

# **Weapon Detection using Deep Learning**

A PROJECT REPORT

*Submitted by*

**Carlin Fernandes [Reg No: RA1911003010129]**

**Adit Mahajan [Reg No: RA1911003010497]**

*Under the Guidance of*

**DR. A. PANDIAN**

Associate Professor, Department of Computing Technologies

*In partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**



**DEPARTMENT OF COMPUTING TECHNOLOGIES  
COLLEGE OF ENGINEERING AND TECHNOLOGY  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR – 603 203**

**MAY 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR – 603 203**

**BONAFIDE CERTIFICATE**

Certified that this B.Tech project report titled “Weapon Detection Using Deep Learning” is the bonafide work of Mr. Carlin Fernandes [RA1911003010129] and Adit Mahajan [RA1911003010497] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

**DR. A. PANDIAN**

**SUPERVISOR**

Associate Professor

Department of Computing Technologies

**DR. A. PANDIAN**

**PANEL HEAD**

Associate Professor

Department of Computing Technologies

**DR. M. PUSHPALATHA**

**HEAD OF THE DEPARTMENT**

Department of Computing Technologies

**SIGNATURE OF INTERNAL EXAMINER**

**SIGNATURE OF EXTERNAL EXAMINER**



Department of Computing Technologies

**SRM Institute of Science and Technology**

**Own Work Declaration Form**

**Degree/ Course** : B.Tech in Computer Science and Engineering

**Student Names** : Carlin Fernandes, Adit Mahajan

**Registration Number:** RA1911003010129, RA1911003010497

**Title of Work** : Weapon Detection using Deep Learning

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own

- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

**DECLARATION:**

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

**Carlin Fernandes [RA1911003010129]**

**Adit Mahajan [RA1911003010497]**

## ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

I am incredibly grateful to my Head of the Department, **Dr. M. Pushpalatha**, Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, **Dr. M. Kanchana**, Assistant Professor, Panel Head, **Dr. A. Pandian**, Associate Professor and members, **Dr. R. Thenmozhi**, **Dr. R. Vidhya**, Assistant Professor and **Dr. P. Velmurugan**, Assistant Professor, and **Dr.S.Jeeva**, Assistant Professor, Department of C.Tech, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Karthikeyan U**, Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. A. Pandian**, Associate Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

I sincerely thank the Department of Computing Technologies' staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

**Carlin Fernandes [RA1911003010129]**

**Adit Mahajan [RA1911003010497]**

## **ABSTRACT**

One of the most fundamental and challenging issues in computer vision is object recognition, which aims to identify instances of objects from the vast categories of already defined and easily accessible natural images. To accomplish machine vision understanding, the object detection approach tries to identify all the objects or entities in the provided image and determine the categories and location information. It is a fundamental problem that identifies objects within images and videos and determining its precise location. The goal of object recognition is to enable machines to automatically understand and interpret the visual world in a similar manner to that of humans. It also has many practical applications such as autonomous driving, surveillance systems and medical imaging. It also does involve processing an input image or video stream and extracting features that can be used to distinguish between different objects. These features are used to train the ML models like SVM, CNN and DNNs. In this project, we build a system that is able to detect weapons in surveillance systems using YOLOv3 algorithm, it's a new state-of-the-art for real-time object detectors released in 2022. The aim of this project is to investigate the effect of training YOLOv3 with two classes "Weapons" and "Shotgun" on detecting weapons in Realtime.

Keywords: YOLO, Object Recognition, Object Detection, Real-time Detection.

# TABLE OF CONTENTS

C.NO	TITLE	PAGE NO.
	Abstract	vi
	List of Tables	
	List of Figures	viii
	List of Symbols and Abbreviations	ix
1.	INTRODUCTION	1
1.1	Problem Statement	1
1.2	Objective	2
1.3	Scope	2
1.4	Machine Learning	3
1.5	Deep Learning	6
1.6	YOLO (You Look Only Once)	8
2	LITERATURE REVIEW	9
2.1	Related Work	9
3	SYSTEM ARCHITECTURE AND DESIGN	16
3.1	Architecture of YOLO Model	17
3.2	Working of the YOLO Object Detection	18
3.3	YOLOv3 Algorithm steps	19
3.4	Drawbacks of YOLOv3 Model	20
3.5	Overcoming the Drawbacks of YOLOV3 Model	21
3.6	Design of the Model	21
3.7	Requirements of Model	23
	3.7.1 Software Requirements	23
	3.7.2 Hardware Requirements	23
4	METHODOLOGY	24
4.1	Features of model	24
4.2	Dataset Collection and Processing	25
4.3	Data preparation, Variable selection	25
4.4	Splitting of Data	26
4.5	Training and validation of the model	29



5	SOURCE CODE AND CODING METHODS	30
5.1	Coding methods and procedures	30
6	RESULTS	47
6.1	Introduction	47
6.2	Accuracy and Performance Evaluation of the model	48
7	CONCLUSION	51
7.1	Conclusion	51
7.2	Future Enhancement	52
8	REFERENCES	53
	APPENDIX 1	58
	PAPER PUBLICATION STATUS	59
	PLAGIARISM REPORT	63

## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>FIGURE NAME</b>	<b>PAGE NO.</b>
1.1	Machine Learning Workflow	4
1.2	Flowchart on Steps of Machine Learning	5
1.3	Deep Learning Layers Diagram	7
3.1	Flow diagram of the model	17
4.1	Features of the YOLO Model	26
4.2	Flowchart of different Models in fps	30
5.1	List of Classes of YOLO	32
5.2	List of Classes of YOLO	32
5.3	Object.Name File of YOLO Model	36
5.4	Object.data of YOLO Model	36
5.5	Darknet Files	38
5.6	Shotgun Image	42
5.7	Video Detection Image	46
5.8	Different Weapon Detection	47
6.1	Video Detection	49
6.2	Handgun Detection	50

## LIST OF SYMBOLS AND ABBREVIATIONS

<b>YOLO</b>	You Look only Once
<b>DL</b>	Deep Learning
<b>CNN</b>	Convolutional neural network
<b>CONFIG</b>	Configuration File
<b>ML</b>	Machine Learning
<b>OpenCV</b>	Open Computer Vision Library
<b>V3</b>	Version 3

# **Chapter 1**

## **INTRODUCTION**

### **1.1 PROBLEM STATEMENT**

This effort is being done with the intention of decreasing incidents by real-time detection of various weapons. An early alarm system can be used to control these situations by notifying the operators and relevant authorities so that quick action can be taken. The majority of modern surveillance systems rely on a human component for monitoring. As a result, monitoring is a very difficult process. It requires a lot of labor and is error-prone. These conventional systems suffer from a number of issues, including poor stability, excessive costs, poor intelligence, and weak security. The majority of these systems are run by people. These operators find it challenging to monitor and assess all potentially hazardous events, especially given the extended observation times and numerous cameras. Therefore, it is essential to review the current surveillance system once more in light of the adoption of new technology. Another way to prevent crimes from occurring is to enable security officials to act quickly by early detection of the situation. Installing security cameras with an alert system is another option. In real time, we humans can quickly recognize the objects in an image, but machines have a harder job before CNN. In 2012, Alex Net introduced CNN, and since then, most things have changed. We can state that modern machines are faster and more accurate than humans because many improvements have been made. Due to the rising crime rate in many large cities and isolated places, security has always been a top priority in every field. Computer vision is widely used in abnormal detection and monitoring to combat numerous issues. The deployment of surveillance in video systems can recognize the events happening and plays a significant part in intelligence monitoring due to the growing demand in protection of safety, security, and personal.

## **1.2 OBJECTIVE**

A Weapon detection system can provide advance detection of dangerous situations that is important for public wellbeing. The way stops it from happening is to detect harmful objects like example shotguns and other weapons in CCTV videos. In order to train the object detector, we need to supervise the learning with box annotations. We draw a box around each object that we want the detector to see and label it with a class that the detector would predict. There are various tools available to do the labelling eg: LabelImg or we can collect the dataset along with their labelled boxes and train the model. The primary objective of an object detection model is to accurately identify the location and class of objects within an image or video stream. Object detection models are trained to detect objects within images and videos by analyzing the input data and identifying specific patterns and features associated with objects of interest. The main goal of object detection models is to achieve high detection accuracy. The specific objectives of an object detection model can vary depending on the application. For example, in autonomous driving systems, the objective may be to detect vehicles, and other objects on roads to avoid collisions. In medical imaging, the objective may be to detect tumors or other abnormalities in medical images to aid in diagnosis. In surveillance systems, the objective may be to detect and track people, vehicles, or other objects of interest. Overall, the primary objective of an object detection model is to accurately and efficiently detect objects within images and video streams, with the ultimate goal of enabling machines to understand and interpret the visual world in a manner similar to humans. The YOLO model is then installed for object detection and the images are then inputted for the detection and the output is produced.

## **1.3 SCOPE**

An Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them.

YOLO, an acronym for 'You only look once,' is an open-source software tool utilized for its efficient

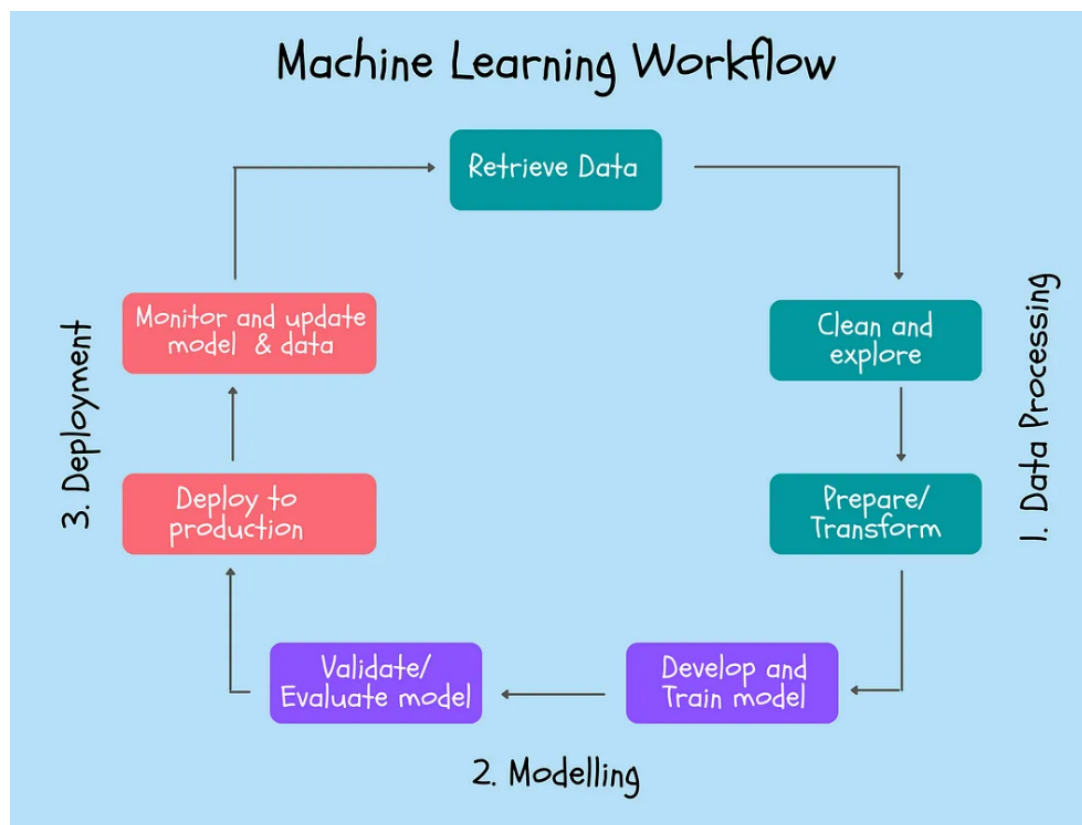
capability of detecting objects in a given image in real time. The YOLO (You Only Look Once) algorithm is an algorithm that detects object which uses convolutional neural networks (CNN) as objects in real time are detected. The algorithm requires a single forward propagation through one neural network in order to detect objects. This means that prediction is done in a single algorithm run. The core working of the YOLO model relies upon its detection technique which brings together different object detection into single neural network.

## **1.4 MACHINE LEARNING**

Machine learning can be defined as the field of study that assists the computers with the ability to learn without the need to hardcode in every scenario. As the name says it is the ability to grant a machine to behave as closely to a human learning ability. In today's environment machine learning is used in wide range of applications. It is concerned with developing programmers that can learn by getting access to the data and doing logic on their own. Until now the data analysis has been in trial and error but this technique will get difficult as the datasets will become larger and larger. This is where machine learning can help to identify alternatives to analyze larger amounts of data. It can also deliver accurate findings by utilizing efficient and fast algorithms as well as real time processing. the main goal of machine learning is to enable computers to learn for themselves automatically without the need for user intervention or human aid. This method starts with data instances like recognizing patterns in data pools and make conclusions based on the examples provided. There are countless uses for machine learning. A few Commonly used models are:

- Support Vector Machines (SVM Support Vector Machine): it is one of the most popular supervised learning algo which is used for classification and regression problems. however, it is used mostly for classification problems in ML. The goal of SVM algo is to create the best line that can segregate n dimensional spaces into classes so that we can easily put the new data point in the correct category.
- Decision trees: It is a powerful and popular tool for classification and prediction. It is a flowchart tree like structure where each internal node denotes a test on attribute and each branch represents a outcome of test and each node holds a class label.

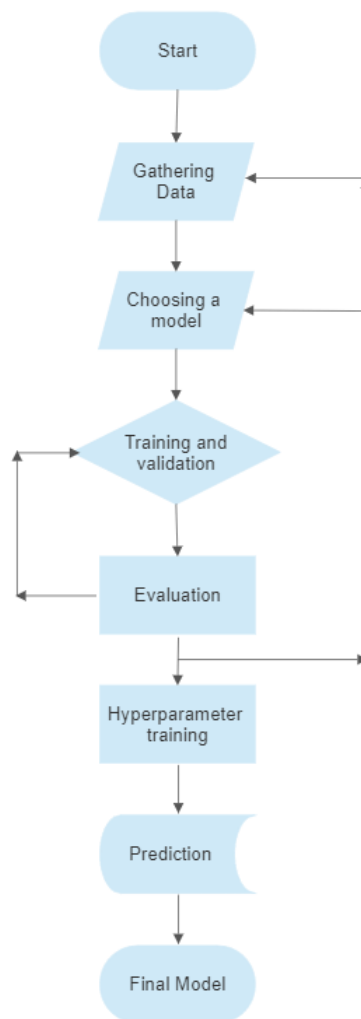
- K means clustering it is model that works by grouping data points into groups based on similar characteristics.
- Neural Networks: it is known as ANNs or SNNs are a subset of machine learning and are at heart of deep learning algo.
- Reinforcement Learning: It is an area of ML that concerned with how intelligent agents take action in environment to maximize the reward.



**Fig 1.1**

## Flowchart on steps of machine learning-

1. Gathering data
2. Choosing a model
3. Training and validation
4. Evaluation
5. Hyperparameter training
6. Prediction
7. Final Model

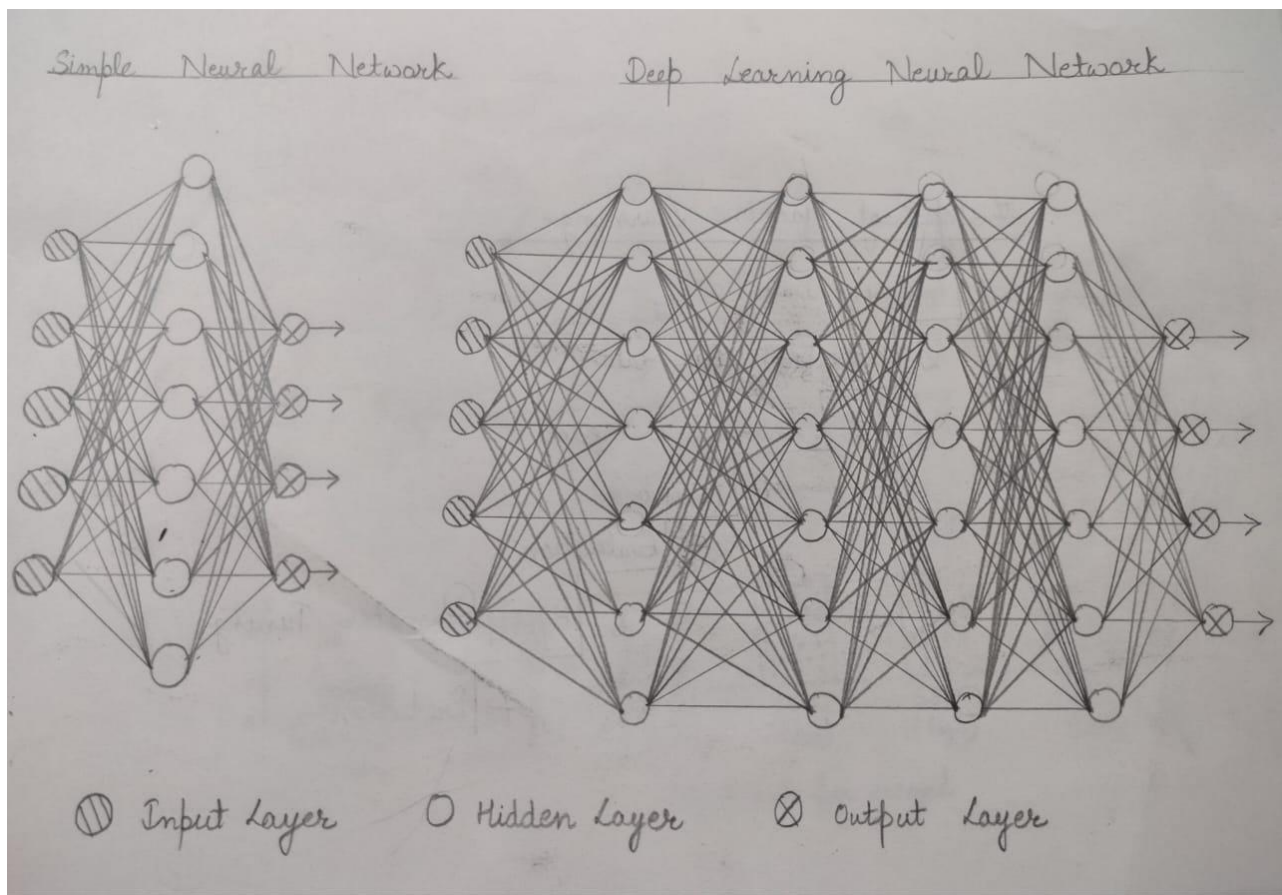


**Fig 2.2**



## 1.5 DEEP LEARNING

It is an AI feature that deals with the simulation of the learning process that a human would utilize to gather information and develop nuanced abilities. At its core, it is a way for automating a prediction-based analysis that human brain is capable of performing, without the need to hardcode every situation and by knowing the paths of outcomes to various logics. A classic machine learning algorithm is a linear learning process. Deep learning algorithms propose a hierarchical stacking strategy depending on the complexity and abstraction of the input. To understand the process of Evolution of deep learning, let us assume that we have some images, each of which contains a category of objects (out of four categories, for the sake of this example), and our requirement is that the algorithm detects the category of an object in any of the images. We first create a data set by labeling the images, so that the network can be trained using this set. The network would then start to recognize unique features and correlate them with a category. Successive layers use data from preceding layers, and pass this on to the next. The complexity of learning and the details increases as data moves from layer to layer. When deep learning is utilized, the method is nearly identical, which is as follows: the algorithm acquires the data, and this data is subjected to non-linear transformations. The transformations are used to learn, and the output is obtained as a model. This is repeated through numerous layers and a plethora of trials until a dependable and accurate output is obtained. When using an ML algorithm, the magnitude of specificity provided by the user must be very high, because the computer cannot interpret what it has to search for with lower levels of complexities. It is difficult to provide such high levels of accuracy because it necessitates manual inputs and the rate of success is totally dependent on the user's ability to give precise differentiation. This is where the utility of DL comes into play, as it can illustrate feature sets on its own, with high levels of accuracy, without the need for manual input or control.



**Fig 1.3**

A DL algorithm follows a course of operation very similar to what can be found in the brain, i.e, a network of neurons. This is why DL networks are also known as deep neural networks. Here, we can see an instance of bio-mimicry, since the path of operation of the algorithms is structured like a set of neurons interlinked to one another, such that a following layer takes the output of the previous one, and so on. Unique countenances of the data are interpreted by unique layers, the collective of which forms the entire network. To extend the understanding of neural networks further, let us assume a scenario, where the ML program is designed to distinguish manually written alphabets. Sequencing of the layers could be in several ways, for example, the initial layer would handle the recognition of grayscale percentage, and chromaticity. A successive layer could handle the recognition of the structure of the alphabet based on contours, and another successive layer could handle recognition of an overall resemblance to an alphabet. This procession continues throughout all the layer, and at the end, an output is obtained. This would be the

possibility of the written alphabet is any one from a to z. Learning is done sequentially. The network determines how to perceive individual characteristics by progressively changing the importance of a certain characteristic data while passing through layers. A certain “weight” is attributed to every link, the equivalence of which can be changed to modify the link’s importance. Ultimately, the closeness of the output with the actual data is determined, after every learning instance.

## **1.6 YOLO (You Look only Once)**

YOLO it is an object detection algorithm that was first introduced in 2016. It revolutionized the field of object detection by being able to perform real-time object detection with a high accuracy on single GPU. There have been several versions of YOLO with each having improvements and modifications over the previous versions. Here is a brief overview of the different versions in YOLO:

1. YOLOv1: This was the first version and it was introduced in 2016. It used a single neural network to perform object detection and classification. The YOLOv1 divided the input image into a grid of cells and predicted bounding boxes and class probabilities for each cell. However, YOLOv1 suffered from accuracy issues, especially for small objects.
2. YOLOv2: YOLOv2 was introduced in 2017 and addressed some of the issues with YOLOv1. YOLOv2 used a more complex network architecture with skip connections and anchor boxes to improve accuracy. It also used a technique called batch normalization to make the network more robust to changes in lighting and contrast.
3. YOLOv3: YOLOv3 was introduced in 2018 and is the most widely used version of YOLO. It further improved on YOLOv2 by using a feature extraction network called Darknet-53 and introducing a technique called multi-scale prediction. This allowed YOLOv3 to detect objects at different scales and improve its accuracy for small objects.

## **CHAPTER-2**

### **LITERATURE REVIEW**

In this section, It is discussed the existing research that has been conducted in the field of object detection

#### **2.1 RELATED WORKS**

The paper "Real-Time Abnormal Object Detection for Video Surveillance" addresses the issue of detecting abnormal objects in video surveillance systems, which is a critical task for ensuring the safety and security of public spaces. The authors propose an approach that combines deep learning and background subtraction techniques to detect abnormal objects in real-time.

The paper begins by discussing the importance of video surveillance in public spaces, and how the detection of abnormal objects can play a vital role in preventing security breaches. The authors then review the existing literature on abnormal object detection, highlighting the limitations of current approaches, including their inability to handle complex scenarios or real-time processing. Next, the paper introduces the proposed approach, which combines a convolutional neural network (CNN) with a background subtraction technique. The CNN is trained on a large dataset of normal objects, and is used to classify objects in real-time video frames as either normal or abnormal. The background subtraction technique is used to subtract the stationary background from the video frames and isolate moving objects, which are then fed into the CNN for classification. The paper then presents the experimental results of the proposed approach, which were conducted on a publicly available dataset of abnormal objects. The results show that the proposed approach achieves high accuracy and detection rates, even in challenging scenarios such as crowded spaces or objects partially occluded by other objects.

Finally, the paper concludes by summarizing the contributions of the proposed approach, including its

real-time processing capabilities, high accuracy, and ability to handle complex scenarios. The authors also suggest potential future directions for research in this area, such as improving the robustness of the approach to variations in lighting and weather conditions. Overall, the paper provides a comprehensive review of the existing literature on abnormal object detection and presents a novel approach that addresses the limitations of current approaches. The experimental results demonstrate the effectiveness of the proposed approach, making it a valuable contribution to the field of video surveillance [1]

The paper "Train your Own Object Detection Model" addresses the issue of object detection, which is a fundamental task in computer vision with a wide range of applications. The authors propose a methodology for training custom object detection models using deep learning, which can be applied to a variety of real-world scenarios. The paper begins by discussing the importance of object detection and the limitations of existing approaches, such as pre-trained models that may not be optimized for specific tasks or scenarios. The authors then review the existing literature on object detection, highlighting the use of deep learning techniques such as convolutional neural networks (CNNs) and the popular frameworks used to implement them, such as TensorFlow and Keras. Next, the paper presents the proposed methodology for training custom object detection models, which involves several key steps such as data preparation, model selection, training, and evaluation. The authors provide detailed explanations and examples of each step, including how to label images with object annotations, how to select and fine-tune pre-trained models, and how to evaluate the performance of the trained model. The paper then presents several case studies to demonstrate the effectiveness of the proposed methodology in real-world scenarios, such as detecting vehicles on a highway, recognizing faces in a crowd, and detecting food items in a restaurant. The authors compare the performance of their custom models to pre-trained models and show that the custom models achieve higher accuracy and are better suited for the specific tasks at hand. Finally, the paper concludes by summarizing the contributions of the proposed methodology, including its flexibility

to adapt to a wide range of scenarios and its ability to achieve high accuracy with relatively small datasets. The authors also suggest potential future directions for research in this area, such as improving the efficiency of the training process and exploring new techniques for data augmentation. Overall, the paper provides a comprehensive review of the existing literature on object detection and presents a practical methodology for training custom models using deep learning techniques. The case studies demonstrate the effectiveness of the proposed methodology in real-world scenarios, making it a valuable contribution to the field of computer vision.[2]

The paper "Weapons Detection for Security and Video Surveillance Using CNN and YOLO-V5" addresses the issue of detecting weapons in video surveillance for security purposes. The authors propose an approach that combines deep learning techniques, specifically convolutional neural networks (CNNs) and YOLO-V5s, for accurate and real-time weapons detection.

The paper begins by discussing the importance of weapons detection in video surveillance and the challenges of accurately detecting weapons in real-world scenarios. The authors then review the existing literature on weapons detection, highlighting the limitations of current approaches, such as the high false positive rates and the inability to detect concealed weapons.

Next, the paper presents the proposed approach, which involves two key steps: pre-processing and deep learning-based detection. The pre-processing step involves image resizing and normalization, followed by region of interest (ROI) extraction. The ROI is then passed through a CNN model that has been trained on a large dataset of images containing weapons. The authors use a YOLO-V5 model to further refine the detection and reduce false positive rates.

The paper then presents the experimental results of the proposed approach, which were conducted on a dataset of images containing both visible and concealed weapons. The results show that the proposed approach achieves high accuracy and detection rates, with low false positive rates. The approach is also shown to be robust to variations in lighting conditions and occlusions. Finally, the paper concludes by

summarizing the contributions of the proposed approach, including its ability to achieve high accuracy and real-time processing, and its ability to detect concealed weapons. The authors suggest potential future directions for research in this area, such as exploring the use of other deep learning techniques and improving the efficiency of the approach for large-scale deployments. Overall, the paper provides a comprehensive review of the existing literature on weapons detection and presents a novel approach that addresses the limitations of current approaches. The experimental results demonstrate the effectiveness of the proposed approach, making it a valuable contribution to the field of video surveillance and security.[3]

The paper "Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4" addresses the issue of detecting whether individuals are wearing face masks in real-time using computer vision. The authors propose an approach that combines deep learning techniques, specifically an improved version of YOLO-v4, for accurate and efficient face mask wearing detection. The paper begins by discussing the importance of face mask wearing detection in the context of the COVID-19 pandemic and the challenges of detecting face masks in real-world scenarios. The authors then review the existing literature on face mask detection, highlighting the limitations of current approaches, such as the inability to detect masks with high accuracy and the lack of real-time performance. Next, the paper presents the proposed approach, which involves several key steps: data preparation, model training, and model testing. The authors use a dataset of images containing individuals with and without face masks to train the improved YOLO-v4 model, which has been optimized for object detection tasks. The paper then presents the experimental results of the proposed approach, which were conducted on a real-world dataset of video frames containing individuals with and without face masks. The results show that the proposed approach achieves high accuracy and detection rates, with low false positive rates. The approach is also shown to be robust to variations in lighting conditions and occlusions. Finally, the paper concludes by summarizing the contributions of the proposed approach, including its ability to achieve high accuracy and real-time processing, and its potential for large-scale deployment in public settings. The authors suggest potential future directions for research in

this area, such as improving the efficiency of the approach and exploring the use of other deep learning techniques. Overall, the paper provides a comprehensive review of the existing literature on face mask detection and presents a novel approach that addresses the limitations of current approaches. The experimental results demonstrate the effectiveness of the proposed approach, making it a valuable contribution to the field of computer vision in the context of the COVID-19 pandemic.[4]

The paper "Passenger flow counting in buses based on deep learning using surveillance video" proposes a deep learning-based approach for counting passenger flow in buses using surveillance video. The authors discuss the importance of passenger counting in public transportation and the challenges associated with accurately detecting passengers in crowded and dynamic environments. The paper presents a comprehensive review of the existing literature on passenger counting in buses. The authors highlight the limitations of traditional approaches, such as manual counting, which are time-consuming and prone to errors, and other automated methods that rely on sensors, which can be expensive and require frequent calibration. The proposed approach involves using deep learning techniques, specifically convolutional neural networks (CNNs), to detect and count passengers in surveillance videos. The authors use a dataset of surveillance videos containing passengers getting on and off buses to train the CNN model, which is optimized for object detection tasks. The experimental results demonstrate the effectiveness of the proposed approach, which achieved high accuracy and detection rates with low false positives. The approach is also shown to be robust to variations in lighting conditions and occlusions, making it suitable for real-world scenarios. The paper concludes by summarizing the contributions of the proposed approach and its potential for large-scale deployment in public transportation settings. The authors suggest future directions for research, such as exploring the use of other deep learning techniques and improving the efficiency of the approach for large-scale deployments. Overall, the paper provides a valuable contribution to the field of computer vision in the context of public transportation. The proposed approach offers a more efficient and accurate method for passenger flow counting, which can help improve the safety and



efficiency of public transportation systems.[5]

The paper "Real-time clothing recognition in surveillance videos" by M. Yang and K. Yu addresses the problem of recognizing clothing items in real-time using surveillance videos. The authors propose an approach that combines feature extraction and classification techniques to identify clothing items in surveillance videos. The paper begins by discussing the importance of clothing recognition in surveillance videos and the challenges associated with accurately detecting and identifying clothing items. The authors review the existing literature on clothing recognition, highlighting the limitations of current approaches, such as the inability to handle variations in lighting conditions and occlusions. Next, the paper presents the proposed approach, which involves several key steps: feature extraction, feature selection, and classification. The authors use a dataset of surveillance videos containing people wearing different types of clothing items to train the model, which has been optimized for clothing recognition tasks. The experimental results demonstrate the effectiveness of the proposed approach, which achieved high accuracy and recognition rates, even in real-time scenarios. The approach is also shown to be robust to variations in lighting conditions and occlusions, making it suitable for real-world scenarios. The paper concludes by summarizing the contributions of the proposed approach and its potential for large-scale deployment in surveillance systems. The authors suggest future directions for research, such as exploring the use of other deep learning techniques and improving the efficiency of the approach for large-scale deployments. Overall, the paper provides a valuable contribution to the field of computer vision in the context of surveillance systems. The proposed approach offers a more efficient and accurate method for clothing recognition, which can help improve the safety and security of public spaces.[6]

The paper "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow" by R. Phadnis, J. Mishra, and S. Bendale addresses the problem of object detection and pattern tracking using TensorFlow. The authors propose an approach that combines deep learning-based object detection and tracking algorithms to track objects in real-time using surveillance videos.

The paper begins by discussing the importance of object detection and pattern tracking in surveillance systems and the challenges associated with accurately detecting and tracking objects. The authors review the existing literature on object detection and tracking, highlighting the limitations of current approaches, such as the inability to handle complex scenes and occlusions.

Next, the paper presents the proposed approach, which involves several key steps: object detection using the Single Shot Multibox Detector (SSD) algorithm, object tracking using the Correlation Filter Tracker (CFT) algorithm, and feature extraction using the Convolutional Neural Network (CNN) algorithm. The experimental results demonstrate the effectiveness of the proposed approach, which achieved high accuracy and tracking rates, even in real-time scenarios. The approach is also shown to be robust to variations in lighting conditions and occlusions, making it suitable for real-world scenarios. The paper concludes by summarizing the contributions of the proposed approach and its potential for large-scale deployment in surveillance systems. The authors suggest future directions for research, such as exploring the use of other deep learning techniques and improving the efficiency of the approach for large-scale deployments. Overall, the paper provides a valuable contribution to the field of computer vision in the context of surveillance systems. The proposed approach offers a more efficient and accurate method for object detection and pattern tracking, which can help improve the safety and security of public spaces.[7]

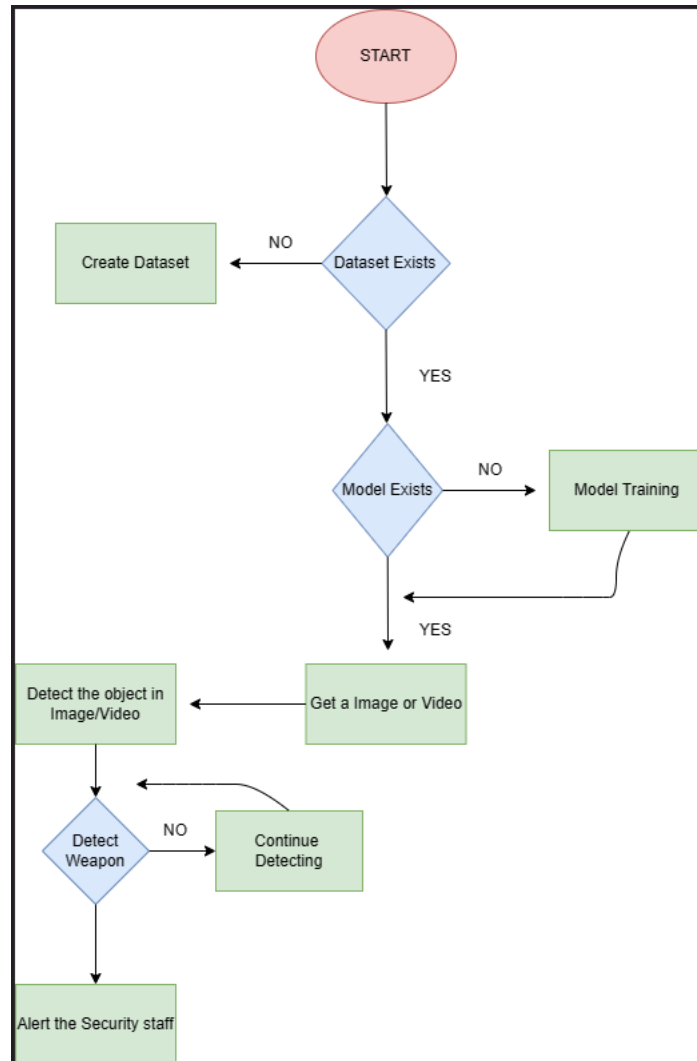
Many different features claim to increase Convolutional Neural Network (CNN) accuracy.

It is necessary to evaluate these feature combinations in practice on sizable datasets and to theoretically support the outcome. While some aspects, like as batch-normalization and residual-connections, are relevant to the majority of models, tasks, and datasets, others only function on specific models, for specific issues, or only for small-scale datasets.[9]

## CHAPTER-3

### SYSTEM ARCHITECTURE AND DESIGN

#### 3.0 WORKFLOW OF PURPOSED MODEL



**Fig 3.1 Flow diagram of the model**

The Fig 3.0.1 shows the flow diagram of the proposed model. First the process is started by the user. Then the dataset is created if it does not exist or downloaded if the dataset exists already then, a check is needed to be made if the model is existing or not. Then the model is trained if not there using the datasets that are fed into the model. Then the images and videos are downloaded to test the model with the dataset that it was trained on. After the image/Video is inputted into the model it is then run for detection if successful it will produce output with the percentage of the object detected.

### **3.1 ARCHITECTURE OF THE YOLO MODEL**

Object detection is a way that is used in computer vision for identification and localization purposes within an image or videos. Localization of images is a process of identifying the correct location of image of one or multiple objects using bounding boxes which are like rectangular shape around the objects. The process that this happens is called image classification or image recognition which predicts the classes of a image or an object that is there within an image into classes or categories. YOLO (You Only Look Once) it is a state of art real time object detection algorithm introduced in 2015 in a research paper. The authors had done it using a regression problem instead of a classification task by making bounding boxes and probabilities to detect images using CNN (convolutional neural network).

YOLO is leading in the competition as it has:

1. Speed
2. Accuracy of Detection
3. Generalization is good
4. Open Sourcing

## **1.Speed**

YOLO is extremely quicker than others because it doesn't deal with complex pipelines and it can process 45 FPS (Frames Per Second). YOLO also able to reach twice the mean precision value compared with the other systems and it makes it ideal for doing real-time processing. From the drawing below we observe that YOLO is way superior than other object detectors having 91 FPS.

## **2. High-Detection Accuracy**

The YOLO is far way beyond ahead in terms of accuracy with a few errors while running in background.

## **3.Better Generalization**

In the newer versions of YOLO, it can provide better generalization of new domains which makes it great for applications that require fast and robust object detection.

## **4. Open Source**

Making YOLO Open Source that is open to the community to improve the model always and this is the reasons why YOLO has made so much changes and improvements in limited time period.

## **3.2 WORKING OF YOLOV3 MODEL**

The YOLO algorithm is working based on four approaches:

1. Residual Blocks
2. Intersection Over Union (IOU)
3. Bounding Box Regression
4. Non- Maximum Suppression

### **1.Residual Blocks**

In order to estimate the class of object that each cell will cover, along with a confidence level, the original image (B) is first divided into  $N \times N$  grid cells with equal shapes, where N can be 4 in a 4 by 4 image.

## 2. Bounding Box Regression

The next step is to choose the bounding boxes, which are rectangles that emphasize all of the image's elements. Regarding the objects in the provided photos, there may be a variety of bounding boxes.

In the following format, YOLO calculates it using a single regression module, where Y is the final vector representation of each box.

$$Y = [pc, bx, by, bh, bw, c1, c2]$$

This is frequently crucial throughout the model's training phase.

1. On a computer, it displays the likelihood rating for the grid containing the object. The probability score for each of the red grids is greater than zero. The chance will be zero if the grids are yellow.
2. bx, by are the grid cell's x and y coordinates relative to the centre of the bounding box.
3. bh, bw represent the bounding box's height and breadth.
4. The numbers c1 and c2 represent the number of classes.

## 3. Intersection Over Unions

In any of the photos, a single object may appear in numerous grid boxes for a prediction, not all of which may be very pertinent. The value of the IOU must be between 0 and 1, and grid boxes should be discarded while keeping those.

### 3.3 YOLOV3 ALGORITHM STEPS

1. The YOLOv3 model is built on the Darknet-53 a model that does feature extraction. It has 53 layers that are stacked on top of one another to create a 106-layer fully convolutional architecture. The 82nd, 94th, and 106th layers of the architecture are where things are detected. A feature pyramid network and a prediction module are both used by the YOLOv3 architecture to recognize objects at various scales.
2. The architecture has three layers where the objects are recognized- the 82nd, the 94th, and the 106th.

3. The YOLOv3 architecture works by splitting the image into  $S \times S$  squares. Each square is in charge of predicting one object in the image.
4. To get the location of the bounding boxes each cell contains five parameters  $x, y, w, h$  and confidence where the  $x$  and  $y$  are the center of the box,  $w$  and  $h$  are the height and widths of the bounding box and the confidence parameter will tell if there is a object present or not
5. The algorithm may assign high probabilities to more than one grid cell for the same object, resulting in multiple overlapping bounding boxes. To eliminate this problem, the Non-Max Suppression technique is applied.
6. The YOLOv3 model is a deep convolutional neural network that can detect and classify objects in images or videos in real time.
7. The YOLOv3 model does not reduce the size of the feature maps. Instead, it uses convolutional layers with a stride of 2 for down sampling. This preserves more information and allows the model to make better predictions.
8. A convolutional layer produces several feature maps by applying different filters to the input. Each filter performs a convolution operation that extracts some features from the input.

### **3.4 DRAWBACKS OF YOLOV3 MODEL**

One of the most remarkable deep learning model architectures in the domain of object detection is YOLOv3. It stands out for its speed, accuracy, and versatility in various use cases. However, it also has its strengths and weaknesses, like any other technology.

1. One of the challenges of YOLOv3 is detecting objects that are small in size. This is because the design of its anchor boxes and the big stride it uses can make it miss or ignore small objects. Here is another way to write the same text using different words.
2. A challenge for devices with limited resources is the high memory requirements

of YOLOv3. To run this algorithm, a large amount of memory is needed, which may not be available on some devices.

3. One of the challenges of YOLOv3 is the long duration of its training process, which depends on having access to abundant data and powerful computing devices.

### **3.5 OVERCOMING THE DRAWBACKS OF YOLOV3 MODEL**

1. A possible way to enhance the accuracy of YOLOv3 is to apply data augmentation methods such as random cutting, mirroring, and spinning. This can help enrich the variety of the training data and prevent overfitting.
2. One way to enhance the performance of YOLOv3 is to apply transfer learning. This means using a model that has already been trained on a large dataset and adjusting it to a smaller one. This can help the model learn faster and with less data.
3. A way to enhance the accuracy of models is to use ensemble models. Ensemble models are based on the idea of combining several models that have been trained with different settings or structures. By aggregating the predictions of these models, you can create a more reliable model.
4. To enhance the results of YOLOv3, a larger dataset is one of the most effective methods. The model will perform better with more data. This can be done by gathering more data or using existing datasets.

#### **ENHANCEMENTS:**

1. We aim to enhance the precision of object detection by applying a stronger loss function such as the focal loss function. This loss function assigns more importance to hard examples, which enables the model to concentrate on learning challenging cases.
2. We aim to improve the speed and efficiency of a network by applying pruning techniques that can reduce the number of parameters and operations without compromising accuracy. This involves eliminating unnecessary filters, cutting out whole layers, or using low-rank approximations.

### **3.6 DESIGN OF THE MODEL**

To create our model with YOLO v3, we have to take into account



some key factors that affect its performance. These factors are explained in more detail in the next sections:

1. **Dataset Collection and Setup:** The model requires a dataset of images and videos that show different kinds of weapons. These can be obtained from various sources. The next step is to label the dataset with categories such as weapon or shotgun. The dataset also needs some preprocessing steps such as resizing the images, changing them to grayscale, and normalizing the pixel values if necessary.
2. **YOLO V3 Architecture and Algorithm:** The models for weapon detection should be based on the YOLO v3 architecture and algorithm. This model uses a deep convolutional neural network called DarkNet-53 to extract features from images and videos. The algorithm employs anchor boxes to locate objects and estimate the bounding boxes and class probabilities for each anchor box.
3. **Fine-Tuning of the Model:** To adapt the model to the weapon dataset, we need to use a transfer learning technique. This means that we start with the YOLO v3 model that has already been trained on a large dataset, and we only update the last few layers of the model with the weapon dataset that we downloaded from Open Images. This way, we can leverage the existing knowledge of the model and fine-tune it to our specific task.
4. **Warning System:** An alert system should be designed that can alert the security staff if weapon is detected. The alert system can be in the form of a loud alarm or flashing lights. The alert system should be activated only when weapon is detected.
5. **Performance Assessment:** The performance for the weapon detection model and alert model should be evaluated using metrics like accurateness, preciseness of the model. The model should also be tested on a separate test set to evaluate its generalization performance.

In conclusion, designing a weapon detection model using YOLOv3 has several critical parameters like dataset collection and preparation, the YOLO V3 algorithm and fine

tuning of the model. By considering all these parameters a robust and reliable system can be designed to improve human safety.

### **3.7 REQUIRMENTS OF MODEL**

#### **3.7.1 Software Requirements**

- Any IDE than can run Python code (jupyter notebook / VB Studio/Google Collab)
- Downloading Necessary libraries and modules
- Open Image Dataset for testing
- Operating System: Ubuntu, macOS, or Windows
- CUDA Toolkit: version 10.0 or higher
- CuDNN: Version 7.0 or higher
- OpenCV
- Python: Version 3.6 or higher
- TensorFlow: Version 1.15 or higher

#### **3.7.2 Hardware Requirements**

- Any computing system
- CPU: Intel i7/i3(11<sup>th</sup> gen)/i5, AMD Ryzen 7 or higher
- GPU: NVIDIA GeForce GTX 1080 or higher, NVIDIA Tesla V100 ori higher
- RAM: 16 GB or higher
- Storage: 100 GB or higher
- Connectivity: Fast Internet speed is required to download the dataset and train the model.
- Browser: Google Chrome, Microsoft Edge etc.

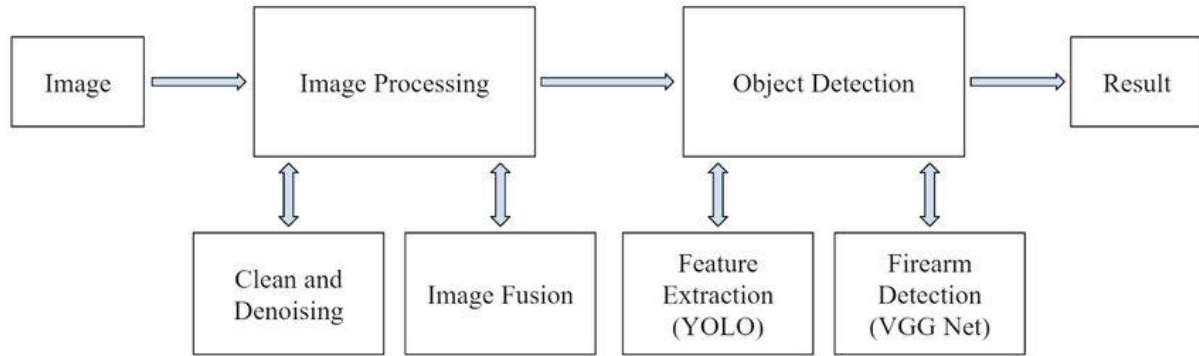
## **CHAPTER-4**

### **METHODOLOGY**

In this project we were able to develop a framework for security that can differentiate between weapons progressively if the correct weapon is identified then an alert can be given to the security officials to arrive at the spot and handle the situation and that can be done with the help of IP Cameras. This model would give the machine a sense of vision that helps it to identify the unsafe object. This information can also be linked to a database where all the activities that are taking place can be recorded and could be retrieved incase if the data needs to be looked at again by the officials. The most important part of any running application is that a dataset is needed to train the learning models. We have chosen Open Image Dataset which has thousands of images with their annotations to train the model. We then combined all of those images in one multiclass for the model to detect both the objects and the images were saved in image form. Incase the images had many extensions, then would be difficult for the model to read all the different types of images and errors would come up while training. Once the training starts the images are processed parts as batches and all are taken in the same size of width and height. Video clips also are taken and read in form of frames per second and the output is shown in the form of percentage in the accuracy of the object detected. Typically, object detection is connected to computer vision, which distinguishes between the items in computerized images. YOLO is a pretrained detector, and it has made significant advancements in deep learning.

#### **4.1 FEATURES OF MODEL**

The convolutional network, sometimes known as CNN or ConvNet, is a particular type of deep neural network. A deep, forward-feeding artificial neural network is used. Keep in mind that the fundamental deep learning models, multi-layer perceptrons (MLPs), are also known as feed-forward neural networks



**Fig 4.1**

#### **4.1.1 DATASET COLLECTION AND PREPROCESSING**

The main step in any machine learning project is to collect and pre-process the data required for the model. In case of weapon detection model using YOLO v3, the data is usually collected through cameras mounted anywhere on the wall doing surveillance in case any person is carrying harmful object. Pre-processing involves cleaning the data, removing any noise or irrelevant information, and formatting the data in some way that the machine learning model could understand. For this research there is no appropriate data set. Since it is not available, we created a custom dataset and trained our models in multiple batches using various cycles. The data set includes the following attributes: weapon and shotgun as the labels. To train the YOLO v3 model, we collected a custom dataset of images containing different weapons and size. The images were collected from the dataset Open Image Dataset by Google. Once the images are preprocessed, they are labeled to indicate the different kinds of weapons. This labelled data will be then used for training the ML model to identify the different weapons that can appear when doing surveillance.

#### **4.1.2 DATA PREPARATION AND VARIABLE SELECTION**

One of the steps in building an ML model is to choose the best variables or features that have the most impact on the model's accuracy. This process is called variable selection and it helps to reduce the complexity and improve the performance of the ML model. In case of weapon detected it can differentiate

between shotgun and if its any other weapon. Feature engineering involves transforming raw variables into meaningful features that the machine learning model can understand. For example, different guns have different features hence the machine learning model needs to understand and detect it. Once the data is collected, they are cleaned and processed. They are also classified into one folder called data folder with one main folder as it is a multiclass for training and validation. For the folders it contains labels and images like key-value pair. The folder is then put into the model to run for predictions of the trained model. The collected images were preprocessed to ensure consistency in image size and resolution in order for it to detect properly. The dataset was then annotated with bounding boxes in order to identify the region of interest.

### 4.1.3 SPLITTING OF DATA

The dataset was split into train and test. The training set was used to train the YOLOv3 model and training incase to assess the performance.

## 4. Workflow Procedure

### a) Using Google Collab and Setting up the environment:

The first step towards implementing a weapon detection model using YOLO is to configure google collab with required GPU in order for Darknet to run. Google drive is then connected to the google collab running session using the below code.

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

then we change the make file to enable the GPU and OPENCV to be enabled using code below.

```
# change makefile to have GPU and OPENCV enabled  
%cd /content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet  
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile  
!sed -i 's/GPU=0/GPU=1/' Makefile  
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

## **b) Label Images with export to YOLO format**

The next step involves labeling the images and exporting the images to YOLO format but first a Toolkit is used called OIDV4 to get the images from Open Images Dataset V4 as it is having over 600 classes and more than 2,000,000 images relating to bounding boxes. The next process is to label the images and then we export them to YOLO format but in order to do this we use a toolkit called OIDV4 to download the images from Open Dataset Version 4 as it is having more than 700 classes of objects and images that are having bounding boxes

- It can create the bounding boxes for each downloaded image and download any one of the 600 classes that are accessible.
- Multiple number of classes can be downloaded from the dataset and create separated folders and bounding boxes for each of them, you can use a Python Toolkit that allows you to do that easily. The Toolkit accepts a list of classes or a text file that contains the names of the classes you want to download. For each class, it will create a folder with the images and a file with the annotations in YOLO format. You can also specify which subset of the dataset (train, validation or test) you want to download from. The ToolKit also has a visualizer option that lets you inspect the downloaded images and their bounding boxes. This way, you can get the data you need for your object detection project without downloading the whole dataset.
- The Toolkit that supports YOLO annotation can be used to get single classes or several classes and it also generates bounding boxes for each of the images and enables you to get any of the classes that are available from the dataset that can be individual or combined.

The Toolkit is a software that allows you to create and customize your own dataset for various purposes. You can specify the location of the folder where you want to save your dataset by using the --Dataset argument. If you do not provide this argument, the default folder name will be

Dataset. This way, you can have multiple folders with different datasets and options inside them.

- The `-classes` argument in the code below will accept the entire list of classes or the location of the text file containing the complete classes that are available. As in this case, we downloaded the validation set's Weapons and Shotgun from the dataset.

```
python3 main.py downloader --classes Weapon Shotgun --type csv validation
```

- The algorithm will download all the files from the aforementioned code and create a directory with the files, photos, and text files associated with them.
- The following code is used to download multiclass like we have since it downloads all the classes into a single folder

```
python3 main.py downloader --classes Apple Orange --type csv validation --multiclass 1
```

### c) Making the Detections

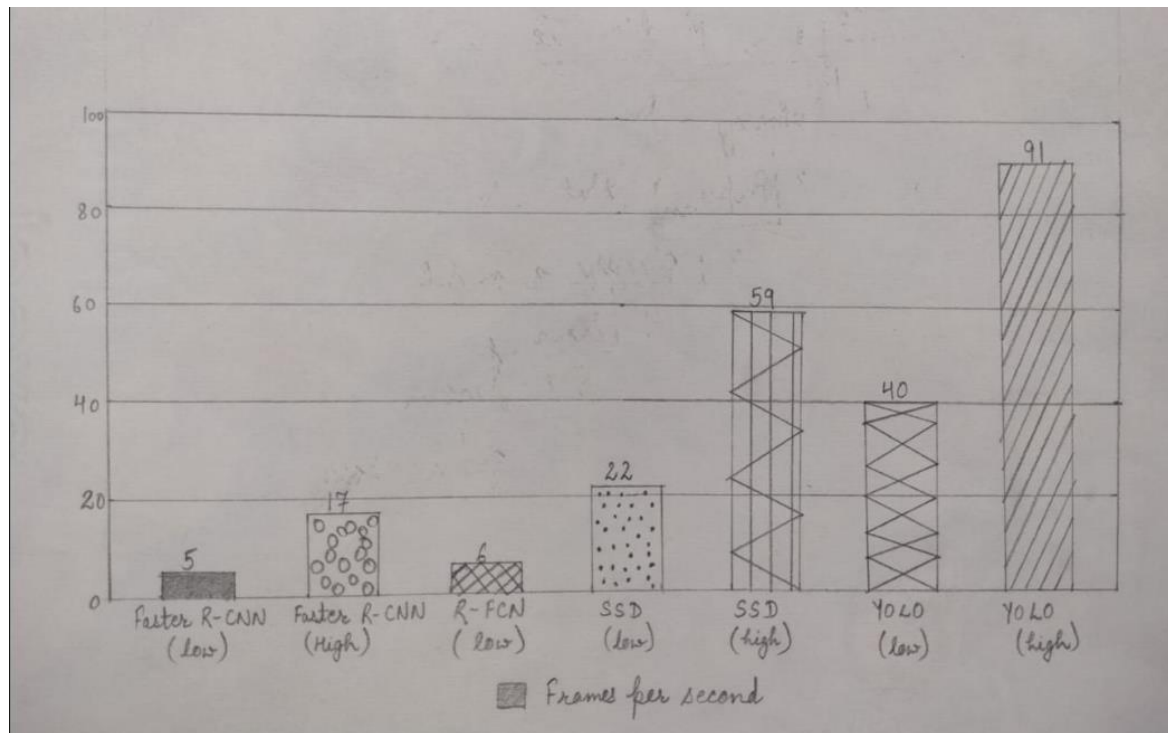
We use a grid-based approach to detect objects in an image. For each cell in the grid, we generate bounding boxes and scores that indicate how likely the box contains an object. The user can also fine-tune the model on their own dataset to increase the detection performance. The model is then run over 4000 times over the dataset to be trained

### d) Performance of the model

To improve the performance of the model the user can also use model quantization and pruning. Model quantization will reduce the precision of the model metric which speeds up the inference. Model pruning will also remove unnecessary parameters from the model which reduces its size and speeds it up. The user can also use hardware accelerator like powerful GPUs to speed up inference.

## 4.1.4 TRAINING AND VALIDATION OF MODEL

After the dataset is preprocessed and the model architecture is selected the model is then trained on the dataset. One of the main steps in building a machine learning model is training. This means finding the optimal values for the parameters of our model, such as weights and biases, that minimize the loss function. The loss function is a measure of how well our model predicts the drowsiness state of a driver, compared to the actual state. We want our model to make accurate predictions, so we want to keep the loss function as low as possible. Validation of the model is to check if whether it is detecting the required image inputted to the model. It also involves that the model isn't overfitting to the training data. The models hyperparameters like learning rate, batch size and number of times runs are based on validation set.



**Fig 4.2**

Graph depicting the fps of different models, where high yolo is found.



## **CHAPTER-5**

### **SOURCE CODE AND CODING METHODS**

#### **5.1 CODE METHODS AND PROCEDURES**

To implement the trained YOLOV3 model the requirements are from the library of the model that is the available config file present in the YOLOV3 model which tells in detail about the layers and other information like the number of filters that are present in each of the layers in the model, the learning rate of the model, classes present and the size of input of each of the layers and the channels. The yolov3.config file gives us a better understanding of the structure of the model by defining the number of batch sizes and whether the model should be trained and tested. With the help of the config file one can also edit the config file based on their requirements in training the model and by saving it as yolov3-custom.config as this file will serve as the custom configuration to the model that one could be training on. One can also go ahead and then collect the dataset which is pretrained like COCO Dataset or Alex Net and one could also use Open Images Datasets V7 that we have used in this project all of these datasets can be used for detection purposes.

The Open Images Dataset V7 contains the dataset of millions of images that are interpreted with labels, bounding box and segmentation masks.

- The dataset has a total of 16 million bounding boxes and over 600 classes of objects on a total of 1.9 million images making it one of the largest datasets with object annotations.
- It does also give visual relationships that indicate that the object is of some certain relation or similar.
- In the version 5 the segmentation masks were also added for 2.8 million of the objects and segmentation masks were also added for level of details

```

1 person
2 bicycle
3 car
4 motorbike
5 aeroplane
6 bus
7 train
8 truck
9 boat
10 traffic light
11 fire hydrant
12 stop sign
13 parking meter
14 bench
15 bird
16 cat
17 dog
18 horse
19 sheep
20 cow
21 elephant
22 bear
23 zebra
24 giraffe
25 backpack
26 umbrella
27 handbag
28 tie
29 suitcase
30 frisbee
31 skis
32 snowboard
33 sports ball
34 kite
35 baseball bat
36 baseball glove
37 skateboard
38 surfboard
39 tennis racket

```

**Fig 5.1**

```

40 surfboard
39 tennis racket
40 bottle
41 wine glass
42 cup
43 fork
44 knife
45 spoon
46 bowl
47 banana
48 apple
49 sandwich
50 orange
51 broccoli
52 carrot
53 hot dog
54 pizza
55 donut
56 cake
57 chair
58 sofa
59 pottedplant
60 bed
61 diningtable
62 toilet
63 tvmonitor
64 laptop
65 mouse
66 remote
67 keyboard
68 cell phone
69 microwave
70 oven
71 toaster
72 sink
73 refrigerator
74 book
75 clock
76 vase
77 scissors
78 teddy bear
79 hair drier
80 toothbrush

```

**Fig 5.2**

These are 80 classes the YOLO model was already pre-trained on both the figures.

In this model we have trained the neural network which was a very long process as it required thousands of images and a lot of computational power in graphics processing which took around 7+ hours to train the model. Training was required for the model as it used to detect custom objects which are not yet present or done on which it is trained or the objects that are detected under a certain class as when making just general predictions to which the class the certain object falls. To make the predictions to detect certain objects on which the model was trained for hours, a implantation can be done to get the network running. To do that requires the yolo-v3.config file and also the weight file of the model that is pretrained. A pretrained weights file contains the mathematical relationships between neurons in adjacent layers of a neural network. These relationships, or weights, represent the influence of one neuron on another. When training a model from scratch, random values are initially assigned to these weights. During training, the values are optimized using a loss function and an optimization function to improve the network's performance. When retraining the network, the weights of certain layers can be frozen based on the amount of available data and the specific purpose of the retraining.

```
[net]
# Testing
batch=1
subdivisions=1
# Training
#batch=64
#subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
|
learning_rate=0.001
```

```
burn_in=1000  
max_batches = 4000  
policy=steps  
steps=3200,3600  
scales=.1,.1
```

```
[convolutional]  
batch_normalize=1  
filters=32  
size=3  
stride=1  
pad=1  
activation=leaky
```

```
# Downsample
```

```
[convolutional]  
batch_normalize=1  
filters=64  
size=3  
stride=2  
pad=1  
activation=leaky
```

```
[convolutional]  
batch_normalize=1  
filters=32  
size=1  
stride=1  
pad=1  
activation=leaky
```

```
[convolutional]
```

```
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear
```

```
# Downsample
```

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky
```

#### YOLOv3.Config File

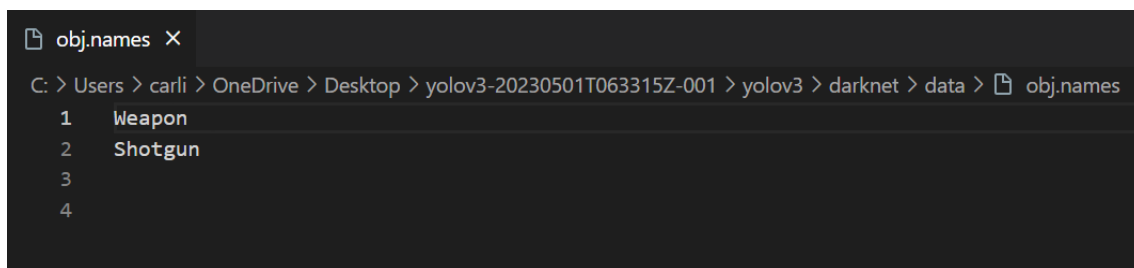
In this implementation of YOLOv3, the config and weights files were obtained from the OpenCV library. The network was used without modification because it was able to detect and track objects in a video. A Python script was written using dependencies such as OpenCV and OS to process the video frames and perform object detection and tracking.

```
[net]
# Testing
batch=1
subdivisions=1
# Training
#batch=64
#subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
```

```
saturation = 1.5
exposure = 1.5
hue=.1
```

Here for the training the dataset for particular classes one should uncomment the training section in the yolov3 custom config file as this will tell the model to train while the directory of the file is given and the code is executed. The batch size will be 64 and the subdivisions will be based on the system requirements as if it shows CUDA error out of memory then it should be changed to 32 otherwise best default should be subdivisions=16.

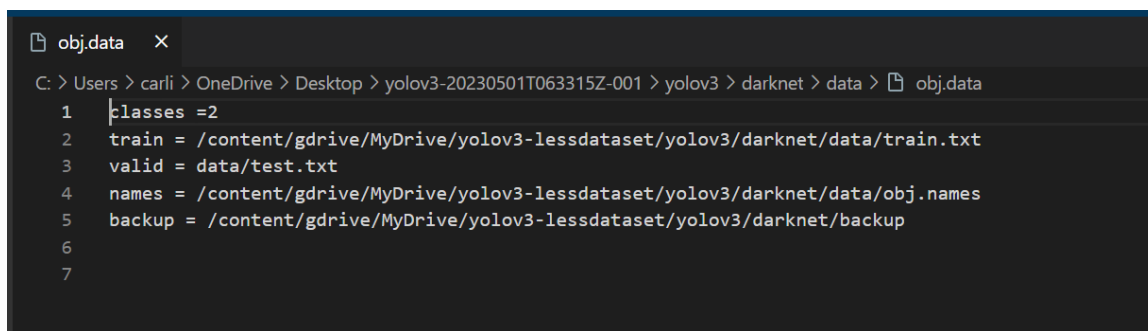
The Max Batches is how many times the training will run through the data so it is how many batches it will go through. If it is 2 classes like this model between weapons and shotgun then it should be classes\*2000 should not be less than 4000. If 3 classes then 6000 the next is the steps should be 80% and 90% of the max batches. Hence all the rest below needs to be changed accordingly.



```
obj.names X
C: > Users > carli > OneDrive > Desktop > yolov3-20230501T063315Z-001 > yolov3 > darknet > data > obj.names
1   Weapon
2   Shotgun
3
4
```

**Fig 5.3**

The obj.name file is created for the data that is downloaded from the dataset and compiled. Hence Obj.data should be placed in Obj folder inside the data file of the model.



```
obj.data X
C: > Users > carli > OneDrive > Desktop > yolov3-20230501T063315Z-001 > yolov3 > darknet > data > obj.data
1   classes =2
2   train = /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/data/train.txt
3   valid = data/test.txt
4   names = /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/data/obj.names
5   backup = /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/backup
6
7
```

**Fig 5.4**

The obj.data is also created which is inside the data folder of the model.

It consists of five variables to link the files in which directory it is in.

- Classes: there are 2 classes weapon and shotgun
- Train: directs to the directory of the train.txt file which is created for all the images that are needed to be trained.
- Names: the directory of the obj.name file in the model
- Backup: in case of any shutdown during training the model can be recovered as the weight files will be stored in the backup folder which is helpful due to long hours of model training

For the training of the model the convolution weights that are already pretrained are used which is from the Darknet53 Model. Darknet is a neural network that is 53 layers which are deep. The version can be loaded pretrained from more than millions of images from the dataset called ImageNet. The pretrained network can classify to almost 1000 object categories such as mouse, pen and different objects. Hence the network has a rich feature to represent a wider range of images and can be used.

### **1)Using Googles Open Image Dataset:**

Gather thousands of images and get their labels generated and by using OVIDV4 toolkit to generate the labels for thousands of images which could be easy and also time efficient. The dataset contains images that are labelled over 600 classes.

### **2)Move the Model to cloud:**

For this model have moved it to cloud such as Google Drive for training as it would reduce the time it takes to transfer the dataset.

```
from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```



























```
!ln -s /content/gdrive/My\ Drive/ /mydrive

# change makefile to have GPU and OPENCV enabled
%cd /content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet
```

### 3) The Darknet files in Google Drive repo

... > yolov3 > darknet ▾

Name ↑	Owner	Last modified ▾
 .circleci	 me	1 May 2023 me
 .git	 me	1 May 2023 me
 .github	 me	1 May 2023 me
 3rdparty	 me	1 May 2023 me
 backup	 me	1 May 2023 me
 build	 me	1 May 2023 me
 cfg	 me	1 May 2023 me
 cmake	 me	1 May 2023 me
 data	 me	1 May 2023 me
 include	 me	1 May 2023 me
 obj	 me	1 May 2023 me
 results	 me	1 May 2023 me
 scripts	 me	1 May 2023 me

**Fig 5.5**

### 4) Cloning and Building Darknet

The following code and cells will clone the darknet from AlexeyAB repo and the GPU and OPENCV will be enabled to build darknet for the model.



```
# clone darknet repo
!git clone https://github.com/AlexeyAB/darknet
```

Output:

```
Cloning into 'darknet'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 13395 (delta 18), reused 22 (delta 11), pack-reused 13357
Receiving objects: 100% (13395/13395), 12.04 MiB | 12.43 MiB/s, done.
Resolving deltas: 100% (9140/9140), done.
```

```
# change makefile to have GPU and OPENCV enabled
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

## 5) Making Darknet into the cloud Repo

```
# make darknet (build)
!make
```

## 6) The Yolov3 pretrained weights are downloaded

```
# get yolov3 pretrained coco dataset weights
!wget https://pjreddie.com/media/files/yolov3.weights
```

## 7) Train the model on the downloaded dataset

```
%cd /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet
!./darknet detector train /content/gdrive/MyDrive/yolov3-
lessdataset/yolov3/darknet/data/obj.data /content/gdrive/MyDrive/yolov3-
lessdataset/yolov3/darknet/cfg/yolov3-custom.cfg darknet53.conv.74 -dont_show
```

```

mkdir -p ./obj/

mkdir -p backup

chmod +x *.sh

g++ -std=c++11 -std=c++11 -include/ -I3rdparty/stb/include -DOPENCV `pkg-config --cflags opencv4 2>
/dev/null || pkg-config --cflags opencv` -DGPU -I/usr/local/cuda/include/ -DCUDNN -Wall -Wfatal-errors
-Wno-unused-result -Wno-unknown-pragmas -fPIC -Ofast -DOPENCV -DGPU -DCUDNN -
I/usr/local/cudnn/include -c ./src/image_opencv.cpp -o obj/image_opencv.o

./src/image_opencv.cpp: In function 'void draw_detections_cv_v3(void**, detection*, int, float, char**,
image**, int, int)':

./src/image_opencv.cpp:910:23: warning: variable 'rgb' set but not used [-Wunused-but-set-variable]

    float rgb[3];
        ^~~

./src/image_opencv.cpp: In function 'void cv_draw_object(image, float*, int, int, int*, float*, int*, int,
char**)':

./src/image_opencv.cpp:1391:14: warning: unused variable 'buff' [-Wunused-variable]

    char buff[100];
        ^~~~~

./src/image_opencv.cpp:1367:9: warning: unused variable 'it_tb_res' [-Wunused-variable]

    int it_tb_res = cv::createTrackbar(it_trackbar_name, window_name, &it_trackbar_value, 1000);
        ^~~~~~

./src/image_opencv.cpp:1371:9: warning: unused variable 'lr_tb_res' [-Wunused-variable]

    int lr_tb_res = cv::createTrackbar(lr_trackbar_name, window_name, &lr_trackbar_value, 20);
        ^~~~~~

./src/image_opencv.cpp:1375:9: warning: unused variable 'cl_tb_res' [-Wunused-variable]

    int cl_tb_res = cv::createTrackbar(cl_trackbar_name, window_name, &cl_trackbar_value, classes-1);
        ^~~~~~

./src/image_opencv.cpp:1378:9: warning: unused variable 'bo_tb_res' [-Wunused-variable]

    int bo_tb_res = cv::createTrackbar(bo_trackbar_name, window_name, boxonly, 1);
        ^~~~~~

g++ -std=c++11 -std=c++11 -include/ -I3rdparty/stb/include -DOPENCV `pkg-config --cflags opencv4 2>
/dev/null || pkg-config --cflags opencv` -DGPU -I/usr/local/cuda/include/ -DCUDNN -Wall -Wfatal-errors

```

3998: 0.270800, 0.318546 avg loss, 0.000010 rate, 4.445639 seconds, 255872 images, 0.165266 hours left

Loaded: 0.000079 seconds

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.776959), count: 4, class\_loss = 0.206493, iou\_loss = 0.256089, total\_loss = 0.462582

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.659596), count: 1, class\_loss = 0.251513, iou\_loss = 0.104540, total\_loss = 0.356052

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.550534), count: 2, class\_loss = 0.142094, iou\_loss = 0.765666, total\_loss = 0.907761

total\_bbox = 52144, rewritten\_bbox = 0.475606 %

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.836909), count: 5, class\_loss = 0.724370, iou\_loss = 0.193208, total\_loss = 0.917578

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.849559), count: 1, class\_loss = 0.085587, iou\_loss = 0.016183, total\_loss = 0.101770

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class\_loss = 0.000006, iou\_loss = 0.000000, total\_loss = 0.000006

total\_bbox = 52150, rewritten\_bbox = 0.475551 %

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.847659), count: 4, class\_loss = 0.178455, iou\_loss = 0.077699, total\_loss = 0.256154

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.610348), count: 4, class\_loss = 0.403399, iou\_loss = 0.566255, total\_loss = 0.969653

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.639717), count: 1, class\_loss = 0.232698, iou\_loss = 0.100065, total\_loss = 0.332763

total\_bbox = 52159, rewritten\_bbox = 0.475469 %

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.721884), count: 2, class\_loss = 0.021227, iou\_loss = 0.128976, total\_loss = 0.150204

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.748722), count: 3, class\_loss = 0.370844, iou\_loss = 0.112534, total\_loss = 0.483378

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.555355), count: 2, class\_loss = 0.404858, iou\_loss = 0.277969, total\_loss = 0.682827

total\_bbox = 52166, rewritten\_bbox = 0.475405 %

v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.805899), count: 2, class\_loss = 0.243529, iou\_loss = 0.027847, total\_loss = 0.271377

4000: 0.112884, 0.080791 avg loss, 0.000010 rate, 4.143253 seconds, 256000 images, 0.119808 hours left

Saving weights to /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/backup/yolov3-custom\_4000.weights

Saving weights to /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/backup/yolov3-custom\_last.weights

Saving weights to /content/gdrive/MyDrive/yolov3-lessdataset/yolov3/darknet/backup/yolov3-custom\_final.weights

If you want to train from the beginning, then use flag in the end of training command: -clear

## 8) Detecting the object for a Shotgun image uploaded

```
!./darknet detector test /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/data/obj.data /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/cfg/yolov3-custom.cfg /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/backup/yolov3-custom_3000.weights  
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Shotgun-3.jpg
```

```
from google.colab.patches import cv2_imshow  
import cv2 as cv
```

```
img = cv.imread('/content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/predictions.jpg')#mentioning a path of an image  
cv2_imshow(img)
```



**Fig 5.6**

conv 32 3 x 3/1 416 x 416 x 3 -> 416 x 416 x 32 0.299 BF  
 1 conv 64 3 x 3/2 416 x 416 x 32 -> 208 x 208 x 64 1.595 BF  
 2 conv 32 1 x 1/1 208 x 208 x 64 -> 208 x 208 x 32 0.177 BF  
 3 conv 64 3 x 3/1 208 x 208 x 32 -> 208 x 208 x 64 1.595 BF  
 4 Shortcut Layer: 1, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.003 BF  
 5 conv 128 3 x 3/2 208 x 208 x 64 -> 104 x 104 x 128 1.595 BF  
 6 conv 64 1 x 1/1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF  
 7 conv 128 3 x 3/1 104 x 104 x 64 -> 104 x 104 x 128 1.595 BF  
 8 Shortcut Layer: 5, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF  
 9 conv 64 1 x 1/1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF  
 10 conv 128 3 x 3/1 104 x 104 x 64 -> 104 x 104 x 128 1.595 BF  
 11 Shortcut Layer: 8, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF  
 12 conv 256 3 x 3/2 104 x 104 x 128 -> 52 x 52 x 256 1.595 BF  
 13 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 14 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 15 Shortcut Layer: 12, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 16 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 17 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 18 Shortcut Layer: 15, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 19 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 20 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 21 Shortcut Layer: 18, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 22 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 23 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 24 Shortcut Layer: 21, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 25 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 26 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 27 Shortcut Layer: 24, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 28 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 29 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 30 Shortcut Layer: 27, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 31 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 32 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 33 Shortcut Layer: 30, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 34 conv 128 1 x 1/1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF  
 35 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF  
 36 Shortcut Layer: 33, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF  
 37 conv 512 3 x 3/2 52 x 52 x 256 -> 26 x 26 x 512 1.595 BF  
 38 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF  
 39 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF  
 40 Shortcut Layer: 37, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF  
 41 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF  
 42 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF  
 43 Shortcut Layer: 40, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF  
 44 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF  
 45 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF  
 46 Shortcut Layer: 43, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF  
 47 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF  
 48 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF  
 49 Shortcut Layer: 46, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF  
 50 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF  
 51 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF  
 52 Shortcut Layer: 49, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF

```

53 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
54 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
55 Shortcut Layer: 52, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF
56 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
57 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
58 Shortcut Layer: 55, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF
59 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
60 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
61 Shortcut Layer: 58, wt = 0, wn = 0, outputs: 26 x 26 x 512 0.000 BF
62 conv 1024 3 x 3/2 26 x 26 x 512 -> 13 x 13 x 1024 1.595 BF
63 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
64 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
65 Shortcut Layer: 62, wt = 0, wn = 0, outputs: 13 x 13 x 1024 0.000 BF
66 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
67 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
68 Shortcut Layer: 65, wt = 0, wn = 0, outputs: 13 x 13 x 1024 0.000 BF
69 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
70 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
71 Shortcut Layer: 68, wt = 0, wn = 0, outputs: 13 x 13 x 1024 0.000 BF
72 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
73 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
74 Shortcut Layer: 71, wt = 0, wn = 0, outputs: 13 x 13 x 1024 0.000 BF
75 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
76 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
77 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
78 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
79 conv 512 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 512 0.177 BF
80 conv 1024 3 x 3/1 13 x 13 x 512 -> 13 x 13 x 1024 1.595 BF
81 conv 21 1 x 1/1 13 x 13 x 1024 -> 13 x 13 x 21 0.007 BF
82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00,
1.00
83 route 79 -> 13 x 13 x 512
84 conv 256 1 x 1/1 13 x 13 x 512 -> 13 x 13 x 256 0.044 BF
85 upsample 2x 13 x 13 x 256 -> 26 x 26 x 256
86 route 85 61 -> 26 x 26 x 768
87 conv 256 1 x 1/1 26 x 26 x 768 -> 26 x 26 x 256 0.266 BF
88 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
89 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
90 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
91 conv 256 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
92 conv 512 3 x 3/1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
93 conv 21 1 x 1/1 26 x 26 x 512 -> 26 x 26 x 21 0.015 BF
94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00,
1.00
95 route 91 -> 26 x 26 x 256
96 conv 128 1 x 1/1 26 x 26 x 256 -> 26 x 26 x 128 0.044 BF
97 upsample 2x 26 x 26 x 128 -> 52 x 52 x 128
98 route 97 36 -> 52 x 52 x 384
99 conv 128 1 x 1/1 52 x 52 x 384 -> 52 x 52 x 128 0.266 BF
100 conv 256 3 x 3/1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

```



```

101 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv 21 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 21 0.029 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00,
1.00
Total BFLOPS 65.312
avg_outputs = 516922
Allocate additional workspace_size = 134.22 MB
Loading weights from /content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/backup/yolov3-
custom_3000.weights...
seen 64, trained: 192 K-images (3 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Shotgun-33.jpg: Predicted in 2917.166000 s
seconds.
Shotgun: 83%
Unable to init server: Could not connect: Connection refused

(predictions:4666): Gtk-WARNING **: 08:05:55.501: cannot open display:

```

## 9) Detecting the object in video uploaded

```

!./darknet detector demo /content/gdrive/MyDrive/yolov3-
weapon/yolov3/darknet/data/obj.data /content/gdrive/MyDrive/yolov3-
weapon/yolov3/darknet/cfg/yolov3-custom.cfg /content/gdrive/MyDrive/yolov3-
weapon/yolov3/darknet/backup/yolov3-custom_3000.weights
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/male-hunter-shoots-
shotgun.mp4 -dont_show -out_filename results.avi

```



**Fig 5.7**

## 10) Running the model to detect different images

```
!./darknet detector test /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/data/obj.data /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/cfg/yolov3-custom.cfg /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/backup/yolov3-custom_3000.weights  
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Shotgun-3.jpg
```

```
from google.colab.patches import cv2_imshow  
import cv2 as cv
```

```
img = cv.imread('/content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/predictions.jpg')#mentioning a path of an image  
cv2_imshow(img)
```

```
!./darknet detector test /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/data/obj.data /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/cfg/yolov3-custom.cfg /content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/backup/yolov3-custom_3000.weights  
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Different-Weapon.jpeg
```



```
from google.colab.patches import cv2_imshow
import cv2 as cv
```

```
img = cv.imread('/content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/predictions.jpg')#mentioning a path of an image  
cv2_imshow(img)
```



**Fig 5.8**

```
from google.colab.patches import cv2_imshow
import cv2 as cv
```

```
img = cv.imread('/content/gdrive/MyDrive/yolov3-  
weapon/yolov3/darknet/predictions.jpg')#mentioning a path of an image  
cv2_imshow(img)
```



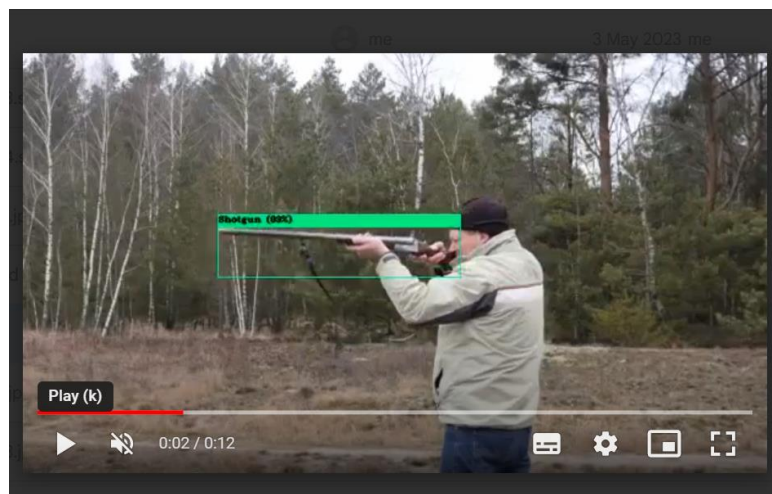
## CHAPTER-6

## RESULTS

### 6.1 Introduction

The dataset that was used to train the model contained over a 1000 images of both classes weapon and shotgun from open dataset.

After the video stream was inputted into the model the output is shown below in Fig 6.1



**Fig: 6.1**

The Image below shows the model identifying the weapon as Shotgun





**Fig 6.2**

**In this figure the model is able to differentiate between shotgun and weapon.**

As seen above the model that was trained using two classes weapon and shotgun the results are successful as the model is able to differentiate between the two classes on any given image or video that is inputted. The prediction score is also accurate on what it displays based on the object in the image. Compared to many earlier models, the predictions we obtained from the model are accurate.

## **6.2 Accuracy and Performance Evaluation of the Model**

After we have done training the model the next step is to look at its accuracy and the performance of the model while testing it on the dataset. The accuracy of the model is measured by the ability to detect the weapons accurately. The performance of the model is based on the output that it gives during the testing phase.

The accuracy obtained for this model

```
[yolo] params: iou_loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
95 route 91 -> 26 x 26 x 256
96 conv 128 1 x 1/ 1 26 x 26 x 256 -> 26 x 26 x 128 0.044 BF
97 upsample 2x 26 x 26 x 128 -> 52 x 52 x 128
98 route 97 36 -> 52 x 52 x 384
99 conv 128 1 x 1/ 1 52 x 52 x 384 -> 52 x 52 x 128 0.266 BF
100 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
101 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv 21 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 21 0.029 BF
106 yolo
[yolo] params: iou_loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.312
avg_outputs = 516922
Allocate additional workspace_size = 134.22 MB
Loading weights from /content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/backup/yolov3-custom_3000.weights...
seen 64, trained: 192 K-images (3 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Handgun-1.jpeg: Predicted in 880.501000 milli-seconds.
Weapon: 47%
Unable to init server: Could not connect: Connection refused

(predictions:122116): Gtk-WARNING **: 18:03:30.280: cannot open display:

97 upsample 2x 26 x 26 x 128 -> 52 x 52 x 128
98 route 97 36 -> 52 x 52 x 384
99 conv 128 1 x 1/ 1 52 x 52 x 384 -> 52 x 52 x 128 0.266 BF
100 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
101 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv 128 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv 256 3 x 3/ 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv 21 1 x 1/ 1 52 x 52 x 256 -> 52 x 52 x 21 0.029 BF
106 yolo
[yolo] params: iou_loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.312
avg_outputs = 516922
Allocate additional workspace_size = 134.22 MB
Loading weights from /content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/backup/yolov3-custom_3000.weights...
seen 64, trained: 192 K-images (3 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
/content/gdrive/MyDrive/yolov3-weapon/yolov3/darknet/Shotgun-3.jpg: Predicted in 2658.358000 milli-seconds.
Shotgun: 90%
Unable to init server: Could not connect: Connection refused

(predictions:8663): Gtk-WARNING **: 04:22:25.090: cannot open display:
```

This image below shows the percentage of accuracy on video files:

```
▶ Shotgun: 90%
↳ FPS:42.0      AVG_FPS:40.4
  cvWriteFrame
  Objects:
  Shotgun: 91%
  FPS:42.0      AVG_FPS:40.4
  cvWriteFrame
  Objects:
  Shotgun: 91%
  FPS:42.0      AVG_FPS:40.4
  cvWriteFrame
  Objects:
  Shotgun: 91%
  FPS:42.0      AVG_FPS:40.4
  cvWriteFrame
  Objects:
  Shotgun: 92%
```

## CHAPTER-7

### CONCLUSION

#### 7.1 Conclusion

The intent of the project was to implement the YOLOV3 in detecting various harmful objects like weapons after doing a comparison of it in other state of the art models. Since they are existing datasets but no model was built hence, we implemented this model. The images that are used for training will be annotated based on its classes and it will be uploaded to the model for testing.

It is well known that people can distinguish one another from one another in important ways, particularly if they are carrying a weapon. People can be distinguishable from one another in significant ways, which is one of these ways. The disparities that may exist between individuals should be taken into account when conducting research on an algorithm based on computer vision. In this paper, we present an algorithm for lowering the number of accidents and injuries brought on by criminals carrying ammunition.

We were able to beat out most of the other models in terms of accuracy ratings because to our success. We were able to achieve this degree of precision by utilising the concepts of deep learning, whereas earlier models were either built on hardware or based on other models with lower accuracy. Using physical sensors may be quite unpleasant and upsetting for the many other innocent individuals because the gear is affixed to the surveillance itself. On the other hand, earlier models were built employing metal detector sensors.

## **7.2 Future Enhancement**

This study implemented the YOLO V3 object detection model to train a dataset for weapon detection. The proposed model aims to provide machine with the ability to identify dangerous weapons and alert the security personnel if any dangerous weapon is around or detected. There is a need to upgrade existing surveillance capabilities with better resources to monitor the performance of human operators. Smart surveillance systems could replace the current infrastructure as better video processing technologies, low-cost storage, and video infrastructure become more available. It is also possible that digital monitoring systems, including robots, may eventually replace current surveillance systems as computing becomes cheaper and video infrastructure and processing technologies improve. The importance of updating surveillance systems with improved resources to support monitoring is emphasized, and the potential for smart surveillance systems to fully replace current infrastructure in the future is mentioned. Additionally, a proposed method for alerting authorities about the presence of weapons in surveillance videos using IOT and Arduino board technology can be implemented in future.

## REFERENCES

- [1] Ingle, P. Y., & Kim, Y. G. (2022, May 19). Real-Time Abnormal Object Detection for Video Surveillance in Smart Cities. *Sensors*, 22(10), 3862. <https://doi.org/10.3390/s22103862>
- [2] Pachitariu, M., & Stringer, C. (2022, November 7). Cellpose 2.0: how to train your own model. *Nature Methods*, 19(12), 1634–1641. <https://doi.org/10.1038/s41592-022-01663-4>
- [3] Hanan Ashraf, A., Imran, M., M. Qahtani, A., Alsufyani, A., Almutiry, O., Mahmood, A., Attique, M., & Habib, M. (2022). Weapons Detection for Security and Video Surveillance Using CNN and YOLO-V5s. *Computers, Materials & Continua*, 70(2), 2761–2775. <https://doi.org/10.32604/cmc.2022.018785>
- [4] Yu, J., & Zhang, W. (2021, May 8). Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4. *Sensors*, 21(9), 3263. <https://doi.org/10.3390/s21093263>
- [5] Hsu, Y. W., Wang, T. Y., & Perng, J. W. (2020, February). Passenger flow counting in buses based on deep learning using surveillance video. *Optik*, 202, 163675. <https://doi.org/10.1016/j.ijleo.2019.163675>
- [6] Al-Azzoa, F., Mohammed, A., & Milanovab, M. (2018). Human Related-Health Actions Detection using Android Camera based on TensorFlow Object Detection API. *International Journal of Advanced Computer Science and Applications*, 9(10). <https://doi.org/10.14569/ijacsa.2018.091002>
- [7] Jimin Yu and Wei Zhang. Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4 (2021), *Sensors* 2021, 21(9)



- [8] Chien-Yao Wang, Alexey Bochkovskiy, HongYuan Mark Liao. Scaled-YOLOv4: Scaling Cross Stage Partial Network (2021), Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13029-13038
- [9] A Bochkovskiy, CY Wang, HYM Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection (2020), arXiv:2004.10934v1 [cs.CV];
- [10] R. Phadnis, J. Mishra and S. Bendale. Objects Talk - Object Detection and Pattern Tracking Using TensorFlow (2018) - Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 1216-1219
- [11] Wang Zhiqiang and Liu Jun. A Review of Object Detection Based on Convolutional Neural Network, Proceedings of the 36th Chinese Control Conference
- [12] Nadia Jmour, SehlaZayen, Abdelkrim. Convolutional Neural Networks for image classification.
- [13] S. A. Velastin, B. A. Boghossian, and M. A. VicencioSilva, “A motion-based image processing system for detecting potentially dangerous situations in underground railway stations,” *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 2, pp. 96–113, 2006.
- [14] H. Mousavi, S. Mohammadi, A. Perina, R. Chellali, and V. Murino, “Analyzing tracklets for the detection of abnormal crowd behavior,” in *Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 148–155, IEEE, Waikoloa, HI, USA, January 2015.
- [15] S. Ren, K. He, R. Girshick and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in Neural Information Processing Systems*, vol. 39, pp. 91–99, 2015.

- [16] ] L. Pang, H. Liu, Y. Chen, and J. Miao, “Real-time concealed object detection from passive millimeter wave images based on the YOLOv3 algorithm,” *Sensors*, vol. 20, no. 6, p. 1678, 2020.
- [17] R. Phadnis, J. Mishra and S. Bendale. Objects Talk - Object Detection and Pattern Tracking Using TensorFlow (2018) - Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 1216-1219.
- [18] F. Pérez-Hernández, S. Tabik, A. Lamas, R. Olmos and F. Herrera, “Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance,” *KnowledgeBased Systems*, vol. 194, pp. 212–223, 2020
- [19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, “Towards Real-Time Object Detection with Region Proposal Networks”, Jan 2016
- [20] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “YOLO, You Only Look Once: Unified, Real time object detection”, June 2015
- [21] A Bochkovskiy, CY Wang, HYM Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection (2020)
- [22] ] Jimin Yu and Wei Zhang. Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4 (2021), *Sensors* 2021.
- [23] A. Egiazarov, V. Mavroeidis, F. M. Zennaro and K. Vishi, “Firearm detection and segmentation using an ensemble of semantic neural networks,” in *European Intelligence and Security Informatics Conf.*, Norway, USA, pp. 70–77, 2020.

- [24] G. L. Hung, M. S. B. Sahimi, H. Samma, T. A. Almohamad and B. Lahasan, “Faster R-CNN deep learning model for pedestrian detection from drone images,” SN Computer Science, vol. 1, no. 2, pp. 1–9, 2020.
- [25] A. Castillo, S. Tabik, F. Perez, R. Olmos, and F. Herrera, ‘ “Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning,” Neurocomputing, vol. 330, pp. 151–161, 2019.
- [26] ] J. Xiong, S. Li, J. Yang, X. Xue and Z. Mao, “A novel Otsu method based on prior area information for concealed target detection in PMMW images,” MATEC Web of Conferences, vol. 59, pp. 81–92, 2016
- [27] Sanping Zhou, Jinjun Wang, Le Wang, Jimuyang Zhang, Fei Wang, Dong Huang, Nanning Zheng, “Hierarchical and Interactive Refinement Network for Edge-Preserving Salient Object Detection”, IEEE JOURNAL,2020.
- [28] Hsu, Y.W.; Wang, T.Y.; Perng, J.W, “Passenger flow counting in buses based on deep learning using surveillance video”, IEEE JOURNAL, 2019
- [29] R. Olmos, S. Tabik, A. Lamas, F. Pérez-Hernández and F. Herrera, “A binocular image fusion approach for minimizing false positives in handgun detection with deep learning,” Information Fusion, vol. 49, no. 2, pp. 271–280, 2019.
- [30] Castillo, S. Tabik, F. Pérez, R. Olmos and F. Herrera, “Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning,” Neurocomputing, vol. 330, no. 9, pp. 151–161, 2019.

- [31] A. Egiazarov, V. Mavroeidis, F. M. Zennaro and K. Vishi, “Firearm detection and segmentation using an ensemble of semantic neural networks,” in European Intelligence and Security Informatics Conf., Norway, USA, pp. 70–77, 2020.
- [32] M. Yang and K. Yu, “Real-time clothing recognition in surveillance videos,” in 18th IEEE Int. Conf. on Image Processing, NY, USA, pp. 2937–2940, 2011.
- [33] N. Kurek, L. A. Darzi and J. Maa, “A Worldwide perspective provides insights into why a US surgeon general annual report on firearm injuries is needed in America,” Current Trauma Reports, vol. 6, pp. 36–43, 2020
- [34] S. Ahmed, H. Wang, and Y. Tian, “Adaptive high-order terminal sliding mode control based on time delay estimation for the robotic manipulators with backlash hysteresis,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 2, pp. 1128–1137, 2021.
- [35] S. B. Kibria and M. S. Hasan, “An analysis of feature extraction and classification algorithms for dangerous object detection,” in Proceedings of the 2017 2nd International Conference on Electrical & Electronic Engineering (ICEEE), pp. 1–4, IEEE, Rajshahi, Bangladesh, December 2017.

## **APPENDIX 1**

This current section has the details on the language, software and packages that were used in the project. This project is developed using python programming language, which is a language that is interpreted, interactive and object oriented. It offers very easily readable code. With the despite of being highly complex with workflows when machine learning is written in Python it can help the developers create robust and intelligent applications and systems.

The list of packages used in our project:


### **1) OpenCV:**



It is an open-source library that is used in computer vision, ML and also image processing. It is also very highly optimized library that does focus on Realtime applications and also supports a variety of programming languages like C++, Java and also Python. It also provides a tool for object recognition systems and face detection and even to do hand recognition. The open cv library plays an important part in real time systems and their operations.

### **2) Darknet:**




DarkNet-19 is a 19-layer deep convolutional neural network. A pre-trained version of this network, trained on the ImageNet database with over a million images, is available for use. This pre-trained network can classify images into 1000 different object categories, including various animals and everyday objects such as keyboards, mice, and pencils. The network has an image input size has a learned rich feature representations for a wide range of images. More pre-trained networks are available in MATLAB. The DarkNet-19 model does classification of new images using classify function. To retrain the network for a new classification task, follow the steps outlined in the “Train Deep Learning Network to Classify New Images” guide and load DarkNet-19. DarkNet-19 is commonly used as a base for object detection problems and YOLO (You Only Look Once) workflows. Other pre-trained networks such as DarkNet-19, DarkNet-53, MobileNet-v2 or ResNet-18 can also be used depending on the requirements of the application.

# PAPER PUBLICATION STATUS

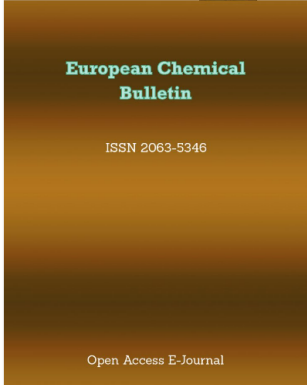
**European Chemical Bulletin**

  
NANO STUDIES

ISSN 2063-5346



About Abstracting and Indexing Archive About Publisher Editorial Policies Contact E-Board More ▾



European Chemical Bulletin

ISSN 2063-5346

Open Access E-Journal

Ayurvedic Management Of Vishada (Depression): A Systematic Review  
Volume -12 | Special Issue-6

ARTIFICIAL INTELLIGENCE IN E-RECRUITMENT – AN EMPIRICAL STUDY ON IT SECTOR  
Volume -12 | Special Issue-6

EXPLORING BIOMARKERS ASSOCIATED WITH TYPE 2 DIABETES MELLITUS  
Volume -12 | Special Issue-6

In-Vitro Antioxidant Activity Of Various Extract Of Leonotis Nepetifolia, Blumea Lacera, And Phyllanthus Acidus

### Submit Article:

Congratulations successfully submitted your article. our team will get back to you soon. Thank you ...!

Name

Email address

Mobile

City

State

Postal Code

Submit article

Indexed by



European Chemical Bulletin

Q3

Chemistry (miscellaneous)

best quartile

SJR 2022

0.25

powered by scimagojr.com



ELSEVIER


ORCID

Crossref

<b>SRM INSTITUTE OF SCIENCE AND TECHNOLOGY</b> (Deemed to be University u/s 3 of UGC Act, 1956)		
<b>Office of Controller of Examinations</b>		
<b>REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES</b> <b>(To be attached in the dissertation/ project report)</b>		
1	Name of the Candidate <b>(IN BLOCK LETTERS)</b>	CARLIN SAVIO JUDE FERNANDES
2	Address of the Candidate	FLAT NO 401, DORINA MANSION HOLY CROSS ROAD I.C. COLONY BORIVALI WEST MUMBAI-400103 <b>Mobile Number:</b> 7506491142
3	Registration Number	RA1911003010129
4	Date of Birth	08 June 1998
5	Department	Computer Science and Engineering
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	Weapon Detection using Deep Learning Model
8	Whether the above project /dissertation is done by	<p><del>Individual</del> or group : (Strike whichever is not applicable )</p> <p>a) If the project/ dissertation is done in group, then how many students together completed the project : 2 (Two)</p> <p>b) Mention the Name &amp; Register number of other candidates :</p>

9	Name and address of the Supervisor / Guide	<p>Dr.A.Pandian</p> <p>Associate Professor Department of Computer Science and Engineering SRM Institute of Science and Technology Kattankullatur - 603 203.</p> <p>Mail ID: <a href="mailto:pandiana@srmist.edu.in">pandiana@srmist.edu.in</a>.</p> <p>Mobile Number: 9150354754</p>
10	Name and address of Co-Supervisor / Co- Guide (if any)	<p>NIL</p> <p><b>Mail ID: Mobile Number:</b></p>



11	Software Used	Turnitin		
12	Date of Verification	14-May 2023		
13	<b>Plagiarism Details: (to attach the final report from the software)</b>			
Chapter	Title of the Chapter	Percentage of similarity index (including self citation)	Percentage of similarity index (Excluding self citation)	% of plagiarism after excluding Quotes, Bibliography, etc.,
1	INTRODUCTION			
2	LITERATURE SURVEY			
3	TECHNICAL SPECIFICATIONS			
4	ARCHITECTURE			
5	MODULES			
6	PROJECT DEMONSTRATION AND RESULTS			
7	CONCLUSION			
8	FUTURE ENHANCEMENTS			
9				
10				
Appendices				
I / We declare that the above information have been verified and found true to the best of my / our knowledge.				
 <b>Signature of the Candidate</b>		<b>Name &amp; Signature of the Staff (Who uses the plagiarism check software)</b>		
<b>Name &amp; Signature of the Supervisor/ Guide</b>		<b>Name &amp; Signature of the Co-Supervisor/Co-Guide</b>		
<b>Name &amp; Signature of the HOD</b>				

# PLAGIARISM REPORT

Weapon\_Detection\_Using\_DeepLearning-ProjectReport-test.docx

## ORIGINALITY REPORT

10%

SIMILARITY INDEX

7%

INTERNET SOURCES

5%

PUBLICATIONS

6%

STUDENT PAPERS

## PRIMARY SOURCES

1

github.com

Internet Source

1%

2

www.projectpro.io

Internet Source

1%

3

Submitted to Thapar University, Patiala

Student Paper

<1%

4

Submitted to Indian Institute of Technology Patna

Student Paper

<1%

5

Submitted to National College of Ireland

Student Paper

<1%

6

www.ijraset.com

Internet Source

<1%

7

Yaa Acquah, Jonathan B. Steele, Balakrishna Gokaraju, Raymond Tesiero, Gregory H. Monty. "Occupancy Detection for Smart HVAC Efficiency in Building Energy: A Deep Learning Neural Network Framework using Thermal

<1%

Imagery", 2020 IEEE Applied Imagery Pattern  
Recognition Workshop (AIPR), 2020  
Publication

8	<a href="http://www.ijsrd.com">www.ijsrd.com</a> Internet Source	<1 %
9	Viktar Atliha. "Improving image captioning methods using machine learning approaches", Vilnius Gediminas Technical University, 2023 Publication	<1 %
10	<a href="http://uk.mathworks.com">uk.mathworks.com</a> Internet Source	<1 %
11	<a href="http://www.mdpi.com">www.mdpi.com</a> Internet Source	<1 %
12	Submitted to CSU, San Marcos Student Paper	<1 %
13	<a href="http://dzone.com">dzone.com</a> Internet Source	<1 %
14	Submitted to Dr. S. P. Mukherjee International Institute of Information Technology (IIIT-NR) Student Paper	<1 %
15	Submitted to SRM University Student Paper	<1 %
16	Hao Xu, Yuan Cao, Qian Lu, Qiang Yang. "Performance Comparison of Small Object Detection Algorithms of UAV based Aerial	<1 %

