**Face Recognition Using MTCNN and InceptionResnetV1**

## **Introduction**

Face recognition is a popular and widely used application of computer vision. It involves identifying and verifying an individual's identity by analyzing and comparing facial features. Real-time face recognition systems can be used for a variety of purposes, such as security, access control, and surveillance. In this report, we will discuss how to build a real-time face recognition system using PyTorch and OpenCV.

## **Approach**

We used a two-step approach for real-time face recognition.

* The first step involves detecting faces in the input video stream using the MTCNN face detection model.
* The second step involves computing the facial feature embeddings of the detected faces using the InceptionResnetV1 face recognition model and comparing them to the embeddings of known faces to recognize them.

To recognize faces, we used PyTorch and OpenCV. We used PyTorch to load and use the pre-trained face detection and face recognition models. We used OpenCV to capture and process video frames, draw bounding boxes around detected faces, and display the results.

## **Implementation**

We implemented the real-time face recognition system in Python using PyTorch and OpenCV. We first loaded the pre-trained face detection and face recognition models using PyTorch. We then opened a camera device using OpenCV and started capturing frames from the camera in a loop. For each frame, we detected faces using the MTCNN face detection model and cropped the faces from the frame. We then computed the facial feature embeddings of the cropped faces using the InceptionResnetV1 face recognition model. Finally, we compared the embeddings to the embeddings of known faces to recognize them.

To compare embeddings, we used the Euclidean distance metric to calculate the distance between the embeddings of two faces. If the distance was less than a threshold, we recognized the face as a known person; otherwise, we classified the face as unknown. We also drew bounding boxes around the detected faces and displayed the recognized name and distance from the stored embeddings.

**Deliverables**

**Face Detection Algorithm**: An implementation of the MTCNN face detection algorithm in Python that can detect human faces in digital images and video.

**Facial Feature Extraction Model**: An implementation of the InceptionResNetV1 model in Python that can extract the facial features from a cropped face image.

**Face Recognition Algorithm**: A Python-based face recognition algorithm that can compare the facial feature embeddings of two faces using the Euclidean distance metric.

**Results**

We tested our real-time face recognition system on a laptop with an AMD Ryzen 7 processor with 16GB of RAM. The system was able to detect and recognize faces in real-time with an average frame rate of 10 frames per second. The recognition accuracy was high, and the system was able to correctly recognize known faces and classify unknown faces. We also tested the system under different lighting conditions and camera angles and found that it was robust to these variations.

**Limitations**

The Limitation of our project:

* Uncertainty for CUDA devices

## **Conclusion**

In this Project, we build a real-time face recognition system using PyTorch and OpenCV. We used a two-step approach that involved detecting faces using the MTCNN face detection model and computing facial feature embeddings using the InceptionResnetV1 face recognition model. We then compared the embeddings to the embeddings of known faces to recognize them. We achieved high recognition accuracy and real-time performance, demonstrating the effectiveness of our approach. This system can be used for a variety of applications, including security, access control, and surveillance.