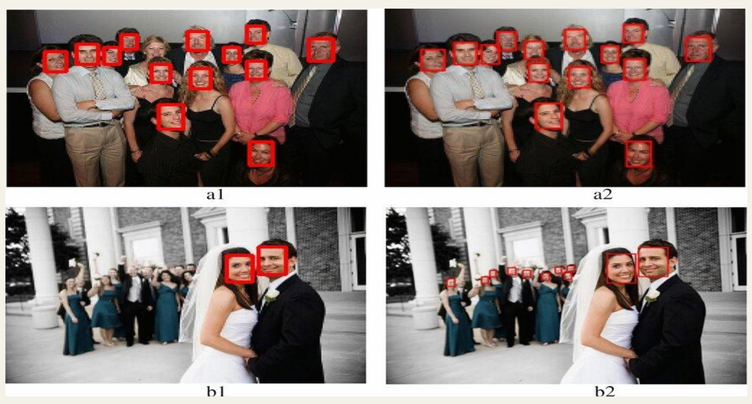
FACE COUNT USING MTCNN

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Introduction

Face detection and counting play a crucial role in numerous modern applications, such as surveillance systems, crowd monitoring, smart classrooms, and human-computer interaction. The ability to accurately detect and count the number of human faces in an image or video feed is vital for analyzing human presence and behavior in a given environment. Traditional face detection methods often struggle with variations in lighting, facial orientation, and occlusion

This project utilizes the **Multi-task Cascaded Convolutional Neural Network (MTCNN)** to perform face detection and counting. MTCNN is a deep learning-based framework that combines three neural networks in a cascade P-Net, R-Net, and O-Net to detect faces and facial landmarks with high precision. The system processes input images, identifies all visible faces, and counts them by drawing bounding boxes around each detected face.



Functional Needs

•**High Precision Face Detection :** MTCNN provides accurate and efficient detection of multiple faces in real time video feeds,even under varying lighting and occulsions.

•**Crowd Size Estimation:** The face count feature healps estimate the number of individuals in a given area, providing real-time information on crowd density and movement.

•**Real-Time Monitoring:** Integrating MTCNN with surveillance cameras allows for immediate tracking of face counts during live events or health monitoring scenarios.

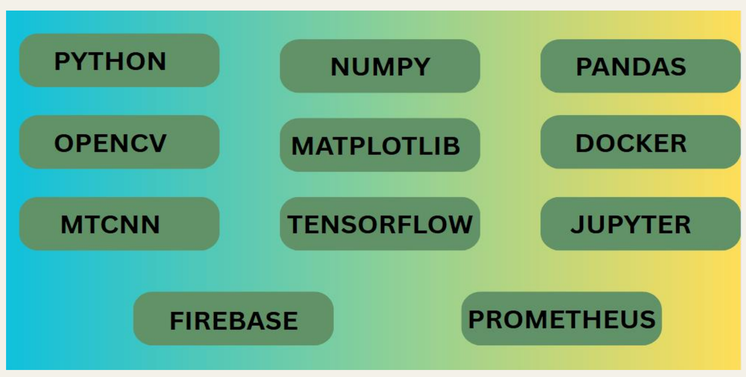
•**Face Recognition:** Can be combined with other facial recognition models to identify specific individuals for more detailed demographic or security analysis**.**

•**Data Collection and Reporting :** Real-time face count data is stored and processed for analysis, offering insights into crowd patterns, behaviour and flow.

•**Privacy Considerations:** The system can be set up to respect privacy by focusing only on face count without capturing detailed personal information unless required for security purposes.

Tech Stack

The tech stack for the Face Count using MTCNN project includes Python for programming, leveraging libraries like OpenCV for real-time video processing and MTCNN for accurate face detection. TensorFlow or PyTorch is used for deep learning tasks, and scikit-learn for additional machine learning needs. For real-time processing, Flask or FastAPI is used to build APIs, while Socket.IO ensures real-time communication. Docker and Tensorflow is used ot stores face count data and logs, and AWS or Google Cloud is used for cloud infrastructure. The system is containerized using Docker and orchestrated with Kubernetes.

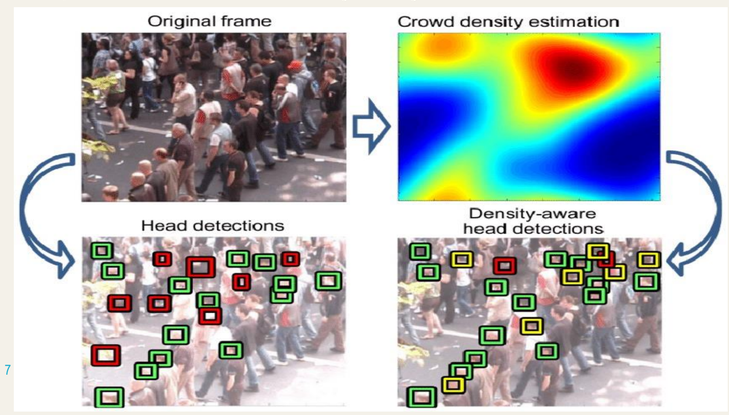


Objectives

1. **Face Detection System Development:** To develop a face detection system that accurately identifies and counts human faces in static images and real-time video streams using the MTCNN.
2. **Performance Evaluation:** To evaluate the performance of the MTCNN-based face detection model in terms of detection accuracy, speed, and robustness across diverse lighting conditions, angles, and occlusions.
3. **Real-Time Interface Design:** To create a user-friendly interface that displays the total face count along with bounding boxes around detected faces, enabling real-time monitoring and analytics applications.

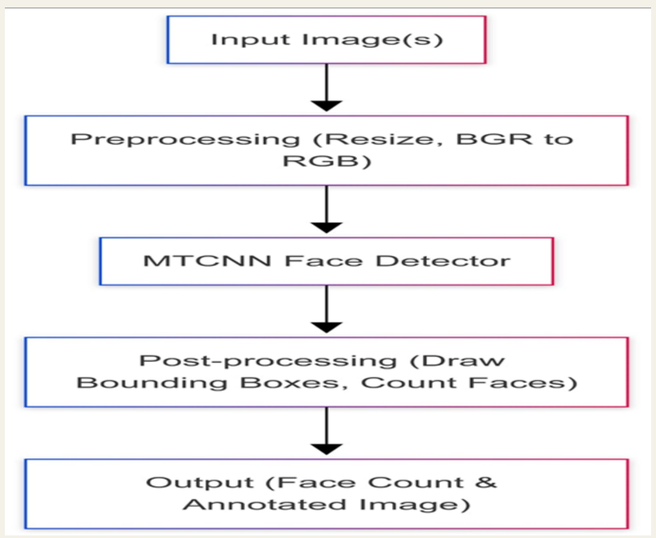
Crowd Density Estimation using MTCNN

Crowd density estimation involves detecting faces in real-time video feeds and calculating the number of people in specific regions of the frame. Using MTCNN for face detection, the video frame is divided into regions of interest (ROI), and the density is calculated by counting faces in each region and normalizing the results across the entire frame. A heatmap is generated to visualize the density distribution, with areas of higher density highlighted in red. Enhancements like facial landmark detection, body and distance estimation can improve accuracy, providing a more comprehensive crowd density analysis.



Workflow

1.Input Acquisition Action: Capture or load image/video frames from the webcam, camera feed, or stored media.
2. Preprocessing Action: Convert the image/video frames from BGR to RGB format for compatibility with MTCNN.
3. Face Detection Action: Use MTCNN (P-Net → R-Net → O-Net) to detect faces in the frames. P-Net: Proposes candidate windows. R-Net: Refines proposals and filters false positives. O-Net: Outputs final bounding boxes for faces.
4. Bounding Box Extraction Action: Extract bounding boxes from the MTCNN output and draw rectangles around the detected faces.
5. Face Counting Action: Count the number of bounding boxes (faces detected) in the frame.
6. Display Output Action: Show the processed image or video feed with faces highlighted by bounding boxes, and display the total face count on the screen.

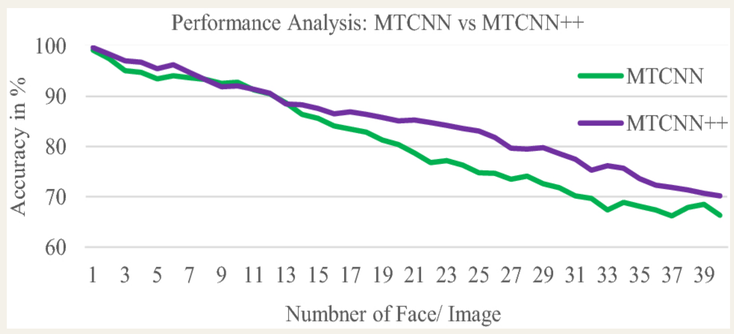


Advantages

High Accuracy	Real-Time Crowd Monitoring
Multi-Stage Processing	Robustness in Crowded Scenes
Easy to Implement	Real time Performance

Scalability and Cost

The MTCNN-based face counting system is highly scalable, as it can be deployed across various platforms ranging from edge devices like Raspberry Pi to powerful cloud servers. Its modular architecture allows seamless integration with existing surveillance systems, enabling expansion to multiple cameras and locations without significant rework. The use of pre-trained models reduces the need for large training datasets, minimizing development time and computational costs. Additionally, real-time processing can be achieved on mid-range GPUs or optimized CPUs, making it cost-effective for medium to large-scale deployments. For large-scale applications, integration with cloud platforms (e.g., AWS, Azure) ensures flexible storage and compute scalability. The cost of deployment varies based on hardware choice—basic setups can run on a local server, while enterprise solutions may require cloud infrastructure and GPU acceleration. Overall, the system offers a balance between performance, scalability, and affordability, making it suitable for both small businesses and enterprise-grade implementations.



Real-Time Applications

1. **Surveillance and Security Monitoring Application:** Monitoring crowded areas in real-time for security purposes. MTCNN helps in detecting faces in video feeds from surveillance cameras, enabling crowd management and alerting security personnel about potential risks like overcrowding or suspicious activities.
2. **Crowd Density Estimation Application:** Estimating crowd density in real-time during events or gatherings. MTCNN can be used to detect faces in video frames, and by counting the faces, the system can estimate crowd density, helping organizers take necessary actions if the density exceeds safety thresholds.
3. **Access Control and Attendance Tracking Application:** Using face counting to track the number of people entering a building or a room in real-time. In access control systems, MTCNN can be integrated with facial recognition technology for entry verification and attendance logging without requiring manual counting.
4. **Social Distancing Enforcement Application:** Enforcing social distancing in public spaces like malls, airports, or stations. Face counting can be used to ensure that a set number of people are within a defined area, alerting authorities if people are too close together, helping prevent overcrowding or violations of distancing norms.

Conclusion

In conclusion, MTCNN provides a robust solution for real-time face detection and counting, making it invaluable for applications such as surveillance, security monitoring. Its multi-stage architecture ensures high accuracy by progressively refining face proposals, even in challenging environments with multiple faces and varied lighting conditions. The ease of implementation using pre-trained models in deep learning frameworks like TensorFlow and PyTorch makes MTCNN a preferred choice for developers. Moreover, the system can scale from personal devices to large surveillance systems, making it versatile for various platforms. MTCNN also offers the flexibility to extend its functionality with additional modules such as face tracking and density estimation. This makes it not only efficient but also adaptable to different real-time needs. Ultimately, MTCNN proves to be a powerful tool for enhancing safety, improving user experience, and streamlining operations across numerous industries.

Team

PVSK Rohit (RA2311027010191)
Rahul Govardhan (RA2311027010201)
YEAR AND SECTION : 2nd YEAR(AN-2 Section)
COURSE FACULTY: Dr.M Anand