National College of Ireland

Detection of Unauthorised Person in a Restricted Place using Deep Learning Algorithms

MSc Research Project
Data Analytics

Mayur Said
Student ID: x21118515

School of Computing
National College of Ireland

Supervisor: Qurrat Ul Ain, PhD

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mayur Said
Student ID:	x21118515
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Qurrat Ul Ain, PhD
Submission Due Date:	15/12/2022
Project Title:	Detection of Unauthorised Person in a Restricted Place using Deep Learning Algorithms
Word Count:	5992
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	27th January 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of Unauthorised Person in a Restricted Place using Deep Learning Algorithms

Mayur Said
x21118515

Abstract

Security has always been a concern for everyone. Especially, in some highly volatile restricted places where only authorised individuals are allowed to enter. If some unauthorised individual breaches the security of such locations, he/she can cause catastrophic damage, potentially endangering people's lives as well. In this research paper, a solution is proposed using a face detection algorithm(MTCNN) and a face recognition algorithm(VGGFace2) to identify an unauthorised person in a restricted place using its video footage only. The proposed solution is implemented and evaluated by carrying out several experiments using video clips on YouTube. The results of these experiments showed that MTCNN along with a pre-trained VGGFace 2 model gives the best performance with precision, recall, and f1-score of 0.872, 0.975, and 0.921 respectively.

1 Introduction

Any organization needs an efficient and trustworthy security system to protect its precious assets, keep its people safe, and prevent or respond to any perceived threats. The majority of businesses have restricted places on their property, such as data centers, control rooms, military outposts, money storage facilities, etc. Such places are only accessible to a small number of authorized persons, and the presence of an unauthorized person in any such locations can result in unprecedented losses. Any firm must therefore have a strong security system in place to detect and control such illegal activities. Most organizations have a security system that makes use of a network of cameras to keep an eye on all actions that take place on the organization's property. This is known as a video surveillance system. The majority of video surveillance systems, however, largely rely on people to regularly watch the system in order to spot unlawful or suspicious behavior occurring on the property. An observation-based system is vulnerable to human mistakes, which could allow some illegal behaviors to go undetected.

The study suggests using deep learning algorithms to automatically detect the presence of an unauthorised individual in the restricted place by examining its camera footage as a solution to this issue. These algorithms might be able to eliminate the possibility of missing unwanted activity, increasing the security of an organization. So, the fundamental question of this research is "How accurately the deep learning algorithm can detect the presence of an unauthorized individual from video footage of restricted place?" To answer this research question, the study aims to achieve two main objectives.

1. Implement face detection using deep learning algorithms to detect faces from a video footage

2. Implement face verification using deep learning algorithms to identify unauthorised individuals from detected faces

The rest of the paper is as follows: In section 2, a literature review is done to comprehend other proposed security systems by researchers and the current state-of-the-art techniques for face detection and face recognition. Section 3 gives a detailed explanation of the methodology used in this research. Section 4 explains the underlying architecture of the models implemented in this research. Section 5 gives the details of project implementation. Section 6 evaluates the proposed solution by performing various experiments. The last section concludes the paper and gives the scope of future work.

2 Related Work

Since 1943, the idea of an artificial neural network (ANN)—the foundation of a deep learning architecture—has existed (Mcculloch and Pitts; 1943). However, it wasn't until after the enormous success of Hinton's proposed architecture called 'A Deep Believe Network' (DBN) for faster learning in 2006 that it began to gain popularity among researchers (Hinton et al.; 2006). Since then, deep learning algorithms have undergone rapid progress for a range of applications, including image classification. Therefore, a comprehensive study of the literature is conducted in this section to gain a better grasp of the most cutting-edge solutions currently available and the measures that should be followed to fulfil the goals of this research.

2.1 Literature Review on Security Systems Proposed by Other Researchers

Researchers are constantly looking for improved ways to construct a robust and trustworthy security system because the demand for one has grown significantly in today's modern society. To prevent accidents when crossing railroad tracks, Singh et al. built a security system integrating by an Arduino UNO, and ultrasonic sensors with the video feed from the camera (Singh et al.; 2020a). When a train is due to arrive, the Arduino and ultrasonic sensors are used to automatically close the railway crossing tracks. The video feed is also subjected to computer vision analysis in order to determine whether there are any persons or vehicles on the track while the gates are closed. When something is detected on the track, the systems promptly warn the train driver and open the gate, allowing them to move off the track(Singh et al.; 2020b). The detection of suspicious activity from video footage is another popular research area. To better understand automatic suspicious detection systems, Soomro et al. compare 42 publications available at Springer, IEEE, and Elsevier online repositories (Soomro et al.; 2022). Their research presented a basic framework for identifying suspicious activities. A group of researchers suggested a context-based method to identify suspicious activity (Wiliem et al.; 2012). The study is founded on the idea that without context, human behavior cannot be categorized as suspicious. Human behavior can appear suspicious in one situation while acting entirely normal in another. According to the study's findings, detecting suspicious behavior can be done more precisely by applying contextually relevant knowledge.

A lot of emphasis of researchers is on integrating machine learning algorithms with IoT, especially for home security systems, algorithms. Istiqomah et al. suggested utilizing an automatic barrier gate to secure homes (Istiqomah et al.; 2021). They created a system that can scan QR codes produced by Android apps and compare them to database data. The device will instantly unlock the barrier gate after finding a perfect match. According to Rathour et al., robust security requires specialized hardware with better computing (Rathour et al.; 2022). KlugOculus, new, specially designed hardware, was created as a result. Using a motion sensor and facial detection algorithm, Dr. Joy Iong Zong Chen creates a smart security system to detect the presence of unauthorized personnel in volatile settings such as hospitals, banks, military areas, etc (Chen; 2020). Motion is detected in the restricted region by the system using the motion detector. As soon as movement is detected, a camera is activated to take a picture of the person, whose face is then compared to those in the database to confirm their identity. However, one drawback of this proposed solution is the possibility that the camera will capturing a blurry image. If images taken by the camera are blurry, the capability of a system would be significantly reduced.

2.2 Literature Review on Face Detection and Face Recognition Algorithms

Face detection and face recognition were the key components of the majority of the proposed designs and solutions in the quest of attaining ironclad security. To solve the face detection problem, Vidit Jain and Erik Learned-Miller developed a benchmark face dataset (FDDB) with more faces and two evaluation techniques for face detection algorithms(Jain and Learned-Miller; 2010). In 2016, Shuo Yang et al. demonstrated that there is a discrepancy between face detection performance as it now stands and the real-world requirements(Yang et al.; 2016). Therefore, they enabled the Wider Face dataset, which is 10 times bigger than the present dataset, in order to train the face identification algorithm and improve its performance. To train the face identification model for better performance, they made available a dataset named Wider Face dataset that was ten times bigger than the current dataset. Since then, these two datasets have been primarily used by researchers to train and evaluate their proposed solutions. In the same year, a face detection technique known as the "Multi-Task Cascaded Convolutional Neural Network" (MTCNN) was proposed by (Zhang et al.; 2016). Like most researchers, they used FDDB (Jain and Learned-Miller; 2010) and WIDER FACE (Yang et al.; 2016) datasets to train and test the proposed technique. The findings of the study demonstrated that the suggested technique regularly outperformed other cutting-edge algorithms.

After face detection, now comes the problem of face recognition. In 1991, research expressed the features of the face as eigenvalues and used them for face recognition (Turk and Pentland; 1991)to achieve real-time performance. Since then, as time went on, algorithms with greater performance were increasingly put forth. In their survey from 2018, Mei Wang and Weihong Dong gave a summary of the facial recognition research done over the previous 30 years (Wang and Deng; 2021). According to the report, a surge of deep learning algorithms for face recognition emerged in 2014 and 2015, outperforming human performance, following AlexNet's breakthrough in 2012 for simpler image classification tasks (Krizhevsky et al.; 2012). The four most influential facial recognition systems were also named: VGGFace(Parkhi et al.; 2015), , DeepID Sun et al. (2014), FaceNet (Schroff et al.; 2015), and DeepFace (Taigman et al.; 2014)

2.3 Identified Gaps

After performing a thorough literature review, it was found that hardware like motion sensors, scanners, ultrasonic sensors, etc. are needed for the majority of the suggested solutions to construct a reliable security system. However, installing such hardware is not always an option due to a number of factors, including the cost of installation, ongoing maintenance, installation time, etc. On the other hand, CCTV cameras are a piece of equipment that almost all businesses have placed on their property. This research suggests solely using video footage from these CCTV cameras to monitor each person continuously while using deep learning algorithms to identify any unauthorized individuals who may be present.

3 Methodology

This section describes the methodology of the research in detail.

3.1 Dataset Description

3.1.1 Video Clips

Due to the lack of a CCTV camera to capture the video footage of a particular restricted zone, this research project has used two short video clips from one of the famous TV shows 'Big Bang Theory' available on YouTube. As per the YouTube fair use guidelines, one can use videos available on YouTube if the use of those videos is transformative in nature and for non-profit educational research[1]. In line with these guidelines, two video clips are selected from YouTube for this research project. The first clip is 140 seconds long with frames per second (FPS) of 23.97[2]. In the whole video clip, there are five different individuals. The second video clip lasts for 58 seconds long with of FPS 29.97[3]. In total, the second video clip has seven different individuals, the first five individuals are the same from the first video clip with the other two new individuals. Frames in Figure 1 and Figure 2 shows all the individuals from the first and second video clip respectively.

Figure 1: Frame of First Video

[1]YouTube Fair Use Guidelines https://support.google.com/youtube/answer/9783148?hl=en

[2]First Video Clip: https://www.youtube.com/watch?v=w9QEoOhNfBE&list=PLEWx3HDIGPVOZNx8QhEox1M63hclerGuo

[3]Second Video Clip: https://www.youtube.com/watch?v=iWEAFdYFKr0&list=PLbliSsgZA_dEjoOEk4WGbzwslbT8BX8yu

Figure 2: Frame of Second Video

3.1.2 Research Assumptions

As already described in the introduction, the main aim of this research is to identify unauthorised individuals from a video clip of a restricted zone. To achieve the same, this research needs to assume some individuals as authorised and some individuals as unauthorised from the two selected video clips. As a result, two assumptions are made, all five individuals (shown in Figure 1) from the first video clip are assumed as authorised, and the rest two individuals from the second video clip (shown in Figure 2) are assumed as unauthorised. The front-face views of all authorised and unauthorised individuals are shown in Figure 3 and Figure 4 respectively

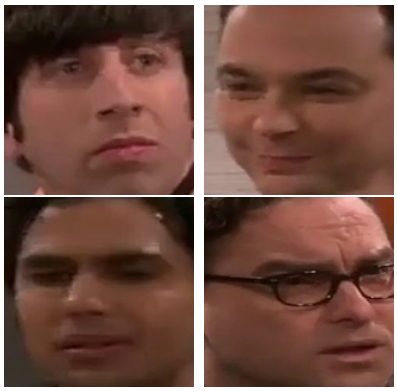

Figure 3: Authorised Faces

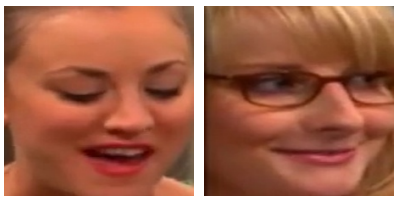

Figure 4: Unauthorised Faces

3.1.3 Face Dataset of Authorised Individuals: Authorised Face Dataset

To identify unauthorised individuals from a video clip of authorised and unauthorised individuals, a face dataset of all authorised individuals needs to be curated. Furthermore,

the curated dataset must have images of each face from different angles to improve the performance of the proposed solution. Based on the assumption made in subsection 3.1.2, the first video clip containing all five authorised individuals is used to curate the same. A few frames are extracted from the first video clip and the face of each individual is manually extracted from those frames. In the end, a total of 45 high-quality face images of each authorised individual were extracted from the first video clip such that there are at least 7 to 8 images of every possible angle of each person.

3.1.4 Test Video Clip

After curating the authorised face dataset, now this research needs a test video clip, such that it has both authorised and unauthorised individuals. As per the assumptions made in subsection 3.1.2, the second video clip has both authorised and unauthorised individuals. Therefore, this research has used it as a test video clip to implement and evaluate the proposed solution.

3.1.5 Dataset to evaluate custom CNN model: Face and No Face Dataset

Later in this research, a Convolutional Neural Network (CNN) model was developed and trained from scratch to classify images predicted by the face detection algorithm (MTCNN) as 'face' or 'no-face' (discussed more in subsection 6.1.2). Therefore, to do the same, a dataset with two classes: face and no-faces was created. For the class face, all the images in authorised face dataset are used. For no-face class, random backgrounds are manually extracted from the frames of the first video clip. A few examples of such images are shown in Figure 5). The size of this dataset was then later increased by using the data augmentation techniques discussed in subsection 3.3. After applying data augmentation, there were a total of 495 images of the class 'face' and 520 images of the class 'no-face'.

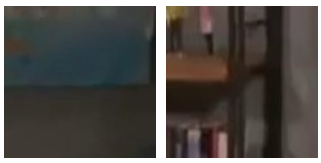

Figure 5: Images of Class 'no-face'

3.2 Data Pre-processing

Before feeding the data to deep learning models, the data needs to be pre-processed and transformed as per the model's requirement as well as to improve its performance. This research has implemented a ResNet-50 model to predict the face embedding of a given input image to the model. As per its architecture (discussed more in subsection 4.3), ResNet-50 requires the size of all input images to be 224x224x3. After resizing the images of detected faces, the pixel values of each image are also transformed by subtracting their channel mean from them.

3.3 Data Augmentation

One way to improve the performance of the deep learning models is by feeding them more data. However, the authorised face dataset that is curated for this research only has 45

images. Data augmentation is a strategy for expanding a dataset by gently changing the current data. Furthermore, it not only helps in increasing the size of the dataset but also aids in the reduction of overfitting of the model. Therefore, this research has incorporated the same to increase the size of its authorised face dataset. Various types of data augmentation techniques are available. This research has used random horizontal and vertical flipping, random rotation, and random contrast as simple but effective data augmentation techniques to increase the size of its dataset and improve the implemented model's performance.

3.4 Image Enhancement

This research has also looked at the effects of image enhancement techniques on the performance of the implemented models for detecting unauthorised faces. Particularly, two techniques are used to enhance the image quality: image denoising and image sharpening. Image denoising is the process of removing noise from a noisy image to improve its quality. To remove noise from detected faces, this research has used a technique called 'Non-Local Means Denoising' which has consistently been shown to remove noise from imagesBuades et al. (2011). One downside of removing noise from images is that it loses some of their details such as edges and texture. Therefore, image sharpening is used to sharpen the edges and improve the texture of a denoised image. To sharpen a denoised image, a 3x3 kernel is used. A Kernel describes how to adjust the value of any given pixel by mixing it with varying amounts of surrounding pixels. Using the convolution method, this 3x3 kernel is applied to the input image to get the final enhanced image. Figure 6 shows the effects of image enhancement.

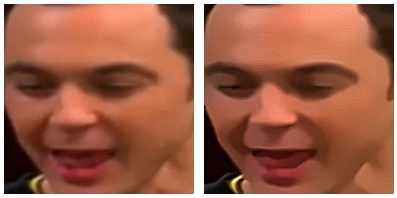

Figure 6: Effect of Image Enhancement Technique on Detected Face

3.5 Proposed Solution

The overview of the proposed solution is shown in Figure 7.

3.5.1 Step 1: Capture Frames from video clips

In step 1, every 10th frame is captured from the test video clip. Each captured frame is now then fed to the face detection block.

3.5.2 Step 2: Detect Faces from Captured Frames Using Face Detection Block

In step 2, faces are detected from the captured frame using a face detection block. Face detection block is the combination of the MTCNN model and the custom CNN model for filtering no-face images from the output of MTCNN. The frames captured from the

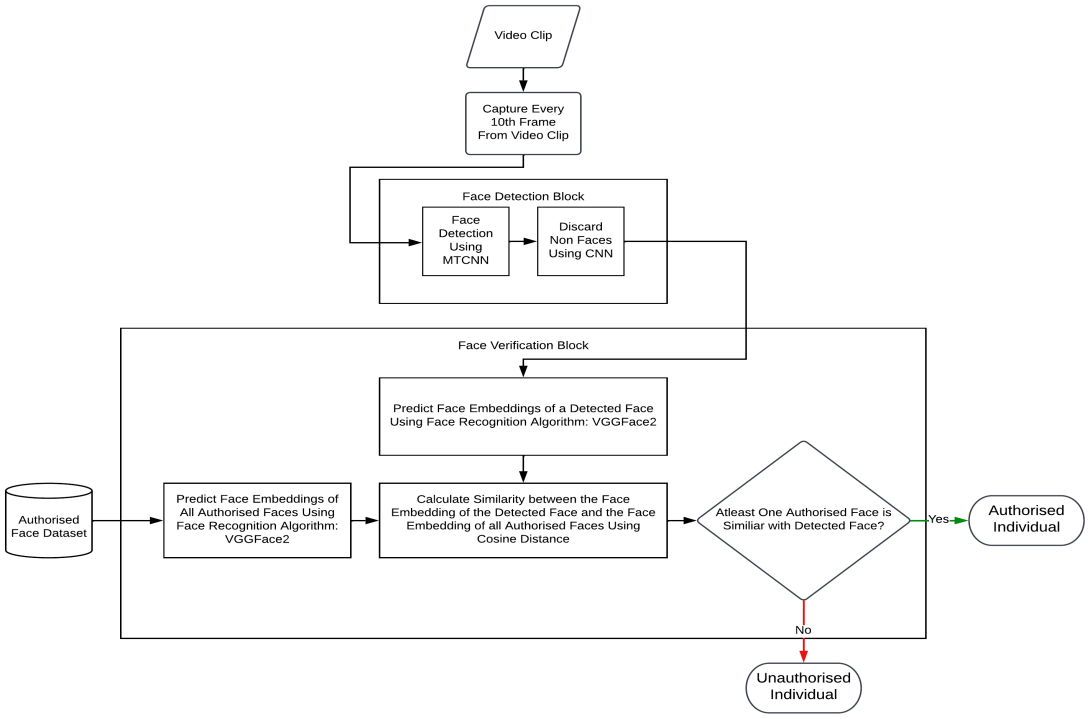

Figure 7: Overview of Proposed Solution

test video clip are fed to a face detection algorithm. As the literature review indicated, MTCNN is one of the best-performing models for face detection as it produced state-of-the-art results on a number of datasetsZhang et al. (2016). Therefore, this research has used the MTCNN algorithm to detect faces from the captured frames. However, not all deep learning algorithms are perfect and MTCNN is no exception as well. After implementing the MTCNN to detect faces, it was observed that not all faces detected by MTCNN were truly face images. Some of the predictions given by MTCNN were some random objects from the captured frame. As a result, to make face detection more robust and reduced false positive rates, a custom CNN model was implemented from scratch for detecting any false positives from the MTCNN. The design of the implemented CNN model is discussed in subsection 4.3. Therefore, in this step, there are two the sub-steps which are as follows:

1. Detect faces from the captured frames using MTCNN
2. Filter no-face images from face images using a custom CNN model.

3.5.3 Step 3: Predicting Detected Face as 'Authorised Face' or 'Unauthorised Face' Using Face Verification Block.

This step predicts the faces detected by the face detection block as 'Authorised Face' or 'Unauthorised Face'. To predict a detected face as 'Authorised Face' or 'Unauthorised Face', each detected face is matched with all the authorised faces. If there is no match for a particular detected face with all the authorised faces, then that particular detected face is predicted as 'Unauthorised Face' or else it is predicted as 'Authorised Face'. To

match faces, face embeddings are predicted for each detected face and each authorised face. Face embeddings are nothing, but a numerical representation of a face predicted by a deep learning model. Now the similarity of the face embedding of the detected face is calculated with all authorised faces. If the face embedding of the detected face is similar to at least one face embedding of an authorised individuals, then that particular face is predicted as an 'Authorised Face'. Otherwise, it is predicted as an 'Unauthorised Face'. There are multiple ways to check the similarity between two face embeddings. In this research, cosine distance is calculated between two face embeddings. If the cosine distance between two face embeddings is less than 0.5, then those two face embeddings are considered similar. Step 3 is further divided into three sub-steps.

1. Predict Face embeddings of all faces from the Authorised Face Dataset using a face recognition algorithm.

2. Predict Face embedding of a detected face using the same face recognition algorithm of Sub-Step 1.

3. Calculate the similarity between the face embeddings of the detected face and all authorised faces using cosine distance. If at least one authorised face is similar to the detected face, predict the detected face as 'Authorised Face'. Otherwise, predict it as 'Unauthorised Face'.

Prediction of Face Embeddings from a Face Using a Face Recognition Algorithm(VGGFace2): The name 'VGGFace2' usually refers to a model trained on the VGGFace2 dataset for face recognition. In this research, a pre-trained ResNet-50 on VGGFace2 dataset is used to predict face embeddings of the detected face. ResNet-50 takes the face image as the input and predicts 2048 long face embeddings of the same. Using this pre-trained ResNet-50, three experiments were done. In the first experiment, the fifth stage of the ResNet-50 was unfreezon and fined-tuned on the authorised face dataset by adding a dense layer of 5 neurons(As there are five unique faces in the Authorised Face Dataset) and then it was used to predict the face embeddings. In the second experiment, a ResNet-50 model pre-trained on the VGGFace2 dataset without any fine-tuning is implemented to predict the face embeddings. In the last experiment, image enhancement techniques are used to improve the image quality of the detected face, and then its face embeddings were predicted. These experiments are discussed more in details in subsection 6.2.

Calculating Similarity between Face Embeddings(Cosine Distance): This research has used cosine distance to identify whether the two face embeddings belong to the same face or two different faces. Before discussing cosine distance, first, let us look at cosine similarity. Cosine similarity is nothing but the cosine of the angle between two vectors. If two vectors are similar or close to each other, the angle between the two vectors is relatively less. Therefore, the cosine of the angle between the vectors is low. To illustrate it more, look at the vectors shown in Figure 8 and Figure 9. The angle between the vectors in Figure 8 and Figure 9 is 60 degrees and 180 degrees respectively. Therefore, the cosine similarity (cosine of the angle) of vectors in Figure 8 and Figure 9 is 0.5 and -1. The range of cosine similarity is from -1 to 1. If two vectors are completely opposite to each other, as in Figure 9, the cosine similarity is -1. On the other hand, if two vectors overlap with each other, the cosine similarity is 1. After understanding the concept of cosine similarity, now let's discuss cosine distance. Cosine distance is inversely proportional to cosine similarity. In other words, higher the cosine similarity, lower the

cosine distance, and vice versa. Now if the two face embeddings belong to the same face, they will be similar to each other and the angle between them will be relatively low. Therefore, the cosine similarity of those face embeddings will be higher and thus, the cosine distance will be lower. This research has considered two face embeddings similar to each other if the cosine distance between two face embeddings is less than 0.5. Otherwise, they are considered different.

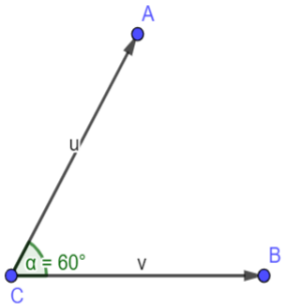

Figure 8: Vectors of Angle 60 Degree

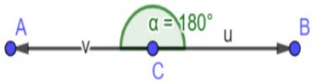

Figure 9: Vectors of Angle 180 Degree

4 Design Specification

This section briefs about the architecture of the models used in this research.

4.1 Multi-task Cascaded Convolutional Network (MTCNN)

Multi-task Cascaded Convolutional Network (MTCNN), a state-of-the-art model for face detection, consists of a cascaded structure with three stages of convolutional networks: Proposal Network (P-Net), Refine Network (R-Net), and Output Network(O-Net) for detecting faces and other facial characteristics such as the eyes and mouth. The structure of all three stages is shown in Figure 9. Figure 9 is from the original MTCNN paperZhang et al. (2016). Before feeding the image to a three-stage cascade structure, first, the input

image is resized to different scales to create an image pyramid, which serves as the input to the three-stage cascaded structure. Now this image pyramid is given as input to the first stage(P-Net) to predict facial regions of all faces, the second stage identifies whether the output of the first stage is face or not and outputs the bounding boxes for all faces. The third stage(O-Net) gives the facial characters of the detected face.

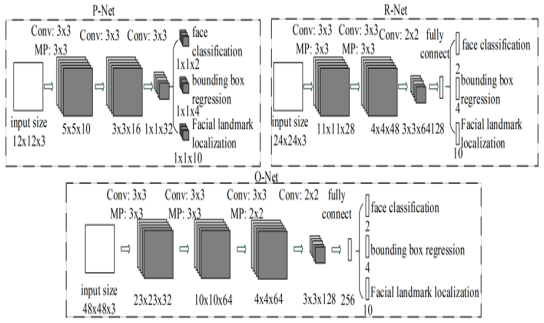

Figure 10: Architecture of MTCNN Algorithm

4.2 Custom Convolution Neural Network (CNN) Model in Face Detection Block

Convolution Neural Network is a type of deep learning algorithm mainly used for image classification. The architecture of the custom CNN model built in this research for classifying face images from non-face images is shown in Table 1. In total, the model has six convolution layers with a kernel size of 3x3 and a 'relu' activation function. All the convolution layers are followed by a 2x2 max pooling layer. After the last max pooling layer, a global average pooling layer is added followed by a dense layer of 2 neurons with a 'softmax' activation function.

4.3 Residual Network-50(ResNet-50

Residual Network(ResNet) is a type of CNN model that overcame the problem of vanishing gradient. During backpropagation, a problem of vanishing gradient is developed. When the neural network training algorithm attempts to find weights that minimize the loss function, if there are too many layers, the gradient becomes extremely tiny until it disappears, and optimization cannot proceed. The unique architecture of ResNet solves this problem. Thus, making it possible to build models with thousands of convolutional layers. The architecture of ResNet-50 is shown in Table 2.

Layer (type)	Output Shape	Param #
sequential_1 (Sequential)	(4, 224, 224, 3)	0
sequential_2 (Sequential)	(4, 224, 224, 3)	0
conv2d (Conv2D)	(4, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(4, 111, 111, 32)	0
conv2d_1 (Conv2D)	(4, 109, 109, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(4, 54, 54, 64)	0
conv2d_2 (Conv2D)	(4, 52, 52, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(4, 26, 26, 64)	0
conv2d_3 (Conv2D)	(4, 24, 24, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(4, 12, 12, 64)	0
conv2d_4 (Conv2D)	(4, 10, 10, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(4, 5, 5, 64)	0
conv2d_5 (Conv2D)	(4, 3, 3, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(4, 1, 1, 64)	0
global_average_pooling2d (GlobalAveragePooling2D)	(4, 64)	0
dense (Dense)	(4, 2)	130

Total params: 167,234
Trainable params: 167,234
Non-trainable params: 0

Table 1: Architecture of Custom CNN Model

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 2: Architecture of ResNet

5 Implementation

This section gives details about the implementation of the research methodology. This research has used python programming language to implement all the techniques discussed in the methodology.

5.1 Implementation of Data Augmentation on Authorised Face Dataset

The data augmentation techniques discussed in subsection 3.3, are implemented on the Authorised Face dataset to increase its size. To implement those data augmentation tech-

niques, the TensorFlow library in python is used. After applying the data augmentation technique to the Authorised Face dataset, its size increased from 45 images to 450 images.

5.2 Capturing Frames from Test Video Clip

To detect faces from the test video clip, every 10th frame is captured. To capture the frames from the test video clip, a python script is written using a python library called 'OpenCV'. After capturing the frames from the test video clip, these frames are now fed to the face detection block for detecting faces.

5.3 Implementation of Face Detection Block

In the face detection block, two models are implemented: MTCNN and a custom CNN model. First, to implement MTCNN, a library in python called 'mtcnn' by 'Ivan de Paz Centeno'[4]is used. This MTCNN model returns a JSON object for each detected face. Each JSON object contains a bounding box, confidence, and key points. The bounding box gives the exact location of the detected face on the frame, the confidence is the probability of the detected face, and the key points give the probable location of the left eye, right eye, nose, mouth left, and mouth right on the frame. Using the bounding box, all the detected faces with a probability of more than 85% are extracted from each frame. Once the faces are detected, these faces are passed to a custom CNN model developed from scratch using the TensorFlow library. This CNN model discards any detected face by the implemented MTCNN model which was not a face and in reality was some other random object from the frame.

5.4 Implementation of Data Pre-processing

Before feeding the detected image to the face verification block, the detected image needs to be pre-processed as per the input parameters of the ResNet-50 model. Therefore, all the detected images are pre-processed using two python libraries 'OpenCV' and 'NumPy'.

5.5 Implementation of Face Verification Block

In Face Verification block, there are three tasks, predicting embedding of authorised faces, predicting embedding of detected face, and calculating cosine distance for checking the similarity between two face embedding. A python library called 'keras_vggface' by Refik Can Malli [5] is used along with TensorFlow library to implement the ResNet-50 model pre-trained on VGGFace2 dataset(discussed more in subsection 6.2).

6 Evaluation

This section evaluates all the implemented models and discusses different experiments carried out to achieve the objectives of the research. To evaluate models, different experiments are carried out and the results of each experiments are evaluated using metrics such as accuracy, precision, recall and f1-score.

[4]MTCNN Library: `https://github.com/ipazc/mtcnn`

[5]keras_vggface library `https://github.com/rcmalli/keras-vggface`

6.1 Evaluation of Face Detection Block

The performance of the face detection block is evaluated by comparing its results with manual face detection. In other words, first, all the faces are detected using the face detection block in each frame and then all the faces in each frame are counted manually. Once done then the accuracy is calculated by dividing the total faces detected accurately by the face detection block with the total faces detected manually.

6.1.1 Experiment 1: Detecting Faces Only Using MTCNN

In the first experiment, only MTCNN was used to detect the faces from the test video clip. At the end of the first experiment, MTCNN detected 368 images as faces from all the frames captured out of which 352 images were actually face images and rest 16 were images of some random objects. As per the manual face detection, there were 378 faces in all the frames. Therefore, MTCNN was able to achieve an accuracy of 93%. Even though this accuracy is not bad, it was observed that some of the images detected by MTCNN are not faces and were some random objects from the frame. Therefore, to improve this, a new experiment was carried out that combined face detection using the MTCNN model and a custom CNN model.

6.1.2 Experiment 2: Detecting Faces Using MTCNN and Custom CNN Model

In the second experiment, outputs from the MTCNN model are fed to a custom CNN model. The architecture of this custom CNN model is described in subsection 4.2. The main job of this custom CNN model is to identify whether a given image contains a face or not. Therefore, it was trained using a dataset(described in subsection 3.1.5) that had two classes: faces and no-face. The main task of this experiment is to train a custom CNN model on a dataset with classes: face and no face and evaluate its performance in identifying whether a given image contains a face. For training and evaluation, the dataset is divided into three parts: Training, Validation, and Testing. The training contains 60% of the whole dataset, validation contains 20% of the whole dataset, and testing contains the rest 20% of the dataset. The model is optimised using Adam optimiser and evaluated using accuracy, precision, and recall metrics. Figure 11 shows training accuracy, validation accuracy, training loss, and validation loss for each epoch. After training for 16 epochs, as the model did not show any major improvements in both training and validation accuracy, the training was stopped. At the end of 16 epochs, the training accuracy and validation accuracy of the custom CNN model was 99.5% and 100% respectively. On the testing dataset, accuracy, precision, and recall were 100%, 1, and 1 respectively. From the results, it's clear that the implemented model can identify a given image as an image with an image face or no face with 100% accuracy.

Now after implementing the custom CNN model, all the faces detected by the MTCNN are passed to the custom CNN model. Thus, the whole face detection block is a combination of these two models. Now again, faces are detected using the whole face detection block. This time, it detected 330 images as faces with 324 images truly containing the faces. Therefore, its accuracy is 87%.

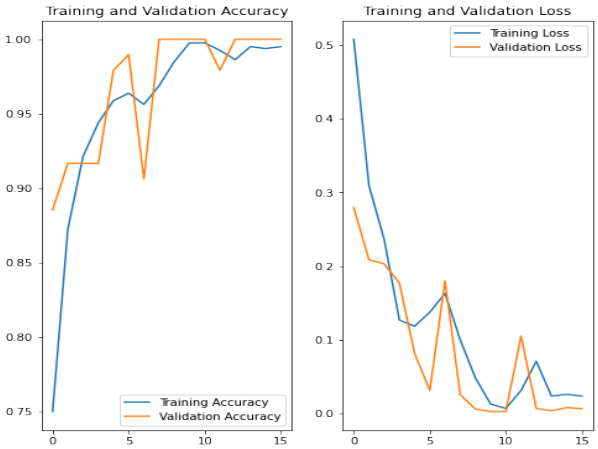

Figure 11: Performance of Custom CNN Model

6.1.3 Discussion of Results

Even though the accuracy of the second experiment is less when compared with the first experiment, the false positive rate of the first experiment is more than the second experiment. This research decided to have a low false positive rate in its proposed solution and thus continued the research with the results of the second experiment.

6.2 Evaluation of Face Verification Block

To Evaluate the face verification block, all the faces detected by the face detection block are manually annotated as an 'Authorised Face' or 'Unauthorised Face' to get the true labels. This is done as per the assumption made in subsection 3.1.2. The face detection block detected a total of 330 faces. After manually annotating each detected face, out of 330 detected faces 211 were 'Authorised Face' and 119 were 'Unauthorised Face'. Now after obtaining the true labels of each detected face, it's time to implement the face verification block and evaluate its performance for predicting each detected face as an 'Authorised Face' or 'Unauthorised Face'.

6.2.1 Experiment 1: Predicting Face Embeddings with a ResNet-50 Model Pre-trained on the VGGFace2 Dataset and Fine Tuned on the Authorised Face Dataset

In this experiment, a pre-trained ResNet-50 model is fine-tuned on the authorised face dataset. To fine-tune, the fifth stage of the ResNet-50 model without the top layer is unfrozen. The output layer of this model is then attached to a dense layer of five neurons as there are five unique faces in the authorised face dataset. For fining tuning, the Authorised Face dataset is divided into training, validation, and testing datasets such that the training dataset is 60%, the validation dataset is 20% and the testing dataset is the rest 20% of the whole dataset. The performance of this model is shown in Figure 12. At the end of the 33rd epoch, the model achieved 100% accuracy on the training and validation dataset. On the testing dataset as well the model achieved an accuracy of 100%.

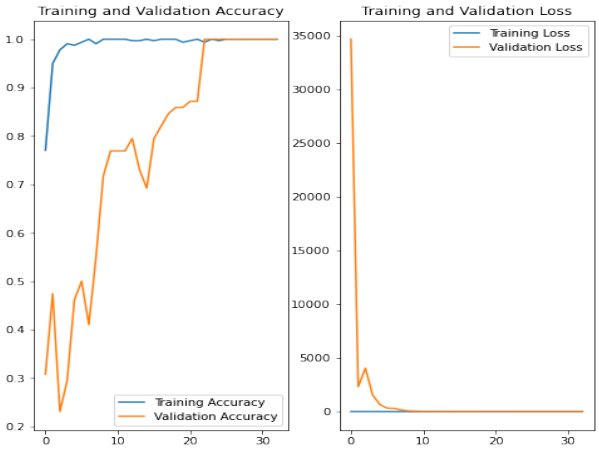

Figure 12: Performance of Fine-tuned ResNet-50 on Authorised Face Dataset

After fine-tuning, the model is used to predict face embeddings in the face verification block. Face embeddings are predicted using the second last layer of the model i.e., the Global Average Pooling Layer. Therefore, the last dense layer of the model is discarded. The output of the last layer (Global Average Pooling Layer) for a given input face image is the predicted face embedded for that face image. Now, this fine-tuned model without the dense layer is incorporated in the face verification block for predicting the detected face as an 'Authorised Face' or 'Unauthorised Face'. Once done, the performance of the face verification block with this new fine-tuned model is evaluated.

At the end of this experiment, the face verification block predicted 293 detected faces as 'Authorised Faces' and 37 detected faces as 'Unauthorised Faces'. Now the predicted labels are compared with the true labels and a confusion matrix is created. The precision, recall and f1-score of the face verification block for identifying unauthorised faces are 0.649, 0.202, and 0.307.

6.2.2 Experiment 2: Predicting Face Embeddings with a ResNet-50 Model Pre-trained on the VGGFace2 Dataset without Any Fine Tuning

In this experiment, the pre-trained ResNet-50 on VGGFace2 dataset is directly used to predict face embeddings of a face image without any fine-tuning on the authorised face dataset. Therefore, this pre-trained model without any fine-tuning is incorporated into the face verification block for prediction. After predicting all the labels for the detected faces, precision, recall, and f1-score was calculated. The precision, recall, and f1-score are are 0.872, 0.975 and 0.921 respectively.

6.2.3 Discussion of Results

From the results of the two experiments performed, it was observed that the results of the second experiment (ResNet-50 model pre-trained on VGGFace2 dataset without any fine-tuning) were much better than the first experiment (ResNet-50 model pre-trained on VGGFace2 dataset with fine-tuning). The main reason for the better performance is the richness of the VGGFace2 dataset. Due to the richness of the VGGFace2 dataset, the ResNet-50 model can generalize all faces much better than the fine-tuned model. Even though the performance fine-tuned was exceptional in classifying all our five authorised

faces, it was not good in predicting the face embeddings of a general face. As a result, bad quality face embeddings were predicted by the fined model compared with the pre-trained model without any fine-tuning.

Even if the results of the ResNet-50 model pre-trained on the VGGFace2 dataset without any fine-tuning were good, there were still some cases identified where the model failed. After inspecting the results of the experiment, it was found that, when the image quality of the detected face was good, the face verification block was successfully able to identify whether that face belongs to an authorised individual or an unauthorised individual. A good quality image is one in which the features of the face are sharp and clearly visible. For example, as per the assumption made, two good-quality face images in Figure 13 are authorised and unauthorised respectively. The face verification block in the second experiment was accurately able to identify them as authorised and unauthorised. On the other hand, when the quality of face images is poor, for example, shown in Figure 14, the face verification block identified these faces as unauthorised even though when they were authorised.

Figure 13: Good Quality Face Images of Authorised and Unauthorised Individuals

Figure 14: Bad Quality Face Images of Authorised Individuals

6.2.4 Experiment 3: Evaluating the Effects of Image Enhancement on the Performance of the Face Verification Block

In this experiment, the faces detected by the face detection block are enhanced with the Image enhancement techniques discussed in subsection 3.4. After enhancing the images, now these images are passed to a face verification block. The rest of the steps in this experiment are the same as that of experiment 2. The precision, recall, and f1-score of experiment 3 are 0.875, 0.95, and 0.911 respectively. The results of experiment 3 are similar to the results of experiment 2. Therefore, it is concluded that the image enhancement techniques implemented in this research failed to improve the performance of the face verification block.

Several experiments were performed in this research to achieve its objective. The summary of results of all experiments performed in the face detection block and face verification bock is shown in Table 3 and Table 4. After critically analyzing the results of all experiments done and discussed in the above sections, it can be concluded that using the proposed solution, this research achieved its main aim of identification of an unauthorised individual from a video clip using deep learning algorithms along with all of its other objectives.

Table 3: Summary of Face Detection Block Experiments

	Accuracy	False Positive
Experiment 1(Only MTCNN)	93%	16 Images
Experiment 2(MTCNN + CNN)	87%	6 Images

Table 4: Summary Face Verification Block Experiments

	Precision	Recall	F1-score
Experiment 1(Fine-Tuned ResNet-50)	0.649	0.202	0.37
Experiment 2(Pre-Trained ResNet-50)	0.872	0.975	0.921
Experiment 3(Image Enhancement)	0.875	0.95	0.911

7 Conclusion and Future Work

Security is an important factor for any organization and undermining the same can cause some severe consequences to the organization. Especially, securing the restricted places and making sure only authorised individuals are present in such places should be of utmost priority. Therefore, this research aimed at accurately identifying unauthorised persons in a restricted place by just using CCTV video footage and proposed a solution for the same using deep learning algorithms. However, because of an absence of a proper CCTV camera, video clips from YouTube were used instead of CCTV footage to achieve the research's main objectives. Several experiments were performed to evaluate the proposed solution and their results showed that deep learning models have the capability of accurately identifying the unauthorised individual from a video clip, thus achieving the research's main objective.

In the future, the same proposed solution can be implemented and evaluated using actual CCTV footage of a restricted place. To match two face embeddings, cosine distance was used. Instead of using a cosine distance, a more sophisticated machine learning or deep learning algorithm can be implemented to match two-face embedding to improve the overall performance of the proposed solution. Furthermore, the experiments showed that the face recognition models performed poorly while identifying a poor-quality face image. Indeed, Image enhancement techniques were used to improve image quality. However, not much improvement was seen in the performance of the proposed solution. Nonetheless, more research and experiment can be done on other types of image enhancement techniques to improve the detected face image quality and evaluate their effects on the performance of the models.

References

Buades, A., Coll, B. and Morel, J.-M. (2011). Non-Local Means Denoising, *Image Processing On Line* **1**: 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm.

Chen, D. (2020). Smart security system for suspicious activity detection in volatile areas, *Journal of Information Technology and Digital World* **02**: 64–72.

Hinton, G. E., Osindero, S. and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets, *Neural Computation* **18**(7): 1527–1554.
URL: *https://doi.org/10.1162/neco.2006.18.7.1527*

Istiqomah, F., Nuurul Izza, D. K., Susila, J., Kindhi, B. A., Indasyah, E. and Adhim, F. I. (2021). Automated barrier gate for housing estate security system using qr code based on android application, *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, pp. 293–297.

Jain, V. and Learned-Miller, E. G. (2010). Fddb: A benchmark for face detection in unconstrained settings.

Krizhevsky, A., Sutskever, I. and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks, *Neural Information Processing Systems* **25**.

Mcculloch, W. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* **5**: 127–147.

Parkhi, O. M., Vedaldi, A. and Zisserman, A. (2015). Deep face recognition, *BMVC*.

Rathour, N., Singh, R., Gehlot, A., Priyadarshi, N. and Khan, B. (2022). Klugoculus: A vision-based intelligent architecture for security system, *Computational Intelligence and Neuroscience* **2022**: 1–13.

Schroff, F., Kalenichenko, D. and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 815–823.

Singh, G., Kumar, P., Mishra, R. K., Sharma, S. and Singh, K. (2020a). Security system for railway crossings using machine learning, *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 135–139.

Singh, G., Kumar, P., Mishra, R. K., Sharma, S. and Singh, K. (2020b). Security system for railway crossings using machine learning, *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 135–139.

Soomro, A., Shaikh, N. and Hussain, R. (2022). Intelligent video surveillance techniques to detect suspicious human activities: A critical review, **2**.

Sun, Y., Wang, X. and Tang, X. (2014). Deep learning face representation by joint identification-verification, *Proc. NIPS* **27**.

Taigman, Y., Yang, M., Ranzato, M. and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification, *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708.

Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces, *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 586–591.

Wang, M. and Deng, W. (2021). Deep face recognition: A survey, *Neurocomputing* **429**: 215–244.

Wiliem, A., Madasu, V., Boles, W. and Yarlagadda, P. (2012). A suspicious behaviour detection using a context space model for smart surveillance systems, *Computer Vision and Image Understanding* **116**(2): 194–209.
URL: *https://www.sciencedirect.com/science/article/pii/S1077314211002001*

Yang, S., Luo, P., Loy, C. C. and Tang, X. (2016). Wider face: A face detection benchmark, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5525–5533.

Zhang, K., Zhang, Z., Li, Z. and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks, *IEEE Signal Processing Letters* **23**: 1499–1503.