

REAL TIME SECURITY MANAGEMENT IN SMART HOME ENVIRONMENT

Submitted in partial fulfilment of the requirements for the award of Bachelor of
Engineering degree in Computer Science and Engineering

By

P.sai surya teja (38110485,)AND

venkat nagendra (38110636)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide
work of P.sai surya teja (38110485,)AND
venkat nagendra (38110636)
who carried out the project entitled "REAL TIME SECURITY MANAGEMENT IN
SMART HOME ENVIRONMENT" under our supervision from NOVEMBER 2019 to
APRIL 2021.

INTERNAL GUIDE

Dr. Bharathi., Ph.D.,

HEAD OF THE DEPARTMENT

Dr. S. VIGNESHWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

**We, P.sai surya teja (38110485,)AND
venkat nagendra (38110636)
hereby declare that the Project Report entitled**

REAL TIME SECURITY MANAGEMENT IN SMART HOME ENVIRONMENT done by
us under the guidance of **Dr.Bharathy M.E., Ph.D.**, is submitted in partial fulfilment of
the requirements for the award of **Bachelor of Engineering** degree in **Computer
Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGMENT

We are pleased to acknowledge my sincere thanks to the Board of **Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean**, School of Computing and **Dr. L. LAKSHMANAN M.E., Ph.D.,** and **Dr. VIGNESHWARI M.E., Ph.D., Head of the Department, Department of Computer Science And Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. Bharathi M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

To design and implement a distributed Internet of Things (IoT) framework called IoT-guard, for an intelligent, resource-efficient, and real-time security management system. The system, consisting of edge-fog computational layers, will aid in crime prevention and predict crime events in a smart home environment (SHE). The IoT-guard will detect and confirm crime events in real-time, using Artificial Intelligence (AI) and an event-driven approach to send crime data to protective services and police units enabling immediate action while conserving resources, such as energy, bandwidth (BW), and memory and Central Processing Unit (CPU) usage. In this study, we implement an IoT-guard laboratory testbed prototype and perform evaluations on its efficiency for real-time security application. The outcomes show better performance by the proposed system in terms of resource efficiency, agility, and scalability over the traditional IoT surveillance systems and state-of-the-art (SoA) approaches.

TABLE OF CONTENTS

ABSTRACT

i

LIST OF FIGURES

v

CHAPTER	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 OBJECTIVE	2
	1.2 NEED FOR SMART HOME ENVIRONMENT	3
2	LITERATURE SURVEY	4
3	SYSTEM ANALYSIS	6
	3.1 EXISTING SYSTEM	6
	3.2 PROPOSED SYSTEM	6
	3.3 SYSTEM ARCHITECTURE	7
	3.4 LIST OF MODULES	8
	3.4.1 USER AUTHENTICATION WITH SURVEILLANCE SYSTEM	9
	3.4.2 OBJECT DETECTION WITH EDGE NODE	9
	3.4.3 FOG SERVER COMPUTATION	10
	3.4.4 SURVEILLANCE ALERT MECHANISM	11
4	RESULTS AND DISCUSSIONS	14

	4.1 DESIGN AND IMPLEMENTATION CONSTRAINTS	14
	4.1.1 CONSTRAINTS IN ANALYSIS	14
	4.1.2 CONSTRAINTS IN DESIGN	14
	4.1.3 CONSTRAINTS IN IMPLEMENTATION	14
	4.2 OTHER NON - FUNCTIONAL REQUIREMENTS	15
	4.2.1 PERFORMANCE REQUIREMENTS	15
	4.2.2 SAFETY REQUIREMENTS	15
	4.3 HARDWARE AND SOFTWARE REQUIREMENTS	16
	4.3.1 HARDWARE REQUIREMENTS	16
	4.3.2 SOFTWARE REQUIREMENTS	16
	4.4 ALGORITHMS USED	16
	4.4.1 REGIONAL BASED CONVOLUTIONAL NEURAL NETWORK	16
	4.5 FRAMEWORKS	17
	4.5.1 PYTHON	17
	4.5.2 IoT	17
5	CONCLUSION AND FUTURE ENHANCEMENTS	21

	5.1 CONCLUSION	21
	5.2 FUTURE ENHANCEMENTS	21
	REFERENCES	22
	APPENDICES	
	A. SOURCE CODE	27
	B. SCREENSHOTS	43
	C. PUBLICATION WITH PLAGIARISM REPORT	45

LIST OF FIGURES

FIGURE NO	NAME	PAGE NO
3.1	SYSTEM ARCHITECTURE	8
3.2	USER AUTHENTICATION MECHANISM	9
3.3	SURVEILLANCE ALERT MECHANISM	12
4.1	REGIONAL BASED CONVOLUTIONAL NEURAL NETWORK	16
4.2	APPLICATIONS OF IoT	19

CHAPTER 1

INTRODUCTION

A smart home environment (SHE) consists of different applications of ubiquitous computing that integrates smartness into dwellings for comfort, healthcare, safety, security, and energy conservation. An SHE is monitored by ambient intelligence to provide context-aware services and to facilitate safety and security management. A security management system is designed to provide complete safety from robbery, sabotage, and intrusion by monitoring the internal and external SHE, using surveillance cameras. Various cyber-physical systems widely adopt the use of intelligent video surveillance (IVS), for automatic and accurate identification of events and objects in a target scene. IVS enables video analytics to predict and interpret the activity of a scenario without human intervention. Meanwhile, with the development of artificial intelligence (AI) and machine learning (ML), surveillance applications and security procedures are being improved with enhanced functions and According to the Uniform Crime Reports published by the Federal Bureau of Investigation (FBI), the 2017 statistics show that in USA, burglaries of residential properties accounted for 67.2 percent of all burglary offenses, and the victims of these offenses suffered an estimated \$3.4 billion in property losses. In addition, 15.5 percent of all robberies in 2017 occurred at commercial properties while residences experienced 16 percent. Due to this rise in property crime, the research community is paying attention to smart home security Protective services and authorities often fail to respond to crime incidents efficiently. They tend to follow a reactive approach which relies mostly on witness reports or closed circuit television (CCTV) footage after the crime takes place. Therefore, in most cases, when an event occurs, authorities visit the location of the incident, retrieve the content manually from the camera, and then proceed to identify relevant footage either by watching the full length of the video or by processing it through specialized video analytics algorithms. Thus reactive approach is naturally inefficient for preventing crimes. An efficient crime predictive system could enable robust security management in an SHE by identifying preventative procedures. Thus, the authorities could reduce

crime incidents and losses. In addition, modern multimedia surveillance systems comprise a wide range of sensors, distributed over multiple sites. The video surveillance system in an SHE consists of many cameras that can produce a large amount of surveillance data, both photo and video. This may result in heavy network congestion and impose complicated processing load on individual devices and systems. In this paper, we will discuss an IoT-integrated intelligent video surveillance framework to provide an effective solution to this problem. The internet of things (IoT) is the internetworking of physical objects, virtual objects, living beings, analytics, user interfaces, and network connectivity that allows these objects to collect and exchange data over an internet-based infrastructure. This internetworking enables advanced IoT applications (e.g., environment monitoring, smart-city, intelligent transportation, healthcare, surveillance, and smart homes). The IoT opens the door to advanced innovations to facilitate modern interactions and provides new opportunities for infrastructures and services that improve the quality of life. Hence, an IoT-based smart-surveillance system can be adapted to reduce the crime rate, especially in a smart building. Although cloud-based IoT architectures are used for processing and storing essential surveillance data, they have issues regarding bandwidth and latency-sensitive video surveillance applications which require IoT nodes near the source of visual data to meet their delay requirements. Fog computing has been introduced to address these issues.

1.1 OBJECTIVE :-

To Design a Home with a Smart Home Environment which protects the people in the home from any kind of robbery, crime, sabotage. Which safeguards by monitoring the events and the moments happening.

1. To protect home from any kind of crime activities.
2. To provide secure environment to the people living in the home.

1.2 NEED FOR SMART HOME ENVIRONMENT :-

A security management system is designed to provide complete safety from robbery, sabotage, and intrusion by monitoring the internal and external SHE, using surveillance cameras. Various cyber-physical systems widely adopt the use of intelligent video surveillance (IVS), for automatic and accurate identification of events and objects in a target scene. IVS enables video analytics to predict and interpret the activity of a scenario without human intervention. Meanwhile, with the development of artificial intelligence (AI) and machine learning (ML), surveillance applications and security procedures are being improved with enhanced functions and accuracy.

CHAPTER 2

LITERATURE SURVEY

- **Shih (2014)** developed an occupancy detection and tracking system for automatic monitoring and commissioning of a building with the help of an image-based depth sensor and a programmable pan tilt-zoom camera.
- **Yang et al. (2018)** developed a device free occupant-activity sensing system using Wi-Fi - enabled IoT devices for smart homes.
- **Hossain (2014)** proposed a framework for a cloud-based multimedia surveillance system that supports the processing overload, storage requirements, access, security, and privacy in large-scale surveillance settings. These studies reveal the capability of cloud computing to satisfy many IoT requirements (e.g., monitoring, sensor stream processing, and visualization tasks). However, the large amount of real-time media data sent by the end devices using high-speed fiber networks leads to a high network deployment cost. Although the situation has changed in recent years with the internetworking ability of IoT, still IoT cloud architecture has issues regarding bandwidth, energy, and latency in real-time video surveillance applications.
- **Ni et al. (2018)** explained the architecture, features, and role of fog computing. Distributed and efficient object-detection architecture in edge computing for real-time surveillance application is also proposed.
- **Li et al. (2018)** showed the design of a novel offloading strategy to optimize IoT DL applications with an edge-computing environment.
- **Cao et al. (2017)** described the design of a self-optimizing, context-driven, and energy-aware IoT wireless video sensor node for surveillance applications.
- **A.J.V. Neto et al. (2018)** proposed a fog framework for intelligent video surveillance to enhance crime assistance and safety in public transportation.
- **Fan et al. (2017)** described a novel visualization mechanism which fuses multimodal information for large-scale intelligent video surveillance, utilizing an event-driven approach. Some architectures and frameworks for event-driven video surveillance approaches are also described, along with energy-aware, event-driven video surveillance solutions.

- **A.J.V. Neto et al. (2018)** described the critical application requirements of an efficient smart-surveillance system, such as real-time and accurate detection of an event, reliable and agile prediction of crime events, and high-performance service deployment. Resource-efficient approaches add benefits in the IoT-based video surveillance architectures because of the ever-increasing number of surveillance nodes.
- **R. Hasan et al. (2017)** proposed a BW-and energy-aware video compression algorithm for IoT-based video surveillance applications.
- **Lee et al. (2015)** studied an on-road pedestrian tracking system across multiple moving cameras [20] and in another article, developed a technique for vehicle tracking and localization based on 3-D constrained multiple-kernel tracking.
- **Chen et al. (2017)** proposed a quality-of-content-based joint source and channel coding system for detecting humans in a mobile surveillance cloud.
- **Ajiboye et al. (2015)** proposed Fused Video Surveillance Architecture (FVSA) that enhances the public safety by utilizing data from privately-owned cameras.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing system various cyber-physical systems widely adopt the use of intelligent video surveillance, for automatic and accurate identification of events and objects in a target scene. IVS enables video analytics to predict and interpret the activity of a scenario without human intervention. Meanwhile, with the development of artificial intelligence (AI) and machine learning (ML), surveillance applications and security procedures are being improved with enhanced functions and accuracy. Protective services and authorities often fail to respond to crime incidents efficiently. Therefore, in most cases, when an event occurs, authorities visit the location of the incident, retrieve the content manually from the camera, and then proceed to identify relevant footage either by watching the full length of the video or by processing it through specialized video analytics algorithms.

3.1.1 Problem Definition:

- Protective services and authorities often fail to respond to crime incidents efficiently.
- Authorities visit the location of the incident, retrieve the content manually from the camera. Manual investigation will take much time.

3.2 PROPOSED SYSTEM

We are going to implement IoT-guard, an event-driven edge-fog-integrated video surveillance framework, to perform real-time security management by aiding in crime prevention and predicting crime events at an SHE. The proposed IoT-guard approach provides a three layer architectural framework that orchestrates event-driven edge devices in an SHE and DL-implemented fog computing nodes to address increasing human security concerns. The system also provides an alert by sending the crime data instantly to the police or protective service, and thus, it ensures a quick response.

Advantages:

- Detection of crime objects within short time.
- Automatic and accurate identification of crime events.

3.3 SYSTEM ARCHITECTURE

The System architecture is illustrated in Figure 3.1. The Proposed System is a Smart Home Environment in which the surveillance cameras are installed at a place, and these surveillance cameras are connected to the Raspberry – pi kit through which the movement captured and the object detection are processed to the fog server and this fog server performs the algorithm proposed on the objects captured and verifies whether the object is a dangerous one or not if it detects a harmful object then it sends the authentication alert message to the respective authority and alert them through the notification.

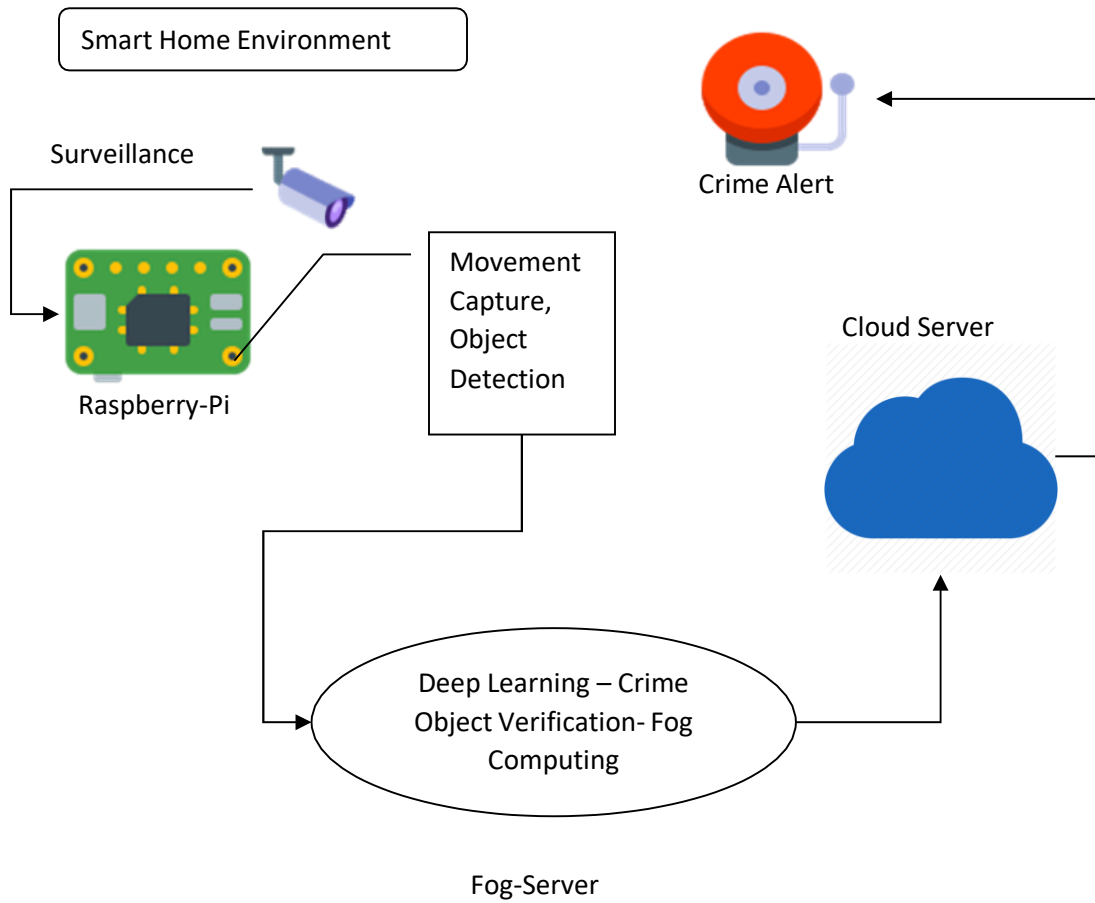


Fig 3.1 System Architecture

3.4 LIST OF MODULES

The Proposed System is composed of the list of modules.

1. User Authentication with Surveillance System
2. Object Detection with Edge Node
3. Fog Server Computation
4. Surveillance Alert Mechanism

3.4.1 USER AUTHENTICATION WITH SURVEILLANCE SYSTEM

The user need to authenticate with the Cloud Server for enabling this security mechanism. The registration of user includes a basic form to get all basic details of the user. This information will be stored in cloud server. So that users can enter into the web portal to see surveillance statistics.

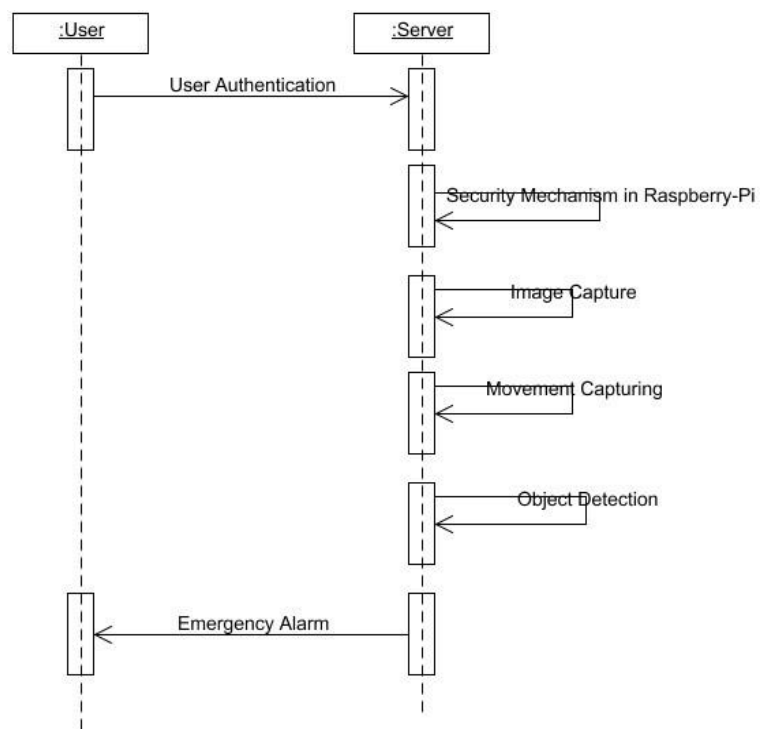


Fig 3.2 User Authentication mechanism

3.4.2 OBJECT DETECTION WITH EDGE NODE

The edge node contains a Raspberry-Pi which will be connected with group of cameras around Smart Home Environment. We are going to interface one camera with the edge node (Raspberry-Pi). When any movement captured in Surveillance camera

then edge node will spot the movement captures the image and analyze if any object is there then send the snap to Fog node for further computing.

The edge node divides its task into three categories: image processing, motion detection, and motion image dispatching. The image-processing functionality finds moving objects in dynamic image sequences by making use of a pixel-based change detection technique. The most intuitive method to detect change is the simple differencing of pixels, followed by thresholding. The detection of pixel level change requires little computational cost. Hence, the system will utilize the pixel-based background subtraction technique for detecting changes or motion in real-time video sequences at a constrained IoT edge node. An advantage of the pixel-based background subtraction technique is that it compensates for the lack of spatial consistency by a constant updating of the model parameters. In addition, the background subtraction method enables robust segmentation by applying a threshold to the individual pixel's difference in subsequent frames, which helps to separate the moving object or foreground from the background.

We implemented this approach at the edge node to identify human motion. We tested the algorithm and set the threshold and sensitivity level for the system to avoid false positive results.

While the edge node continuously scans for motion using the motion detection algorithm, it does not perform any transmission of media data. Whenever it detects motion, it immediately captures motion-detected images and dispatches them to a specific fog node along with its location information. Then, it continues its function of identifying motion as before.

3.4.3 FOG SERVER COMPUTATION

Fog node identifies and confirms the presence of a human and weapon, it will classify the type of weapon and immediately dispatch crime event information to the nearest crime prevention unit instantly. Each fog node is also able to dispatch crime data simultaneously in the form of a mobile phone alert message. Using the crime data sent by the fog node, the crime prevention unit can ensure real-time crime prevention

before the crime actually takes place. The AI enabled event-driven fog node also nullifies any false positive result registered by the edge node. Each crime prevention unit may receive a crime notification from several fog nodes covering a residential area and send them to Cloud Server.

The convolutional neural network (CNN) is implemented by forming layers based on convolution, where input data is convolved to a smaller area, detecting important part within that area. Each of the convolutional layers applies nonlinear activation functions and filters in the order of hundreds to thousands and combine their results to compute the output [17]. A fog node, controlling several edge nodes, receives motion-detected images from them. Using intelligent computational methods, it then applies an object detection algorithm using a pretrained CNN model.

Therefore, we collected gun and knife datasets to train and build a CNN model that is capable of detecting guns and knives. Hence, the CNN model running at a fog node detects and labels the images with the name of the crime objects having the highest probability, and saves those images. The fog node then assembles and sends crime data (i.e., the labeled image, crime event location, and camera position) to the nearest crime assistance or police unit and also sends an alert message to the protective service in real-time. On the other hand, the cloud is responsible for generating updated CNN crime data models (i.e., by using transfer learning methods), so that the fog node can download them whenever they are available.

3.4.4 SURVEILLANCE ALERT MECHANISM

All the fog nodes maintain bidirectional communication with a central cloud server within a smart city for receiving system updates, crime event data mining, statistical analysis, and periodic information storage. Based on the prediction result the alert or notification will be triggered to the represent authority in-order to prevent the crime before it is going to take place.

So with the help of this alert mechanism the respective authority can alert and take the necessary action or can stop the crime event before it happens.

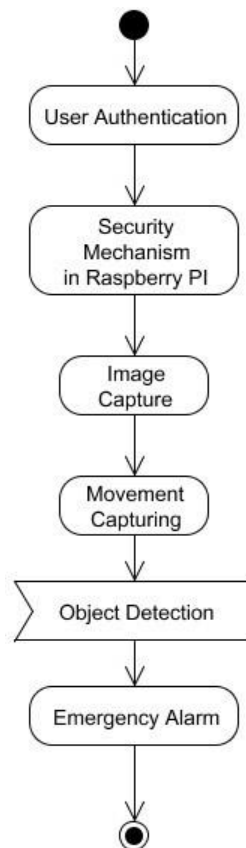


Fig 3.3 Surveillance Alert Mechanism

ANACONDA

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda is a Python-based data processing and scientific computing platform. It has built in many very useful third-party libraries. Installing Anaconda is equivalent to automatically installing Python and some commonly used libraries such as Numpy, Pandas, Scrip, and Matplotlib, so it makes the installation so much easier than regular Python installation. It is painful and you need to consider compatibility, thus it is highly recommended to directly install Anaconda.

IDE Setup

The Machine Learning Concepts can be well implemented through PYTHON. We have a numerous Python Tools but for the DNN implementation Anaconda can meet our needs. Spyder in Anaconda is chosen as the IDE setup. The latest Anaconda Spyder has Python 3.6 version. Python 3.6 is unstable to hold the KERAS backend . Apart from the inability to build over Python 3.6 , KERAS requires two additional library packages THEANO and TENSORFLOW . Tensorflow can be implemented over Python 3.5 only .

A cloning environment is built using the Anaconda Prompt . The Python 3.5 is cloned using the command .

conda create -n py35 python=3.5 anaconda

After activating the cloned environment the Spyder is installed using the command . So the Python3.5 has its own Spyder version , which does not interfere with the base version of Anaconda .

conda install spyder

The required libraries can be set up using the below commands

conda install theano

conda install tensorflow

conda install keras

conda install wordcloud

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Design and Implementation Constraints

4.1.1 Constraints in Analysis

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

4.1.2 Constraints in Design

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses
- Global actions and Constraint Realization

4.1.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.2 Other Nonfunctional Requirements

4.2.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.
- Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.2.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

4.3 HARDWARE AND SOFTWARE SPECIFICATION

4.3.1 HARDWARE REQUIREMENTS

- Hard Disk : 80GB and Above
- RAM : 4GB and Above
- Processor : P IV and Above
- Webcam
- Raspberry-Pi

4.3.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 10 (64 bit)
- Software : Python and Anaconda
- Tools : Jupyter Note Book and Spyder IDE

4.4 ALGORITHMS USED

4.4.1 Region Based Convolutional Neural Network

Recently, the automatic object detection in high-resolution remote sensing images has become the key point in the application of remote sensing technology. The traditional methods, such as bag-of-visual-words (BOVW), could perform well in simple scenes, but when it used in complex scenes, the performance drops quickly.

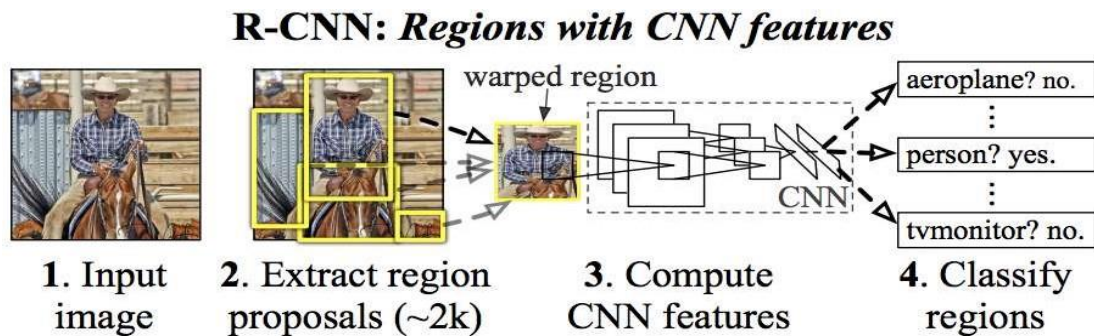


Fig 4.1 Regional based convolutional neural network

4.5 FRAMEWORKS

4.5.1 PYTHON

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

4.5.2 IOT:

IoT refers to an Internet Of Things (IoT). Connecting any device (including everything from cell phones, vehicles, home appliances and other wearable embedded with sensors and actuators) with Internet so that these objects can exchange data with each other on a network. It is interesting to note that there is a difference between IoT and the Internet; it is the absence of Human role. The IoT devices can create information about individual's behaviors, analyze it, and take action (IoT is smarter than Internet: D)

A question would arise in your mind that why we are concerned about IoT? Here is the answer that why you should be concerned about IoT. Say for example you are on your way to a meeting, your car could have access to your calendar and already know the best route to take. If the traffic is heavy your car might send a text to the other party

notifying them that you will be late. What if your alarm clock wakes up you at 6 a.m. and then notifies your coffee maker to start making coffee for you? Being able to turn the lights on in your house or heating before coming home using your smartphone? Yes, all these things are possible because of IoT.

Smart System and the Internet of the Things are driven by a combination for:

1. Sensors & Actuators
2. Connectivity
3. People & Process

INTEROPERABILITY IN IOT

- The Internet of Things (IoT) is an incredibly diverse space, encompassing a large variety of hardware form factors and software ecosystems unlike anything we have seen in technology. Smart watches, connected cameras, drones, thermostats, voice-enabled speakers, smart appliances and more—they all live together within the IoT.
- The diversity and innovation that excites many IoT fans is a big challenge not just for manufacturers and developers, but also (and most importantly) consumers. Which technology options should be used when designing or deploying IoT devices? How do they keep up with updated or new operating systems? What about new software and connectivity technologies coming up? Those are just some of today's challenges.
- Having a single, unified communication and software framework for the IoT seems like an ideal solution, but the diverse and fast-paced nature of the IoT makes this utopia a big challenge. Diversity in the IoT is not something to be solved, but an aspect that must be embraced and managed.

Application areas for the Internet of Things

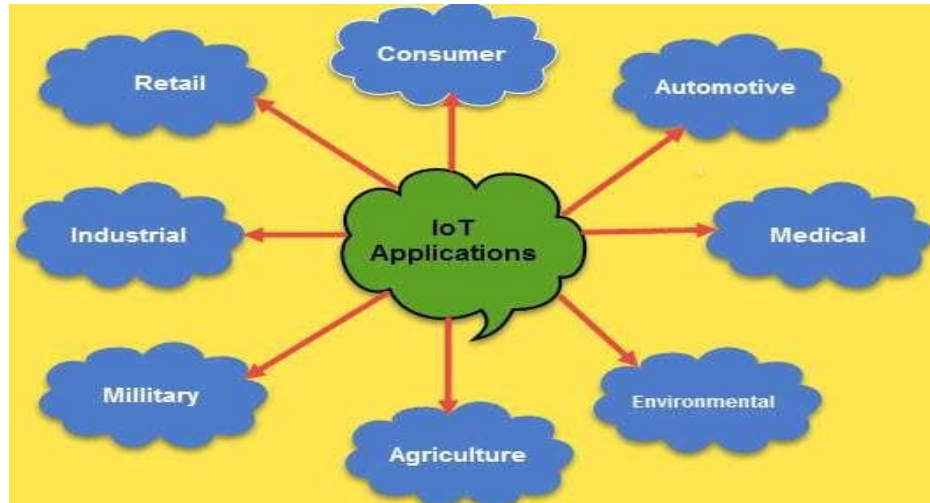


Fig 4.2 Applications of IoT

Smart Home

The concept of Smart Home is brought up to save time, energy and money. With the introduction of Smart Homes, we would be able to switch on air conditioning before reaching home or switch off lights even after leaving home or unlock the doors to friends for temporary access even when you are not at home.

Smart cities

Smart surveillance, automated transportation, smarter energy management systems, water distribution, urban security and environmental monitoring all are examples of internet of things applications for smart cities. IoT will solve major problems faced by the people living in cities like pollution, traffic congestion and shortage of energy supplies etc. By installing sensors and using web applications, citizens can find free available parking slots across the city. Also, the sensors can detect meter

tampering issues, general malfunctions and any installation issues in the electricity system.

Challenges to IoT

In the era of IoT, everything is connected, linked up—much more than we see in and around us. IoT is certainly opening door to a lot of opportunities but also to many challenges.

Security Challenges

Security is a big issue with IoT devices. With billions of devices being connected together over Internet, how can people be sure that their information is secure.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 CONCLUSION

We have presented the design, deployment, and performance evaluation of the IoT-guard, which is an event driven and fog-based smart-surveillance system for real-time crime detection and security management. The application targets security management within a smart home environment under the smart-city paradigm.

We evaluated the suitability and feasibility of the proposed system by deploying a laboratory testbed of the IoT-guard and observed its performance.

We implemented the proposed architecture with and without a video compression algorithm and compare the performance between them along with other IoT-based video surveillance architectures. Our system outperformed others with greater efficiency in terms of energy, bandwidth, and percentage of CPU usage. Although the video compression algorithm helped our proposed IoT-guard architecture to save 20 percent more storage, it reduced its efficiency, notably in terms of energy and percentage of CPU usage. Therefore, the proposed system is far more efficient even without the video compression algorithm.

5.2 FUTURE ENHANCEMENTS

We can upgrade the proposed system in the future by adding other types of crime objects or threat events to the model without changing the system configuration. Moreover, it can be further trained to detect more features in the future, for instance, utilizing transfer learning, and thus enabling it to differentiate between resident members and intruders using facial recognition features. This system could include more intelligence and services in the future for other video surveillance applications by utilizing its efficient workload management ability.

REFERENCES

- [1] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A review of smart homes_Past, present, and future," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1190_1203, Nov. 2012.
- [2] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of Things build our future: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2011_2027, 3rd Quart., 2018.
- [3] V. Gouaillier and A.-E. Fleurant, "Intelligent video surveillance: Promises and challenges," CRIM Technopôle Defence Secur., Montreal, QC, Canada, Technol. Commercial Intell. Rep., Mar. 2009, vol. 456, p. 468. [Online]. Available: <http://docshare04.docshare.tips/files/7051/70518279.pdf>
- [4] T. Sultana, M. W. Alam, and K. A. Wahid, "Reliability analysis of IoT based intelligent video surveillance system," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process. (MMSP)*, Aug. 2018, pp. 1_4.
- [5] S. Din, A. Paul, A. Ahmad, B. B. Gupta, and S. Rho, "Service orchestration of optimizing continuous features in industrial surveillance using big data based fog-enabled Internet of Things," *IEEE Access*, vol. 6, pp. 21582_21591, 2018.
- [6] *FBI_Robbery*. Accessed: Apr. 12, 2019. [Online]. Available: <https://ucr.fbi.gov/crime-in-the-u.s/2017/crime-in-the-u.s.-2017/topic-pages/robbery>
- [7] G. Kioumourtzis, M. Skitsas, N. Zotos, and A. Sideris, "Wide area video surveillance based on edge and fog computing concept," in *Proc. 8th Int. Conf. Inf., Intell., Syst. Appl. (IISA)*, Aug. 2017, pp. 1_6.
- [8] A. J. V. Neto, Z. Zhao, J. J. P. C. Rodrigues, H. B. Camboim, and T. Braun, "Fog-based crime-assistance in smart IoT transportation system," *IEEE Access*, vol. 6, pp. 11101_11111, 2018.
- [9] M. A. Hossain, "Framework for a cloud-based multimedia surveillance system," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 5, May 2014, Art. no. 135257.
- [10] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601_628, 1st Quart., 2018.

- [11] T. Sultana and K. A. Wahid, "Choice of application layer protocols for next generation video surveillance using Internet of video things," *IEEE Access*, vol. 7, pp. 41607_41624, 2019.
- [12] M. W. Alam, M. H. A. Sohag, A. H. Khan, T. Sultana, and K. A. Wahid, "IoT-based intelligent capsule endoscopy system: A technical review," in *Intelligent Data Analysis for Biomedical Applications*, 1st ed. New York, NY, USA: Academic, 2019, pp. 1_20.
- [13] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27_32, Oct. 2014.
- [14] M. Chiang, S. Ha, C.-L. I, F. Risso, and T. Zhang, "Clarifying fog computing and networking: 10 questions and answers," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 18_20, Apr. 2017.
- [15] B. Skrbic, D. Radovanovic, S. Tomovic, L. Lazovic, Z. Zecevic, and I. Radusinovic, "A decentralized platform for heterogeneous IoT networks management," in *Proc. 23rd Int. Sci.-Prof. Conf. Inf. Technol. (IT)*, Feb. 2018, pp. 1_4.
- [16] S. Wu, J. B. Rendall, M. J. Smith, S. Zhu, J. Xu, H. Wang, Q. Yang, and P. Qin, "Survey on prediction algorithms in smart homes," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 636_644, Jun. 2017.
- [17] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traf_c control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432_2455, 1st Quart., 2017.
- [18] H.-C. Shih, "A robust occupancy detection and tracking algorithm for the automatic monitoring and commissioning of a building," *Energy Buildings*, vol. 77, pp. 270_280, Jul. 2014.
- [19] J. Yang, H. Zou, H. Jiang, and L. Xie, "Device-free occupant activity sensing using WiFi-enabled IoT devices for smart homes," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3991_4002, Oct. 2018.
- [20] K.-H. Lee and J.-N. Hwang, "On-road pedestrian tracking across multiple driving recorders," *IEEE Trans. Multimedia*, vol. 17, no. 9, pp. 1429_1438, Sep. 2015.

- [21] K.-H. Lee, J.-N. Hwang, and S.-I. Chen, "Model-based vehicle localization based on 3-D constrained multiple-kernel tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 1, pp. 38_50, Jan. 2015.
- [22] X. Chen, J.-N. Hwang, D. Meng, K.-H. Lee, R. L. de Queiroz, and F.-M. Yeh, "A quality-of-content-based joint source and channel coding for human detections in a mobile surveillance cloud," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 19_31, Jan. 2017.
- S. O. Ajiboye, P. Birch, C. Chatwin, and R. Young, "Hierarchical video surveillance architecture: A chassis for video big data analytics and exploration," *Int. Soc. Opt. Photon.*, vol. 9407, Mar. 2015, Art. no. 94070K.
- [24] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 964_975, Jan. 2018.
- [25] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Netw.*, vol. 32, no. 6, pp. 137_143, Nov./Dec. 2018.
- [26] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput. Pract. Exper.*, vol. 28, no. 10, pp. 2991_3005, 2016.
- [27] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1126_1139, May 2018.
- [28] A. Rego, A. Canovas, J. M. Jiménez, and J. Lloret, "An intelligent system for video surveillance in IoT environments," *IEEE Access*, vol. 6, pp. 31580_31598, 2018.
- [29] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96_101, Jan. 2018.
- [30] N. Cao, S. Bin Nasir, S. Sen, and A. Raychowdhury, "Self-optimizing IoT wireless video sensor node with *in-situ* data analytics and contextdriven energy-aware real-time adaptation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2470_2480, Sep. 2017.

- [31] C.-T. Fan, Y.-K. Wang, and C.-R. Huang, "Heterogeneous information fusion and visualization for a large-scale intelligent video surveillance system," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 4, pp. 593_604, Apr. 2017.
- [32] D. Kieran and W. Yan, "A framework for an event driven video surveillance system," in *Proc. 7th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Aug./Sep. 2010, pp. 97_102.
- [33] A. Dimou, A. Axenopoulos, D. Matsiki, and P. Daras, "A user-centric approach for event-driven summarization of surveillance videos," in *Proc. 6th Int. Conf. Imag. Crime Prevention Detection*, Jul. 2015, pp. 1_6.
- [34] A. Sakaushi, K. Kanai, J. Katto, and T. Tsuda, "Edge-centric video surveillance system based on event-driven rate adaptation for 24-hour monitoring," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 651_656.
- [35] N. Al-Yamani, S. Qaisar, A. Alhazmi, S. Mohammad, and A. Subasi, "An event driven surveillance system," in *Proc. 5th Int. Conf. Electron. Devices, Syst. Appl. (ICEDSA)*, Dec. 2016, pp. 1_4.
- [36] T. Mekonnen, E. Harjula, A. Heikkinen, T. Koskela, and M. Ylianttila, "Energy efficient event driven video streaming surveillance using Sleepy-CAM," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Aug. 2017, pp. 107_113.
- [37] R. Hasan, S. K. Mohammed, A. H. Khan, and K. A. Wahid, "A color frame reproduction technique for IoT-based video surveillance application," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1_4.
- [38] L. Li and M. K. H. Leung, "Integrating intensity and texture differences for robust change detection," *IEEE Trans. Image Process.*, vol. 11, no. 2, pp. 105_112, Feb. 2002.
- [39] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709_1724, Jun. 2011.
- [40] J. Kang, D. V. Anderson, and M. H. Hayes, "Face recognition for vehicle personalization with near infrared frame differencing," *IEEE Trans. Consum. Electron.*, vol. 62, no. 3, pp. 316_324, Aug. 2016.

- [41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [42] R. Olmos, S. Tabik, and F. Herrera, "Automatic handgun detection alarm in videos using deep learning," *Neurocomputing*, vol. 275, pp. 66_72, Jan. 2018.
- [43] Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," *Remote Sens.*, vol. 9, no. 9, p. 907, 2017.
- [44] A. Sammoud, A. Kumar, M. Bayoumi, and T. Elarabi, "Real-time streaming challenges in Internet of video things (IoVT)," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1_4.
- [45] *CCTV Image of Man Wanted in Connection with Scunthorpe Armed Robbery Credit: Humberside Police*. Accessed: Jul. 17, 2019. [Online]. Available: <https://www.itv.com/news/calendar/topic/armed-robbery/>

APPENDIX A

SOURCE CODE

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import zipfile
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
import cv2
cap = cv2.VideoCapture(0)
# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
# ## Object detection imports
# Here are the imports from the object detection module.
# In[3]:
from utils import label_map_util
from utils import visualization_utils as vis_util
# # Model preparation
# ## Variables
# Any model exported using the `export_inference_graph.py` tool can be loaded here
simply by changing `PATH_TO_CKPT` to point to a new .pb file.
# By default we use an "SSD with Mobilenet" model here. See the [detection model
zoo](https://github.com/tensorflow/models/blob/master/object_detection/g3doc/detection
```

_model_zoo.md) for a list of other models that can be run out-of-the-box with varying speeds and accuracies.

```
# In[4]:
```

```
MODEL_NAME = 'ssd_mobilenet_v2_coco_2018_03_29'
```

```
# What model to download.
```

```
#MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'
```

```
MODEL_FILE = MODEL_NAME + '.tar.gz'
```

```
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
```

```
# Path to frozen detection graph. This is the actual model that is used for the object detection.
```

```
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
```

```
# List of the strings that is used to add correct label for each box.
```

```
PATH_TO_LABELS = os.path.join('data', 'mscoco_complete_label_map.pbtxt')
```

```
NUM_CLASSES = 90
```

```
# ## Download Model
```

```
# In[5]:
```

```
opener = urllib.request.URLopener()
```

```
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
```

```
tar_file = tarfile.open(MODEL_FILE)
```

```
for file in tar_file.getmembers():
```

```
    file_name = os.path.basename(file.name)
```

```
    if 'frozen_inference_graph.pb' in file_name:
```

```
        tar_file.extract(file, os.getcwd())
```

```
# ## Load a (frozen) Tensorflow model into memory.
```

```
# In[6]:
```

```
detection_graph = tf.Graph()
```

```
with detection_graph.as_default():
```

```
    od_graph_def = tf.GraphDef()
```

```
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
```

```

serialized_graph = fid.read()

od_graph_def.ParseFromString(serialized_graph)

tf.import_graph_def(od_graph_def, name=")

# ## Loading label map

# Label maps map indices to category names, so that when our convolution network
predicts `5`, we know that this corresponds to `airplane`. Here we use internal utility
functions, but anything that returns a dictionary mapping integers to appropriate string
labels would be fine

# In[7]:
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories      =      label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

# ## Helper code

# In[8]:
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)

# # Detection

# In[9]:
# For the sake of simplicity we will use only 2 images:

# image1.jpg
# image2.jpg

# If you want to test the code with your images, just add path to the images to the
TEST_IMAGE_PATHS.

PATH_TO_TEST_IMAGES_DIR = 'test_images'

TEST_IMAGE_PATHS      =      [      os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3) ]

# Size, in inches, of the output images.

```

```

IMAGE_SIZE = (12, 8)

# In[10]:
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        while True:
            ret, image_np = cap.read()

            # Expand dimensions since the model expects images to have shape: [1, None, None,
3]
            image_np_expanded = np.expand_dims(image_np, axis=0)
            image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

            # Each box represents a part of the image where a particular object was detected.
            boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

            # Each score represent how level of confidence for each of the objects.
            # Score is shown on the result image, together with the class label.
            scores = detection_graph.get_tensor_by_name('detection_scores:0')
            classes = detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections = detection_graph.get_tensor_by_name('num_detections:0')

            # Actual detection.
            (boxes, scores, classes, num_detections) = sess.run(
                [boxes, scores, classes, num_detections],
                feed_dict={image_tensor: image_np_expanded})

            # Visualization of the results of a detection.
            vis_util.visualize_boxes_and_labels_on_image_array(
                image_np,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index,
                use_normalized_coordinates=True,

```

```

        line_thickness=8)
cv2.imshow('object detection', cv2.resize(image_np, (800,600)))
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    break

```

MODULE 2:

```

#!/usr/bin/env python
# coding: utf-8
# # Object Detection Demo
# # Imports
# In[ ]:
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from distutils.version import StrictVersion
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):
    raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')

```

```

# ## Env setup

# In[ ]:

# This is needed to display the images.
get_ipython().run_line_magic('matplotlib', 'inline')

# ## Object detection imports

# Here are the imports from the object detection module.

# In[ ]:

from utils import label_map_util

from utils import visualization_utils as vis_util

# # Model preparation

# ## Variables

# Any model exported using the `export_inference_graph.py` tool can be loaded here
simply by changing `PATH_TO_FROZEN_GRAPH` to point to a new .pb file.

# By default we use an "SSD with Mobilenet" model here. See the [detection model
zoo](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/
detection_model_zoo.md) for a list of other models that can be run out-of-the-box with
varying speeds and accuracies.

# In[ ]:

# What model to download.

MODEL_NAME = 'ssd_mobilenet_v2_coco_2018_03_29'

MODEL_FILE = MODEL_NAME + '.tar.gz'

DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for the object
detection.

PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.

PATH_TO_LABELS = os.path.join('data', 'object-detection.pbtxt')

# ## Download Model

# In[ ]:

opener = urllib.request.URLopener()

```

```

opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

tar_file =
tarfile.open("E:/aaa/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz")

for file in tar_file.getmembers():

    file_name = os.path.basename(file.name)

    if 'frozen_inference_graph.pb' in file_name:

        tar_file.extract(file, os.getcwd())

### Load a (frozen) Tensorflow model into memory.

# In [ ]:

detection_graph = tf.Graph()

with detection_graph.as_default():

    od_graph_def = tf.GraphDef()

    with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:

        serialized_graph = fid.read()

        od_graph_def.ParseFromString(serialized_graph)

        tf.import_graph_def(od_graph_def, name="")

### Loading label map

# Label maps map indices to category names, so that when our convolution network
predicts `5`, we know that this corresponds to `airplane`. Here we use internal utility
functions, but anything that returns a dictionary mapping integers to appropriate string
labels would be fine

# In [ ]:

category_index =
label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)

### Helper code

# In [ ]:

def load_image_into_numpy_array(image):

    (im_width, im_height) = image.size

    return np.array(image.getdata()).reshape(

```

```

        (im_height, im_width, 3)).astype(np.uint8)
# # Detection
# In[ ]:
# For the sake of simplicity we will use only 2 images:
# image1.jpg
# image2.jpg
# If you want to test the code with your images, just add path to the images to the
TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images1'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3) ]
# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
# In[ ]:
def run_inference_for_single_image(image, graph):
    with graph.as_default():
        with tf.Session() as sess:
            # Get handles to input and output tensors
            ops = tf.get_default_graph().get_operations()
            all_tensor_names = {output.name for op in ops for output in op.outputs}
            tensor_dict = {}
            for key in [
                'num_detections', 'detection_boxes', 'detection_scores',
                'detection_classes', 'detection_masks'
            ]:
                tensor_name = key + ':0'
                if tensor_name in all_tensor_names:
                    tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
                        tensor_name)

```

```

if 'detection_masks' in tensor_dict:
    # The following processing is only for single image
    detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
    detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
    # Reframe is required to translate mask from box coordinates to image coordinates
    and fit the image size.
    real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
    detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])
    detection_masks = tf.slice(detection_masks, [0, 0, 0], [real_num_detection, -1, -1])
    detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
        detection_masks, detection_boxes, image.shape[1], image.shape[2])
    detection_masks_reframed = tf.cast(
        tf.greater(detection_masks_reframed, 0.5), tf.uint8)
    # Follow the convention by adding back the batch dimension
    tensor_dict['detection_masks'] = tf.expand_dims(
        detection_masks_reframed, 0)
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')
    # Run inference
    output_dict = sess.run(tensor_dict,
                           feed_dict={image_tensor: image})
    # all outputs are float32 numpy arrays, so convert types as appropriate
    output_dict['num_detections'] = int(output_dict['num_detections'][0])
    output_dict['detection_classes'] = output_dict[
        'detection_classes'][0].astype(np.int64)
    output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
    output_dict['detection_scores'] = output_dict['detection_scores'][0]
    if 'detection_masks' in output_dict:
        output_dict['detection_masks'] = output_dict['detection_masks'][0]
return output_dict

```

```

# In[ ]:
for image_path in TEST_IMAGE_PATHS:
    image = Image.open(image_path)
    # the array based representation of the image will be used later in order to prepare the
    # result image with boxes and labels on it.
    image_np = load_image_into_numpy_array(image)
    # Expand dimensions since the model expects images to have shape: [1, None, None,
3]
    image_np_expanded = np.expand_dims(image_np, axis=0)
    # Actual detection.
    output_dict = run_inference_for_single_image(image_np_expanded, detection_graph)
    # Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        output_dict['detection_boxes'],
        output_dict['detection_classes'],
        output_dict['detection_scores'],
        category_index,
        instance_masks=output_dict.get('detection_masks'),
        use_normalized_coordinates=True,
        line_thickness=8)
    plt.figure(figsize=IMAGE_SIZE)
    plt.imshow(image_np)
MODULE 3:
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile

```

```
import tensorflow.compat.v1 as tf
import zipfile
from datetime import datetime
import json
import requests
import os
from requests.auth import HTTPBasicAuth
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
# import ujson as json
import pandas as pd
import numpy as np
import time
import sys
from firebase import firebase
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config)
import cv2
# cam = 0 # Use local webcam.
# cam = "http://192.168.0.101:8081/video"
# cap = cv2.VideoCapture(cam)
# cap = cv2.VideoCapture(1)
sys.path.append("..")
fixefixed_interval = 3
firebase = firebase.FirebaseApplication('https://iot-guard-918ce.firebaseio.com', None)
```

```

count=1
# In[3]:
from utils import label_map_util
from utils import visualization_utils as vis_util

def diffImg(t0, t1, t2):
    d1 = cv2.absdiff(t2, t1)
    d2 = cv2.absdiff(t1, t0)
    return cv2.bitwise_and(d1, d2)

#url='rtsp://admin:888888@192.168.0.164:10554/tcp/av0_0'
threshold = 100000 # Threshold for triggering "motion detection"
cap = cv2.VideoCapture(0) # Lets initialize capture on webcam
# Read three images first:
t_minus = cv2.cvtColor(cap.read()[1], cv2.COLOR_RGB2GRAY)
t = cv2.cvtColor(cap.read()[1], cv2.COLOR_RGB2GRAY)
t_plus = cv2.cvtColor(cap.read()[1], cv2.COLOR_RGB2GRAY)
# Lets use a time check so we only take 1 pic per sec
timeCheck = datetime.now().strftime('%Ss')
# In[4]:
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v2_coco_2018_03_29'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
# Path to frozen detection graph. This is the actual model that is used for the object
detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

```

```

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'object-detection.pbtxt')
NUM_CLASSES = 90

# ## Download Model

# In[5]:
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
    file_name = os.path.basename(file.name)
    if 'frozen_inference_graph.pb' in file_name:
        tar_file.extract(file, os.getcwd())

# In[6]:
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

# In[7]:
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
category_index = label_map_util.create_category_index(categories)

# In[8]:
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)

```

```

# In[9]:
PATH_TO_TEST_IMAGES_DIR = 'test_images'

TEST_IMAGE_PATHS = [os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3)]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)

count = 1

# In[10]:
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        while(cap.isOpened()):
            # Read first and next image
            ret,image = cap.read()
            ret2,image2 = cap.read()

            if image is not None and image2 is not None and ret is True and ret2 is True:
                cv2.imshow('frame', image)

                if cv2.countNonZero(diffImg(t_minus, t, t_plus)) > threshold and timeCheck !=
datetime.now().strftime('%Ss'):
                    dimg = image

                    #cv2.imwrite(path + datetime.now().strftime('%Y%m%d_%Hh%Mm%Ss%f')
+ '.jpg', dimg)

                    image_np = dimg

                    # Expand dimensions since the model expects images to have shape: [1,
None, None, 3]
                    image_np_expanded = np.expand_dims(image_np, axis=0)
                    image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

                    # Each box represents a part of the image where a particular object was
detected.

                    boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

                    # Each score represent how level of confidence for each of the objects.

```

```

# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
    [boxes, scores, classes, num_detections],
    feed_dict={image_tensor: image_np_expanded})
# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)

df = pd.DataFrame(boxes.reshape(100, 4), columns=['y_min', 'x_min',
'y_max', 'x_max'])
df1 = pd.DataFrame(classes.reshape(100, 1), columns=['classes'])
df2 = pd.DataFrame(scores.reshape(100, 1), columns=['scores'])
df5 = pd.concat([df, df1, df2], axis=1)
df6 = df5.loc[df5['classes'] == 1]
df7 = df6.loc[df6['scores'] > 0.50]
if int(len(df7.index)) > 0:
    people = int(len(df7.index))
else:
    people = 0

```

```

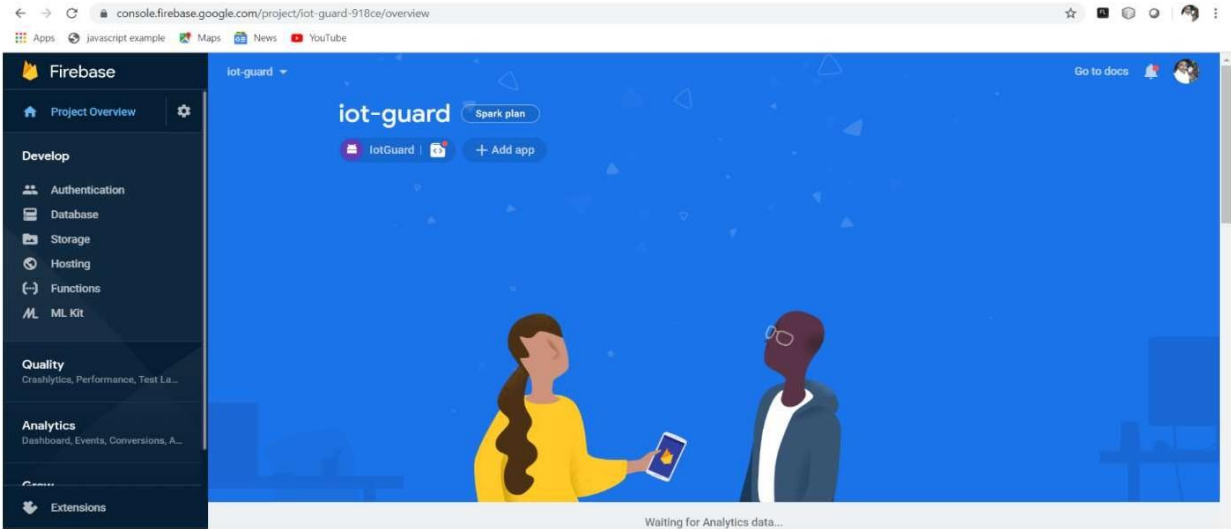
columns = ['LocationID','DeviceID','DeviceTime', 'Class', 'Count']
index = [0]
timenow = datetime.utcnow()
df_ = pd.DataFrame(index=index, columns=columns)
df_.loc[0, 'LocationID'] = '1'
df_.loc[0, 'DeviceID'] = '32'
df_.loc[0, 'DeviceTime'] = timenow
df_.loc[0, 'Class'] = 1
df_.loc[0, 'Count'] = people
jn = df_.to_json(orient='records', lines=True)
jn1 = json.loads(jn)
print(jn)
firebase.put("", 'IOT guard/Location 1', jn1)
print("pixelDiff:" + str(cv2.countNonZero(diffImg(t_minus, t, t_plus))))
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    #    break
timeCheck = datetime.now().strftime('%Ss')
# Read next image
t_minus = t
t = t_plus
t_plus = cv2.cvtColor(image2, cv2.COLOR_RGB2GRAY)

if ret is False:
    cap.release()
    cv2.destroyAllWindows()
    cap = cv2.VideoCapture(0)

```

APPENDIX B

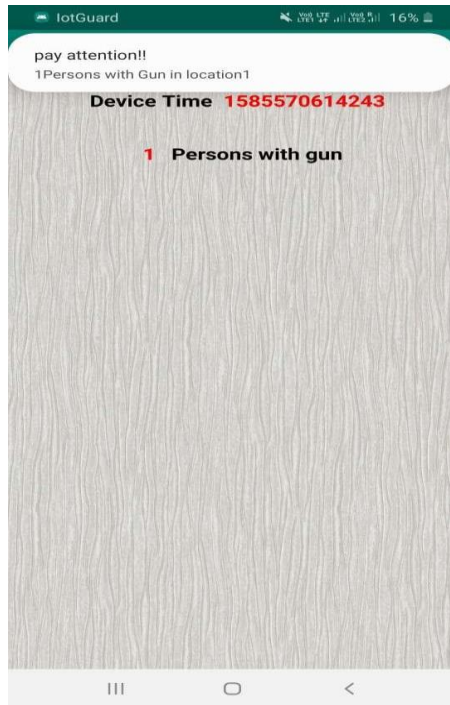
SCREENSHOTS



Screenshot 1



Screenshot 2



Screenshot 3



Screenshot 4

APPENDIX C

PLAGIARISM REPORT

REAL-TIME SECURITY MANAGEMENT IN SMART HOME ENVIRONMENT

ORIGINALITY REPORT

0%

SIMILARITY INDEX

0%

INTERNET SOURCES

0%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes Off

Exclude matches Off

Exclude bibliography On