Table of Contents

[ABSTRACT 4](#_Toc90401087)

**Chapter-1**

[Introduction 5](#_Toc90401088)

[1.1 Objective 5](#_Toc90401089)

[1.2 Problem Statement: 5](#_Toc90401090)

[1.3 Proposed Solution 6](#_Toc90401091)

[1.4 Organization of Report 6](#_Toc90401092)

[1.5 Hardware and Software Requirements 6](#_Toc90401093)

[Hardware Specifications: 6](#_Toc90401094)

[Software Specifications: 6](#_Toc90401095)

[Operating System: 8](#_Toc90401096)

**Chapter-2**

[Literature Survey 9](#_Toc90401097)

[2.1 Related work 9](#_Toc90401098)

[Automatic handgun detection alarm in videos using deep learning 9](#_Toc90401099)

[Object Detection Binary Classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance 9](#_Toc90401100)

[Face Detection & Face Recognition Using Open Computer Vision Classifiers 9](#_Toc90401101)

[SSDMNV2: A real time DNN-based face mask detection system using single shot multi-box detector and MobileNetV2 10](#_Toc90401102)

**Chapter-3**

[MODULES 11](#_Toc90401103)

[3.1 NumPy: 11](#_Toc90401104)

[3.1.1 NumPy in Image Processing: 11](#_Toc90401105)

[3.2 OpenCV - Open-Source Computer Vision: 12](#_Toc90401106)

[3.3 Keras 12](#_Toc90401107)

[3.3.1 Keras for Deep Learning: 13](#_Toc90401108)

[3.4 smtplib 14](#_Toc90401109)

[3.5 pickle 15](#_Toc90401111)

[3.6 winsound 16](#_Toc90401112)

**Chapter-4**

[Algorithms 17](#_Toc90401113)

[4.1 Convolutional Neural Networks 17](#_Toc90401114)

[4.1.1 Architecture of Artificial Neurons 17](#_Toc90401115)

[4.1.2 Training a CNN 18](#_Toc90401116)

[4.1.3 Building a CNN 19](#_Toc90401117)

[4.2 YOLO Algorithm 20](#_Toc90401118)

[4.2.1 Working of Yolo Algorithm: 21](#_Toc90401119)

**Chapter-5**

[IMPLEMENTATION 23](#_Toc90401120)

[5.1 Data Collection 23](#_Toc90401121)

[5.2 Data Preprocessing 24](#_Toc90401122)

[5.3 Face Detection and Identification Model 24](#_Toc90401123)

[5.4 Weapon Detection Model 25](#_Toc90401124)

[5.5 Capturing Frames 25](#_Toc90401125)

[5.6 Calculating Score 28](#_Toc90401126)

[5.7 Sending Emails 29](#_Toc90401127)

[5.8 Alert Sound 30](#_Toc90401128)

**Chapter-6**

[CONCLUSION 31](#_Toc90401129)

[6.1 Sample Code 31](#_Toc90401130)

[6.2 Output 38](#_Toc90401131)

[6.2.1 Sending Emails 39](#_Toc90401132)

[6.3 Accuracy: 40](#_Toc90401133)

[6.4 Results 40](#_Toc90401134)

[6.5 Conclusion: 41](#_Toc90401135)

[References 42](#_Toc90401136)

**List of Figures:**

* Fig 1.1: TensoraFlow multi-dimenal arrays
* Fig 3.1: Code for cropping image using NumPy
* Fig 3.2: Basic SMTP model
* Fig 4.1: Architecture of artificial neurons
* Fig 4.2: Image processing using CNN
* Fig 4.3: residuals blocks in the image
* Fig 4.4: Bounding box of an object
* Fig 4.5: Overlapping phenomenon
* Fig 4.6: Combining three techniques of yolo
* Fig 5.1: Sample Images from the first Dataset(weapons)
* Fig 5.2: Sample Images from the second dataset (people with and without masks)
* Fig 6.1: There is no human presence and no weapon detected
* Fig 6.2: When person is detected but is not hiding his face
* Fig 6.3 & 6.4: When the person with hiding his face is detected and a weapon is detected in the frame.
* Fig 6.5: Sample of email sent to owner
* Fig 6.6: Accuracy of Face detection model

## ABSTRACT

**THEFT DETECTION SYSTEM**

**Theft Detection System** is a house safety technology which helps in alerting owner of the house in case of robbery. With our busy lives, it is not possible to monitor it 24\*7. Basically, this system is similar to smart camera which will be able to detect any suspicious weapons or suspicious people entering the house. This system is capable of detecting people with inappropriate gesture or visiting at unusual time and alerts the user via mail.

This system is developed using Image processing technique in which a dataset of weapons images is created first. Then, a model is built on the dataset using CNN which will be able to detect any weapons used during robbery. This model is used fed to the camera which will be able to alert the owner via mail. This system is further improved to also detect the people with suspicious face gestures like wearing masks that cover parts of the face. Based on the weapon detection and gestures of the guests visiting the guest, a score will be calculated. If the score is very high, an alarm sound is played which alerts the neighbours.

Chapter-1

# Introduction

Robberies, burglaries, and thefts continue to be a problem across the country. According to the National Crime Records Bureau (NCRB), there were 2,44,119 occurrences of robbery, theft, burglary, and dacoity at residential premises in 2017. This was an increase of more than 10% from 2016. The financial losses incurred because of these thefts and burglaries are enormous. Property stolen from residential premises in 2017 was valued at over Rs. 2065 crores, up 40% from Rs. 1,475 crores stolen the previous year. Dealing is burglary being a major part, restoring the lost property is another headache for the victims of burglary like the cost of replacing the lost property or asset, the expenses to associated with the litigations, etc., It has so much impact on the victims financially, emotionally, and mentally. It is seen in most of the cases that people who has been the victims of burglary have faced physiological issues for several years.

## 1.1 Objective

Our motive is to reduce this number of crimes and the impacts of burglary via an automated system which can detect these kinds of acts and immediately alert the owner as well the neighbors of the house who house is about to be robbed. In particular, the idea is to equip security cameras at homes and shops with system. This allows the owners of the residence to act quickly and alert concerned authority members.

To achieve this, we need a model which is trained to detect any weapons along with the person or suspicious face of the person using image processing when trying to enter the house or a shop. Basically, our idea is to develop a machine learning model which is capable of identify if any burglary is about to happen and if it is confirmed, the owner should be immediately informed.

## 1.2 Problem Statement:

CCTV camera systems are installed to protect your home and shops from burglary and break-ins. But with our busy lives, it is not possible to monitor it 24\*7. So, we need a more reliable and robust smart security system that can notify us when someone breaks into our shop or home.

## 1.3 Proposed Solution

The proposed solution is a security system which tries to analyze all the guests entering the house and predict if they are a threat to the house. The system continuously takes images of the scene visible to the camera and analyze those images to rectify any suspicious guests or if the people entering the house are carrying any weapon.

## 1.4 Organization of Report

Chapter-1: Gives the brief description of the project, its important and identifies the problem and purpose of this project. It also describes the software and hardware requirements of the project.

Chapter -2: literature Survey discusses all the work that has been done related to the proposed work and identifies the pros and cons of the works.

Chapter -3: Describes all the modules used in project, basic functionalities of the modules, and reasons for choosing those modules

Chapter-4: Describes all the algorithms that have been implemented in the project and explains how these algorithms work.

Chapter- 5: comprehend the workflow of the project with the code explanation

Chapter -6: concludes the documents with the code, outputs and the result.

References

## 1.5 Hardware and Software Requirements

### Hardware Specifications:

* Processor - Intel Atom® processor or Intel® Core™ i3 processor
* RAM - 4GB
* Disk space - 2 to 3 GB

### Software Specifications:

**Programming Language – Python:**

Python is a high-level, general-purpose programming language that is widely used. The latest version Python 3 is used in web development, machine learning applications, etc., Python is a programming language that may be used to build a variety of applications. It's a preferred choice of most of the developers for projects involving artificial intelligence (AI), machine learning, and deep learning. Python is the best suited language for this project because of several reasons like:

* It has ample libraries and frameworks for all functions- The most popular libraries like numpy, TensorFlow, Keras are used in the project.
* Its simplicity as it is a readable language even for the new developers which is very essential in the project.
* It is much easier to code in python than in any other language as it has simple syntax that is easy to understand.

**TensorFlow**

TensorFlow is an open-source Python library which is developed and released by Google for faster numerical computation. It is a foundation framework that can be used to develop Deep Learning models concepts in the simplest way possible, either directly or utilizing wrapper libraries built on top of TensorFlow to make the process easier. It combines computational algebra and optimization approaches to make many mathematical equations simple to calculate.

TensorFlow was created with various machine learning and deep learning algorithms in mind for both research and development. TensorFlow can train and execute deep neural networks for various purposes like handwriting recognition, image recognition, converting handwriting notes to digital notes etc.,

**Features of TensorFlow:**

* It has a functionality that uses multi-dimensional arrays called tensors to design, optimize, and calculate arithmetic equations.

A picture containing text, electronics

Description automatically generated

Fig 1.1 TensoraFlow multi-dimenal arrays

* Deep neural networks and machine learning approaches are supported by TensorFlow programming.
* It has a highly scalable computational capability that can work with a variety of data sets.
* TensorFlow automates management by utilizing GPU computation. It also has a one-of-a-kind feature that optimises the utilisation of the same memory and data.

### Operating System:

* Windows 7 or higher or
* Linux **64-bit RHEL or higher or**
* Mac OS X 10.11 or higher.

CHAPTER-2

# Literature Survey

## 2.1 Related work

### Automatic handgun detection alarm in videos using deep learning

The main are of focus in this project was to reduce the crimes in the country that happen with guns and recognized the problem of reducing the number of false alarms. The developers have developed an automatic gun detection models which helps in alerting the security guards and police to avoid the crime from happening. They had rephrased this detection algorithm as a problem of minimizing false positives and solve it by first constructing the vital training data set depending on the results of a VGG-16 based classifier, and then evaluating the ideal classification model using two approaches: the sliding window approach and the region proposal approach.

### Object Detection Binary Classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance

This work focuses mainly on reducing the false positives by identify the object which might be misunderstood as weapons by the automated systems used in security cameras. This work presented ODeBiC, a two-level deep learning-based system for detecting small items that can be handled identically. In surveillance footage, they employed the detection of small objects that could be mistaken for a firearm or a knife as a case study. They had created the Sohas weapon training database, which comprises six objects that can be mistaken for a weapon since they are regularly handled in the same way. Their experiment showed that the number of false positives has been reduced by more than 50% by using ODeBiC methodology based on an aggregation method of OVO.

### Face Detection & Face Recognition Using Open Computer Vision Classifiers

This is a model for the face detection and face recognition using several classifiers available in Computer Vision Classifiers (OpenCV). These faces are detected using Haar-cascade classifier and recognized using different applications like Eigenface, Fisherface and Local binary pattern histogram (LBPH) algorithms. Apart from implementing these application, the project also compares the results obtained from these algorithms. Faces classifier objects are created using cv2.CascadeClassifier() and eye classifier objects are also created by using OpenCV XML files.

### SSDMNV2: A real time DNN-based face mask detection system using single shot multi-box detector and MobileNetV2

This model aims at detecting the people who are not wearing masks and violating covid norms using the COVID-19 pandemic situation. In this model the image dataset for both training and testing in which contains images of people wearing the mask and not wearing the mask is developed. The OpenCV deep neural networks is used to train the model and obtained successful results. The MobilenetV2 image classifier was used to effectively classify images. The SSDMNV2 model is compared with several pre-existing models like LeNet – 5, AlexNet, VGG-16. Comparison is done in terms of accuracy, F1scores and average performance in Frames per second(FPS), and it is found that SSDMNV2 excels in all the comparisons.

CHAPTER-3

# MODULES

## 3.1 NumPy:

NumPy also called as Numerical Python is the most important Python library which is used for scientific mathematical computations. It has a number of features, including the following:

* powerful N-dimensional array object
* Advanced (broadcasting) capabilities
* Linear algebra and random numbers capabilities
* Integration tools for C/C++ and Fortran code

NumPy can also be used to store generic data in a multi-dimensional container. Numpy allows arbitrary data types to be created, allowing NumPy to connect with a wide range of databases smoothly and quickly.

### 3.1.1 NumPy in Image Processing:

Numerous image processing can be done with NumPy functions after reading the image as a NumPy ndarray.  ndarray function can be used to get and set pixel values, editing images and concatenating, and so on. Those who are familiar with NumPy can perform a variety of image processing tasks without the need for tools like OpenCV.  Even when using OpenCV for Python, picture data is treated as an ndarray, therefore knowing how to utilize NumPy is essential. Many libraries, such as scikit-image also treat images as ndarrays. There are transformations like editing, padding, rotating, shifting, flipping that can be done using NumPy arrays.

Once the image is converted to NumPy format, it makes the calculations faster and also flexible compared to “Python Images”.

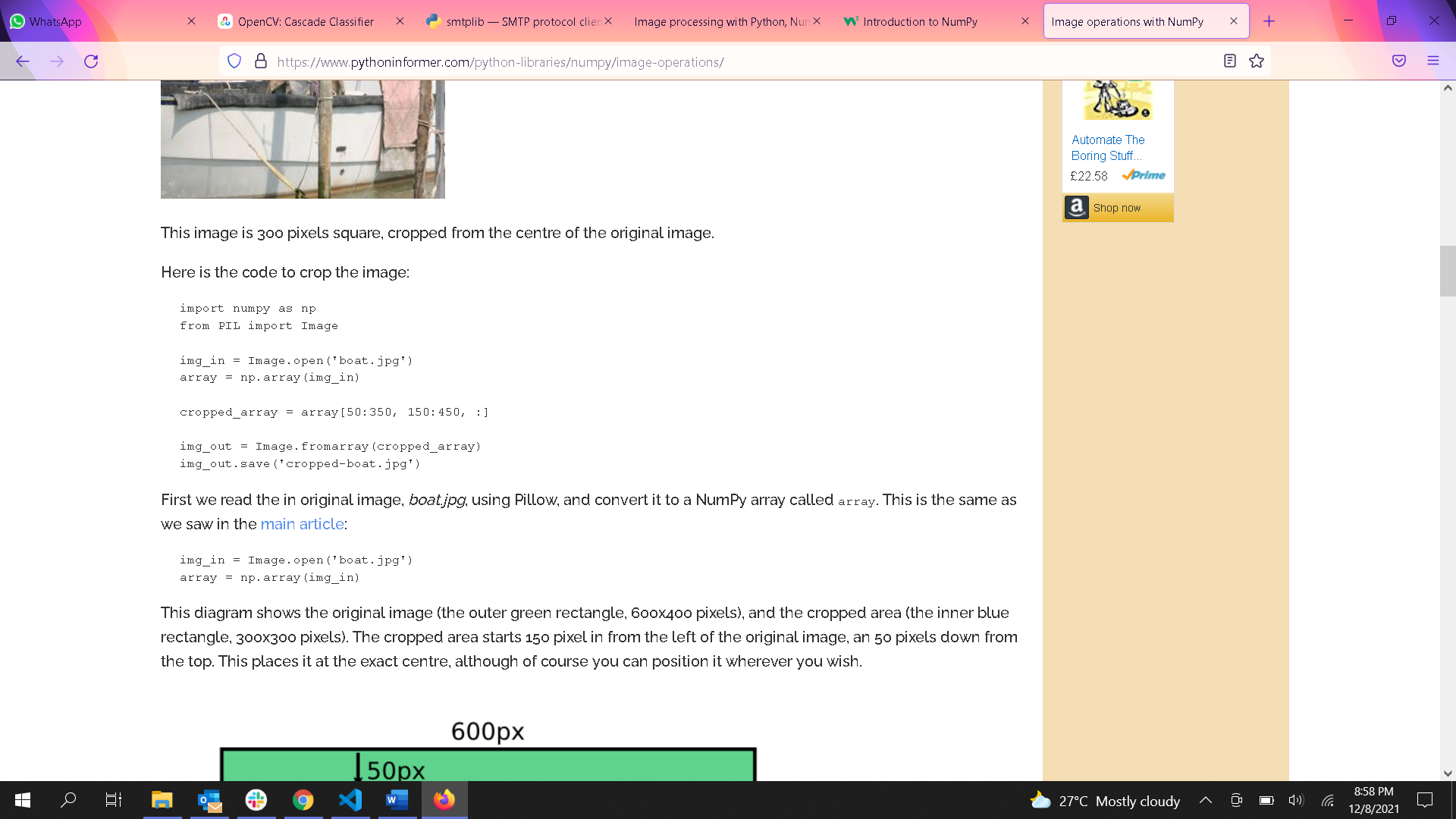


Fig 3.1 Code for cropping image using NumPy

## 3.2 OpenCV - Open-Source Computer Vision:

OpenCV is a large open-source library for image processing, computer vision, and machine learning. Python, C++, Java, and other programming languages are supported by OpenCV. OpenCV was established to provide a standard infrastructure for computer vision applications and to accelerate the use of machine perception in consumer products. OpenCV made it easy for organizations to utilise and modify the code. OpenCV is one of the many predefined modules and libraries that make our lives easier.

It can analyse photos and videos to recognise and analyse objects, faces, and even human handwriting. When OpenCV is combined with other libraries, such as Numpy, a highly efficient library for numerical operations, any operations that Numpy can perform can be combined with OpenCV. In our project, we use OpenCV combined with NumPy module that made our work more sophisticated.

There are several applications of OpenCV module out of which some are used in our daily life frequently. These include:

* Medical image analysis
* object recognition
* counting vehicles on highways and identifying their speed
* counting number of people present in an image
* face recognition

OpenCv is used in our project for several purposes like image processing, to load readNet, for video objects and for capturing frames. The Haar classifier of the OpenCV are also used to identify human faces in the images in this project.  
3.2.1 Haar-cascade classifier for face detection

Haar features are like kernals in Convolution which are often used to identify the presence of a specific features in a given image. The cv::CascadeClassifier::load method in OpenCV allows you to read pre-trained models. The models are saved as XML files in the repository, which may be accessed using OpenCV Functionalities. Face detection models, eye detection models, upper and lower body identification models, license plate detection models, and so on are among them.

## 3.3 Keras

Keras is a lightweight Python deep learning package that runs on top of TensorFlow. It was created to make developing deep learning models for research and development as simple and quick as feasible. It operates on Python 2.7 or 3.5 and, thanks to the underlying frameworks, can run on both GPUs and CPUs. Keras is built on four fundamental principles:

**Modularity**: A model can be viewed as a single sequence or graph. A deep learning model's issues are all separate components that can be merged in arbitary way.

**Minimalism**: The library delivers only what is necessary to achieve a goal, with no unnecessary frills and extra readability.

**Extensibility**: New components are designed to be simple to install and utilise within the framework, allowing researchers to experiment with new concepts.

**Python:** No separate model files with specific file formats. Everything is written entirely in Python.

Keras is the most popular and compulsory module while developing a deep learning algorithm because of following reasons:

* Keras is very easy and simple to understand
* It has a large community support
* It is very quick to build network models.
* Keras can be trained on a single GPU or multiple GPUs simultaneously

### 3.3.1 Keras for Deep Learning:

To create a network is model in Keras, we have followed the following steps

**1. Loading the Data:** After importing the packages necessary for themodel, we have to load the data into the model. For doing this, we need to use the OS module as the data would be large and in CSV format, it would make it easier to read.

**2. Defining the keras model:** Defining the model is nothing but defining the layers sequentially. The commonly used layers in keras are Convolutional Layer, Max Pooling Layer, Dense Layer, Drop out layer. The configurations for each layer are defined in trial and error method.

**3. Compiling the keras model:** The model is compiled using efficient numerical libraries (the so-called backend) like TensorFlow. The backend automatically selects the optimum way of representing the network for training and prediction on your hardware, whether it's a CPU, GPU, or distributed architecture.

**4. Fitting the model:** By using the fit() function of the model, we could train or fit our model on our loaded data. Training takes place in epochs, with each epoch divided into batches.

Epoch: A single run across the training dataset's rows.

Batch: Before weights are adjusted, the model considers one or more samples within an epoch.

**5. Evaluating the model:** This will tell us how well we modelled the dataset, but not how well the algorithm would perform on new data. For training and evaluation of the model, you should split your data into train and test datasets. we can use the evaluate() method on your model to evaluate the model on training dataset, passing it the same input and output that you used to train it.

## 3.4 smtplib

SMTP stands for Simple Mail Transfer Protocol. The client who wants to send the email establishes a TCP connection with the SMTP server and then sends the email via it. The smtplib module provided by Python creates an SMTP client session object that may be used to deliver mail to any system on the Internet that has an SMTP listener daemon. When sending emails using Python, make sure your SMTP connection is secured so that your message and login credentials are not accessible to outsiders. TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are two technologies that would be used to encrypt an SMTP session.

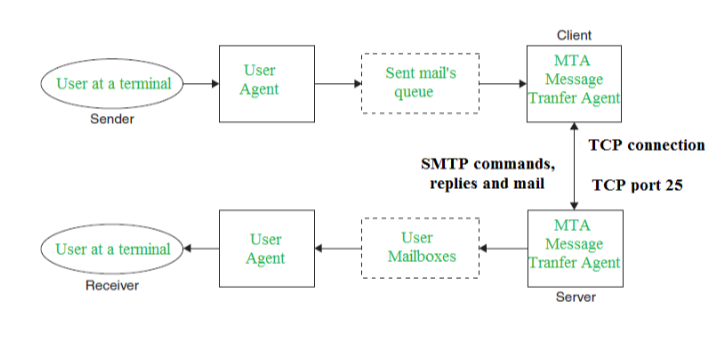


Fig 3.2 Basic SMTP model

You can establish a secure connection with your email server in one of two ways:

* Begin an SMTP connection using SMTP SSL() that is encrypted from the start .
* Start an unencrypted SMTP connection that can be encrypted later with the .starttls() .

If you're using SMTP SSL(), you'll need to connect to port 465, and if you're using .starttls(), you'll need to connect to port 587. We have used .startttls() for our project. For doing so, Create an instance of smtplib.SMTP, which encapsulates an SMTP session and provides access to its methods, to do this. After you've established a secure SMTP connection with one of the techniques listed above, you can send your email with .sendmail(). A Sample code of this would be:

## 

import smtplib, ssl

sender\_email = "my@gmail.com"

receiver\_email = "your@gmail.com"

password = #password

message = """\

Subject: Hi there"""

with smtplib.SMTP("smtp.gmail.com", 587) as server:

server.starttls(context=context)

server.login(sender\_email, password)

server.sendmail(sender\_email, receiver\_email, message)

## 3.5 pickle

The pickle module supports binary serialisation and de-serialization protocols for Python class structures. Pickling is the process of converting a Python object structure into a byte stream, while unpickling is the process of converting a byte stream again into an object structure.

The pickle module interface include:

* Pickle.dumps(*obj*, *protocol=None*, *\**, *fix\_imports=True*, *buffer\_callback=None*) – As a *bytes* object, return the pickled format of the object *obj*.

The *protocol* argument, a number, instructs the pickler to utilize the specified protocol; protocols ranging from 0 to HIGHEST PROTOCOL are supported. DEFAULT PROTOCOL is the default if none is supplied. HIGHEST PROTOCOL is chosen if a negative integer is supplied.

Pickle will map the modules names from old versions to the new version of Python, if *fix\_imports* is true and protocol is smaller than 3, such that the pickle data streamcan be read by the older version.

Buffer views are serialized into a file if buffer callback value is set to None. If *buffer\_callback* is not set to None, a buffer view can call it an unlimited number of times.

* pickle.load(*file*, *\**, *fix\_imports=True*, *encoding='ASCII'*, *errors='strict'*, *buffers=None)* – This function is called to de-serialize a data stream. This function is similar to Unpickle() function.This method returns the rebuilt object hierarchy provided after reading a pickled object representation from the open file object file. The protocol version is automatically recognised, therefore no protocol parameter is required. Bytes beyond the pickled representation of the object are disregarded.

The pickle.load() is modified as to pickle.loads()when the python version is updated. The arguments passing to the function are also modified.

## 3.6 winsound

In Python 3, the winsound module acts as an interface for interacting with Windows' sound-playing hardware.

**winsound. Beep( ):** This method's functionality is to make a 'Beep' noise. The user must, however, provide the sound's frequency and duration (these are parameters that shall be given when calling you call the function).

Other winsound methods exist, the majority of them are task-specific and deal with runtime parameters.

CHAPTER-4

# Algorithms

## 4.1 Convolutional Neural Networks

Deep learning techniques are built on neural networks, which are a subset of machine learning. Node layers are made up of an input layer, one or more hidden levels, and an output layer. Each node has a weight and a threshold that it links to another node from different layer. If a node's output exceeds a certain threshold, the node is activated, and data is sent to the next tier of the network. If this is not the case, no data is sent on to the network's next tier.

Convolutional neural network (CNN/ ConvNets) is so important in deep learning compared to other neural network because of its high performance in images, and audio processing. A Convolutional Neural Network is a Deep Learning algorithm that takes an image as input, assign value to various aspects/objects in the image, and distinguish between them. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less. While basic approaches require hand-engineering of filters, ConvNets can learn these characteristics with appropriate training. Convolutional Neural Networks were able to capture the Spatial and Temporal dependencies which no other neural network could do.

### 4.1.1 Architecture of Artificial Neurons

Several layers of artificial neurons make up convolutional neural networks. Artificial neurons are computational units that calculate the weighted sum of various inputs and deliver an activation value, similar to their biological equivalents. Each neuron's behavior is determined by its weights. The artificial neurons of a CNN extract numerous visual properties when supplied with pixel values.

Each of the layers in a ConvNet produces different activation maps when you feed it an image. Activation maps emphasize the image's most important characteristics. Each neuron takes a pixel block as data, multiplies its color values by its weights, adds them together, and runs them through the activation function.

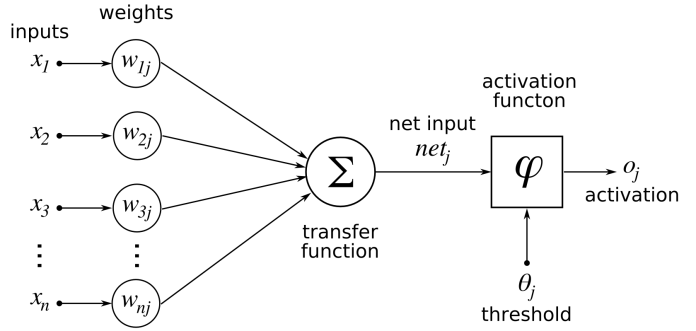


Fig 4.1 Architecture of artificial neurons

Basic characteristics such as horizontal, vertical, and diagonal edges are generally detected by the CNN's bottom most layers. The output generated in this layer is sent into the next layer, which extracts more complicated characteristics like corners and edge patterns. The layers recognize higher-level characteristics such as objects, faces, and more as you progress further into the convolutional neural network.

### 4.1.2 Training a CNN

"Training" the neural network refers to adjusting the weights of each neurons to retrieve the relevant characteristics from pictures. This is one of the most difficult challenge for the researchers. Initially, the weights of the neurons are assigned randomly. A CNN is generally trained on a large set of images along with their labels called dataset. The CNN begins with a set of random weights. During training, the developers submit a big collection of photos labelled with their matching classifications to the neural network. Each image is processed using random weights, and the outcome is compared to the image's original class. If the network's output does not match the correct label—which is probable at the start of the training process—it adjusts the values of the neurons to get its output nearer to the right label when it encounters the same image.

An "epoch" is a loop of the full training dataset that the ConvNet goes through during training, updating its weights in small increments. Each epoch improves the neural network's ability to categorize the training pictures. The weight modifications made by the CNN get closer and closer as it learns. The network "converges" at a certain stage, which implies it become as good as it can be.

The developers can use a test dataset to validate the correctness of the CNN once it has been trained. The test dataset consists of a collection of categorised images that were not used in the training phase. Each image is sent into the ConvNet, and the output is verified with the image's original class. The test dataset is used to measure how effective the neural network has gotten at identifying images it has never seen before. When a CNN performs well on training data but poorly on test data, it is considered to be "overfitted." This frequently arises if the training data isn't varied enough or if the ConvNet is trained on a dataset with several epochs.

### 4.1.3 Building a CNN

There are four different layers using which a CNN model can be built. They are

1. Convolutional layer

2. Pooling Layer

3. Fully Connected Layer

4. Soft-Max layer

**Input the Image:** Before feeding an image to the Convolutional neural network, we must convert the picture into RGB image or gray scale image. Then image is then resized, and basic pre-processing is done on the image.

**Convolutional layer:** After the image is preprocessed, it is fed into the network. The first layer the image goes through convolutional layer. The convolutional layer is the fundamental element of CNN. The goal of the layer is find the collection of features from this image. As a result, the convolutional layer takes multiple images as input and calculates the convolution of each with each filter.

Filters are the characteristics we are looking for in the images. We obtain a feature map with each pair (image, filter) that shows us where the features are in the image: the greater the value, more the corresponding location in the image matches the feature.

**Pooling Layer:** This layer is frequently layered between two convolutional layers: it receives many feature maps and perform the pooling operation to all of them. The pooling procedure reduces the size of the images while maintaining their vital features. We obtain the same amount of feature maps in output as we did in input, but they are substantially smaller. The pooling layer minimizes the number of parameters and operations. This increases network performance and prevents over-fitting. The pooling process sweeps a filter across the whole input, like convolutional layer, however this filter do not contain any weights. There are 2 types of pooling:

* Average Pooling: The average value inside the receptive field is calculated as the filter passes over the input and sent to the output array.
* Max-Pooling: The filter picks the pixel with the highest value to transmit to the output array as it filter across the input. In comparison to average pooling, this strategy is employed more frequently.
* **The ReLU correction layer:** The real non-linear function defined by ReLU(x)=max is referred to as ReLU (Rectified Linear Units) (0,x). All negative values obtained as inputs are replaced by zeros by the ReLU correction layer. It serves as a mode of activation.

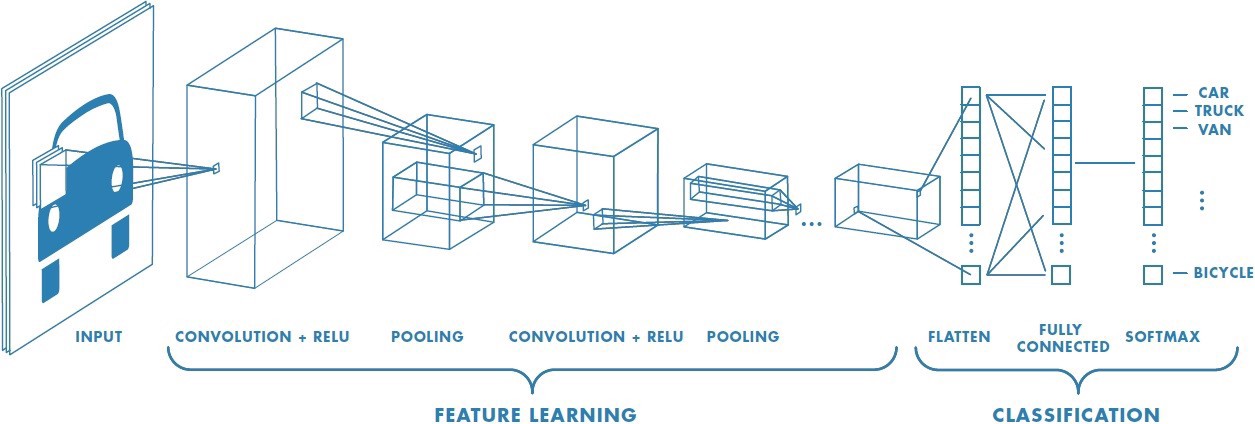


Fig 4.2 Image processing using CNN

**Fully connected Layer:** This layer takes an input vector and turns it into a new output vector. The input values are subjected to a linear combination and, perhaps, an activation function. The image is classified as an input to the network by the final fully connected layer, which produces a vector of size N, where N is the number of labels in our image classification problem. Each item of the vector represents the likelihood that the supplied image belongs to a specific category.

The fully-connected layer multiplies each input item by value, adds the results, and then adds an activation function to determine the probabilities. The phrase fully-connected refers to the fact that each input value is linked to all output values. The fully connected layer establishes the link between the image's feature positions and a classification.

**Soft-max Layer:** It's mostly used to fit the output of networks between zero and one. It's used to express the network output's "probability" of accuracy. The normalization is computed by dividing the investigated output's exp value by the total of all probable outputs' exp values.

Soft-max (xi) = exp(xi) ÷ (xi)

## 4.2 YOLO Algorithm

YOLO stands for ‘You Only Looking Once’. This is an algorithm for detecting and recognizing different items in a real-time image. Object detection in YOLO is done as a regression problem, and the identified images' class probabilities are provided. YOLO is a real-time object recognition technique that use neural networks. Because of its speed and reliability, this algorithm is quite popular. It has been used to identify pedestrians, parking metres, and animals in a variety of applications.

Object detection is a computer vision phenomenon in which numerous items are detected in digital photos or video recordings. People, automobiles, chairs, stones, houses, and animals are among the items observed. Object detection is frequently connected with self-driving cars, which use a combination of computer vision, LIDAR, and other technologies to create a multidimensional description of the road and all of its users. It's also commonly used in video surveillance, particularly in crowd monitoring to avoid terrorist attacks, counting people for general statistics, and analysing consumer experience with shopping centre walking routes.

Convolutional neural networks (CNN) are used in the YOLO method to recognise objects in real time. To identify objects, the approach just takes a single forward propagation across a neural network. This indicates that a single algorithm run is used to forecast the whole image. The CNN is used to forecast multiple bounding boxes and class probabilities at the same time.

### 4.2.1 Working of Yolo Algorithm:

It works using the following techniques:

* **Residual blocks:** The given is divided into several equal and small grids with dimensions SxS. Each grid will detect any object that is present within its boundaries. The boundaries of the grid will change its color indicating that an object is present inside it as shown below.

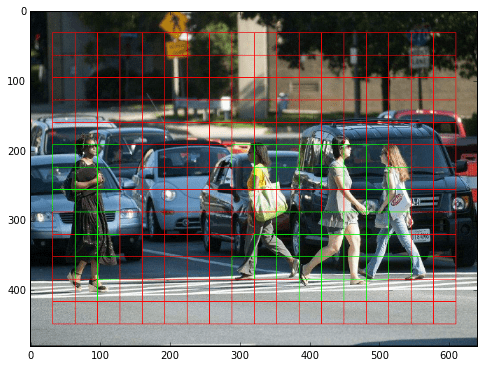


Fig 4.3 residuals blocks in the image

* **Bounding box regression:** Combining these residual blocks with objects, a bounding box is formed with the attributes: Width(bw), Height(bh), label or class (c), and the bounding box center (bx, by). Yolo uses only single bounding box regression to identify the height and width of any object.

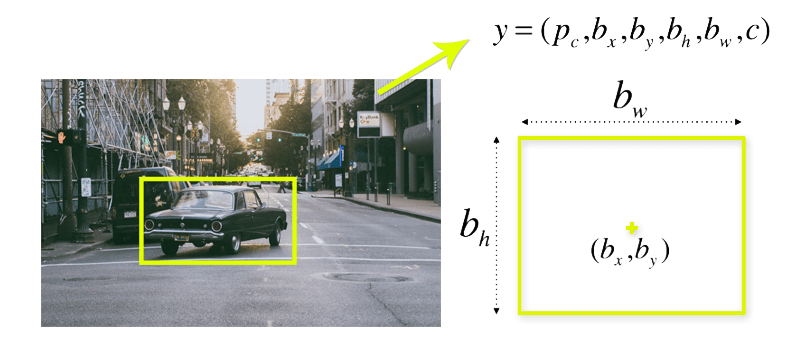


Fig 4.4 Bounding box of an object

* **Intersection over Union (IOU):** IOU phenomenon in object detection is occurred when tow or more bounding boxes overlap. The yolo alogirthm uses IOU to provide the object dimensions accurately.



Fig 4.5 Overlapping phenomenon

By combining all these 3 techniques, yolo will do the object detection efficiently. The final detection of yolo will be the separate bounding boxes for each object in a given image.

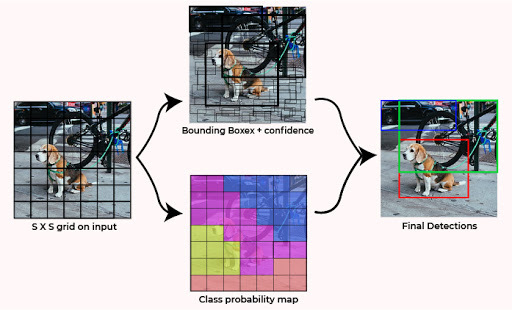


Fig 4.6 Combining three techniques of yolo

CHAPTER - 5

# IMPLEMENTATION

## 5.1 Data Collection

The data required for the proposed project is of 2 kinds, a dataset of different weapons and dataset of people wearing and not wearing masks. The dataset of weapons is from Google Images and Kaggle, which is an online community that allows users to find and publish dataset of different kinds. Google images are scraped using the selenium package and Chrome Driver extension This dataset consist is built on two types of weapons namely pistols or handguns and knives. We use these images to train the weapon detection model. The second data which is the dataset of people wearing masks is obtained from GitHub which provides Internet hosting for software developers. I have obtained the dataset from a project that has been developed to detect if people are wearing masks or not for COVID-19 situation.

A picture containing weapon

Description automatically generatedA black and white photo of an object

Description automatically generated with medium confidenceA picture containing weapon

Description automatically generatedA picture containing weapon

Description automatically generatedA picture containing weapon

Description automatically generatedA knife with a handle

Description automatically generated with low confidenceA picture containing weapon, knife, ax, tool

Description automatically generatedA picture containing weapon, knife

Description automatically generatedA knife with a black handle

Description automatically generated with medium confidenceA picture containing weapon, knife, floor, several

Description automatically generated

Fig 5.1 Sample Images from the first Dataset(weapons)

A picture containing person, crowd

Description automatically generatedA group of people wearing face masks

Description automatically generated with low confidenceA picture containing text, person, child, outdoor

Description automatically generatedA picture containing person, ground, outdoor, person

Description automatically generatedA baby in a stroller

Description automatically generated with medium confidenceA person wearing a mask

Description automatically generated with low confidenceA group of people standing in front of a sign

Description automatically generated with medium confidenceA group of people in clothing

Description automatically generated with low confidenceA person and person wearing masks

Description automatically generated with medium confidenceA picture containing text, person, floor, indoor

Description automatically generated

Fig 5.2 Sample Images from the second dataset (people with and without masks)

## 5.2 Data Preprocessing

The collected data is clean and had no missing values. But these images are of different sizes which might affect the accuracy of the model. So, these images are all converted to same size. These images are converted into 64\*64\*1 dimensions. These preprocessed images are loaded into the system and sent to the model.

These images are divided into training and testing sets. The model is first trained on the training set by extracting the features of these images and then testing on the testing sets. Both the datasets are divided in 70:30 for training and testing respectively.

## 5.3 Face Detection and Identification Model

Thieves will try to hide their faces to hide their identity. They will try to cover their faces, so they do not get captured in any surveillance cameras. We use this logic of them to detect if the person entering the house is a thief or not. Here we built to identify if people present in the images are wearing masks or not. This model is trained to identify if the person present in the image is hiding his face or not. As mentioned earlier, this model is built on a dataset consisting of pictures of human beings with and without wearing face masks from different angles. The images collected are from different angles and different distance to increase the accuracy of detection of the model. Here, we perform the transfer learning with Keras and Deep learning.

Before identifying the face masks, these images are preprocessed. All the images are converted to same size i.e., 64\*64\*1 and in RGB scale. The Convolutional Neural networks come into picture at this part of the process. These images are fed into the CNN algorithm to extract the features and study the nature of the images along with its labels, supervised learning. For training we have used 2 convolutional layers(convo2D) with ReLu activation with 200 filters, (3,3) kernel size and 100 filters, (3,3) kernel size respectively. The dimensionality of the kernel thus obtained is reducing using Max-Pooling of strides (2,2) with padding window size (2,2). The max pooling is also done twice after each convolution layer. The output of this layer would be the feature vector which holds the features of different classes. These features are fed into the Fully connected layer to classify the into the classes. These are normalized by SoftMax layer.

The MobileNetV2 model is trained using ImageNet values, and Transfer Learning is implemented. We do not use the Fully connected layer and soft-max layer which are mentioned above as we do not want to store the labels instead, we are storing the features of the labels. We use this model to identify of the person entering the house is trying to hide his face, and if so, he is recognized as a threat to the house.

## 5.4 Weapon Detection Model

Robbers while entering the house are most likely to bring a weapon along with the either for breaking-in or for self-defense or to blackmail the anyone inside the house or with the intensions of hurting the victims. This model is trained to identify if there are any weapons present in each image. This model too is trained on CNN algorithm. Unlike mask detection, transfer learning is used for training the model. The MobileNetV2 architecture is trained on the weights of ImageNet to realize better accuracy during a short time.

Once the images are pre-processed and fed to the network to extract the features from the images. The output of this model would also be features i.e., feature vector rather than labels. When the images are preprocessed there are fed to the yolo algorithm. Once the yolo algorithm detects the co-ordinates of the objects those co-ordinates are sent to the weapon detection model. The feature extraction is done during the convolutional layer and pooling layer in CNN algorithm. We use this model to identify if the person trying to enter the house is carrying any weapon with him which might be a threat to the house.

## 5.5 Capturing Frames

Once the models are trained, we use these models to scan a person and determine if the person is acceptable to enter the house or not. The workflow of our project goes like:

First, we load the weights from the above models into the variables and we load the haar cascade classifier to identify the faces.

Workflow of the Theft Detection System.

Score= score-1

Object detected?

no

yes

yes

no

no

yes

yes

yes

yes

Play alarm

Send email

Face mask model

Is score=15?

Is score=7?

Score = score+1

Is weapon?

Face detected?

Yolo algorithm

Resize and reshape

Capture frames

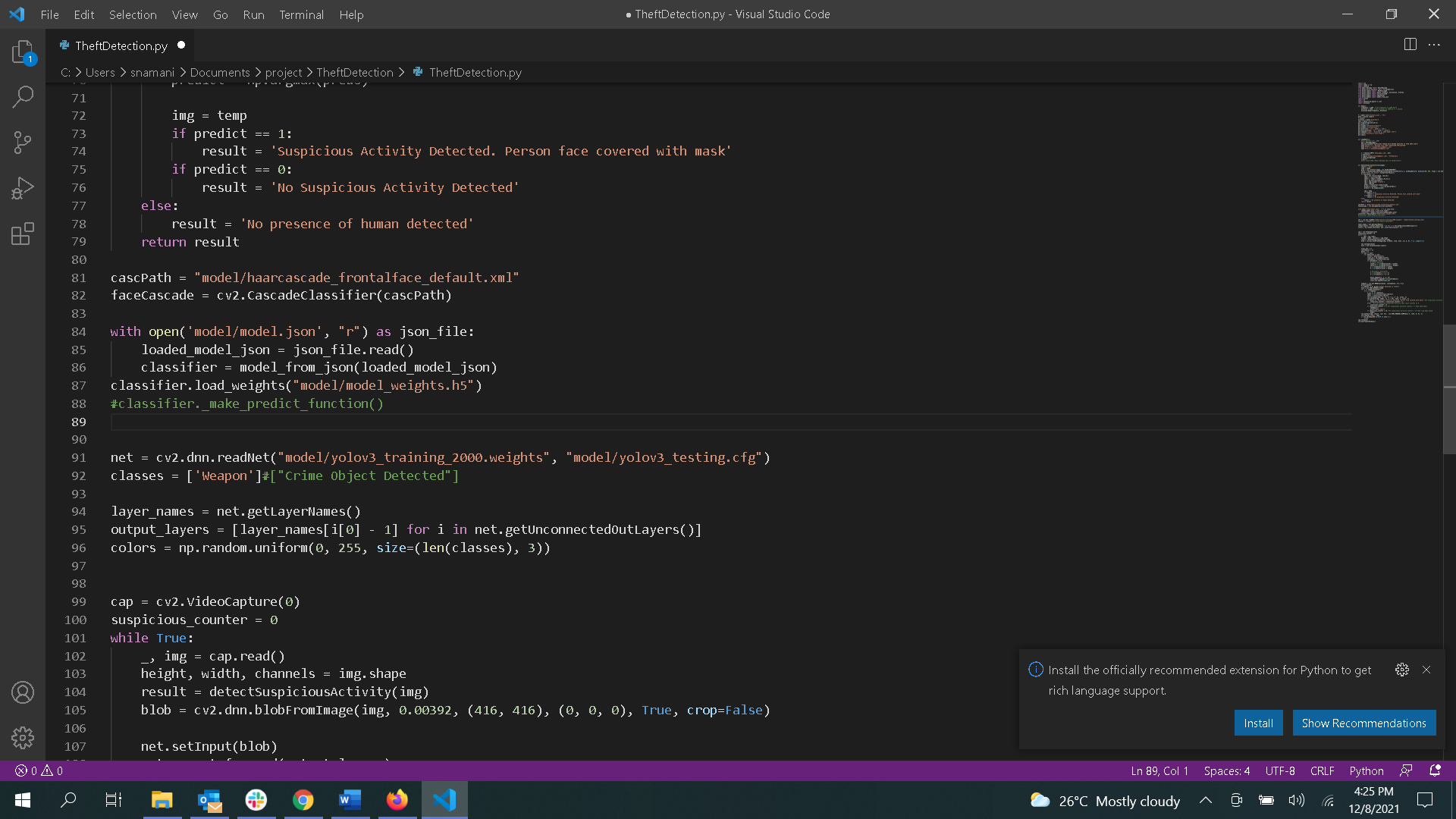
Weapon detection model

Weapon detection

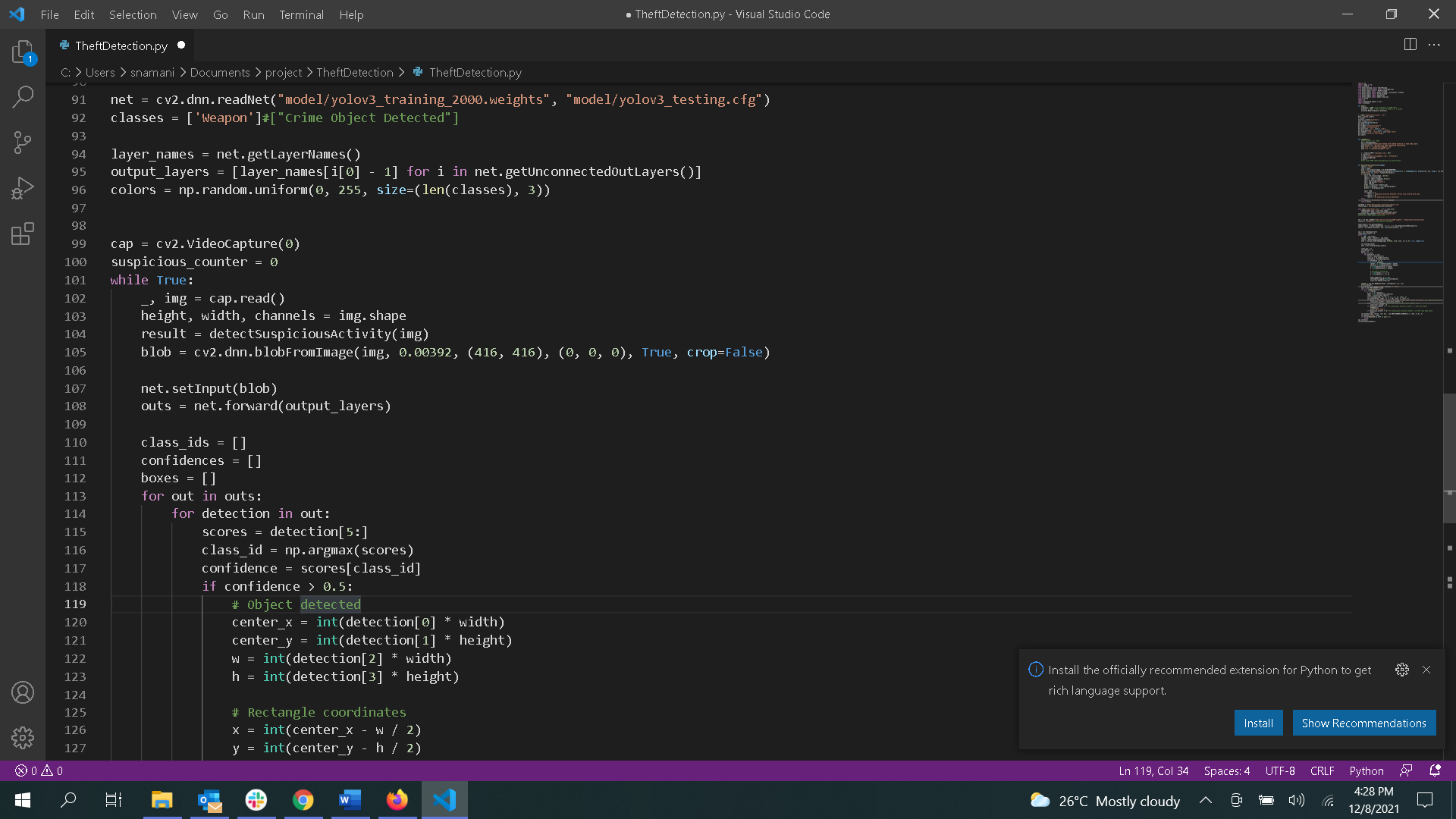
is hiding?

Haar cascade classifier

face detection model



We then start our VideoCapture object of OpenCV module to start capturing the frames. In an infinite loop, these frames are resized and recolored to ‘RGB’ mode for better performance.

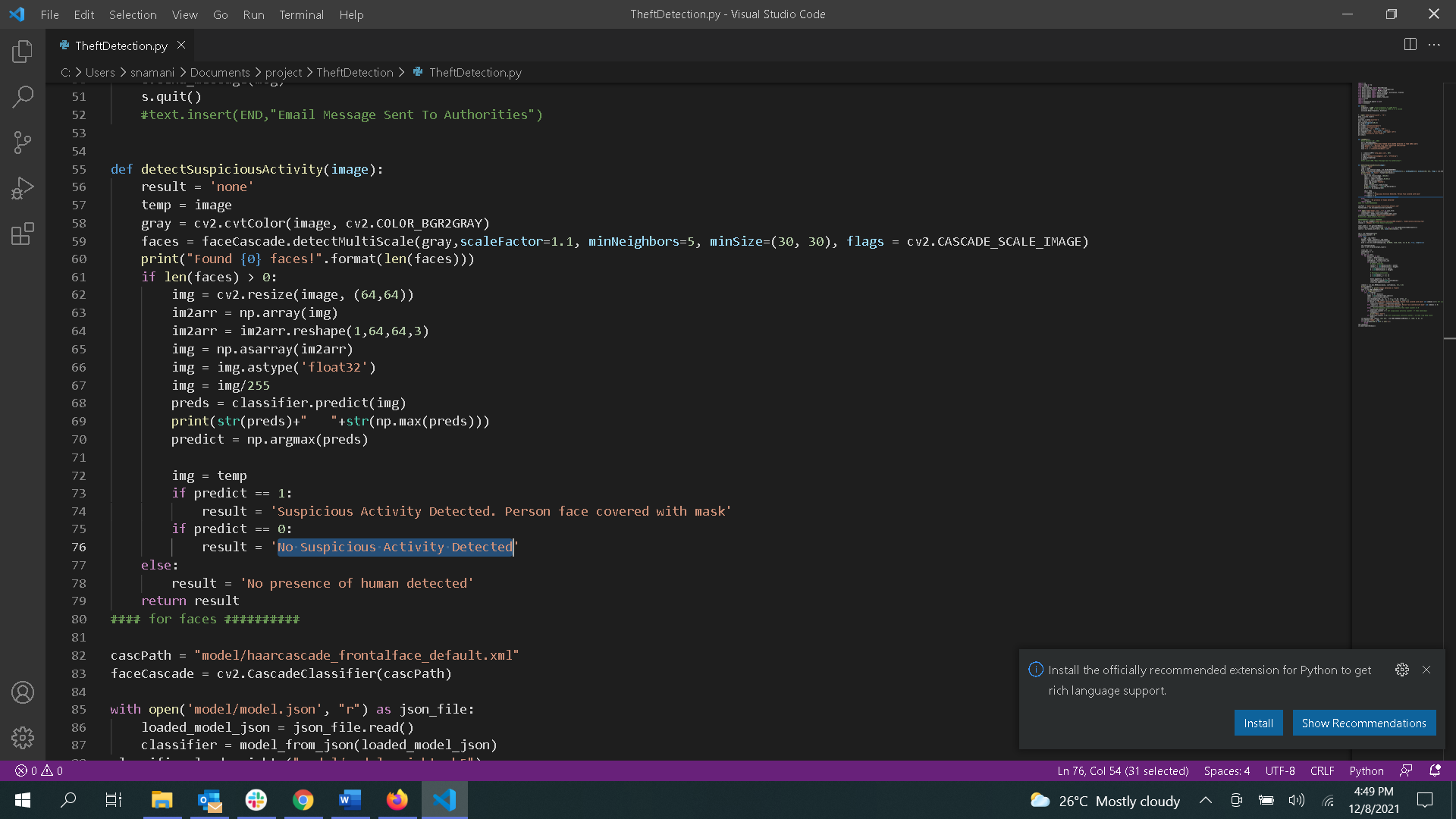


These frames are first fed into the face detection model. To do so, we first identify if there is any human presence in the frame using a pre-trained Haar-cascade classifier for faces as mentioned earlier. If this gives that human presence is existing in the frame, this frame will be further sent to identify if the person present in the frame is trying to hide is face or not.

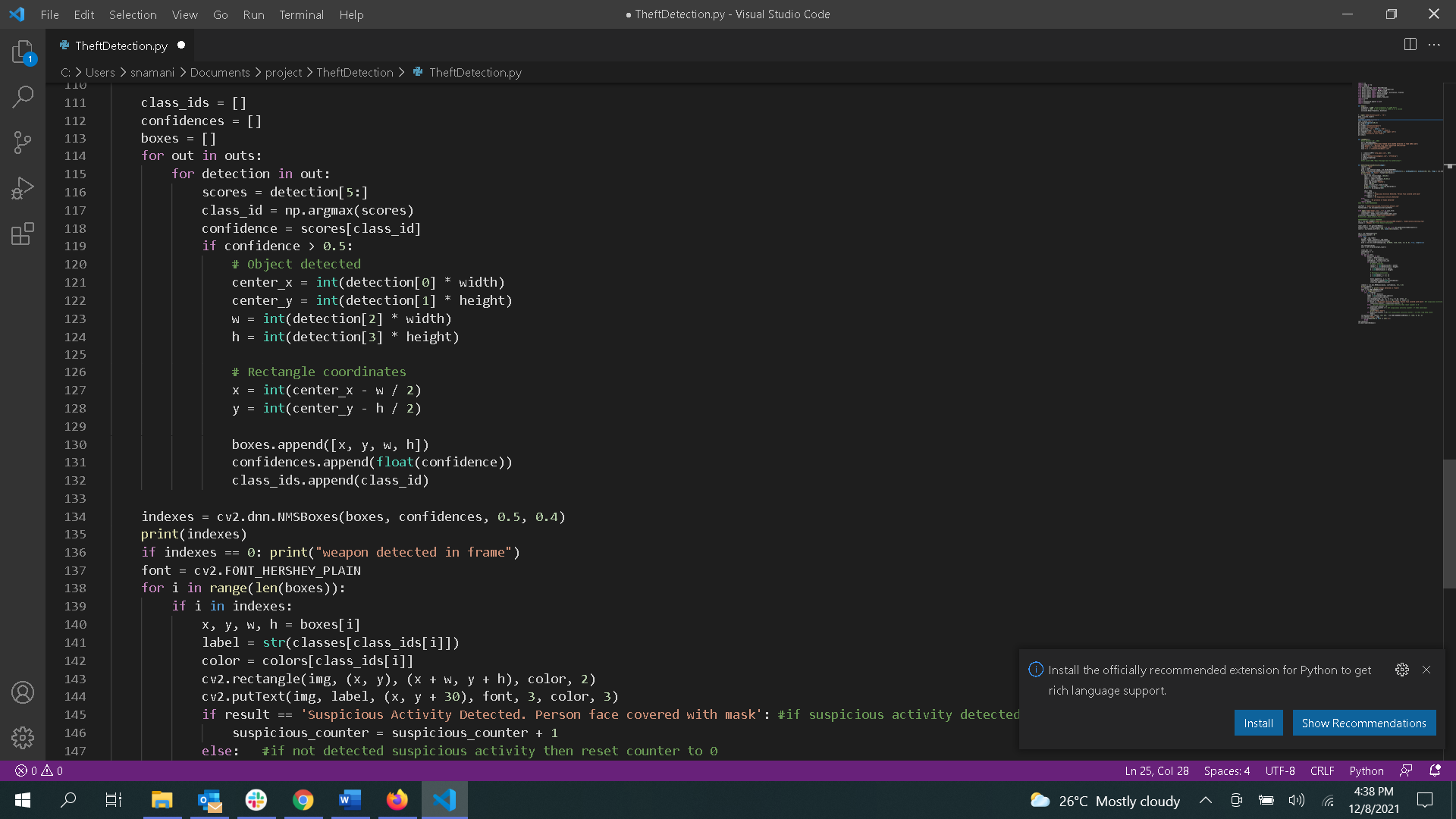
Case-1: If he is trying to hide his face, he is determined as a threat to the house. In this case, the message on the screen states “Suspicious Activity Detected. Person face covered with mask”

Case-2: If there is human presence, but he is not trying to hide his face. Then the message says “No Suspicious Activity Detected”

Case-3: If there is no human presence at all, the message states “No human detected”



Then these frames are fed into the weapon detection model to verify if there is any weapon present in the frame. For doing so, we first need to identify any kind of objects in the frame. We use yolo algorithm for identifying any objects in the frame. As discussed earlier, yolo algorithm has the capability of identifying any object in a given image in a single run. If any object is present the dimensions of the object are taken and the features from these dimensions are mapped to the weights, we got from the weapon detection model. This confirms if the object identified is a weapon of not. If the object is identifed as a weapon, then a message stating “weapon detected” is displayed on the screen.



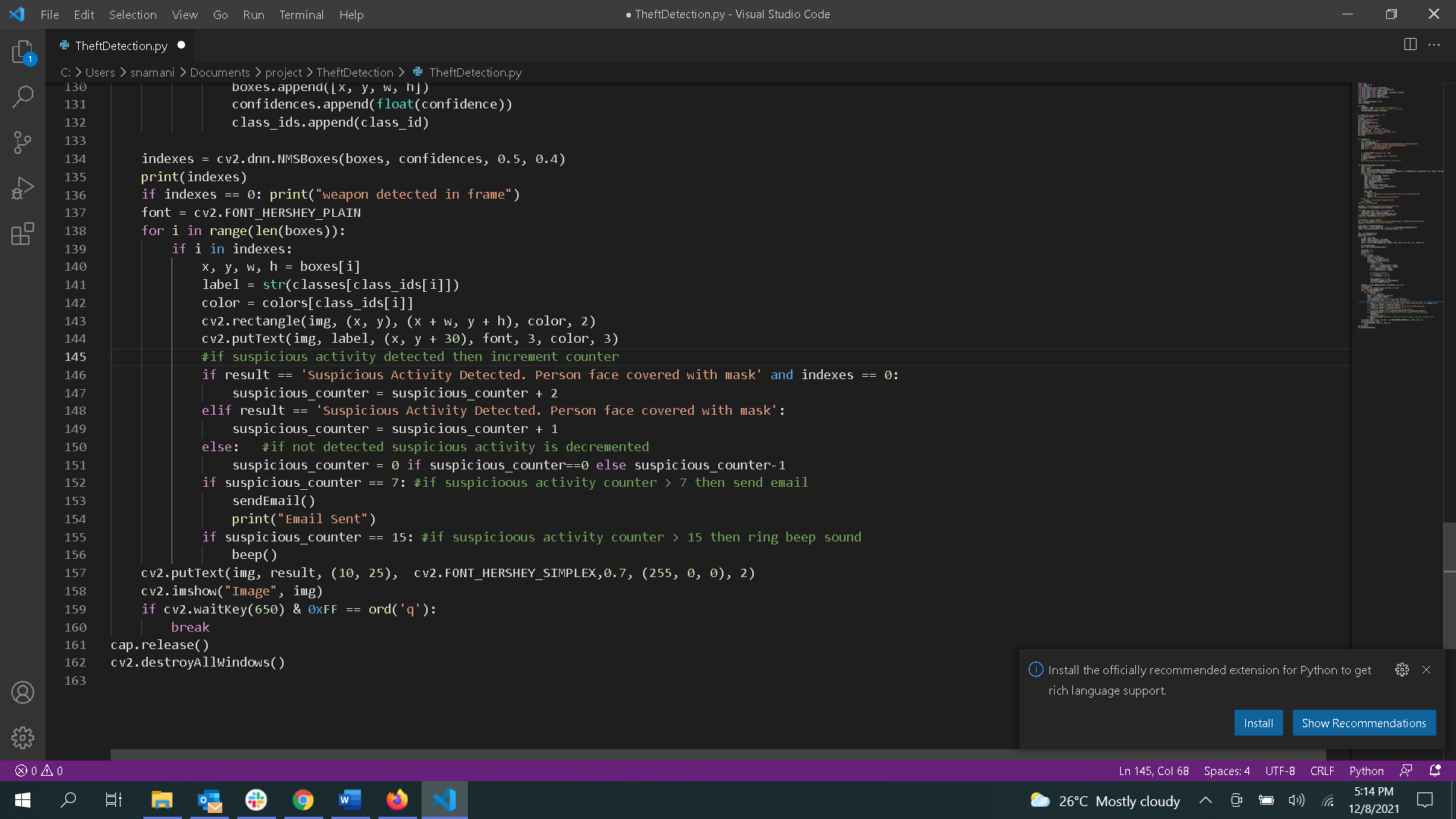
## 5.6 Calculating Score

The aim of the project is not only identifying the treat but also alerting the owner that his house is at stake. For doing, we must confirm if the person detected as a threat to the house is actually a threat or not. Before alerting the owner, we assign a score value. This score value is incremented or decremented based on the outcomes from both the models.

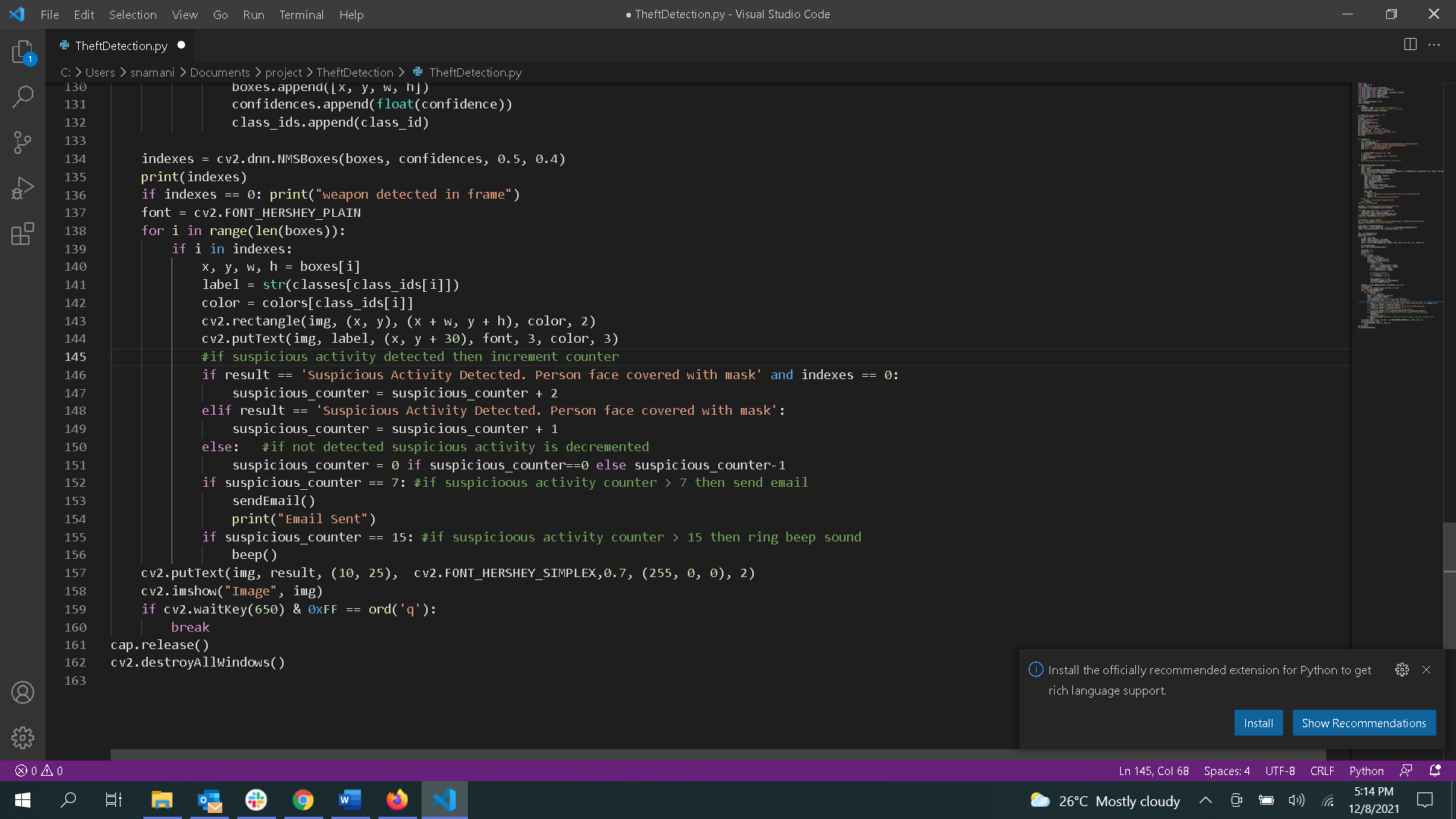
1. If the frame has both human presence with a mask and a weapon is detected, the score is incremented by 2 per frame.

2. If the frame has only a human with no weapon, the score is incremented by 1 per frame.

3. If neither of them is detected, the score value starts decreasing till it reaches 0.



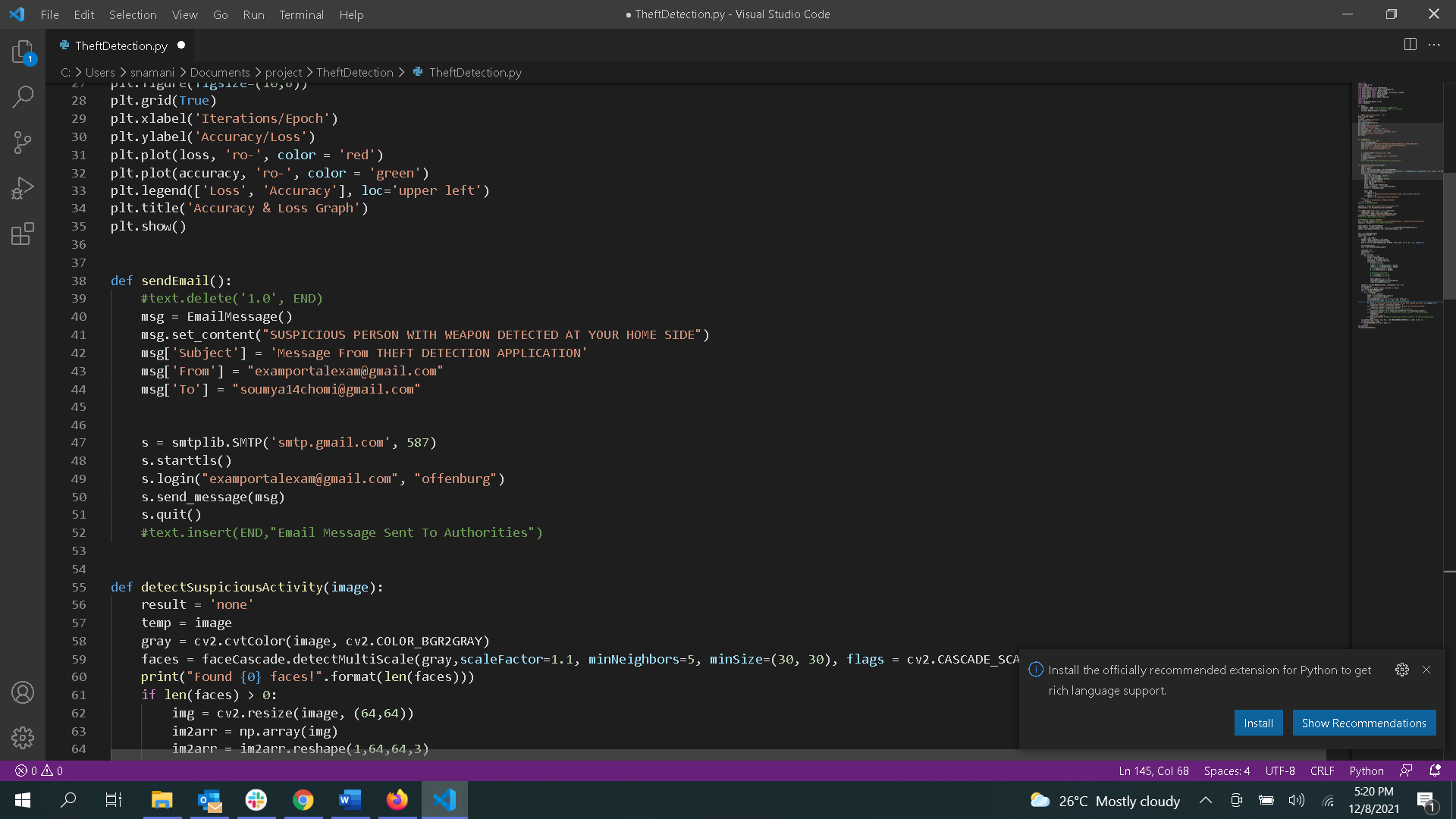
Two threshold values as set for this score, of the score crosses the first threshold value, we would inform the owner via mail that something suspicious is going on at his home. If the second threshold value is reached an alarm sound is played through which the neighbors of the house will get a hint if theft.



The threshold values are set as 7 and 15 experimentally, intending the first threshold value would be reached if the in 2-3 seconds within the person trying to break into the house arrives. The second threshold would be reached approximately after 5-6 seconds of his arrival.

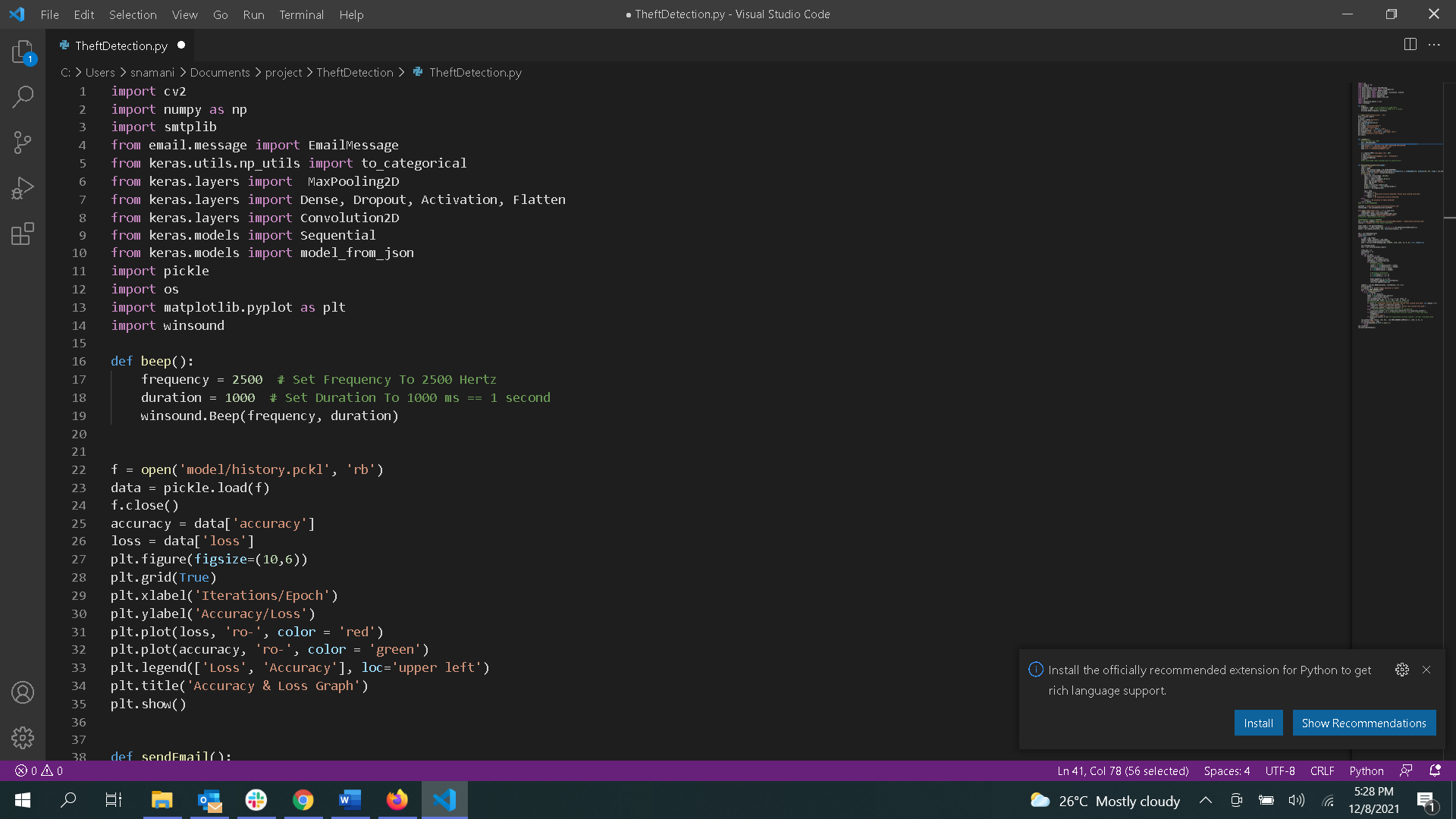
## 5.7 Sending Emails

As discussed earlier, if there are any suspicious activity going on and the first threshold value is reached, the owner is alerted via mail. To send these mails we have used STMP module in python. As discussed earlier, STMP module makes it easier to send and read mails through python. To achieve this, I have created a demo account through which mails can be sent to the owner. An SMTP instance is first created that encapsulates the SMPT connection. The credentials of the demo account are given to login into the account using this connection. Later, the message is sent from the demo account to the registered owner account using a method called send\_message() with the mail stating “SUSPICIOUS PERSON WITH WEAPON DETECTED AT YOUR HOME SIDE”.



## 5.8 Alert Sound

An alarm sound is played to alert the user surrounding the house which is at the risk of theft when a second threshold is reached, as discussed earlier. To play this sound, we can use many modules in python like pygame.mixer module, winsound module etc.,. I have chosen winsound module to play this alarm sound. It is played using a method called Beep() with specified frequency and for specified duration.



CHAPTER-6

# CONCLUSION

## 6.1 Sample Code

import cv2

import numpy as np

import smtplib

from email.message import EmailMessage

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

import os

import matplotlib.pyplot as plot

import winsound

def beep():

frequency = 2500 # Set Frequency To 2500 Hertz

duration = 10000 # Duration is set to 1000 i.e., == 10 seconds

winsound.Beep(frequency, duration)

f = open('model/history.pckl', 'rb')

data = pickle.load(f)

f.close()

accuracy = data['accuracy']

loss = data['loss']

plot.figure(figsize=(10,6))

plot.grid(True)

plot.xlabel('Iterations/Epoch')

plot.ylabel('Accuracy/Loss')

plot.plot(loss, 'ro-', color = 'red')

plot.plot(accuracy, 'ro-', color = 'green')

plot.legend(['Loss', 'Accuracy'], loc='upper left')

plot.title('Accuracy & Loss Graph')

plot.show()

def sendEmail():

#text.delete('1.0', END)

msg = EmailMessage()

msg.set\_content("SUSPICIOUS PERSON WITH WEAPON DETECTED AT YOUR HOME SIDE")

msg['Subject'] = 'Message From THEFT DETECTION APPLICATION'

msg['From'] = "examportalexam@gmail.com"

msg['To'] = "soumya14chomi@gmail.com"

s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("examportalexam@gmail.com", "offenburg")

s.send\_message(msg)

s.quit()

#text.insert(END,"Email Message Sent To Authorities")

def detectSuspiciousActivity(image):

result = 'none'

temp = image

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray,scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags = cv2.CASCADE\_SCALE\_IMAGE)

print("Found {0} faces!".format(len(faces)))

if len(faces) > 0:

img = cv2.resize(image, (64,64))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,64,64,3)

img = np.asarray(im2arr)

img = img.astype('float32')

img = img/255

preds = classifier.predict(img)

print(str(preds)+" "+str(np.max(preds)))

predict = np.argmax(preds)

img = temp

if predict == 1:

result = 'Suspicious Activity Detected. Person face covered with mask'

if predict == 0:

result = 'No Suspicious Activity Detected'

else:

result = 'No presence of human detected'

return result

#### for faces ##########

cascPath = "model/haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

classifier.load\_weights("model/model\_weights.h5")

#classifier.\_make\_predict\_function()

###########for weapons ########

net = cv2.dnn.readNet("model/yolov3\_training\_2000.weights", "model/yolov3\_testing.cfg")

classes = ['Weapon']#["Crime Object Detected"]

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

colors = np.random.uniform(0, 255, size=(len(classes), 3))

cap = cv2.VideoCapture(0)

suspicious\_counter = 0

while True:

\_, img = cap.read()

height, width, channels = img.shape

result = detectSuspiciousActivity(img)

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

class\_ids = []

confidences = []

boxes = []

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

# Object detected

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

# Rectangle coordinates

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

class\_ids.append(class\_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

print(indexes)

if indexes == 0: print("weapon detected in frame")

font = cv2.FONT\_HERSHEY\_PLAIN

for i in range(len(boxes)):

if i in indexes:

x, y, w, h = boxes[i]

label = str(classes[class\_ids[i]])

color = colors[class\_ids[i]]

cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)

cv2.putText(img, label, (x, y + 30), font, 3, color, 3)

#if suspicious activity detected then increment counter

if result == 'Suspicious Activity Detected. Person face covered with mask' and indexes == 0:

suspicious\_counter = suspicious\_counter + 2

elif result == 'Suspicious Activity Detected. Person face covered with mask':

suspicious\_counter = suspicious\_counter + 1

else: #if not detected suspicious activity is decremented

suspicious\_counter = 0 if suspicious\_counter==0 else suspicious\_counter-1

if suspicious\_counter == 7: #if suspicioous activity counter > 7 then send email

sendEmail()

print("Email Sent")

if suspicious\_counter == 15: #if suspicioous activity counter > 15 then ring beep sound

beep()

cv2.putText(img, result, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0), 2)

cv2.imshow("Image", img)

if cv2.waitKey(650) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

## 6.2 Output

The below are the screenshots captured when in various situations.

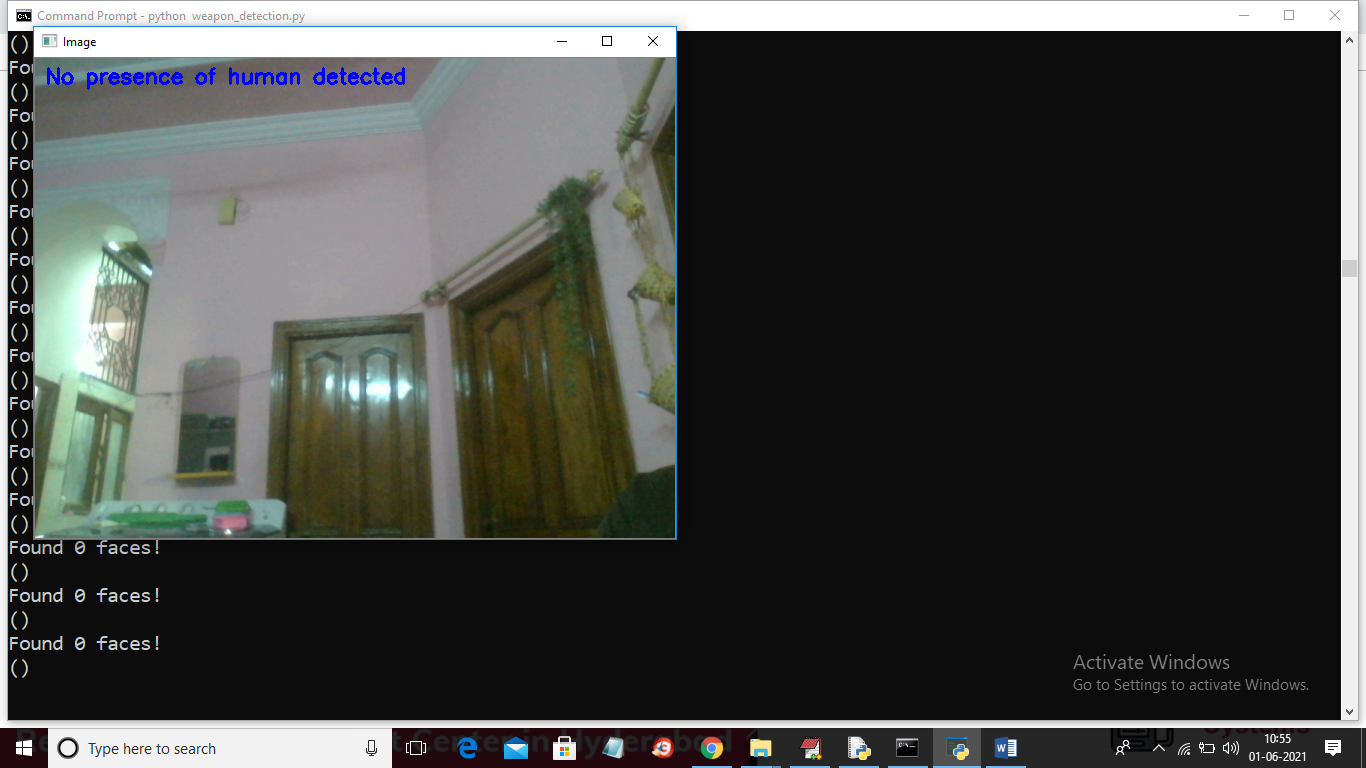


Fig 6.1. There is no human presence and no weapon detected

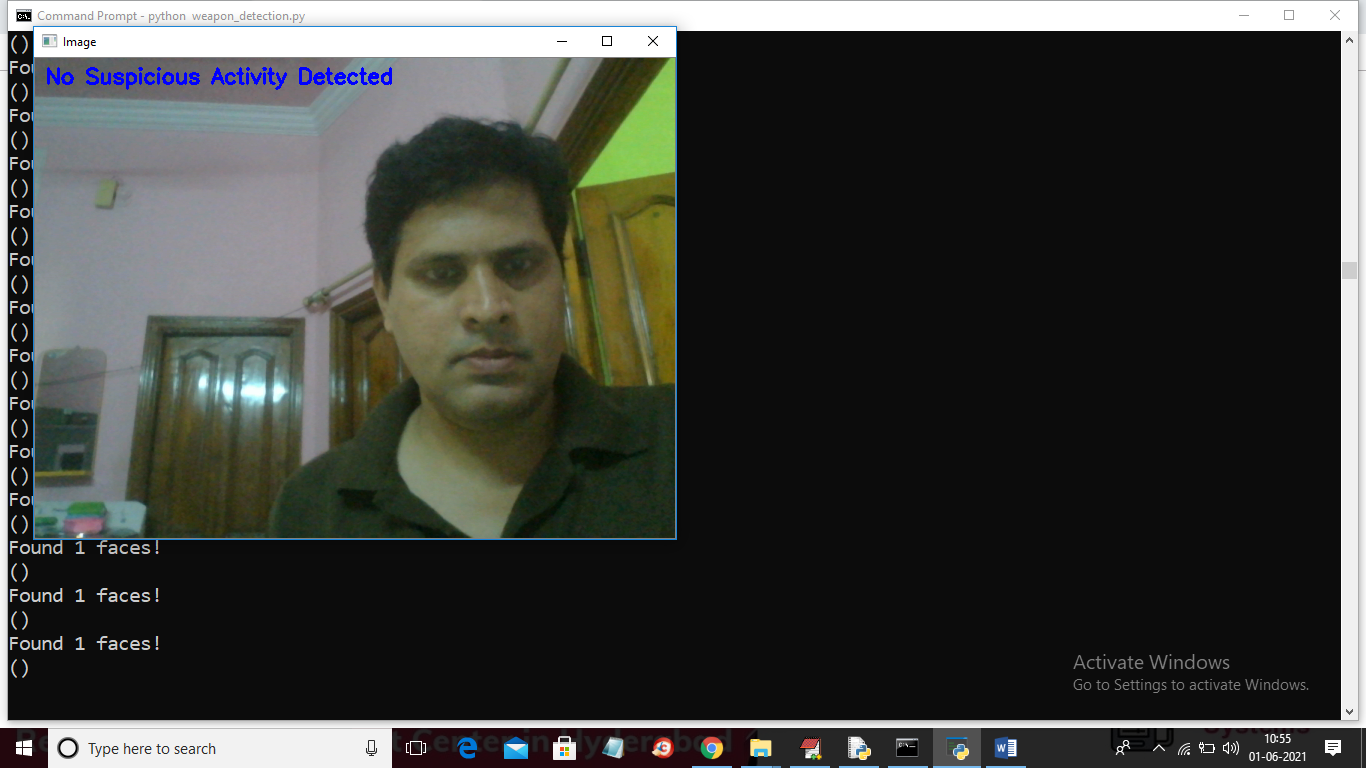


Fig 6.2. When person is detected but is not hiding his face



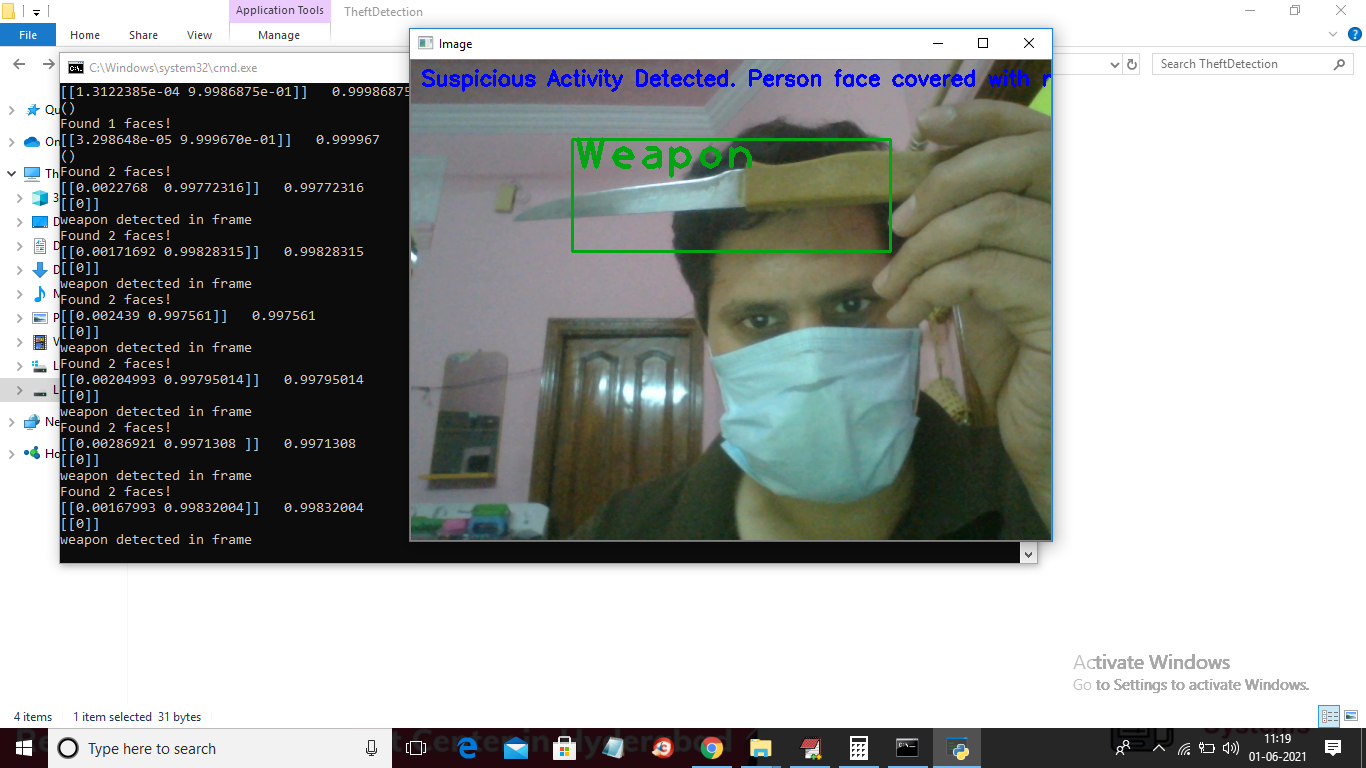


Fig 6.3 & 6.4 When the person with hiding his face is detected and a weapon is detected in the frame.

### 6.2.1 Sending Emails

Below is the sample mail that would be sent to the owner when any suspicious activity occur around the house.

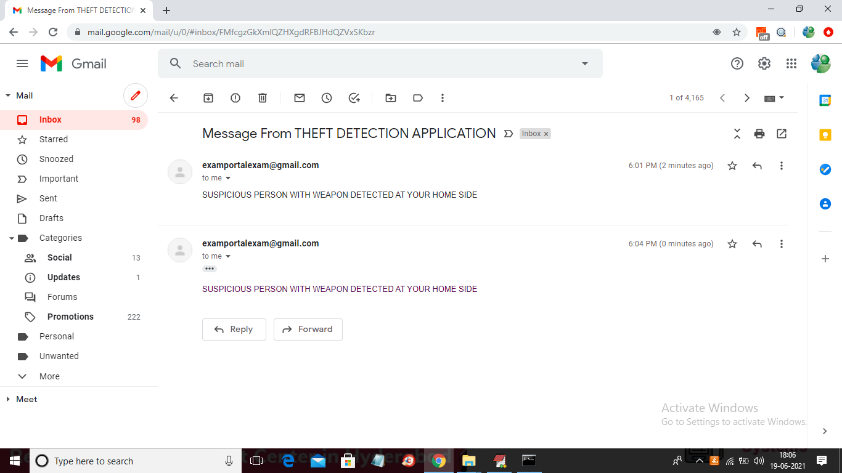


Fig 6.5 Sample of email sent to owner

## 6.3 Accuracy:

The accuracy of the model is found using 3 different metrics: Accuracy and loss. These metrics can be used interchangeably, which means both shows the same values. The accuracy of the model built for identifying of the persons is wearing mask or not is obtained in 10 epochs.

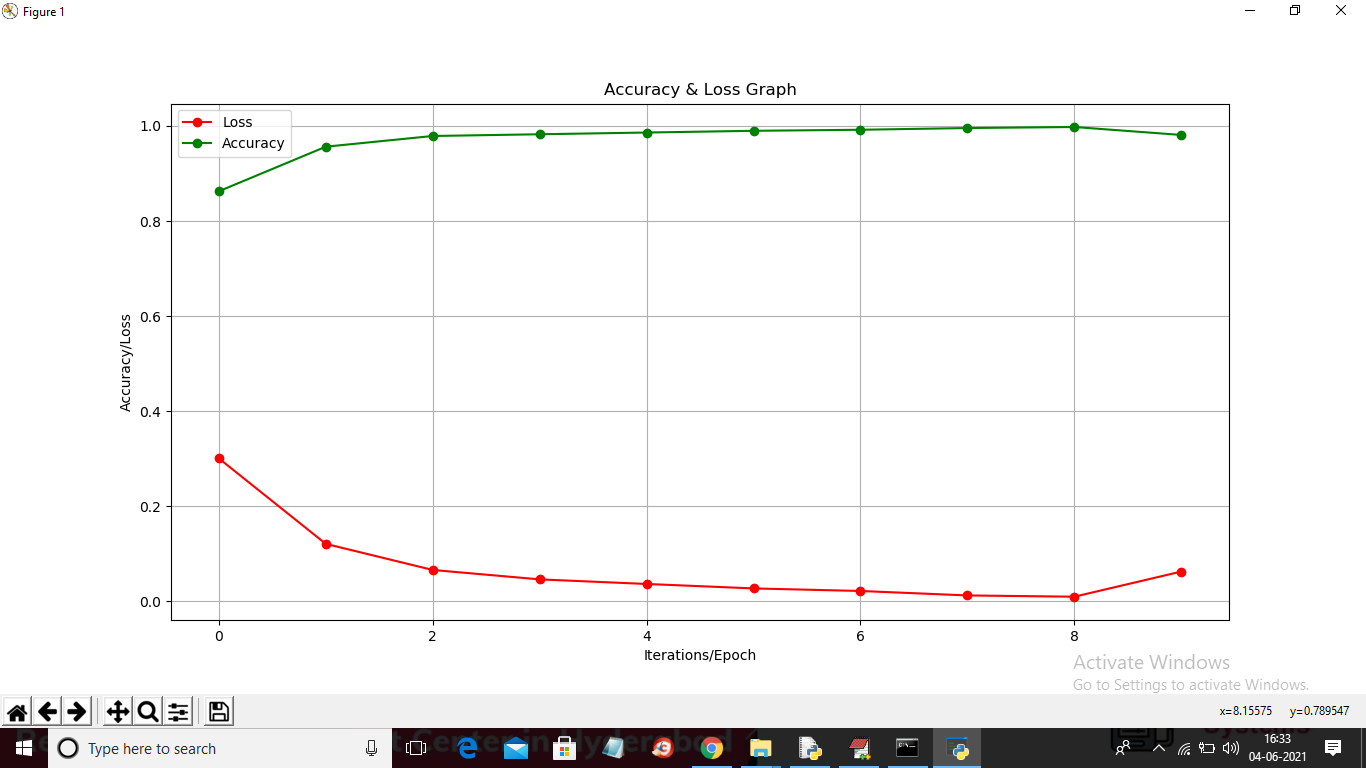


Fig 6.6 Accuracy of Face detection model

The accuracy of the weapon detection model is not used as the images of the weapon are very clear and the accuracy was 100 % for all the epochs.

## 6.4 Results

In this project we were able to derive the expected outcomes. We were able to identify a suspicious person with the help of face mask detection model that we have built and successfully evaluated the model. The weapon detection model which is built to detect any object in the a image has shown fruitful results. Both the models that were built are CNN models and transfer learning is used in both the algorithms for effective results. This project is a combination of major part of Machine Learning and Deep Learning and some of the other components of the python too for sending emails etc., We were about to successfully establish the SMTP connection for sending emails. The winsound which is used for playing alar sound had given a little problem at the begging later it did work as expected.

## 6.5 Conclusion:

The work presented in this project is primarily focused on developing and implementing effective and useful observation frameworks for resolving security issues that can help decrease or prevent theft. The device will only take pictures if there is a human in the frame. As a result, the amount of data will be processed is reduced. It will also save data storage by avoiding the capture of static photos that do not generally contain the object of interest. Users of this system do not need to worry about constantly monitoring the cameras; however, the system will alert them if anything suspicious goes around the house and allows them to act quickly before the property is lost. After completing the project, it can be integrated into a smart surveillance system, which would be extremely useful in detecting auto theft for security reasons.

This work can be further enhanced by trying to capture the facial expressions of the person trying to enter the house and reading those expressions. It is obvious that person trying to break-in to the house will be tensed and in hurry. These symptoms might produce effective results.

Also, the alert message which is sent via mail can be improved to send via other social media like WhatsApp as it is unlikely for someone to check their mail as frequently as the check their other social media accounts. This alert message can also include the image of the visitor which might help the owner identify if the visitor is a threat or not.

# References

1. Roberto Olmos, Siham Tabik, Francisco Herrera, Automatic handgun detection alarm in videos using deep learning, Neurocomputing, Volume 275, 2018, Pages 66-72, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2017.05.012.
2. F. Pérez-Hernández, S. Tabik, A. Lamas et al., Object Detection Binary Classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance, Knowledge-Based Systems (2020) 105590, <https://doi.org/10.1016/j.knosys.2020.105590>.
3. Dinalankara, Lahiru. "Face detection & face recognition using open computer vision classifies." ResearchGate (2017).
4. Nagrath, Preeti, et al. "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2." Sustainable cities and society 66 (2021): 102692.
5. Suresh K1 , Palangappa MB2 , Bhuvan S, “Face Mask Detection by using Optimistic Convolutional Neural Network.” Proceedings of the Sixth International Conference on Inventive Computation Technologies [ICICT 2021] IEEE Xplore Part Number: CFP21F70-ART; ISBN: 978-1-7281-8501-9
6. Rhowel Dellosa, “Development of an Anti-Theft Device using Motion Detection and Body Temperature”, Asia Pacific Journal of Multidisciplinary Research, December 2014. P-ISSN 2350-7756 | E-ISSN 2350-8442
7. Xinyi Zhou, Wei Gong, WenLong Fu, Fengtong Du, “Application of Deep Learning in Object Detection”, International Journal of Innovative Research in Advanced Engineering (IJIRAE), June 2014. ISSN: 2349-2163
8. Yong-Deuk Shin, Jae-Han Park, Ga-Ram Jang, Moon Hong Baeg, “Moving Objects Detection using Freely Moving Depth Sensing Camera”, 21st International Conference on Pattern Recognition, November 2012.
9. Ashwini Patil, Shobha Mondhe, Tejashri Ahire, Gayatri Sonar, “Auto-Theft Detection using Raspberry Pi and Android App”, International Journal of Research in Engineering Application & Management (IJREAM), October 2016. ISSN: 2494-9150
10. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
11. <https://docs.python.org/3/library/smtplib.html>
12. <https://www.guru99.com/keras-tutorial.html>