SMART HOME SECURITY SYSTEM WITH FACE RECOGNITION AND WEAPON DETECTION

A PROJECT REPORT

Submited by

K. DINESH [Reg No: RA2011003011858]

D.PAVAN KUMAR [Reg No: RA2011003011863]

D.SAI SIDDARDHA [Reg No: RA2011003011887]

Under the guidance of

Mrs. K.DEEBA

Assistant Professor, Department of Computer Science and Engineering

in par al fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled "SMART HOME SECURITY SYSTEM WITH FACE RECOGNITION AND WEAPON DETECTION" is the bona fide work of **K.DINESH(RA2111003011858), D.PAVANKUMAR(RA2111003011863),D.SAISIDDARDHA(RA21110030118870)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	SIGNATURE
GUIDE NAME : Mrs. K . DEEBA	Dr. M. Pushpalatha
GUIDE	HEAD OF THE DEPARTMENT
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

Today home automata on systems are popular in households. The control of electric fixtures like fans and lights is possible with the help of Internet of Things (IOT). The problem arises due to intrusion of burglars. The security of such systems has been done using computer vision and IOT. Here we aim to enhance this system by use of image processing for object detection. The system uses cameras at the door for face recognition on as access control. Also, vibrant on and door magnet sensors are installed at the entry points to detect when the burglar tries to barge inside. PIR sensors are employed to detect human presence. A vibrant on sensor is also used to give alert if any shock nearby is detected. The system allows entry only if authorized person like owner or person registered on the database arrives. The person may be identified through valid proof of identiy. It sends a message to the owner in case it doesn't recognize´ the person within 20 seconds and the owner can monitor the ac vi es via live feed from the camera. All sensor signals are checked and status of the system is updated continuously. In case the burglar tries to break inside, siren is ac vated and alert messages are redirected to the owner and the police.

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	Ar ficial Intelligence
GAN	Genera ve Adversarial Networks
DDoS	Distributed Denial of Service
PDoS	Permanent Denial of Service
WSN	Wireless Sensor Networks
IOT	Internet of Things
SVM	Support Vector Machines
PIR	Passive Infrared
DVR	Digital Video Recorder
PoE	Power over Ethernet
GPIO	General Purpose Input Output
CNN	Convolu onal Neural Networks
HOG	Histogram of Oriented Gradients
PCA	Principle Component Analysis
LDA	Linear Discriminant Analysis
LBPH	Linear Binary Pa ern Histograms
P2P	Peer-to-Peer
API	Applica on Programmer Interface
MAP	Mean Average Precision
TPU	Tensor Processing Unit
LTE	Long Term Evolution
VPU	Vision Processing Unit

CHAPTER 1

INTRODUCTION

1.1 Background of the project

In this era, technology is constantly changing the world. Human brain has been studied extensively in the previous century. Scientists were able to mimic the brain function using complex mathematical models. However, they could not be applied in real life un l recently, due to lack of good computing systems. With development of advanced mathematical models, now a computer is able to learn to distinguish between images in real- me. The mystery of why computers could communicate but couldn't recognize a subject in a given image was long unsolved un l the introduction of Deep Neural Networks which lead to a new branch of study called Deep Learning. Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554 was the first paper which lead to the shift from shallow to deep approach towards Neural Networks. This approach uses various matrix operations in sequence to extract more and more useful features from a given image based on the intuition of how the human brain is able to segregate parts of an image and find useful relations among them.

Home security is one of the most significant aspects of our lives. It has been es mated that home security system market is expected to become worth 74.75 USD by 2023 according to a report ”Home Security System Market by Home Type (Independent Homes, Apartments), System Type (Professionally Installed & Monitored, SelfInstalled & Professionally Monitored, Do-It-Yourself), Offering (Products, Services), and Geography - Global Forecast to 2023”. As the homes are ge ng smarter due to the increase in home automation, the security risks possessed by them become a major concern for businesses and consumers alike. The Smart City Mission envisioned by Ministry of Urban Development (M.o.U.D.) for the term 2015 to 2020 comprises of enabling ci es with high quality infrastructure for driving economic growth. This means that internet access will be extended to each corner of the city. The modern homes demand for smart control of fans, lights, ACs, doors, kitchen appliances, etc. In order to meet these needs, each device has to be connected to internet and be remotely accessible by means of a wireless sensor network at any place. Home automation also deals with energy efficiency which can help save on costs. Sensors like thermostats are used to control ambient conditions by using a threshold to govern the indoor conditions. Major brands in home security market have implemented multifactor authentication on using biometrics. Video analysis is a major component of most systems where the data generated by cameras is processed for security purposes. The recent trends in home market security indicate that focus is now shifting to providing large scale solutions. The number of devices that can be connected are increasing multi fold. The systems are able monitor indoors and give high quality surveillance feed. They provide protection on to the home owners by monitoring levels of humidity, temperature, presence of toxic gases like carbon monoxide, etc. Hence, with the use of improved data driven algorithms, the security systems can be can be enhanced and made even smarter than existing systems in future.

1.2 Challenges in exis ng system

For an average household owner, there is no means of monitoring the indoor ac vi es remotely. Most houses don't have intrusion detec on systems. Thus cases of the have increased day by day. Surveillance systems are largely used by offices of banks, government organisa ons, educa onal ins tu ons and product based industries.

The exis ng systems are capable of delivering good performance at much higher costs. There is a scope of reducing costs in terms of both capital and computa onal performance.

Presently, systems are able to do simple object detec on for only surveillance and no market solu on is offering face access control in home space.

In the recent years, more number of solu ons relied on cloud services than on edge compu ng pla orms which have seen their entry only recently. Cloud services are unable to deliver low latency.

Most systems which are able to offer good speed and accuracy have never been employed in home security space. The most sophis cated large scale surveillance systems to do recogni on are employed to monitor road traffic and public spaces. They are capable of whitelis ng members of a family. Hence, they have good poten al as a component of home security system.

Like any other so ware, AI powered systems are prone to a acks by using AI powered a acks. Images looking very similar to each other but having differences in pixels can be used to create GAN based a acks. This could cause even a high accuracy system to misclassify an otherwise authorized person. This can be tackled by performing adversarial training on the algorithm using hard nega ve examples.

The security of smart homes can be compromised very easily by means of D.D.o.S, P.D.o.S and device hijacking a acks. The iden ty the of a person allows an intruder to bypass a system. This can be solved by mul factor authen ca on. The system must be accessible through physical means only. The tampering of the system can be detected.

If the above challenges are not addressed then, there could be catastrophic consequences including threat to lives of the inhabitants.

1.3 Applica on

This project finds its applica ons for providing security to households, shops, classified rooms, where entry has to be provided for specific individuals, while restric ng others from doing so. The major applica ons areas for this project:

- Access Control for Households, private spaces.
- Face recogni on for accessing ATM machines.
- Automated a endance using Face data.

- Providing secure access to all devices connected to WSN network
- Replacement for digital signature by delivery systems
- Ac vity monitoring by video analy cs
- Securing transporta on by authorized logis cs personnel
- Using Face access instead of keys for unlocking vehicles
- Drone surveillance by close manoeuvring
- Pet and child monitoring

It can help securing banks and ATM spaces from intruder a acks, especially during night- me.

1.4 Need of the project

The exis ng systems offer mul factor authen ca on via biometrics which means more number of parameters for access. To simplify this, we introduce a single factor of authen ca on using face recogni on.

Security systems have provided protec on against camera tampering, but it is a late response to a poten al intruder. This can be improved using weapon detec on along with it.

The primary idea was to develop a system with state of the art algorithm on a low end embedded device. The concept can then be used to make efficient security cameras and thereby reducing costs without compromising on accuracy.

The need of a system arises as the video data captured can be used for analy cs to give details of ac vi es in a mely manner. Most state of the art algorithms have a bo leneck when processing in real me without dedicated hardware. Hence, a need for the system to achieve a reliable result even without high end hardware appeared. The instrumenta on of the sensors has to be done so that the house is secured from all sides and not just a single point of entry. This requires the system to be able to detect human presence, if any shock (vibra on) is there, when the door or window is moved and an alert signal in case of any emergency.

1.5 Methodology

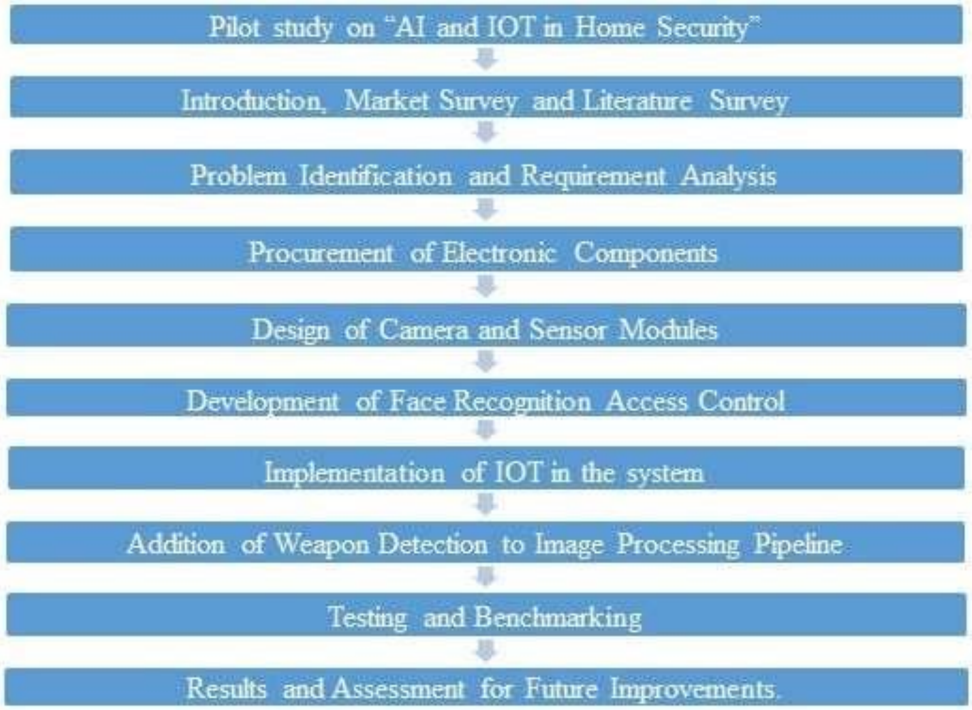

Figure 1.1: Methodology of the project

1.6 Organiza on of the report

Chapter 1 provides the background informa on of the use of AI and IOT in home security, need of the project, challenges of exis ng systems, applica ons, methodology and organiza on of report.

Chapter 2 describes the literature survey in the area of deep learning for face recogni on and object detec on.

Chapter 3 describes the so ware and hardware requirements of the system with specifica ons of the packages and apparatus used to build it.

Chapter 4 explains the methodology which was used for instrumenta on and control of the system. This involves selec on of various sensors, their func ons, development of image processing algorithms and internet of things for no fica ons.

Chapter 5 contains the test results and benchmarks obtained a er evalua on of the system.

Chapter 6 gives the conclusions and future scope of the project.

CHAPTER 2

LITERATURE SURVEY

Schroff, Florian, Dmitry Kalenichenko, and James Philbin ("Facenet: A unified embedding for face recogni on and clustering." Proceedings of the IEEE conference on computer vision and pa ern recogni on. 2015) in the paper tled "Facenet: A unified embedding for face recogni on and clustering" have developed a unified model of face verifica on, recogni on and clustering. The method promises to eliminate performance issues due an intermediate bo leneck layer which is present in previous deep learning models. The Euclidian distances between the 128 dimensional face embedding are calculated. During the training phase, the L2 distances represent face similarity with faces of the same person having smaller distances and faces of dis nct people have larger distances. Face verifica on is done by crea ng thresholds between two embedding. Recogni on task is translated into a k-NN classifica on problem. Clustering is done using off-the-shelf techniques such as agglomera ve and k-means clustering algorithms. The processing of CNN can be improved by removing the layer dedicated to process image for extrac ng faces and introducing a new loss func on. The 128-D embedding is produced using a triplet based loss func on. For increasing the model efficiency, hard training examples are used which can be discarded by the loss func on. Hence it appears to perform in a fashion similar to large margin classifiers like SVM. However, the model's performance decreases with decrease in number of parameters without loss of accuracy. The model has achieved an accuracy of 99.39% on Labelled Faces in the Wild (LFW) dataset and 95.12% on YouTube Faces DB dataset.

Kazemi, Vahid, and Josephine Sullivan. ("One millisecond face alignment with an ensemble of regression trees." Proceedings of the IEEE conference on computer vision and pa ern recogni on. 2014) in the paper tled "One millisecond face alignment with an ensemble of regression trees" have approached the problem of Face Alignment for a single image. The method provides a high accuracy and fast way to perform landmark detec on on faces. The image is pre-processed using highly accurate regression func ons which are trained using gradient boos ng. The pixel intensi es are indexed rela ve to the shape es mated ini ally. To avoid the effects of noise produced by illumina on levels so that reliable features can be extracted, the shape vector is obtained by regressing over features in a normalized coordinate system and not the global coordinate system of the image. It also deals with finding the best possible shape to fit the image data. Since the nature of the problem is non-convex, there is no single global op ma but many local op ma. The solu on lies in assuming the subspace to be linear which can be done by methods like training an SVM over principal components extracted using PCA algorithm. The number of possibili es of shape outcomes are reduced significantly while inferencing and there no other constraints are needed. A squared error loss func on performs with high efficiency via gradient boos ng. A set of sparse data (pixel set) is fed to the regression algorithm's input. This pixel set is selected using gradient boos ng as well as prior probability on the distance between input pixel pairs. Hence an ensemble of regression trees is able to find the best facial landmarks when given mean face pose as ini al parameters. The method can tackle the issue of missing labels or uncertain labels. The method is capable of producing high quality predic ons.

Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. ("Deep face recogni on." bmvc. Vol. 1. No. 3. 2015) in the paper tled "Deep face recogni on." have developed a novel approach to obtain a large scale dataset using the combined efforts of both humans and automa on for face recogni on. The approach further discusses on how to improve the training of deep neural networks by suitable configura ons. The dataset is reduced by three order of magnitude and yet is able to achieve state of the art performance on LFW and YTF face benchmarks. The images were collected by using both frontal and profile photos of subjects from internet using Google and Bing search engines. The classifier used for faces has a simple vector representa on wherein one hot encoding is obtained from fully connected layer. The ideas of face embedding are based on triplet loss func on where posi ve examples have distances closer to the anchor and nega ve examples have distances far away in the Euclidian space. Different configura ons of architecture were developed for understanding effects on the model's performance. Component analysis was done by checking the effects of data cura on, alignment, architecture and triplet-loss embedding on the performance. The ROC curves indicate that the proposed model is able to achieve performance comparable to DeepID3 model.

Liu, Wei, et al. ("Ssd: Single shot mul box detector." European conference on computer vision. Springer, Cham, 2016) in the paper tled "Ssd: Single shot mul box detector" describes a novel approach to counter problems related to two-stage detec on which demand high computa on and not suitable for embedded systems. This approach deals with object detec on with only one deep neural network. The bounding boxes of all objects are given as output and adjusted according to the matrix dimensions for the object. It is able to combine predic ons taken from various feature maps so that it can handle objects of different sizes in a single frame. The advantage of the model is that it has good balance between speed vs accuracy trade-off even on image inputs with smaller size. The training of the model is easy compared to two-stage object detectors. It can be deployed on low-end embedded devices where the computa on environment is highly constrained. A feed-forward convolu onal network is employed that processes fixed-size groups of bounding boxes as well as predic ons for respec ve object classes. The layers in the network decrease successively and allow predic ons of detec ons at mul ple scales. Hard nega ve mining ensures at most 3:1 ra o of the nega ves and posi ves. This leads to faster op miza on and a more stable training. Data augmenta on is used to make model robust to various input object shapes and sizes. The model is very promising as a framework for high quality real-me detec on but it has a lot worse performance on smaller objects than bigger objects. The conclusion states that this detector is capable to be a part of a system using recurrent neural networks to detect and track objects in video simultaneously.

Lin, Tsung-Yi, et al. ("Focal loss for dense object detec on." Proceedings of the IEEE interna onal conference on computer vision. 2017.) in the paper tled "Focal loss for dense object detec on" have described a state of the art approach to perform object detec on by improving the accuracy of other one-stage detectors which are faster and simpler. The primary issue was the presence of class imbalance which was addressed by modifying the conven onal cross entropy loss so that it can be trained to perform be er on nega ve examples during dense sampling of the poten al object loca ons.

The Focal loss term penalises the normal loss func on as it performs too accurately on easy examples. The approach claims to achieve both speed and accuracy of one-stage and two-stage detectors respec vely. The loss func on is ini alized with a prior value of p eliminate the dominance of a single class in early training. The first subnetwork performs classifica on on the output of the backbone and the second subnet performs regression. The total focal loss of an image is calculated as the sum of the focal loss for all 100k anchors, followed by normaliza on by the number of anchors assigned to a ground-truth box. The training loss is the sum of the focal loss and the standard smooth

L1 loss used for box regression.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 So ware Requirements

The project was developed using Python. This chapter describes about the so ware packages and libraries which were used in the project. The project uses Raspberry Pi 3 Model B, hence, the opera ng system is Linux based Raspbian OS. Image Processing was implemented with the help OpenCV. Intel OpenVINO Toolkit R5.1 for Linux is used for hardware accelera on of image processing. IOT was implemented using IFTTT android applica on.

3.1.1 The Python Programming Language

Python was developed by Guido van Rossum in the early 1990s. It has a lot of advantages over other object-oriented programming languages like C++ and Java. The primary advantages of the language are as follows:

- Due to its popularity among scien sts for numerical computa on, python was adapted rapidly for machine learning and ar ficial intelligence. Some of the libraries which are available are Scikit-learn, Scipy, Numpy, Tensorflow, Keras, Pytorch, Matplotlib, etc.
- It gives the user more flexibility in terms of applica on development as compared to MATLAB even though the la er is recommended for faster prototyping and diverse in terms of tools available.
- It is simple to program in python and learning curve is not steep. The syntax is easy to use and the use of strict indenta on to improve code readability.
- It is independent of pla orm. Hence, the programs built and tested on Raspbian OS, could also be tested on Windows pla orm.

3.1.2 Raspbian OS

Raspberry Pi uses a Debian-based opera ng system. The project uses Raspbian Stretch among other op ons like Ubuntu MATE and Windows 10 IOT Core as it is the na ve OS and more stable. The OS highly suited for the Raspberry PiâA˘ Zs low performance´ ARM CPUs.

The user interface supports GUI with Python pre-installed. It is oriented to help users who do not use Linux for development. The file system, networking, process handling and access to peripherals using Linux kernel.

3.1.3 OpenCV Library

OpenCV (Open Source Computer Vision Library) is an image processing library made by Willow Garage of Intel. It was built for real- me computer vision applica ons. It was designed for efficient computa on. It is wri en in C/C++ and can take advantage of mul -core processing.

OpenCL gives it support for hardware accelera on. OpenCV's applica on areas include:

- Egomo on es ma on
- Facial recogni on system
- Mo on understanding
- 2D and 3D feature toolkits
- Segmenta on and recogni on
- Structure from mo on (SFM)
- Mo on tracking
- Gesture recogni on
- HumanâAS¸computer interac on (HCI)˘
- Object iden fica on
- Stereopsis stereo vision: depth percep on from 2 cameras
- Augmented reality

OpenCV uses a sta s cal machine learning library that contains:

- Boos ng
- Ar ficial neural networks
- Decision tree learning
- Gradient boos ng trees

- Expecta on-maximiza on algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

OpenCV has several sta c libraries. The following modules are available:

- High-level GUI (highgui)
- Video I/O (videoio)
- Video Analysis (video)
- Image Processing (imgproc)
- 2D Features Framework (features2d
- Core func onality (core)
- Object Detec on (objdetect)
- Camera Calibra on and 3D Reconstruc on (calib3d)
- Some other helper modules, such as FLANN and Google test wrappers and others.

3.1.4 Intel Distribu on of OpenVINO Toolkit

OpenVINO toolkit, short for Open Visual Inference and Neural Network Op miza on toolkit, provides refined neural network inference on select Intel processors to create real- me vision applica ons. The toolkit enables to do inference in deep learning and speedier inference across mul ple Intel pla orms (CPU, Intel Processor Graphics) deployable on edge devices. It takes away image processing workloads to Intel hardware thereby maximizing performance.

The Intel Distribu on of OpenVINO toolkit:

- Allows uniform execu on across Intel CV accelerators, using a common API for the CPU, GPU, and Vision Processing Units. Enables CNN-based inferences on the processors.
- Includes efficient func ons adhering to computer vision standards, with OpenCV, OpenCL, and OpenVX.
- Fast me-to-market through an easy-to-use library of computer vision func ons and kernels op mized beforehand.

3.1.5 Numpy

NumPy is the essen al toolkit for scien fic compu ng in Python. It contains but is not limited to:

- A highly impac ul N-dimensional array object
- helpful Fourier transform, linear algebra, and random number capabili es
- tools for use with Fortran code and C/C++
- sophis cated (broadcas ng) func ons

It can also be used as an effec ve mul -dimensional generic data container. Customized and less-restricted data-types can be defined. Thus, it can be easily used with a broad range of databases.

NumPy is licensed under the BSD license, enabling reuse with few restric ons.

3.1.6 Dlib's C++ Library by Davis E. King

Dlib is a contemporary C++ u lity having machine learning algorithms and tools for solving problems. It is used in a wide range of domains including embedded devices, robo cs and large high performance computa on.

3.1.7 Face Recogni on API for Python and Command Line by Adam Geitgey

The model is using Dlib's state of the art face iden fica on developed with deep learning. It has 99.38% accuracy on the Labelled Faces in the Wild dataset benchmark. It has a well-defined command line tool called face_recogni on which can work on an image collec on. It has capability to perform real- me face recogni on.

3.1.8 IFTTT

If This Then That is a free web-based service to create chains of simple condi onal statements, called applets. An applet is triggered by a web-based service like Facebook, Twi er, Gmail, Telegram, Twilio, Messenger, etc. It is available for both iOS and Android. It offers WebHooks, a service which allows users to send triggers from python to usually (but not limited to) IFTTT app.

3.2 Hardware Requirements

The components required for the project include PIR sensor, Door Magnet sensor, Vibra on sensor, circuit break detec on module for wire connec ons, panic switch, Raspberry Pi 3 Model B board, Pi Camera module, Neural Compute S ck 2, servo motor, Hikvision Network Camera (Wired), Network Switch, PoE cables, Ethernet cables, DVR, JioFi device(router), SD card, monitor, keyboard, mouse, breadboards, jumper wires, extension wires, LED modules, buzzers, 12V DC ba ery, Arduino Uno board and 5V Power Supply.

3.2.1 PIR Sensor

PIR sensor (Passive Infrared) sensor measures the change in temperature when an object radia ng heat (human in this case) passes in front of it. It is a passive sensor which means it can only read the values.

Figure 3.1: PIR Sensors

Table 3.1: Technical Specifica ons of Texecom PIR Sensor

Input Power	9 VDC to 15 VDC (15 VDC nominal @ 10.6 mA) Power ra ng 0.16 W
Peak to Peak Ripple	2 V (at 12 VDC)
Detector Start-up Time	60 seconds
Normal Current Consump on (mA)	max. 8.7
Current Consump on in Alarm (mA)	max. 7.5
Max Current Consump on (mA)	max. 10.6
Moun ng Height	Compact IR, XT QD Min. 1.5 m (4.1 .), Max. 3.1 m (10 .) Compact PW Min. 1.5 m (4.1 .) Max. 2.3 m (7.5 .)
Target Speed Range	30 cm/s to 3 m/s (1 ./s to 10 ./s)
Alarm Relay	< 24 V 50 mA (NC) max. 34 Îl' resis ve load
Tamper Relay	< 24 V 50 mA (NC)
Alarm Time	> 2 seconds
Pet Immunity (Compact PW)	Pulse Count 1 Up to <20kg/44lbs Pulse Count 2 Up to <35kg/77lbs
Opera ng Temperature	-10°C to +55°C (14°F to 130°F)
Dimensions (HxWxD)	95 mm x 63 mm x 40.5 mm
Rela ve Humidity	Max. 95%
Weight	73 g

Power Supply	Rated 94HB

3.2.2 Door Magnet Sensor

Door Magnet Sensor creates a magne c seal between the contact surfaces, forming a circuit. The surfaces are mounted on door and the frame. As the door is opened, the circuit is in open because of the separa on of contacts from each other.

Figure 3.2: Door Magnet Sensor

3.2.3 Vibra on Sensor

The vibra on sensor operates on the principle of resonance of frequency. The alarm triggers when the external impulse frequency due to shock matches with the resonance frequency of the sensing element. False alarms can be avoided with appropriate calibra on of the intensity controller. It can be placed at walls, windows, ceilings, door frames, etc.

Figure 3.3: Vibra on Sensor

Table 3.2: Vibra on Sensor

Electrical	
Opera ng voltage	12V DC
Supply Current Standby	20 mA
Supply Current Triggered	10mA
Output	Normally closed poten al free contact
Sensor	Piezo electric
Environmental	
Opera ng Temperature	0°C to 45°C
Storage Temperature	-20°C to 60°C
Maximum Humidity	95% non-condensing
Environmental	Residen al/Commercial/Light Industrial
Physical	
Dimensions	93 mm x 30 mm x 28 mm

3.2.4 Circuit Break Detec on

A pin is set to low in the Arduino board when the wires are connected. As soon as the wires are cut, the signal reverses to high, triggering an alert to the owner.

Figure 3.4: Circuit Break Detec on

3.2.5 Panic Switch

Panic switch simply triggers an alarm/alert when it is pressed, so that when the user can seek help when no other alerts are accessible or opera onal.

Figure 3.5: Panic Switch

3.2.6 Raspberry Pi 3 Model B Development Board

The Raspberry Pi is a small single-board computer used in many other applica ons such as robo cs, image processing, security systems, etc.

Figure 3.6: Raspberry Pi 3 Model B

3.2.7 Pi Camera

The Pi Camera v1.3 has a 5MP 1080p 30fps 25x23x8 Image size-2592ÃU1944 which˚ is designed to work with the Raspberry Pi 3 Model B. The camera has found its applica ons in numerous security purposes.

Figure 3.7: Pi Camera v1.3

3.2.8 Neural Compute S ck 2

The Intel Movidius Neural Compute S ck 2 is a small fan-less deep learning device. NCS2 is powered by high performance Intel Movidius Myriad X VPU with a performance boost of 8 mes over the previous genera on.

Figure 3.8: Intel Movidius Neural Compute S ck 2

Table 3.3: Technical Specifica ons of Neural Compute S ck 2

Processor	Intel Movidius Myriad X Vision Processing Unit (VPU)
Supported frameworks	Tensorflow and Caffe
Connec vity	USB 3.0 Type-A
Dimensions	2.85 in. x 1.06 in. x 0.55 in. (72.5 mm x 27 mm x 14 mm)
Opera ng temperature	0°C to 40°C
Compa ble opera ng systems	Ubuntu* 16.04.3 LTS (64 bit), CentOS* 7.4 (64 bit), and Windows 10 (64 bit)

3.2.9 Servo Motor

The servo motor is linear or rotary actuator that allows for precise control of accelera on, velocity, angular or linear posi on. It is used in the applica on of robo cs, posi on control, automated manufacturing and CNC machining.

3.2.10 Hikvision Network Camera (Wired)

The Hikvision Network Dome Camera is a 1.3 Megapixel Dome shaped, CMOS based Vandal-proof camera. It is used for surveillance and video data is stored using DVR or NVR. It has support for IR for a range of approx. 10 to 30 metres. It also features intrusion and mo on detec on. It is powered via PoE using a network switch.

Table 3.4: Technical Specifica ons of Network Camera (Wired)

Model	DS-2CD2110F-I(W)(S)
Camera	
Image Sensor	1/3" Progressive Scan CMOS
Min. Illumina on	0.01Lux @ (F1.2, AGC ON) , 0 Lux with IR 0.028 Lux at (F2.0, AGC ON) ,0 Lux with IR
Shu er Speed	1/3 s to 1/100,000 s
Lens	4mm @ F2.0 (2.8mm, 6mm op onal) Angle of view: 92.5°(2.8mm), 73.1°(4mm) , 46°(6mm)
Lens Mount	M12
Day and Night	IR cut filter with auto switch
Digital Noise Reduc on	3D DNR
Wide Dynamic Range	Digital WDR
Angle Adjustment	Pan:0°- 355°, Tilt: 0°- 75°, Rota on: 0-355°
General	
Opera ng Condi ons	-30 °C âAS¸ 60 °C (-22 °F to 140 °F)˘ Humidity 95% or less (non-condensing)
Power Supply	12 V DC±10% PoE (802.3af)
Power Consump on	Max. 5W
Material	Base: Metal; Top Cover: Plas c
Ingress Protec on level	IP67
IR Range	30 meters
Impact Protec on	IK10
Dimensions	111 x 82 (4.4" ÃU 3.2")˚
Weight	⌀ 500g (1.1 lb)

Figure 3.9: Hikvision Network Camera (Wired)

3.2.11 Network Switch

A switch is a device in a computer network that forms connec ons between other devices. In order to enable communica on between different networked devices,

more than one data cable can be plugged into a switch. Each networked device, to which switch is connected, is known by its network address. This way the switch can channelize the flow of traffic enhancing the security and efficiency of the network to the maximum extent. The flow of data across a network is managed by transmi ng a received packet only to the one or more intended devices.

3.2.12 PoE Cable

PoE (Power over Ethernet) cable provides power supply as well as data communica on between devices which include wireless cables, IP Cameras, VoIP phones.

3.2.13 Ethernet Cable

Ethernet cables are connected to RJ-45 ports of devices. It is used to connect network switch to the computer.

3.2.14 DVR

DVR (Digital Video recorder) records any video in digital format into a disk-drive, USB drive, SD card, SSD or any storage device or cloud. DVR comes with a setup of video player and TV gateways with record capabili es and digital camcorders. Mostly personal computers are connected for the use of surveillance purposes.

3.2.15 Jio-Fi Device

A Jio-Fi is a portable mini-wifi router which is used as a hotspot for devices. Jio-Fi has a Jio sim inside which provides a high speed 4G internet.

3.2.16 SD-Card

SD Card (Secure Digital) is device used to store data. A micro SD card is commonly used as a portable external storage for smartphones, digital cameras, tablets and other embedded devices. It is used as primary storage for single-board computers.

3.2.17 Arduino Uno Development Board

Arduino Uno is open source for so ware, hardware, projects which is designed for single board microcontrollers and microcontroller kits that can be used for the interac on between various components, sense and control the ac ons of the specific devices. It has GPIO (Input/Output) pins that let other devices/sensors interact and make respec ve ac on with surroundings.

Figure 3.10: Arduino Uno Development Board

3.2.18 Power Supply

Power supply is necessary for the sensor and lock to func on for which a dedicated power source is used for the lock system and sensor module. A 5V supply from the Arduino board is provided to data pins of sensors and servo lock. A 12V power supply is used for power and ground pins of the sensors.

CHAPTER 4

METHODOLOGY

4.1 Overview and conceptualiza on

A er a pilot study conducted on "AI and IOT in Home Security", it was concluded that the system has to be designed such that it can be deployed in a small form factor such as a single-board computer. Performance and cost of the system usually present themselves as trade-offs. To combat this, we propose a low-end system which can perform surveillance as well as image processing effec vely.

An embedded system is a resource constrained environment where off-the-shelf algorithms donâA˘ Zt work perfectly. Keeping this in mind, the hardware was chosen such´ that it met performance requirements and a customized algorithm was developed for the same.

The access control methods used previously have been through means like smart cards for residen al purposes. Later, sophis cated algorithms were used for biometrics based access control which include fingerprint, re na scanners, etc. However, with advancement in research towards face recogni on, the access control methods witnessed the rise of its use in everyday life - the best example being in case of smartphones. It was less secure in its ini al phase and is under constant improvement since its introduc on into the market.

The sensors which were generally used by mul na onal corporate firms for employees for a endance have inspired ideas for access control solu ons. Industries use video analy cs for monitoring ac vi es inside the plants. Gradually, these technologies were introduced in homes to make them âAŸsmartâ˘ A˘ Z. Security of smart homes requires´ design of intrusion detec on systems. This may not include physical barriers or traps which are beyond the scope of cost effec ve solu ons. The other sensors which are incorporated into home automa on systems include those for illumina on, humidity, temperature, fire, seismic ac vity, etc.

The model of a smart home was simplified to include sensors which are part of intrusion detec on system only. The proposed system was designed to accomplish two tasks intrusion detec on and access control.

Most systems don't include a method to deal with unforeseen situa ons like camera tampering. This can be resolved by detec ng if wire was cut using the circuit break detec on module. An experimental and dis nguishing feature of using camera to detect weapons is also proposed to provide alerts about suspicious ac vi es. The proposed system in its current state also incorporates a standard security camera in order to record ac vi es throughout the day which is a common component in most setups.

4.2 System Design

The system comprises of 4 simple modules namely-

- Sensor module
- Camera module
- Face Access Control Lock
- No fica on service

4.2.1 Sensor Module

Spread all around the peripherals and entry points of the house, the sensor module is able to record signals whenever the sensors are triggered. The main door is equipped with door magnet sensor. When, in absence of the owner, the door is opened/forcibly broken, alarms will start ringing in the house and the owner would be alerted with specific no fica ons. If the intruder tries to force open the door, the vibra on sensors equipped on the door will detect the same. Vibra on sensor is also integrated with the window panes, a common entry point for burglars. The grill of the window is covered by a wire running through all the parts of the grill. If the burglar tries to cut/weld the grill, the wire will be cut and in turn, a signal will be generated. A PIR sensor which will be mounted either inside or outside of the house, will detect human presence where there should normally be none. Also present is an emergency panic switch which would be used in emergency situa ons, either regarding possible burglary while the owner is s ll inside. Another important aspect for using a panic switch is for senior ci zens who are inside the house alone and in need of medical a en on.

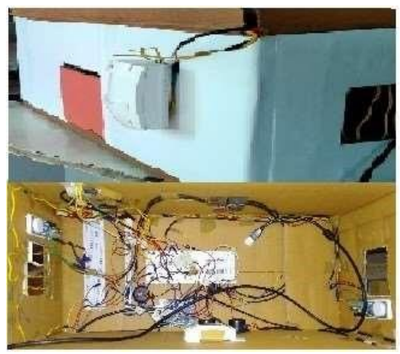

Figure 4.1: Sensor Module

4.2.2 Camera Module

The camera module consists of one primary camera and one secondary camera. The primary camera is Pi Cam, which has been equipped on Raspberry Pi model 3B. This camera is used for face detec on as well as recogni on. The algorithms are implemented on the Raspberry Pi 3 Model B board.

The secondary camera, on the other hand has its own recording and monitoring system.

It is a Hikvision network camera equipped with a DVR having the capacity of 1TB HDD. This network camera captures and sends pictures/videos as no fica ons to the phone whenever any ac vity is detected.

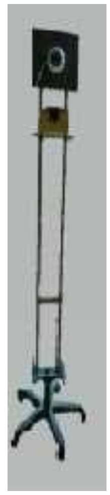

Figure 4.2: Camera Module

4.2.3 Face Access Control Lock

This is door lock which is simulated by a servo motor to which the miniature door model is a ached. The servo is connected to Raspberry Pi 3 Model B board via GPIO pins and is fed signal for actua on. Face recogni on program runs in loop and as soon as a known face is detected a servo is actuated for access. If the face detected is unknown then, no permission is granted. The person can be dis nguished from a thief by detec ng if s/he possesses any weapon. For the demonstra on purpose, simple tools like hammer, scissors, knife, etc. which are easily available in public datasets are available. This reduces me to development because pre-trained models trained on those datasets are readily available.

Figure 4.3: Servo Motor used for simula ng door lock

4.2.4 No fica on Service

The no fica ons to the owner can either be on his phone or any remote device. A smartphone/feature phone as the primary device for ge ng alerts in the form of No fica ons, SMS, Calls, etc. using IFTTT app on Android and iOS. Alerts can also be created to no fy the police in case of burglary. This module no fies ac vity with sensors and sends messages when an intruder with weapon is captured by the camera.

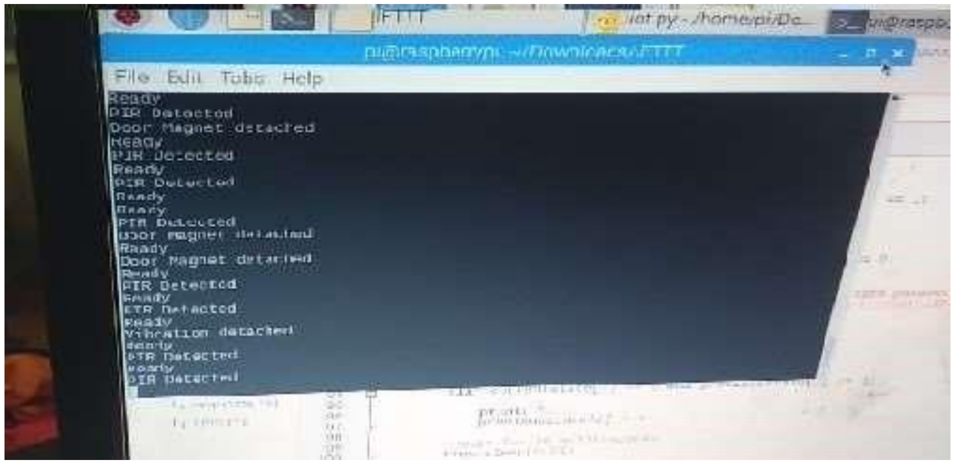

Figure 4.4: No fica on sevice using IFTTT app

4.2.5 Circuit Diagram

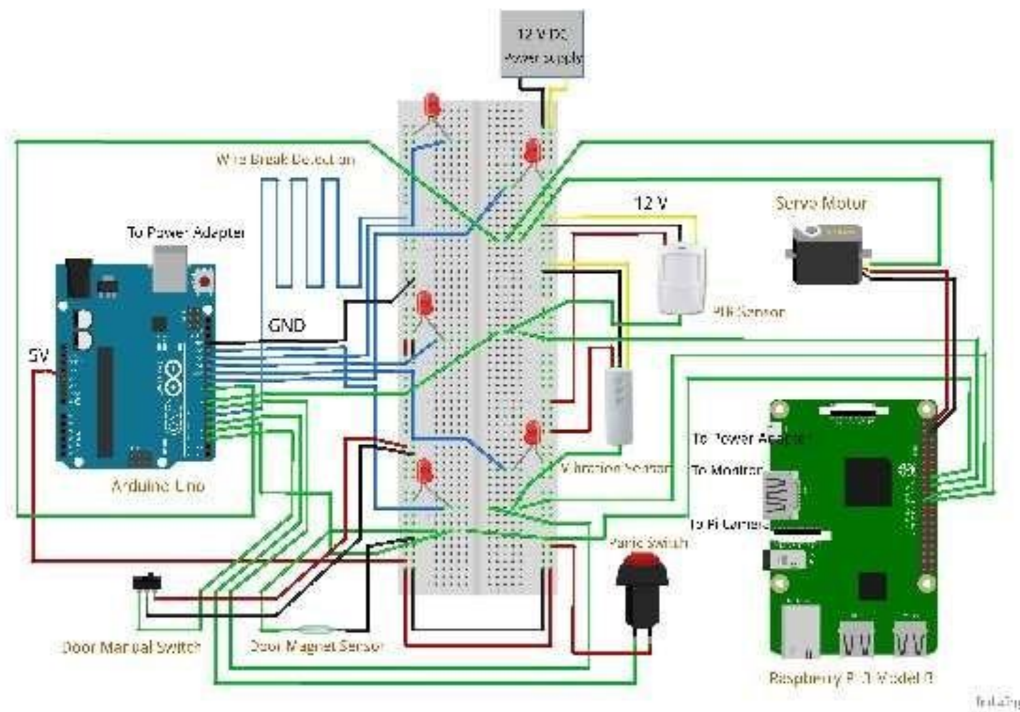

Figure 4.5: Circuit Diagram of the System

4.3 Image Processing Algorithms

The algorithm governing the tasks of face access control and weapon detec on were developed. A study on the state-of-the-art methods was carried, giving away clues to the working of these algorithms.

Image processing involves ge ng the image/video data and performing series of numerical computa on to give desired output. The en re processing pipeline involves several stages to achieve outputs in various ways such as bounding box detec on, ROI landmark detec on, labelling of the bounding box, pose es ma on, op cal flow, gathering informa on about the ac vity, tracking a par cular object in real me. The image has to be pre-processed before passing it to the algorithm so that inferences are faster. Pre-processing stages include cropping, resizing, blurring, scaling, colour transforma ons, etc. Post-processing is required a er obtaining predic ons, in order to display the frame in its original form.

The goal of the Face Access Control Lock module is to recognize faces in real- me scenarios from the video captured by Raspberry Pi Camera as input and grant permission if the person is authorized. The moduleâA˘ Zs image processing has two sub-modules -´

- Face Recogni on
- Weapon Detec on

The Face Recogni on task involves detec on of faces in each frame of the video object created for live feed. Detec on can be done using Deep Learning based CNN model pre-trained on faces and available as open-source so ware. The results of the detec on give the loca on of the faces detected in each frame. For each face detected, a 128-D feature vector is generated by compu ng the dis nguishing distance metrics which can be rela ve distances between eyes, nose, mouth, chin, jaw line, forehead etc. and also their measurements as obtained using face landmarks. These vectors are then matched with faces which are known with their pre-computed feature vectors. If they match then, the corresponding name stored with the vectors are returned as labels. The end result is a bounding box over the face with name label on top. It is to be noted that the above model used for implemen ng the above method is a frontal face detector and does not work for other orienta ons of the faces.

Weapon detec on is a form of object detec on where the key task is to iden fy the weapons in the frame among the objects captured in the video. Object detec on is carried by processing the frames through object detec on model which is another Deep Learning based CNN model pre-trained on objects and covers sufficient classes of objects. The objects in the scene have to detected by segmenta on and detec ons are passed as coordinates for recogni on task. The detec ons of the results are complemented by an addi onal parameter in the form of a number which is the class id of the object detected. This id is matched with label mapping where the names of the object along with corresponding ids. If the ids match, then the corresponding names are returned as labels. When the label falls under class of weapons, an alert is pushed as no fica on to the owner. Both the above tasks above require predic ons to be made on images. Earlier, primi ve methods have been used to make these predic ons such as object classifica on, Image classifica on and object localiza on which are relevant even at present. The drawbacks of these were that they were computa onally expensive and inaccurate bounding boxes were returned due to mismatch between shape of the box and the object of interest.

Recently, methods have been proposed and implemented which can run faster with less number of computa ons and deliver op mal accuracy. The previous approaches involved cropping of images into equal parts and passing them one at a me to the neural network for detec on. However, modern approaches simplify the above step by passing the en re image to the network instead of cropped images. An addi onal layer in the network is used to compute the edges of the objects all at once.

For improving accuracy of the network, âAŸYou Look Only Onceâ˘ A˘ Z method is used.´ According to this method, the image is par oned into mul ple grids. The loca on and classifica on of the objects is done simultaneously for each grid. These give acceptable results and cons tute the group of object classifica on algorithms came to be known as single-shot detectors. However, they ignore the background informa on of the image during training. This reduces the accuracy when comparison with two-stage detectors like mask-RCNN networks. The most advanced methods propose tweaking the loss func on and not by modifying the network to achieve the desired results.

4.3.1 Face Recogni on

Various detec on algorithms have been created to detect faces. The first method was developed by Paul Viola and Michael Jones in the early 2000's and widely adopted. It gives the detected eyes and faces as output. More advanced methods are used at present. Some of the methods are as follows-

- Haar-cascades: These methods involve extrac ng Haar-like features from the given image to convert feature points to feature vectors. One of the popular ways is Histogram of Oriented Gradients (HOG). The method marks arrows on each pixel by comparing the brightness levels of neighbouring pixels on the grayscale image. The collec on of such arrows give direc on of the pixel intensity as it varies from dark to bright regions. However, the dimensions will be huge for processing. The dimensions can be reduced by consider larger areas of the image instead of going for each pixel.
- Eigen Faces: This method reduces dimensionality of the images by deriving only relevant informa on from the image. The method uses principle components (PCA) of the image, which are those features with maximum variance, for feature extrac on. This reduces the image representa on complexity and also saves me and space during computa on.
- Fisher Faces: This approach promises to solve disadvantages of the Eigen Faces method. External factors such as illumina on affect the PCA, wherein components which may contain discrimina ve informa on may also get eliminated. The solu on lies in applying LDA (Linear Discriminant Analysis) to reduce variance among the classes instead of maximizing the overall variance. This essen ally segregates the same classes from the ones which are different. Hence it is u lized to recognize faces.
- Linear Binary Pa ern Histograms: The methods described above work well with lower dimensional data. However, they fail if the amount of data is reduced and noise generated by factors in non-ideal condi ons. According to this algorithm, each pixel is given a value of 1 or 0 based on whether it's intensity increases or decreases in comparison to previous pixel's intensity value. The neighbouring pixels create a batch of 3x3 neighbourhood. The pa ern of numbers generated in this manner give a fine representa on of the image known as LBP codes. These codes can help recognize faces. More recently, using the above approaches new sta s cal and probabilis c models have been developed. These approaches are Deep Learning based CNNs which were modified for the purpose of face detec on.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recogni on and clustering." Proceedings of the IEEE conference on computer vision and pa ern recogni on. 2015 have proposed a model for detec ng faces in images and recognizing them using a unified approach. In this work, Euclidean distances are measured and project same faces near to each other and different faces apart from each other in the Euclidean space.

Figure 4.6: Flowchart depic ng Face Recogni on pipeline

The above diagram represents the series of steps through which each frame from the video has to undergo during the pipeline. The term âAŸpipelineâ˘ A˘ Z refers to the´ fact that the above steps canâA˘ Zt be executed in parallel but in a sequen al manner.´ However, asynchronous processing of the video thread can boost the speed of the algorithm. The video object can make use of mul -threading for higher number of frames per second. Pose Es ma on refers to the process of determining the posi on and orienta on of the head. This is essen al as the neural network regards the same person with different head orienta ons as different. Landmark detec on is a preprocessing step for measuring the distance metrics of the face. Kazemi, Vahid, and Josephine Sullivan. ("One millisecond face alignment with an ensemble of regression trees." Proceedings of the IEEE conference on computer vision and pa ern recogni on. 2014) invented a method to mark landmarks on faces. Their method uses 68 points marked on the faces invariant to the head pose and also resistant to small obstacles on the faces.

A dataset of authorized people, with 10 to 20 images per person is used to generate encodings i.e. 128 dimensional vectors for each person. These encodings are later used for comparison with those of faces captured through live feed. Classifica on involves using a linear classifier such as an SVM (Support Vector Machine) to make predic ons when the given face is represented as feature vector. This is similar to one vs all classifica on as in case of k-means clustering.

The accuracy of the face detec on model is very high as it was trained on millions of images of faces. Hence, even when very few images of a person are presented it can learn the embedding very quickly. Thus it can classify a person for whom it was never trained before, by just seeing very few examples of the face and the corresponding embedding.

4.3.2 Weapon Detec on

Object detec on is a commonly used algorithms in many machine vision tasks for both industrial as well as non-industrial purposes alike. Many methods have been introduced for ge ng faster and accurate outputs. As the amount of data is increasing day by day, researchers have invested me in building large datasets by carefully pu ng them in under categories and ensuring that they have rich informa on for training object classi-

fiers.

The no on of using a convolu on of opera ons in succession to extract meaningful features from images has been prevalent long back since 1960âA˘ Z's but could not reach´ its poten al due to limita ons on computa on.

With modern machines, a 'deep' neural network can be trained in a frac on of the me than what was possible before. The object detec on algorithms can be used for detec ng persons, animals, faces, vehicles and other everyday items. To extract dis nguishing features the image is pre-processed and passed through several layers of the deep convolu onal neural network. Each layer is a matrix which is the resultant of a mathema cal opera on performed on the matrix from previous layer.

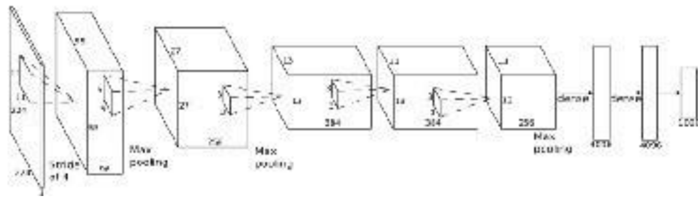

Figure 4.7: An example of a convolu onal neural network architecture

Earlier, a sliding window approach was used to hover over each segment of the image and check if the frame contains an object. Later, the neural networks were made âAŸdeeperâ˘ A˘ Z by adding layers to do the detec on automa cally, even when the object´ is not centred in the sliding window. This is possible with large amount of training examples. The use of convolu on in neural networks is crucial for any modern deep learning algorithm, be it face recogni on or object detec on.

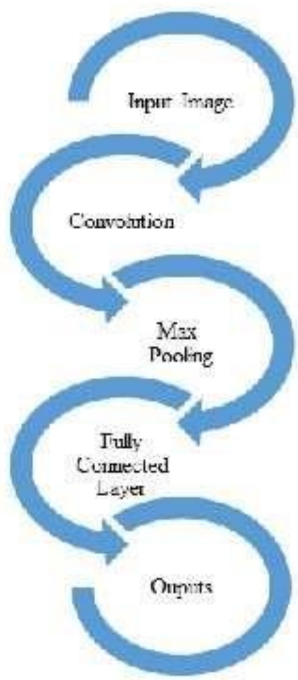

Figure 4.8

The convolu on process consists of dividing the en re image into ny equal parts which can be termed as a le. Each such le is traversed through a small neural network and saved into a separate array. If any useful informa on is available, it is marked for using a erwards.

The array is able to map out different parts of the image where useful features are available. The array is high dimensional. So, it can be projected to lower dimensions by a technique called down-sampling. Here, the array is processed in square grids cons tu ng a single batch. The maximum value of those grids is stored in the output array. This opera on is known as max-pooling.

The matrix which was extracted using max-pooling is an array of numbers and can be unrolled to give a feature vector which has a single column of numbers. This is basically an array which is further passed into another neural network called fully connected layer. The role of this neural network is to make a predic on of whether the given image matches the object in ques on.

The above 3 steps are repeated in various combina ons to get highly complex networks.

This way helps to learn even more complex features, as the number of convolu on layer increases. The goal of the process is to reduce the image into simpler feature vectors which can help in dis nguishing the images.

4.4 Internet of Things

Internet of Things can be defined as a network of devices called 'things' connected and controlled via the internet. The connec on can be local or global.

The 'things' are mainly analogue data sources. They can be simple devices such as smartphones, smartwatch or even sophis cated such as machines, tools, cars, clothes, people, animals, buildings, etc.

'Things' are connected to data acquisi on systems which store the sensor data. These systems transmit the data through Internet Gateways. The processing of this data must be done so that it is 'cleaned' and only desired informa on is kept for records. Finally, the processed data is stored in Data Centre also known as 'Cloud'. This architecture helps in loading off the processing burden on the 'things'.

All the above steps are managed by an Analy cs Management Control system. It helps visualize the data being transmi ed from each stage.

The sensors/actuators can be controlled by using a network known as Wireless Sensor Network (WSN). This type of network is created by allowing each of the things to have an independent wireless adapter.

The sensors are connected in mesh like structure such that the informa on can easily travel in P2P (peer-to-peer) manner. The sensor is then known as sensor node. Sensor nodes are connected to rou ng nodes which are basically routers. This is analogous to the internet connec on supplied by a service provider in an apartment. Each tower can share a common router cable connec on line. The informa on travels through router nodes in a way similar to the routers of each flat, the difference being that in case of the la er the point of origin is same whereas the former is flexible in terms of the path through which the informa on passes.

The above architecture was implemented by configuring Raspberry Pi board as the router node and the sensors connected to it as sensor nodes. No Gateways were configured specifically as a dedicated API was used to program the no fica on service.

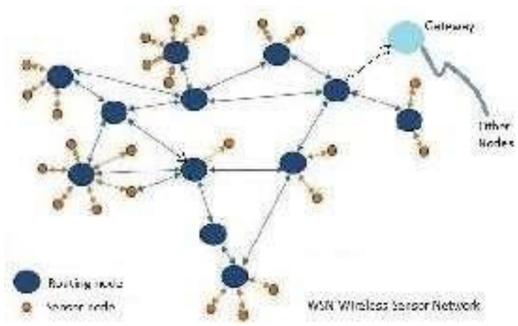

Figure 4.9: Wireless Sensor Network

IFTTT is an applica on which helps connec ng different apps and services together. It eliminates the need of any other expensive communica on devices. The Webhooks service is used in the program to read a sensor value from Raspberry Pi and push a no fica on using a POST request.

Internet of Things can also be implemented for sending images of an intruder captured in the cameraâA˘ Zs live feed to WhatsApp by a service called Twilio. This service helps´ in automa ng chat messages. It is a convenient way to redirect messages to the owner. The sensor data can also be sent and an event can be triggered. In this system, the event (alert) gets triggered when the sensor value reads a high signal value.

The alerts can be triggered to call the police if the person tries to break into the house. The cyber security aspects become pivotal in determining the effec veness of the system. No system is fail-safe at some point. Hence, to avoid a acks which can be used to spoof the access control and the manipula on of sensor values, mul factor authen ca on is proposed as a preven ve measure.

The systems must be provided with an -DDoS so ware modules. This however requires years of experience in Cyber Security and is beyond the scope of this project. The modules are expensive and hence were not incorporated for this scaled down model of the system. However, proposed system can incorporate this feature, given that the required infrastructure consis ng of hardware and addi onal costs is in place.

CHAPTER 5

CODING AND TESTING

5.1 Requirement Analysis

The primary objec ves of the system are:

- Perform Face Recogni on using higher accuracy models
- Detect weapons in video stream
- Send mely alerts triggered by sensors
- Detect tampering of the system

The evalua on of the system was done by using simple metrics such as speed, latency, accuracy, etc. All the parameters considered depend highly on the methodology followed.

5.2 Tes ng and Benchmarks

5.2.1 Face Recogni on Model

The Face Recogni on model uses deep metric learning present in Dlib C++ Library. The model reached an accuracy of 99.38% on the Labelled Faces in the Wild (LFW) dataset. This network was trained on 3 million images. It uses a ResNet having 29 convolu onal layers.

The number of frames per second was considered as evalua on metric. The me taken to generate a frame also includes the me required to perform inference on a single frame. The trained model for face recogni on is converted to inference model

Figure 5.1: Authorized person recognized with name as label

Figure 5.2: Unauthorized person labelled as unknown

which can be used with the program. Inference means that the trained model is tested on various test cases and if the performance is sa sfactory then, it is ready for deployment.

The system's performance was evaluated on 3 different working environments.

Table 5.1: Face Recogni on results using face_recogni on by Adam Geitgey

Configura on	Frames per second (inference me inclusive)
Raspberry Pi Model B + Neural Compute S ck 2 (OpenCV 4 + OpenVINO)	0.5
Intel Core i3-4500U CPU	1.03
Nvidia GeForce GTX 1050 Ti GPU	14.75

The above results were obtained as the model was taken directly without any modifica ons in its architecture or inference model. The above results indicate that even though we added Neural Compute S ck 2 hardware accelerator to Raspberry Pi, the performance dropped as the forward propaga on of the network was done through the ARM processor and not the hardware accelerator.

The deep learning model does not perform well on low-end CPUs. The GPU performance is as expected as the forward propaga on is much faster in an Nvidia GPU with CUDA support.

In order to improve speed another model called DeepFace was adopted. It is present under Caffe Model Zoo API and is an implementa on of the paper Wen, Yandong, et al. "A discrimina ve feature learning approach for deep face recogni on." European conference on computer vision. Springer, Cham, 2016.

Figure 5.3: Tes ng DeepFace model as an alterna ve

The model was tested for all 3 configura ons. It was observed that it could handle non-frontal faces. It was not a highly accurate model and required retraining. The speed was enhanced over 10 mes as both face detec on and embedding (face descriptor) models were defined using OpenCVâA˘ Zs dnn module. This could easily take advantage´ of the hardware accelerator. The performance results improved due to which the video frames appeared as real- me.

Table 5.2: Face Recogni on results using DeepFace model

Configura on	Frames per second (inference me inclusive)
Raspberry Pi Model B + Neural Compute S ck 2 (OpenCV 4 + OpenVINO)	4.7
Intel Core i3-4500U CPU	3.51

Nvidia GeForce GTX 1050 Ti GPU	21

However, this model had a considerable amount of false posi ves and false nega ves during tes ng. Hence, the precision and recall values were low for this model. The custom dataset used for genera ng the face embedding were same.

DeepFace model had face embedding trained on an SVM which was used to generate probabili es for a match between faces and name label. This is not effec ve when compared to the Dlib Library's face descriptor which uses exact face measurements and computes the feature vector using sta s cal approach instead of a probabilis c model.

Neither of the models above fall under ’sweet spot’ in speed vs accuracy trade off. The acceptable solu on was to put accuracy as a priority. Hence, the face descriptor from Dlib was chosen as the final model to be deployed.

5.2.2 Weapon Detec on Model

Weapon Detec on uses the Re naNet object detec on model adapted in Keras library. It is an extension to SSD-ResNet model with an addi onal convolu onal neural network called Feature Pyramid Network as backbone which is used for bounding box

regression.

The performance was poor for this model. This model architecture being computa onally intensive and with a mAP score of 32 on the COCO dataset requires high performance GPUs with lots of memory to give faster throughput results. It is a highly accurate model. However, there were difficul es during classifica on for various orienta on of the same object. This could be improved with retraining.

Table 5.3: Weapon Detec on results using keras-re nanet model

Configura on	Frames per second (inference me inclusive)

Raspberry Pi Model B + Neural Compute S ck 2 (OpenCV 4 + OpenVINO)	No Support
Intel Core i3-4500U CPU	0.07
Nvidia GeForce GTX 1050 Ti GPU	0.24

The model is officially not supported to run Raspberry Pi and hence no results could be obtained for the same.

The solu on for increasing CPU performance is to use OpenVINO library and covert the model into intermediate representa on file which can be used across various plugins.

The OpenVINO library supports high end VPUs for image processing. It is expensive to deploy in a scaled down model. Hence, there were no further experiments conducted in this regard.

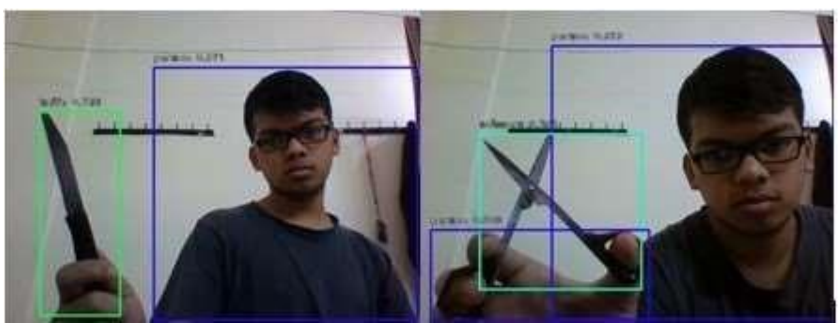

Figure 5.4: Weapons detected and labelled with their names

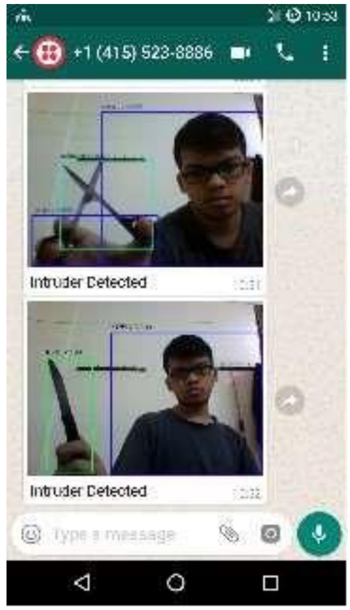

Figure 5.5: No fica on alerts received via Twilio API

5.2.3 Sensor Module

There were three sensors namely- PIR sensor, Vibra on Sensor, Door Magnet sensor. These were tested rigorously for sending alerts with low latency. For the purpose of demonstra on, their range and sensi vity were adjusted. The range was reduced whereas sensi vity was kept high. This allowed for faster tes ng.

The IFTTT app was used to receive no fica ons in the app itself as it was the most reliable way. It is also configurable for SMS alerts.

The alerts were slow during ini al development due to the use of Arduino IDE for sending no fica ons. Later, with the help of Python requests module, the POST method was used and it gave very quick no fica ons.

Figure 5.6: Working of Sensor Module

Figure 5.7: No fica on alerts received using IFTTT app

The system was also able to detect if there is a circuit break using Wire Break Detec on which is placed on the grill of the windows.

Hence, all the objec ves were successfully met by the system.

PROGRAMMING

Project Structure

```
Face Access
|---------dataset
|    |--------Person1
|    |      |-------P1_image1.jpg
|    |      |-------P1_image2.jpg
|    |      |-------.....
|    |--------Person2
|    |      |-------P2_image1.jpg
|    |      |-------P2_image2.jpg
|    |      |-------.....
|    .      .
|    .      .
|----------face_detection_model
|    |------------deploy.prototxt
|    |------------res10_300x300_ssd_iter_140000.caffemodel
|----------encode_faces.py
|----------encodings.pickle
|----------recognition.py

Weapon Detection
|----------weapon.py
|----------resnet50_coco_best_v2.1.0.h5
Notification
|----------iot.py
```

Program 1: encode_faces.py

```python
# USAGE
# When encoding on laptop, desktop, or GPU (slower, more accurate):
# python encode_faces.py --dataset dataset --encodings encodings.pickle --
detection-method cnn
# When encoding on Raspberry Pi (faster, more accurate):
# python encode_faces.py --dataset dataset --encodings encodings.pickle --
detection-method hog

# import the necessary packages
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
    help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
    help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
    help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# grab the paths to the input images in our dataset
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # load the input image and convert it from RGB (OpenCV ordering)
    # to dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
    # corresponding to each face in the input image
    boxes = face_recognition.face_locations(rgb,
        model=args["detection_method"])

    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(rgb, boxes, num_jitters=100)

    # loop over the encodings
    for encoding in encodings:
        # add each encoding + name to our set of known names and
        # encodings
        knownEncodings.append(encoding)
        knownNames.append(name)

# dump the facial encodings + names to disk
```

```
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
f.close()
```

Program 2: recognition.py

```
# USAGE
# python3 recognition.py --detector face_detection_model --
encodings encodings.pickle

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import RPi.GPIO as GPIO
import numpy as np
import argparse
import imutils
import pickle
import time
import cv2
import os


# Servo init
servoPIN = 25
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)

p = GPIO.PWM(servoPIN, 50)
p.start(2.5) # Initialization

process_this_frame = True

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--detector", required=True,
    help="path to OpenCV's deep learning face detector")
ap.add_argument("-e", "--encodings", required=True,
    help="path to serialized db of facial encodings")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
detector.setPreferableTarget(cv2.dnn.DNN_TARGET_MYRIAD)

# load the known faces and embeddings
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())

# initialize the video stream, then allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(usePiCamera=True).start()
```

```python
time.sleep(2.0)

# loop over frames from the video file stream
while True:
    # start the FPS throughput estimator
    fps = FPS().start()
    # grab the frame from the threaded video stream
    frame = vs.read()

    # resize the frame to have a width of 600 pixels (while
    # maintaining the aspect ratio), and then grab the image
    # dimensions
    frame = imutils.resize(frame, width=600)
    (h, w) = frame.shape[:2]

    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(frame, (300, 300)), 1.0, (300, 300),
        (104.0, 177.0, 123.0), swapRB=False, crop=False)

    # apply OpenCV's deep learning-based face detector to localize
    # faces in the input image
    detector.setInput(imageBlob)
    detections = detector.forward()
    rects = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections
        if confidence > args["confidence"]:
            # compute the (x, y)-coordinates of the bounding box for
            # the face
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

            #append coordinates to rects list
            rects.append(box.astype("int"))

    # OpenCV returns bounding box coordinates in (x, y, w, h) order
    # but we need them in (top, right, bottom, left) order, so we
    # need to do a bit of reordering
    boxes = [(t, r, b, l) for (l, t, r, b) in rects]

    if process_this_frame:
        # compute the facial embeddings for each face bounding box
        encodings = face_recognition.face_encodings(frame, boxes)
        names = []

        # loop over the facial embeddings
        for encoding in encodings:
            # attempt to match each face in the input image to our known
            # encodings
            matches = face_recognition.compare_faces(data["encodings"],
                encoding, tolerance=0.4)
            name = "Unknown"

            # check to see if we have found a match
            if True in matches:
                # find the indexes of all matched faces then initialize a
                # dictionary to count the total number of times each face
                # was matched
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}
```

```python

                # loop over the matched indexes and maintain a count for
                # each recognized face face
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1

                # determine the recognized face with the largest number
                # of votes (note: in the event of an unlikely tie Python
                # will select first entry in the dictionary)
                name = max(counts, key=counts.get)

            # update the list of names
            names.append(name)

    process_this_frame = not process_this_frame

    # loop over authorised names and give access through servo actuator
    for name in names:
        if name != 'Unknown':
            p.ChangeDutyCycle(12.5)
            time.sleep(1.5)
            print("Hi, "+name)
            p.ChangeDutyCycle(2.5)
            time.sleep(0.5)

    # update the FPS counter
    fps.update()

    # stop the FPS counter
    fps.stop()

    # loop over the recognized faces
    for ((top, right, bottom, left), name) in zip(boxes, names):
        y = top - 10 if top - 10 > 10 else top + 10
        cv2.rectangle(frame, (left, top), (right, bottom),
            (0, 0, 255), 3)
        cv2.rectangle(frame, (left-2, y-15), (left+80, y+10),
            (0, 0, 255), -1)
        cv2.putText(frame, name, (left+2, y),
            cv2.FONT_HERSHEY_SIMPLEX, 0.55, (255, 255, 255), 1)

    text = "{:.2f} FPS".format(fps.fps())
    cv2.putText(frame, text, (w-80,20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 255), 1)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# stop the timer and display FPS information
#fps.stop()
#print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
#print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Program 3: weapon.py

```
# USAGE
# python weapon.py

import cloudinary
from cloudinary.uploader import upload
from cloudinary.api import delete_resources_by_tag, resources_by_tag
from cloudinary.utils import cloudinary_url

from imutils.video import VideoStream
from imutils.video import FPS
import imutils

# import keras
import keras

# import keras_retinanet
from keras_retinanet import models
from keras_retinanet.utils.image import read_image_bgr, preprocess_image, resi
ze_image
from keras_retinanet.utils.visualization import draw_box, draw_caption
from keras_retinanet.utils.colors import label_color

# import miscellaneous modules

import cv2

import numpy as np
import time

# set tf backend to allow memory to grow, instead of claiming everything
import tensorflow as tf

from twilio.rest import Client

# Account Sid and Auth Token from twilio.com/console
account_sid = 'AC7bc49744715cf96745557735635 1a027'
auth_token = '097a78044fa4fcd7132727133b26193c'
client = Client(account_sid, auth_token)

# Account Key and Secret from cloudinary.com/console
cloudinary.config(
  cloud_name = 'diabloesh',
  api_key = '534458418544597',
  api_secret = '0-AwrffWoiLjPmlL-cDLgPxXDQ'
)

def get_session():
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    return tf.Session(config=config)

# use this environment flag to change which GPU to use
#os.environ["CUDA_VISIBLE_DEVICES"] = "1"

# set the modified tf session as backend in keras
keras.backend.tensorflow_backend.set_session(get_session())

# adjust this to point to downloaded/trained model
# models can be downloaded here: https://github.com/fizyr/keras-
retinanet/releases
model_path = 'resnet50_coco_best_v2.1.0.h5'

```

```
# load retinanet model
model = models.load_model(model_path, backbone_name='resnet50')

# if the model is not converted to an inference model, use the line below
# see: https://github.com/fizyr/keras-retinanet#converting-a-training-model-to-inference-model
#model = models.convert_model(model)

#print(model.summary())

# load label to names mapping for visualization purposes
labels_to_names = {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat',
9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog',
17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21: 'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella',
26: 'handbag', 27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat',
35: 'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis racket', 39: 'bottle', 40: 'wine glass', 41: 'cup', 42: 'fork',
 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange', 50: 'broccoli', 51: 'carrot',
 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table',
 61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven',
 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear',
 78: 'hair drier', 79: 'toothbrush'}


# initialize the video stream, then allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# start the FPS throughput estimator
fps = FPS().start()

while True:
    frame = vs.read()

    draw = frame.copy()
    draw = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)

    # preprocess image for network
    frame = preprocess_image(frame)
    frame, scale = resize_image(frame)

      # process image
      start = time.time()
      boxes, scores, labels = model.predict_on_batch(np.expand_dims(frame, axis=0))
      print("processing time: ", time.time() - start)

      # correct for image scale
      boxes /= scale

      # visualize detections
      for box, score, label in zip(boxes[0], scores[0], labels[0]):
          # scores are sorted so we can break
          if score < 0.5:
              break

          color = label_color(label)
```

```python

            b = box.astype(int)
            draw_box(draw, b, color=color)

            caption = "{} {:.3f}".format(labels_to_names[label], score)
            draw_caption(draw, b, caption)

            # to check if weapon is detected and send notifications in WhatsApp

            # considering only scissors and knife
            if label == 43 or label == 76:
                draw = cv2.cvtColor(draw, cv2.COLOR_RGB2BGR)
                cv2.imwrite("out.jpg",draw)
                cloudinary.uploader.upload("out.jpg", public_id="intruder", tags="uploaded")
                cloudinary.utils.cloudinary_url("intruder.jpg")
                message = client.messages.create(
                                    media_url = 'http://res.cloudinary.com/diabloesh/image/upload/intruder.jpg',
                                    body= 'Intruder Detected',
                                    from_='whatsapp:+14155238886',
                                    to='whatsapp:+917601837265'
                                   )
                delete_resources_by_tag("uploaded")
                draw = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)

        draw = cv2.cvtColor(draw, cv2.COLOR_RGB2BGR)
        cv2.imshow("Frame", draw)

        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

        # update the FPS counter
        fps.update()

    # stop the FPS counter
    fps.stop()
    print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

    # do a bit of cleanup
    cv2.destroyAllWindows()
    vs.stop()
```

Program 4: iot.py

```python
# Imports
import RPi.GPIO as GPIO
import time
import requests

# Set the GPIO naming convention
GPIO.setmode(GPIO.BCM)

# Turn off GPIO warnings
GPIO.setwarnings(False)

# Set a variable to hold the GPIO Pin identity
```

```
pinpir = 24
pinmag = 23
pinvib = 9

# Set GPIO pin as input
GPIO.setup(pinpir, GPIO.IN)
GPIO.setup(pinmag, GPIO.IN)
GPIO.setup(pinvib, GPIO.IN)

# Variables to hold the current and last states
currentstate = [0,0,0]
previousstate = [0,0,0]

try:
    print("Waiting for sensors to settle ...")

    # Loop until all sensors' output is 0
    while GPIO.input(pinpir) == 1 or GPIO.input(pinmag) == 1 or GPIO.input(pinvib) == 1:

        currentstate[0] = 0
        currentstate[1] = 0
        currentstate[2] = 0

    print("    Ready")

    # Loop until users quits with CTRL-C
    while True:

        # Read PIR state
        currentstate[0] = GPIO.input(pinpir)
        currentstate[1] = GPIO.input(pinmag)
        currentstate[2] = GPIO.input(pinvib)

        # If the PIR sensor is triggered
        if currentstate[0] == 1 and previousstate[0] == 0:
            print("PIR Detected")

            # IFTTT URL with event name, key and json parameters (values)
            r = requests.post('https://maker.ifttt.com/trigger/sens_1/with/key/nwKa2db3VvfkdHE8kM6hU47RJkYkPmXXDq3KO4aF1sm', params={"value1":"none","value2":"none","value3":"none"})

            # Record new previous state
            previousstate[0] = 1

            time.sleep(1)

        # If the PIR sensor has returned to ready state
        elif currentstate[0] == 0 and previousstate[0] == 1:

            print("Ready")
            previousstate[0] = 0

        # If the Door Magnet sensor is triggered
        if currentstate[1] == 1 and previousstate[1] == 0:
            print("Door Magnet detached")

            # IFTTT URL with event name, key and json parameters (values)
            r = requests.post('https://maker.ifttt.com/trigger/sens_2/with/key/nwKa2db3VvfkdHE8kM6hU47RJkYkPmXXDq3KO4aF1sm', params={"value1":"none","value2":"none","value3":"none"})

            # Record new previous state
            previousstate[1] = 1

```

```
            time.sleep(1)

        # If the Door Magnet sensor has returned to ready state
        elif currentstate[1] == 0 and previousstate[1] == 1:

            print("Ready")
            previousstate[1] = 0

        # If the Vibration sensor is triggered
        if currentstate[2] == 1 and previousstate[2] == 0:
            print("Vibration detected")

            # IFTTT URL with event name, key and json parameters (values)
            r = requests.post('https://maker.ifttt.com/trigger/1234/with/key/n
wKa2db3VvFkdHtBkMhbU47RJkYkPmXXOqDKO8aF1om', params={"value1":"none","value2":
"none","value3":"none"})

            # Record new previous state
            previousstate[2] = 1

            time.sleep(5)

        # If the Vibration sensor has returned to ready state
        elif currentstate[2] == 0 and previousstate[2] == 1:

            print("Ready")
            previousstate[2] = 0

        # Wait for 10 milliseconds
        time.sleep(0.01)

except KeyboardInterrupt:
    print("    Quit")

    # Reset GPIO settings
    GPIO.cleanup()
```

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.0.1 Conclusion

The system developed in this work proposes a simplified way to handle access control and intrusion detec on in smart homes. The use of face recogni on for access control is an efficient way for this environment. The sensors used are capable of providing instant alerts, so that the owner is never le to unforeseen circumstances. This project clearly shows how state-of-the-art methods can be used even with limita ons on hardware. The design of the system for achieving the âĂŸsweet spotâ˘ A˘ Z between speed and perfor- ´ mance is an itera ve process. It requires further inves ga on with customized models and training on large datasets. Therefore, the best performance can be achieved by redefining image processing algorithms and not using off-the-shelf models.

One of the important highlights of the system is weapon detec on using image processing. The primary advantage of it is that it helps avoid expenditure on any physical components, thus reducing costs significantly. However, it is an experimental feature which cannot be incorporated into a single model alongside face recogni on. This method promises to improve the detec on rates of a poten al intruder and improve the overall efficiency of the access control system.

The system also implements a fail-safe method by having extra components powered by electric power backup in case there is a circuit failure. The tampering of the circuit is also detectable, which makes it suitable for emergencies. With all the above features, the system is apt for use in modern homes and forms an essen al avenue for building smart ci es.

The system can be further used in other industrial areas such as logis cs, schools and ATM facili es for improving security. This will reduce the amount of me taken to iden fy threats to the buildings where the system is employed.

6.1 Future Scope

Even though, the final design was found as sa sfactory, it was realized there is always scope for improvement. The system is not en rely break-proof in spite of providing failsafe method.

Some of the aspects worth no ng for future improvements are as follows-

- Liveness Detec on System: During the process of development of Face Recogni on system, it was observed that the system can be spoofed by using a photo of the owner for access. Hence, a liveness detec on system was incorporated during the tes ng of the system. The system is able to dis nguish faces as valid or invalid based on the whether the captured face is from a phone or the person himself. However, the disadvantage of the system was that it doesnâA˘ Zt generalise´ well on all faces. Actual methods also include detec on eyes blinking, mo on sensing, how pixels change and 3D depth sensing. Depth sensing required cameras with depth sensors for determining if the face is a 2D image or a 3D solid object. Due to lack of suitable hardware and low recall of the method, it was discarded in the final design of the system. Nonetheless, it forms a crucial part in improving the system.

- Non-Frontal Face Detec on: Only front faces were detected in the final model. The alterna ve model using DeepFace could allow for non-frontal faces as well. The descriptor of the current model can be combined with it to achieve recogni on of non-frontal faces with very low latency.

- System Performance: The lower performance of the system can be compensated by using dedicated GPUs present in embedded pla orms like Nvidia Jetson boards and Google Coral TPU board.

- Camera Tampering Detec on: It can be created by using image processing and predic ng by rate at which frames change.

- Indoor Surveillance: Intrusion and access control methods provided above can be also used with indoor surveillance where the cameras are used for monitoring infants in case parents are away and check if pets are safe.

- Fire alarm: The system can be given an extension to alert the owner if a fire was set in the house when they are away. This could help in immediate evacua on.

- Sensor Nodes: In order to make sensors portable and accessible throughout the globe, they can be assigned with individual IPs for their wireless and 4G LTE adapters. This could help in accessing their data globally.

- Drone surveillance: The above system can be incorporated in drones for close quarter surveillance opera ons and iden fica on of people from database.

REFERENCES

. Dubal, P., Mahadev, R., Kothawade, S., Dargan, K., and Iyer, R. (2018). “Deployment of customized deep learning based video analy cs on surveillance cameras.” arXiv preprint arXiv:1805.10604.

. Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). “A fast learning algorithm for deep belief nets.” Neural computa on, 18(7), 1527–1554.

. Kazemi, V. and Sullivan, J. (2014). “One millisecond face alignment with an ensemble of regression trees.” Proceedings of the IEEE conference on computer vision and pa ern recogni on. 1867–1874.

. Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). “Focal loss for dense object detec on.” Proceedings of the IEEE interna onal conference on computer vision. 2980– 2988.

. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). “Ssd: Single shot mul box detector.” European conference on computer vision, Springer. 21–37.

. Mao, J., Lin, Q., and Bian, J. (2018). “Applica on of learning algorithms in smart home iot system security.” Mathema cal Founda ons of Compu ng, 1(1), 63–76.

. Parkhi, O. M., Vedaldi, A., Zisserman, A., et al. (2015). “Deep face recogni on..” bmvc, Vol. 1. 6.

. Schroff, F., Kalenichenko, D., and Philbin, J. (2015). “Facenet: A unified embedding for face recogni on and clustering.” Proceedings of the IEEE conference on computer vision and pa ern recogni on. 815–823.

. Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). “A discrimina ve feature learning approach for deep face recogni on.” European conference on computer vision, Springer. 499– 515.