_recognition.face_encodings(rgb_img2)[0]
```

Comparison of images

With a single line, we make a simple face comparison and print the result. If the images are the same it will print True otherwise False.

```
12.  result = face_recognition.compare_faces([img_encoding], img_encoding2)
13.  print("Result: ", result)
```

Encode all face in the dataset

Now we have to encode all the images in our database, so that through the webcam video stream if it finds the match it shows the name otherwise it says “name not found”.

This is a function of the file I have prepared and it simply takes all the images contained in the **images/** folder and encodes them. In our case, there are 5 images.

```
4.  # Encode faces from a folder
5.  sfr = SimpleFacerec()
6.  sfr.load_encoding_images("images/")
```

```python

# Load Camera
cap = cv2.VideoCapture(0)

# Encode faces from a folder
sfr = SimpleFacerec()
sfr.load_encoding_images("images/")

while True:
    ret, frame = cap.read()

    # Detect Faces
    face_locations, face_names = sfr.detect_known_faces(frame)
    for face_loc, name in zip(face_locations, face_names):
        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

        cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

    cv2.imshow("Frame", frame)

    Key = cv2.waitKey(1)
    if Key == 27:
        break

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

CONTRIBUTION

IMPROVED PUBLIC SAFETY

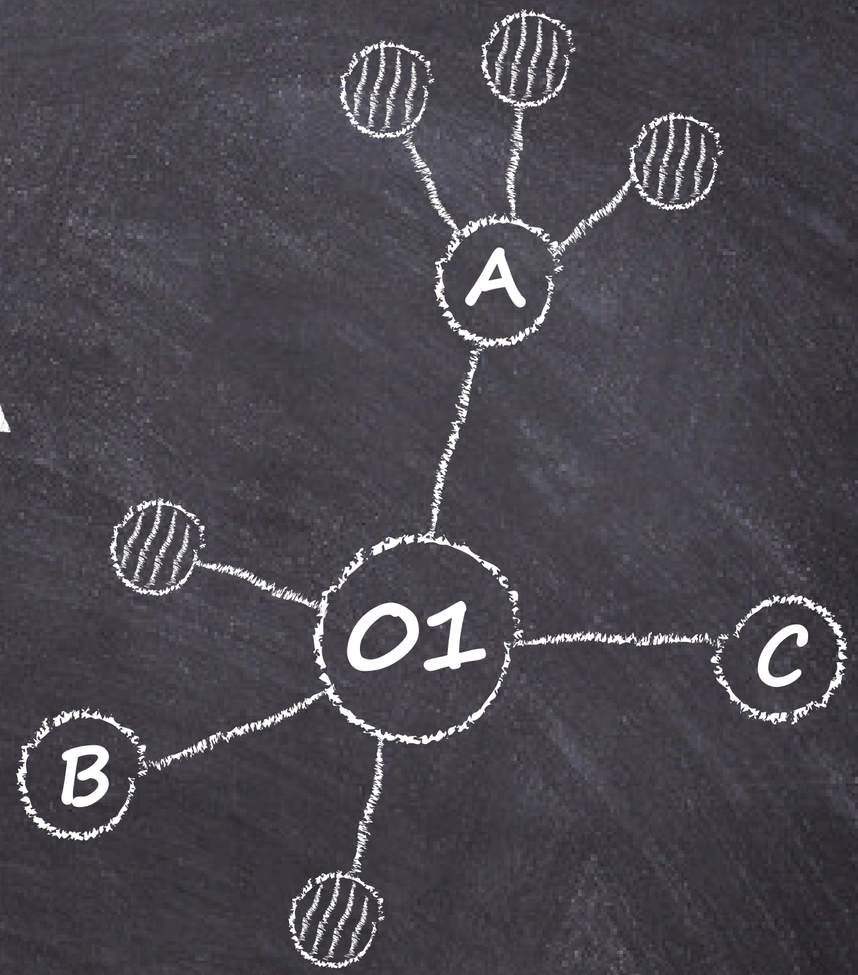
FASTER PROCESSING AND SEAMLESS INTEGRATION

AVIATION AND TRANSPORT

Facial recognition algorithms can help in diagnosing some diseases using specific features on the nose, cheeks and other part of the human face.

APPLICATION AREAS

1. Automobile Security
2. Access Control
3. Immigration
4. Education
5. Retail
6. Healthcare



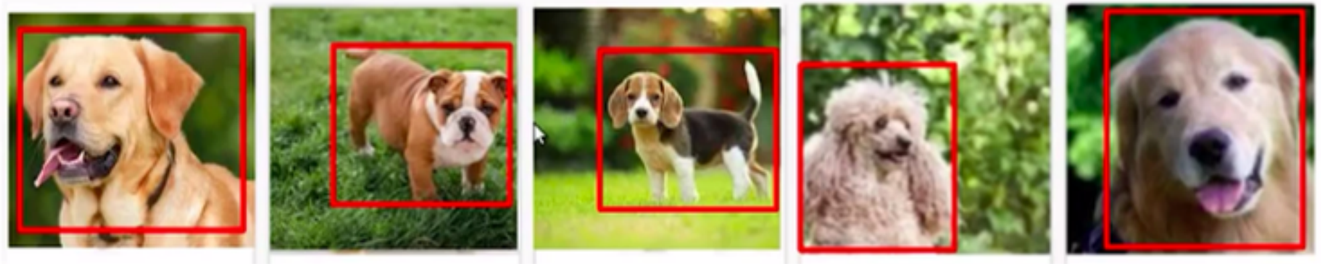
Object Detection




Introduction

- Object detection, within computer vision, involves identifying objects within images or videos. These algorithms commonly rely on machine learning or deep learning methods to generate valuable outcomes. Now let's simplify this statement a bit with the help of the below image.

- So instead of classifying, which type of dog is present in these images, we have to actually locate a dog in the image. That is, I have to find out where is the dog present in the image? Is it at the center or at the bottom left? And so on. Now the next question comes into the human mind, how can we do that? So let's start.Well, we can create a box around the dog that is present in the image and specify the x and y coordinates of this box.



FOR NOW, CONSIDER THAT THE LOCATION OF THE OBJECT IN THE IMAGE CAN BE REPRESENTED AS COORDINATES OF THESE BOXES. SO THIS BOX AROUND THE OBJECT IN THE IMAGE IS FORMALLY KNOWN AS A BOUNDING BOX. NOW, THIS BECOMES AN IMAGE LOCALIZATION PROBLEM WHERE WE ARE GIVEN A SET OF IMAGES AND WE HAVE TO IDENTIFY WHERE IS THE OBJECT PRESENT IN THE IMAGE.


THANK
YOU