

WEAPON DETECTION SYSTEM

WEAPON DETECTION SYSTEM



**Session 2017-2020**

**Project Advisor**

**Mr. Farhan Ahmed Siddiqui**

**Submitted By**

|  |  |
| --- | --- |
| Muhammad Sheharyar | B16101114 |

Department of Computer Science

University of Karachi

Karachi

BS

April 07, 2004

A report submitted to the

Department of Computer Science

in partial fulfillment of the requirements for the

degree

Batchelor of Science

in

Computer Science

by

Muhammad Shaheryar

University of Karachi

April 07, 2021

Acknowledgments

I truly acknowledge the cooperation and help make by Dr. Farhan Ahmed Siddiqui, Associate professor of Department of Computer Science, University of Karachi. He has been a constant source of guidance throughout the course of this project. We are also thankful to our friends and families whose silent support led us to complete our project.

(Signed)

Muhammad Sheharyar B-16101114

Date

April 21, 2021

Abstract :

The weapon detection system will detect the weapon in image that comes from video. The weapon detection system is built on YOLO(You Only Look Once) model. YOLO is a real-time, single-stage object detection model which is basically a Convolutional neural networks. The model is trained on 4,500 images. When the model detects the gun, it informs the guard. The weapon detection system will continuously watches images coming from CCTV cameras.

List of Figures :

[Figure 1 : A sample MNIST Dataset 3](#_Toc68479455)

[Figure 2 : Robbery in shop 1a 5](#_Toc68479456)

[Figure 3 : Robbery in shop 1b 6](#_Toc68479457)

[Figure 4 : A CIFAR-10 dataset images 8](#_Toc68479458)

[Figure 5 : Shapes of gun at different angles 9](#_Toc68479459)

[Figure 6 : Data Flow Diagram (Zero Level) 10](#_Toc68479460)

[Figure 7 : Large number of images 12](#_Toc68479461)

[Figure 8 : First Screen 13](#_Toc68479462)

[Figure 9 : Second Screen 14](#_Toc68479463)

[Figure 10 : Original Image 22](#_Toc68479464)

[Figure 11 : The image in which gun is detected by model 23](#_Toc68479465)

List of table

[Table 1 : Confusion Matrix 20](#_Toc68479466)

Table of contents

[Acknowledgments ii](#_Toc68479469)

[Abstract : iii](#_Toc68479470)

[List of Figures : iv](#_Toc68479471)

[List of table v](#_Toc68479472)

[Table of contents vi](#_Toc68479473)

[Chapter 1 : Introduction 1](#_Toc68479474)

[1 Introduction : 2](#_Toc68479475)

[1.1 History : 2](#_Toc68479476)

[1.1.1 Father of Computer Vision(1960) : 2](#_Toc68479477)

[1.1.2 1970’s – MIT’s Artificial Intelligence : 2](#_Toc68479478)

[1.1.3 1980’s – Machine vision starts to take off in the world of research, with new theories and concepts emerging: 2](#_Toc68479479)

[1.1.4 1990’s – Machine vision starts becoming more common in manufacturing environments leading to creation of machine vision industry: 2](#_Toc68479480)

[1.2 Image Recognition And Detection : 2](#_Toc68479481)

[1.3 Different Classifiers : 3](#_Toc68479482)

[Chapter 2 : Requirements and Specifications 4](#_Toc68479483)

[2 Problem Statement : 5](#_Toc68479484)

[2.1 Tense Environment in Robbery : 5](#_Toc68479485)

[Chapter 3 : Analysis and Design 7](#_Toc68479486)

[3 Convolutional Neural Networks(CNN) : 8](#_Toc68479487)

[3.1 Working of CNN : 8](#_Toc68479488)

[3.2 Change In Shape of Guns : 8](#_Toc68479489)

[3.3 Design : 10](#_Toc68479490)

[Chapter 4 : Implementation 11](#_Toc68479491)

[4 Working : 12](#_Toc68479492)

[4.1 Large Dataset(4500 images) : 12](#_Toc68479493)

[4.2 Opencv modules : 12](#_Toc68479494)

[4.3 Tkinter (GUI library for Python) : 13](#_Toc68479495)

[4.4 Code Explanation : 14](#_Toc68479496)

[4.4.1 yolo\_images\_2 Module : 14](#_Toc68479497)

[4.4.2 New\_window module(Graphical User Interface Library) : 16](#_Toc68479498)

[Chapter 5 : Quality Assurance 19](#_Toc68479499)

[5 Different testing techniques 20](#_Toc68479500)

[5.1 Precision : 20](#_Toc68479501)

[5.2 Recall : 20](#_Toc68479502)

[5.3 Accuracy : 20](#_Toc68479503)

[5.4 F1 score : 20](#_Toc68479504)

[Chapter 6 : Results and Conclusion 21](#_Toc68479505)

[6 Result of the model : 22](#_Toc68479506)

[6.1 Some Images From Tained Model : 22](#_Toc68479507)

[6.2 Conclusion : 23](#_Toc68479508)

[Chapter 7 : Future Work 24](#_Toc68479509)

[7 Expansion of the Project : 25](#_Toc68479510)

[References : 26](#_Toc68479511)

Chapter 1 : Introduction

# Introduction :

## History :

### Father of Computer Vision(1960) :

It is commonly accepted that the father of Computer Vision is Larry Roberts, who in his Ph.D. thesis (1960) at MIT discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks (polyhedra) . Many researchers, at MIT and elsewhere, in Artificial Intelligence, followed this work and studied computer vision in the context of the blocks world.

### 1970’s – MIT’s Artificial Intelligence :

1970’s – MIT’s Artificial Intelligence Lab opens a “Machine Vision” course – Researchers begin tackling “real world” objects and “low-level” vision tasks (i.e. edge detection and segmentation: In 1978 a breakthrough was made by David Marr (at the MIT AI lab) who created a bottom up approach to scene understanding through computer vision. This approach starts with a 2D sketch which is built upon by the computer to get a final 3D image.

### 1980’s – Machine vision starts to take off in the world of research, with new theories and concepts emerging:

Optical character recognition (OCR) systems were initially used in various industrial applications to read and verify letters, symbols, and numbers. Smart camera’s were developed in the late 80’s, leading to more wide spread use and more applications.

### 1990’s – Machine vision starts becoming more common in manufacturing environments leading to creation of machine vision industry:

over 100 companies begin selling machine vision systems. LED lights for the machine vision industry are developed, and advances are made in sensor function and control architecture, furthering advancing the abilities of machine vision systems. Costs of machine vision systems begin dropping.

## Image Recognition And Detection :

Image Recognition and detection is a classic machine learning problem. It is a very challenging task to detect an object or to recognize an image from a digital image or a video. Image Recognition has application in the various field of computer vision, some of which include facial recognition, biometric systems, self-driving cars, emotion detection, image restoration, robotics and many more. Deep Learning algorithms have achieved great progress in the field of computer vision. Deep Learning is an implementation of the artificial neural networks with multiple hidden layers to mimic the functions of the human cerebral cortex. The layers of deep neural network extract multiple features and hence provide multiple levels of abstraction. As compared to shallow networks, this cannot extract or work on multiple features. Convolutional neural networks is a powerful deep learning algorithm capable of dealing with millions of parameters and saving the computational cost by inputting a 2D image and convolving it with filters/kernel and producing output volumes.

## Different Classifiers :

The scientist evaluated the performance on MNIST datasets using 3 different classifiers: SVM (support vector machines), KNN (Knearest Neighbor) and CNN (convolutional neural networks). The Multilayer perceptron didn't perform well on that platform as it didn't reach the global minimum rather remained stuck in the local optimal and couldn't recognize digit 9 and 6 accurately. Other classifiers, performed correctly and it was concluded that performance on CNN can be improved.

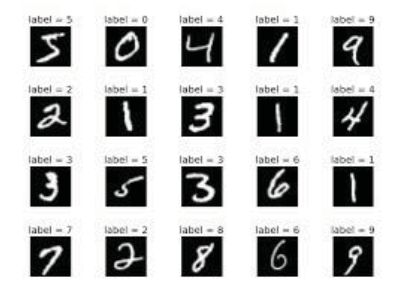


Figure : A sample MNIST Dataset

(Chauhan, Ghanshala, & R.C, December 2018)

Convolutional neural networks are deep learning algorithms that take input as images and convolves it with filters or kernels to extract features. A NxN image is convolved with a fXf filter and this convolution operation learns the same feature on the entire image. The window slides after each operation and the features are learnt by the feature maps. The feature maps capture the local receptive field of th eimage and work with shared.

Chapter 2 : Requirements and Specifications

# Problem Statement :

## Tense Environment in Robbery :

When robbery is occurred in any shop, people are stood still because of the guns hold by robbers. They cannot move because gun is pointed on them.



Figure : Robbery in shop 1a

(Protection)



Figure : Robbery in shop 1b

(Protection)

For this reason, I made weapon detection system which get the images through video ,and process these images by increasing or decreasing brightness and contrast. After these processed images, the model detects guns in these images. If gun detect in an image then software will call police that there are robbers in this shop.

Chapter 3 : Analysis and Design

# Convolutional Neural Networks(CNN) :

Deep Learning algorithms are designed in such a way that they mimic the function of the human cerebral cortex. These algorithms are representations of deep neural networks i.e. neural networks with many hidden layers.

## Working of CNN :

Convolutional neural networks are deep learning algorithms that can train large datasets with millions of parameters, in form of 2D images as input and convolve it with filters to produce the desired outputs. In this article, CNN models are built to evaluate its performance on image recognition and detection datasets. The algorithm is implemented on MNIST and -10 dataset and its performance are evaluated. The accuracy of models on MNIST is 99.6 %, CIFAR-10 is using real-time data augmentation and dropout on CPU unit

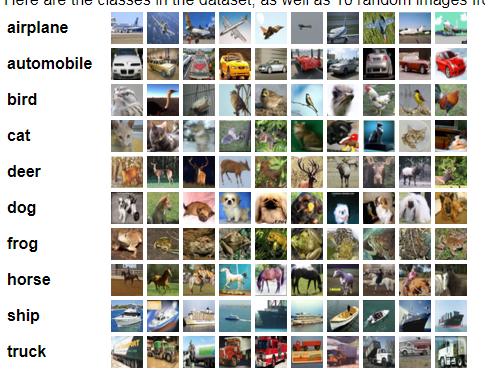


Figure : A CIFAR-10 dataset images

(Chauhan, Ghanshala, & R.C, December 2018)

## Change In Shape of Guns :

The problem, which discusses above, can be solved by convolution neural network. In convolution neural network, the model is trained on images, and these images are tagged through some software.

The convolution neural network model is trained on images. After training the model, the model will detect guns in images which are coming through video. These images are processed by increasing or decreasing brightness and contrast.

The gun is changed its shape at different angles. So it is difficult to detect guns in images. So I made five classes. These five classes have different shapes at angles:

1) Zero

2)45

3)90

4)275

5)315

6)Upper 270 degree

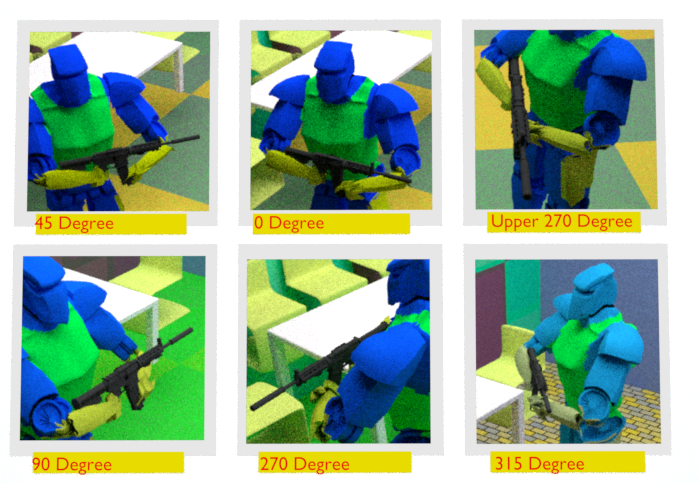


Figure : Shapes of gun at different angles

Each class has almost thousand images.

As you can see that the weapon changes its shape at different angles

## Design :

To understand the system, data flow diagram(DFD) is used. Images will move into system then deep neural network will detect gun in image. Then this image will move into openCV module to show the detected image.

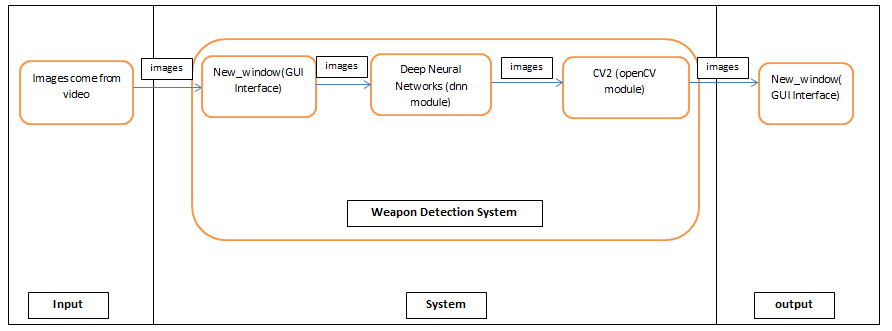


Figure : Data Flow Diagram (Zero Level)

Chapter 4 : Implementation

# Working :

To implement above design, darknet framework is used. it has yolov3 single-stage object detection model.

## Large Dataset(4500 images) :

To Train yolov3, I need images. I cannot find image which have gun at specific position.

So I decided to make images. I used blender to make images. Blender is 3d software for animation, 3d modelling and other artistic works. I have skill in blender. So I made animation in which soldier hold gun at specific position.

The animation produces 4500 images almost.

After making images, I have to tag these images on "lableimg" software. And this part is most time consuming.

After tagging of images, now I have to train model. I use Google colab for training. And it is also very time consuming.



Figure : Large number of images

## Opencv modules :

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

To use opencv in python, CV2 is imported. After importing cv2, images are loaded by glob.glob().cv2.imread() method loads an image from the specified file.

cv2.addWeighted changes the contrast and brightness of image to detect image with more clearly.

cv2.dnn.blobFromImage() takes image, and other parameters to preprocess the image.

After that, self.net.forward() takes image from blob and detects gun in that image. If it finds gun then it provides some value between 0 to 1. If the value is greater or equal to threasold value then the alarm will trigger.

cv2.imshow() will create two windows ,in which one is original and other is modeified.

Tkinter is used for Grapical User Interface. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

## Tkinter (GUI library for Python) :

Tkinter is used to create main window and buttons for user. In main window, there is next button to goto next window.



Figure : First Screen

In 2nd window, there is start button to start the detection.

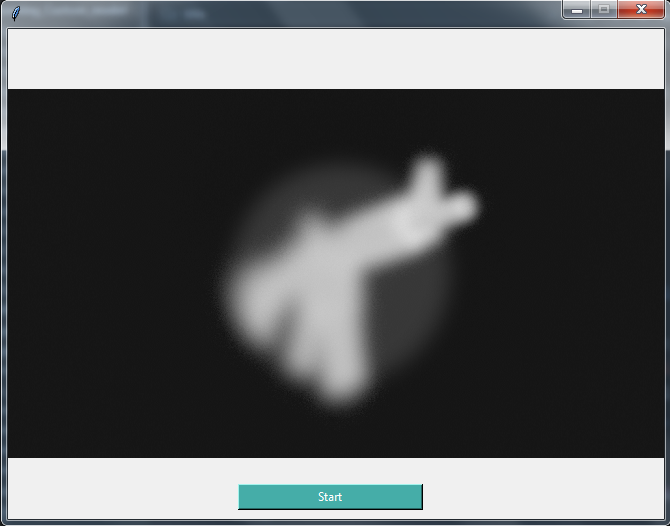


Figure : Second Screen

## Code Explanation :

The code is written in python, and it follows Object Oriented paradigm.Now following modules are explained :

### yolo\_images\_2 Module :

It is custom module in which following modules and classes are used :

**import** cv2

**import** tkinter **as** tk

# import imageProcessing

**import** yolo\_images\_2

# Code to add widgets will go here...

**import** tkinter **as** tk

**class** **Window\_1st:**

**def** \_\_init\_\_**(**self**,** master**):**

self**.**master **=** master

self**.**master**.**geometry**(**'656x490'**)**

self**.**master**.**title**(**""**)**

self**.**master**.**resizable**(False,** **False)**

self**.**filename **=** tk**.**PhotoImage**(**

file **=** "C:/Users/DELL/PycharmProjects/deep\_learning\_Custom\_model/gun4.png"

**)**

self**.**background\_label **=** tk**.**Label**(**

self**.**master**,** image**=**self**.**filename

**)**

self**.**background\_label**.**place**(**

x**=**0**,** y**=**0**,** relwidth**=**1**,** relheight**=**1

**)**

self**.**HelloButton **=** tk**.**Button**(**

self**.**master**,**

text**=**'next'**,**

width**=**25**,**

command**=**self**.**new\_window**,**

**)**

# self.HelloButton.pack()

self**.**HelloButton**.**grid**(**

row**=**3**,** column**=**2**,** padx**=**230**,**pady**=**455

**)**

self**.**HelloButton**.**configure**(**

bg**=**'#45ada8'**,** fg**=**'white'

**)**

#

**def** close\_windows**(**self**):**

self**.**master**.**destroy**()**

self**.**new\_window

**def** new\_window**(**self**):**

self**.**master**.**destroy**()** # close the current window

self**.**master **=** tk**.**Tk**()** # create another Tk instance

self**.**app **=** Window\_2nd**(**self**.**master**)** # create Window\_2nd window

self**.**master**.**mainloop**()**

**class** **Window\_2nd:**

**def** \_\_init\_\_**(**self**,** master**):**

self**.**master **=** master

self**.**master**.**geometry**(**'656x490'**)**

self**.**master**.**title**(**""**)**

self**.**master**.**resizable**(False,** **False)**

self**.**filename **=** tk**.**PhotoImage**(**

file **=** "C:/Users/DELL/PycharmProjects/deep\_learning\_Custom\_model/gun66.png"

**)**

self**.**background\_label **=** tk**.**Label**(**

self**.**master**,** image**=**self**.**filename

**)**

self**.**background\_label**.**place**(**

x**=**0**,** y**=**0**,** relwidth**=**1**,** relheight**=**1

**)**

self**.**quitButton **=** tk**.**Button**(**

self**.**master**,**

text **=** 'Start'**,**

width **=** 25**,**

command **=** self**.**start\_canvas

**)**

self**.**quitButton**.**grid**(**

row**=**3**,** column**=**2**,** padx**=**230**,** pady**=**455

**)**

self**.**quitButton**.**configure**(**bg**=**'#45ada8'**,**fg**=**'white'**)**

self**.**master**.**configure**(**background**=**'red'**)**

**def** close\_windows**(**self**):**

self**.**master**.**destroy**()**

**def** start\_canvas**(**self**):**

**global** lable\_image

obj\_detect **=** yolo\_images\_2**.**Detection**()**

**for** img\_path **in** obj\_detect**.**images\_path**:**

images **=** obj\_detect**.**detect\_images**(**img\_path**)**

# if(obj\_detect.IsDetected):

# winsound.Beep(1000, 1000) # Beep at 1000 Hz for 100 ms

# Rearrang the color channel

# b, g, r = cv2.split(images)

# img = cv2.merge((r, g, b))

# Convert the Image object into a TkPhoto object

# im = Image.fromarray(img)

# imgtk = ImageTk.PhotoImage(image=img)

obj\_detect**.**IsDetected **=** **False**

**if** **(**cv2**.**waitKey**(**1**)** **&** 0xFF **==** ord**(**'q'**)):**

cv2**.**destroyAllWindows**()**

**break**

**def** main**():**

root **=** tk**.**Tk**()**

app **=** Window\_1st**(**root**,)**

root**.**mainloop**()**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

### New\_window module(Graphical User Interface Library) :

In this module, two classes are created for :

1. Window\_1st
2. Window\_2nd

In Window\_1st, the first window will open in which next button is displayed. In Window\_2nd, second window is created by this class in which start button is placed. This button will start the detetcting the guns in images.

**import** cv2

**import** yolo\_images\_2

**import** tkinter **as** tk

# Code to add widgets will go here...

**class** **Window\_1st:**

**def** \_\_init\_\_**(**self**,** master**):**

self**.**master **=** master

self**.**master**.**geometry**(**'656x490'**)**

self**.**master**.**title**(**""**)**

self**.**master**.**resizable**(False,** **False)**

self**.**filename **=** tk**.**PhotoImage**(**

file **=** "C:/Users/DELL/PycharmProjects"

**+**"/deep\_learning\_Custom\_model/gun4.png"

**)**

self**.**background\_label **=** tk**.**Label**(**

self**.**master**,** image**=**self**.**filename

**)**

self**.**background\_label**.**place**(**

x**=**0**,** y**=**0**,** relwidth**=**1**,** relheight**=**1

**)**

self**.**HelloButton **=** tk**.**Button**(**

self**.**master**,**

text**=**'next'**,**

width**=**25**,**

command**=**self**.**new\_window**,**

**)**

# self.HelloButton.pack()

self**.**HelloButton**.**grid**(**

row**=**3**,** column**=**2**,** padx**=**230**,**pady**=**455

**)**

self**.**HelloButton**.**configure**(**

bg**=**'#45ada8'**,** fg**=**'white'

**)**

#

**def** close\_windows**(**self**):**

self**.**master**.**destroy**()**

self**.**new\_window

**def** new\_window**(**self**):**

self**.**master**.**destroy**()** # close the current window

self**.**master **=** tk**.**Tk**()** # create another Tk instance

self**.**app **=** Window\_2nd**(**self**.**master**)** # create Window\_2nd window

self**.**master**.**mainloop**()**

**class** **Window\_2nd:**

**def** \_\_init\_\_**(**self**,** master**):**

self**.**master **=** master

self**.**master**.**geometry**(**'656x490'**)**

self**.**master**.**title**(**""**)**

self**.**master**.**resizable**(False,** **False)**

self**.**filename **=** tk**.**PhotoImage**(**

file **=** "C:/Users/DELL/PycharmProjects/"**+**

"deep\_learning\_Custom\_model/gun66.png"

**)**

self**.**background\_label **=** tk**.**Label**(**

self**.**master**,** image**=**self**.**filename

**)**

self**.**background\_label**.**place**(**

x**=**0**,** y**=**0**,** relwidth**=**1**,** relheight**=**1

**)**

self**.**quitButton **=** tk**.**Button**(**

self**.**master**,**

text **=** 'Start'**,**

width **=** 25**,**

command **=** self**.**start\_canvas

**)**

self**.**quitButton**.**grid**(**

row**=**3**,** column**=**2**,** padx**=**230**,** pady**=**455

**)**

self**.**quitButton**.**configure**(**bg**=**'#45ada8'**,**fg**=**'white'**)**

self**.**master**.**configure**(**background**=**'red'**)**

**def** close\_windows**(**self**):**

self**.**master**.**destroy**()**

**def** start\_canvas**(**self**):**

**global** lable\_image

obj\_detect **=** yolo\_images\_2**.**Detection**()**

**for** img\_path **in** obj\_detect**.**images\_path**:**

images **=** obj\_detect**.**detect\_images**(**img\_path**)**

# if(obj\_detect.IsDetected):

# winsound.Beep(1000, 1000) # Beep at 1000 Hz for 100 ms

# Rearrang the color channel

# b, g, r = cv2.split(images)

# img = cv2.merge((r, g, b))

# Convert the Image object into a TkPhoto object

# im = Image.fromarray(img)

# imgtk = ImageTk.PhotoImage(image=img)

obj\_detect**.**IsDetected **=** **False**

**if** **(**cv2**.**waitKey**(**1**)** **&** 0xFF **==** ord**(**'q'**)):**

cv2**.**destroyAllWindows**()**

**break**

**def** main**():**

root **=** tk**.**Tk**()**

app **=** Window\_1st**(**root**,)**

root**.**mainloop**()**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

Chapter 5 : Quality Assurance

# Different testing techniques

The ‘You Only Look Once’ v3 (YOLOv3) method is among the most widely used deep learning-based object detection methods. It uses the k-means cluster method to estimate the initial width and height of the predicted bounding boxes.

## Precision :

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision(what percent of Positive prediction made were correct):

Precision = TP/TP+FP = 86/(86+10) = 0.89 = 89 %

|  |  |  |
| --- | --- | --- |
|  |  | . |

Precision = TP/TP+FP = 86/(86+10) = 0.89 = 89 %

## Recall :

Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

|  |  |  |
| --- | --- | --- |
|  |  | . |

Recall = TP/TP+FN=86/(86+387) = 86/473 = 0.181 = 18.1 %

## Accuracy :

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

|  |  |  |
| --- | --- | --- |
|  |  | . |

Accuracy =(TP + TN)/(TP+FP+FN+TN)= 86+20/(498)= 0.172 = 17.2 %

## F1 score :

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

|  |  |  |
| --- | --- | --- |
|  |  | . |

F1 = 2\*R\*P/(R+P) = 2\*0.181\*0.89/(0.181+0.89) = 0.32218/1.071 = 0.300 = 30.0 %

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Predicted** | |
| **Actual** |  | **No** | **Yes** |
| **No** | 20[TN] | 10[FP] |
| **yes** | 387[FN] | 86[TP] |

Table : Confusion Matrix

Chapter 6 : Results and Conclusion

# Result of the model :

## Some Images From Tained Model :

The first image is original image taken from mobile camera. And the second image is the gun the detected by model.



Figure : Original Image

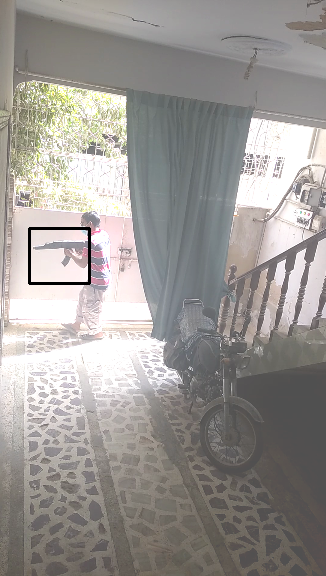


Figure : The image in which gun is detected by model

## Conclusion :

If we see the result of precision which is “89”, then the project is very efficient. But recall is very low ,18%.

Therefore, F-1 provides a way to combine both precision and recall into a single measure that captures both properties.The f1 score is 30% which is very low.

To improve the model, we have to tain it on large set of data like each class will have 10,000 images, so five classes will have 50,000 images.

Chapter 7 : Future Work

# Expansion of the Project :

In the future work, it calls the police when it detects the gun in CCTV video, and further if we expand, then it eliminate the terrorist by drone.

The system will continuously watch the environment. If the robbers enter in shop, mall or other places, then they show guns, then the high power computer which controls drones will send to stop by electric shock or by eliminating the target.

References :

[1] Chauhan, R., Ghanshala, K. K., & R. J. (December 2018). *Convolutional Neural Network (CNN) for Image Detection and Recognition.*

[2] Protection, A. S. (n.d.). *Active Self Protection*. Retrieved from Active Self Protection: https://www.youtube.com/watch?v=V88k2iaL0Fw