A Project report

on

ARM DETECTION IN SURVEILLANCE VIDEOS BY USING DEEP LEARNING ALGORITHMS

Submitted in partial fulfilment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

N MANASA VENKATA NAGA SAI	**(193N1A0540)**
K PRAMOD SAI	**(193N1A0524)**
T SAI DARSHAN	**(193N1A0552)**
S RAGHAVENDRA	**(193N1A0551)**
B KEERTHI	**(19711A0509)**

Under the esteemed guidance of

C REKHA, M.Tech.,

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

P.V.K.K Institute of Technology

An ISO 9001-2015 Certified Institute

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu.)

Sanapa Road, Rudrampeta, Anantapuramu-515001

Andhra Pradesh

2022-2023

P.V.K.K Institute of Technology

An ISO 9001-2015 Certified Institute
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu.)
Sanapa road, Rudrampeta, Anantapuramu-515001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that project report entitled **"ARM DETECTION IN SURVEILLANCE VIDEOS BY USING DEEP LEARNING ALGORITHMS "** is bonafide work done

by

N MANASA VENKATA NAGA SAI	**(193N1A0540)**
K PRAMOD SAI	**(193N1A0524)**
T SAI DARSHAN	**(193N1A0552)**
S RAGHAVENDRA	**(193N1A0551)**
B KEERTHI	**(19711A0509)**

Under the supervision of, Ms. **C REKHA,** M.Tech., Assistant Professor of CSE in the partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** with specialization in **COMPUTER SCIENCE AND ENGINEERING** in **P.V.K.K Institute of Technology**, Sanapa Road, Rudrampeta, Anantapuramu during the academic year 2022-2023.

Signature of the Project Guide	**Signature of H.O.D**
C REKHA, M.Tech.,	Dr. K Bhargavi, M.Tech, Ph.D.,
Assistant Professor	Professor & Head
Dept. of CSE	Dept. of CSE

External Viva-voice held on:___________________

External Examiner

DECLARATION

We here by declare that the project work entitled "**ARM DETECTION IN SURVEILLANCE VIDEOS BY USING DEEP LEARNING ALGORITMS**" is a genuine work carried out by us under the guidance of **Ms. C. REKHA,** **M.Tech.,** **Asst. Professor, Department of Computer Science & Engineering**, in partial fulfilment for the award of the degree of **B.Tech** of **Jawaharlal Nehru Technological University Anantapur, Anantapuramu**. This has not been submitted in part of fulfilment towards any other degree.

N MANASA VENKATA NAGA SAI

193N1A0540

K PRAMOD SAI	T SAI DARSHAN
193N1A0524	193N1A0552
S RAGHAVENDRA	B KEERTHI
193N1A0551	19711A0509

ACKNOWLEDGEMENT

At the outset, we thank our Honourable Founder, Correspondent & Secretary **Dr. Palle Raghunatha Reddy** M.Sc, M. Phil, Ph.D., garu, Honourable Chairman **Dr. Palle Venkata Krishna Kishore Reddy** M.Tech, MS (USA), AMIE...Ph.D., garu, and our Treasurer Smt. **P. Sindhura Reddy** M.Tech., garu of Sri Balaji Educational Society for providing us with good faculty and all the required equipment in C.S.E labs and for their moral support throughout the course.

I wish to express our sense of gratitude to **Dr. B. RAMESH BABU** **B.Tech, M.S, Ph.D.,** garu Principal, **P.V.K.K INSTITUTE OF TECHNOLOGY**, for providing necessary facilities.

With a great sense of pleasure, we extend our gratitude to **Dr. K. BHARGAVI** **M.Tech, Ph.D**, garu, Head of the Dept., Department of Computer Science & Engineering for her co-operation and providing necessary help for completion of this project.

It is a great pleasure to express my heartful gratitude and thankfulness to my guide **Ms. C. REKHA** **M.Tech.,** garu Assistant Professor, Department of Computer Science & Engineering for her valuable guidance and helpful counsel rendered in the due course of the project.

Finally, we would like to thank one and all in the C.S.E department who helped us directly and indirectly along with my family members and friends for their cooperation to complete this project.

PROJECT ASSOCIATES

N MANASA VENKATA NAGA SAI
K PRAMOD SAI
T SAI DARSHAN
S RAGHAVENDRA
B KEERTHI

ABSTRACT

Security and safety are big concerns for today's modern world. For a country to be economically strong, it must ensure a safe and secure environment for investors and tourists. Having said that, Closed Circuit Television (CCTV) cameras are being used for surveillance and to monitor activities i.e., robberies but these cameras still require human supervision and intervention. We need a system that can automatically detect these illegal activities. Despite state-of-the-art deep learning algorithms, fast processing hardware, and advanced CCTV cameras, weapon detection in real-time is still a serious challenge. Observing angle differences, occlusions by the carrier of the firearm and persons around it further enhances the difficulty of the challenge.

This work focuses on providing a secure place using CCTV footage as a source to detect harmful weapons by applying the state of the art open-source deep learning algorithms. No standard dataset was available for real-time scenario so we made our own dataset by making weapon photos from our own camera, manually collected images from internet, extracted data from YouTube CCTV videos, through GitHub repositories, data by university of Granada and Internet Movies Firearms Database (IMFDB) imfdb.org. Some of the algorithms used are VGG16, Inception-V3, Inception-ResnetV2, SSDMobileNetV1, Faster-RCNN Inception-ResnetV2 (FRIRv2), YOLOv3, and YOLOv4. Precision and recall count the most rather than accuracy when object detection is performed so these entire algorithms were tested in terms of them. Yolov4 stands out best amongst all other algorithms and gave a F1-score of 91% along with a mean average precision of 91.73% higher than previously achieved.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

Acronym	Abbreviation
CCTV	Closed Circuit Television
CNN	Convolutional Neural Network
SIFT	Scale Invariant Feature Transform
FREAK	Fast Retina Key point
VGG	Visual Geometric Group
ILSVRC	Image net Large Scale Visual Recognition Challenge
CIFAR-10	Canadian Institute for Advance Research
COCO	Common Objects in Context
RCNN	Region Based Convolutional Neural Network
DCN	Deep Convolutional Network
IMFDB	Internet Movie Firearms Data Base
AATPI	Alarm Activation Time Per Interval
ML	Machine Learning
NLP	Natural Language Processing
RNN	Recurrent Neural Nets
RGB	Red Green Blue
ReLU	Rectified Linear Unit
SGD	Stochastic gradient Descent
YOLO-v3	You Only Look Once -Version 3
ROI	Region of Interest
FPS	Frame rate Per Second
IOU	Intersection Over Union
AR	Augmented Reality
GPU	Graphic Processing Unit
P-4	Pentium 4 Quad core
OpenCV	Open Source Computer Vision Library
CUDA	Compute Unified Device Architecture
CPU	Control Processing Unit

Acronym	Abbreviation
NN	Neural Network
IDE	Integrated Development Environment
IDLE	Integrated Development Learning Environment
CVS	Chorionic Villus Sampling
UML	Unified Modelling Language
JPEG	Joint Photographic Expert Group
PNG	Portable Network Graphics
MAP	Mean Average Precession
PR	Precision-Recall
SSD	Solid State Drive

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

The crime rate across the globe has increased mainly because of the frequent use of handheld weapons during violent activity. For a country to progress, the law-and-order situation must be in control. Whether we want to attract investors for investment or to generate revenue with the tourism industry, all these needs is a peaceful and safe environment. The crime ratio because of guns is very critical in numerous parts of the world. It includes mainly those countries in which it is legal to keep a firearm. The world is a global village now and the associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh. What we speak or write has an impact on the people. Even if the news they heard is crafted having no truth but as it gets viral in a few hours because of the media and especially social media, the damage will be done. People now have more depression and have less control over their anger, and hate speeches can get those people to lose their minds. People can be brainwashed and psychological studies show that if a person has a weapon in this situation, he may lose his senses and commit a violent activity.

This project presents an automatic detection and classification method of weapons for real-time scenario using state of the art deep learning models. For real-time implementation relating the problem question of this work ''detecting weapons in real-time for potential robbers/terrorist using deep learning'', detection and classification was done for pistol, revolver and other shot handheld weapons as in single class called pistol and related confusion objects such as cell phone, metal detector, wallet, selfie stick in not pistol class. A major reason behind this was our research done on weapons used in robbery cases and it further motivated us to choose pistol and revolver as our target object.

We go through several CCTV captured robbery videos on YouTube and found that almost 95% of cases have pistol or revolver as the weapon used.

Deep learning models faced several below mentioned challenges for detection and classification task:

- The first and main problem is the data through which CNN learn its features to be used later for classification and detection.
- No standard dataset was available for weapons.

- For real-time scenarios, making a novel dataset manually was a very long and time-consuming process.
- Labelling the desired database is not an easy task, as all data needs to be labelled manually.
- Every algorithm requires different labelling and pre-processing operations for the same-labelled database.
- As for real-time implementation, detection systems require the exact location of the weapon so gun blocking or occlusion is also a problem
- Different detection algorithms were used, so a labelled dataset for one algorithm cannot be utilized for the other one. That arises frequently and it could occur because of self, inter-object, or background blocking.

CHAPTER 2
SYSTEM ANALYSIS

CHAPTER 2
SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

We need a system that can automatically detect these illegal activities. Despite state-of-the-art deep learning algorithms, fast processing hardware, and advanced CCTV cameras, weapon detection in real-time is still a serious challenge. Observing angle differences, occlusions by the carrier of the firearm and persons around it further enhances the difficulty of the challenge. This work focuses on providing a secure place using CCTV footage as a source to detect harmful weapons by applying the state of the art open-source deep learning algorithms. We have implemented binary classification assuming pistol class as the reference class and relevant confusion objects inclusion concept is introduced to reduce false positives and false negatives. The crime ratio because of guns is very critical in numerous parts of the world. It includes mainly those countries in which it is legal to keep firearm. The world is a global village now and what we speak or write has an impact on the people. Even if the news they heard is crafted having no truth but as it gets viral in a few hours because of the media and especially social media, the damage will be done.

2.2 ADVANTAGES OF EXISTING SYSTEM:

- Ensure the safety of people on site.
- Give people entry and exit process agility.
- Avoid Embarrassment.
- Prevent incoming weapons and sharp objects.

2.3 LIMITATIONS OF EXISTING SYSTEM:

- Some dry and wet foods can create false positives.
- The ability to detect all metal contaminants.

2.4 PROPOSED SYSTEM:

Deep learning is a branch of machine learning inspired by the functionality and structure of the human brain also called an artificial neural network. The methodology adopted in this work features the state of art deep learning, especially the convolutional neural networks due to their exceptional performance in this field. The aforementioned techniques are used for both the classification as well as localising the specific object in a frame so both the object classification and detection algorithms were used and because our object is small with other object in background so after experimentation, we found the best algorithm for our case. Sliding window/classification and region proposal/object detection algorithms were used, and these techniques will be discussed later in this section. We had started by doing the classification using different deep [learning models and achieved good precision but for the real-time scenarios, the low frame per seconds of classification models were the real issue in implementation. Oxford VGG, Google Inceptionv3 and InceptionResnetv2 were trained using the aforementioned approach.

Different datasets were made keeping in mind the classification and detection problem as both have a separate requirement for performing the tasks to achieve high accuracy, mean average precision as well as frame per second for the real time implementation.

To understand object classification and detection let us first briefly understand object recognition as both the aforementioned types come under the umbrella of this and combined classification and localisation make detection possible for any kind of detection problem giving class name as well as the region where our desired object is in the frame.

2.5 ADVANTAGES OF PROPOSED SYSTEM:

- Crime prevention
- Rapid response.
- It is able to produce a faster result.
- Safety for people

2.6 REVIEW OF LITERATURE:

[1] CCTV as an automated sensor for firearms detection: Human-derived performance as a precursor to automatic recognition.

CCTV operators are able to detect firearms, via CCTV, but their capacity for surveillance is limited. Thus, it is desirable to automate the monitoring of CCTV cameras for firearms using machine vision techniques. The abilities of CCTV operators to detect concealed and unconcealed firearms in CCTV footage were quantified within a signal detection framework. Additionally, the visual search strategies adopted by the CCTV operators were elicited and their efficacies indexed with respect to signal detection performance, separately for concealed and unconcealed firearms. Future work will automate effective, human visual search strategies using image processing algorithms.

[2] Automatic image analysis process for the detection of concealed weapons.

The goal of this research is to develop a process, using current imaging hardware and without human intervention, that provides an accurate and timely detection alert of a concealed weapon and its location in the image of the luggage. There are several processes in existence that are able to highlight or otherwise outline a concealed weapon in baggage but so far those processes still require a highly trained operator to observe the resulting image and draw the correct conclusions. We attempted three different approaches in this project. The first approach uses edge detection combined with pattern matching to determine the existence of a concealed pistol. Rather than use the whole body of the weapon which varies significantly, the trigger guard was used since it is fairly consistent in dimensions. While the processes were reliable in detecting a pistol's presence, on any but the simplest of images, the computational time was excessive and a substantial number of false positives were generated. The second approach employed Daubechie wavelet transforms but the results have so far been inconclusive. A third approach involving an algorithm based on the scale invariant feature transform (SIFT) is proposed.

[3] A computer vision-based framework for visual gun detection using Harris interest point detector.

Today's automatic visual surveillance is prime need for security and this paper presents first step in the direction of automatic visual gun detection. The objective of our paper is to develop a framework for visual gun detection for automatic surveillance. The proposed framework exploits the color-based segmentation to eliminate unrelated object from an image using k-mean clustering algorithm. Harris interest point detector and Fast Retina Keypoint (FREAK) is used to locate the object (gun) in the segmented images. Our framework is robust enough in terms of scale, rotation, affine and occlusion. We have implemented and tested the system over sample images of gun, collected by us. We got promising performance of our system to detect a gun. Further, our system performs very well under different appearance of images. Thus, our system is rotation, scale and shape invariant.

[4] Deep residual learning for image recognition.

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

[5] Automatic detection of concealed pistols using passive millimeter wave imaging.

A method is proposed for automatic detection of the concealed pistols detected by passive millimeter in security applications. In this paper, we extend four half-surrounded Hear-like features and use integral image to rapidly calculate the rectangle features. Then we obtain a multi-layer classifier cascaded by several strong classifiers using AdaBoost algorithm to detect the contraband. Various passive millimeter images from both published literatures and our own measurements are used for training and testing. The experimental results show that the metallic pistols in different sizes, shapes, and angles can be accurately detected, so this method is useful for automatic detection of pistols.

[6] A handheld gun detection using faster RCNN deep learning.

Today's, most of the criminal activities are taken place using handheld arms particularly gun, pistol and revolver. Several surveys revealed that hand held gun is the foremost weapon used for diverse crimes like burglary, rape, etc. Therefore, automatic gun detection is a prime requirement in current scenario and this paper presents automatic gun detection from cluttered scene using Convolutional Neural Networks (CNN). We have used Deep Convolutional Network (DCN), a state-of-the-art Faster Region-based CNN model, through transfer learning, for automatic gun detection from cluttered scenes. We have evaluated our gun detection over Internet Movie Firearms Database (IMFDB), a benchmark gun database. For detecting the visual handheld gun, we got propitious performance of our system. Moreover, we demonstrate that, against the number of several training images, CNN model magnifies the classification accuracy, which is most advantageous in those practices where generous liberal is often not available.

[7] Automatic handgun detection alarm in videos using deep learning

Current surveillance and control systems still require human supervision and intervention. This work presents a novel automatic handgun detection system in videos appropriate for both, surveillance and control purposes. We reformulate this detection problem into the problem of minimizing false positives and solve it by Building the key training data-set guided by the results of a deep Convolutional Neural Networks (CNN) classifier.

Assessing the best classification model under two approaches, the sliding window approach and region proposal approach.

The most promising results are obtained by Faster R-CNN based model trained on our new database. The best detector shows a high potential even in low quality YouTube videos and provides satisfactory results as automatic alarm system. Among 30 scenes, it successfully activates the alarm after five successive true positives in a time interval smaller than 0.2 s, in 27 scenes. We also define a new metric, Alarm Activation Time per Interval (AATpI), to assess the performance of a detection model as an automatic detection system in videos.

[8] A dynamic sliding window approach for activity recognition.

Human activity recognition aims to infer the actions of one or more persons from a set of observations captured by sensors. Usually, this is performed by following a fixed length sliding window approach for the features extraction where two parameters have to be fixed: the size of the window and the shift. In this paper we propose a different approach using dynamic windows based on events. Our approach adjusts dynamically the window size and the shift at every step. Using our approach, we have generated a model to compare both approaches. Experiments with public datasets show that our method, employing simpler models, is able to accurately recognize the activities, using fewer instances, and obtains better results than the approaches used by the datasets authors.

[9] Data preprocessing for supervised leaning.

Many factors affect the success of Machine Learning (ML) on a given task. The representation and quality of the instance data is first and foremost. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. It is well known that data preparation and filtering steps take considerable amount of processing time in ML problems. Data pre-processing includes data cleaning, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set. Thus, we present the most well know algorithms for each step of data pre-processing so that one achieves the best performance for their data set.

CHAPTER 3
SOFTWARE REQUIREMENT ANALYSIS & SPECIFICATIONS

CHAPTER 3
SOFTWARE REQUIREMENT ANALYSIS & SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS:

Processor	-	P–IV
RAM	-	4 GB (min)
Hard Disk.	-	40 GB (min)

3.2 SOFTWARE REQUIREMENTS:

Operating system	-	Windows 7 Ultimate or above.
Scripting Language.	-	Python.
Front-End	-	OpenCV
Back-End	-	YoloV3, DarkNet
Python IDE.	-	PyCharm

3.2.1 PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

3.2.2 YOLO V3:

YOLO V3 (You Only Look Once Version3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect an object. Versions 1-3 of YOLO were created by Joseph Redmon and Ali Farhadi, and the third version of the YOLO machine learning algorithm is a more accurate version of the original ML algorithm. The first version of YOLO was created in 2016, and version 3, which is discussed extensively in this article, was made two years later in 2018. YOLOv3 is an improved version of YOLO and YOLOv2. YOLO is implemented using the Keras or OpenCV deep learning libraries. The official successors of YOLOv3 are YOLOv4, and the newly released YOLOv7 (2022), which marks the current state-of-the-art object detector in 2023.

3.2.3 DARKNET:

Darknet is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation."

Darknet-53 is a convolutional neural network that acts as a backbone for the YOLOv3 object detection approach. The improvements upon its predecessor Darknet-19 include the use of residual connections, as well as more layers.

Darknet is mainly for Object Detection, and have different architecture, features than other deep learning frameworks. It is faster than many other NN architectures and approaches like Faster CNN etc. You have to be in C if you need speed, and most of the DNN frameworks are written in C.

3.2.4 PYCHARM:

PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains.

It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

In addition, the IDE provides capabilities for professional Web development using the Django framework. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.

Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages

Python code refactoring: including rename, extract method, introduce variable, introduce constant, pull up, push down and others

Support for web frameworks: Django, web2py and Flask, Integrated Python debugger, Integrated unit testing with line-by-line coverage, Google App Engine Python development.

Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with change lists and merge

Scientific tools integration: integrates with IDLE Python Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.

3.3 SOFTWARE ENVIRONMENT:

3.3.1 INTRODUCTION TO CNN:

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

A convolution is essentially sliding a filter over the input. One helpful way to think about convolutions is this quote from Dr Prasad Samarakoon: "A convolution can be thought as "looking at a function's surroundings to make better/accurate predictions of its outcome."

Rather than looking at an entire image at once to find certain features it can be more effective to look at smaller portions of the image.

3.3.1.1 Common uses for CNNs:

The most common use for CNNs is image classification, for example identifying satellite images that contain roads or classifying hand written letters and digits. There are other quite mainstream tasks such as image segmentation and signal processing, for which CNNs perform well at.

CNNs have been used for understanding in Natural Language Processing (NLP) and speech recognition, although often for NLP Recurrent Neural Nets (RNNs) are used.

A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement.

3.3.1.2 Interesting uses for CNNs other than image processing:

More and more diverse and interesting uses are being found for CNN architectures. An example of a non-image-based application is "The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference" by Lex Flagel et al. This is used to perform selective sweeps, finding gene flow, inferring population size changes, inferring rate of recombination.

There are researchers such as Professor Gerald Quan at the Quantitative biology lab, using CNNs for generative models in single cell genomics for disease identification.

CNNs are also being used in astrophysics to interpret radio telescope data to predict the likely visual image to represent the data.

Deep mind's Wave Net is a CNN model for generating synthesized voice, used as the basis for Google's Assistant's voice synthesizer.

3.3.1.3 Convolutional Kernels:

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map.

These are linear transformations; each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two-dimensional matrix) and a 3x3 convolutional kernel (a two- dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

In other non-vision applications, a one-dimensional convolution may slide vertically over an input matrix.

3.3.1.4 Creating a feature map from a convolutional kernel:

Below is a diagram showing the operation of the convolutional kernel.

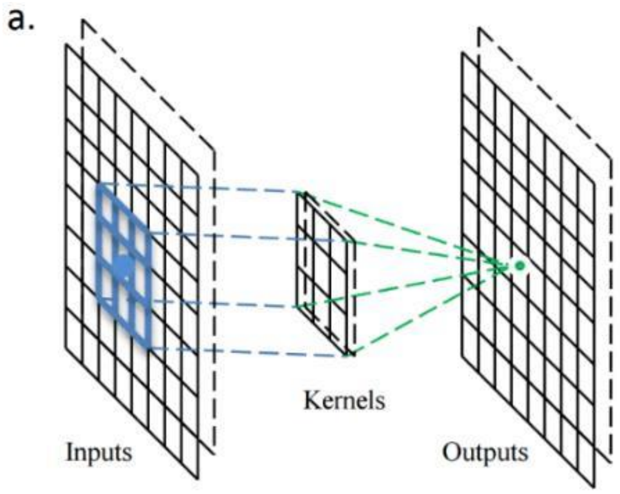


Fig. 3.1: The operation of the convolutional kernel.

A stride one 3x3 convolutional kernel acting on an 8x8 input image, outputting an 8x8 filter/channel shown in (Fig- 3.1) .
Below is a visualisation from an excellent presentation, (Fig-3.2) showing the kernel scanning over the values in the input matrix.

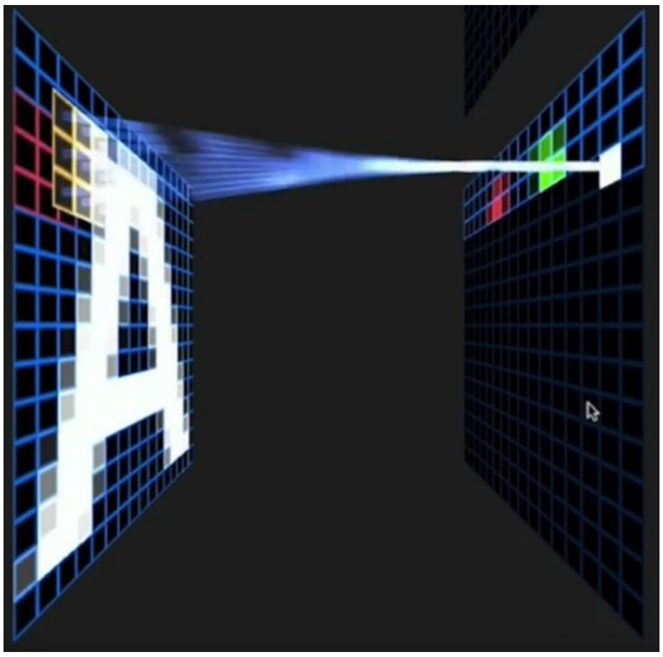


Fig. 3.2: Kernel scanning over the values in the input matrix

3.3.1.5 Padding:

To handle the edge pixels there are several approaches

- Reflection padding
- Losing the edge pixels
- Padding with zero value pixels

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from an input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

3.3.1.6 Creating multiple channels/feature maps with multiple kernels:

When multiple convolutional kernels are applied within a convolutional layer, By the image of (Fig-3.3) it has many channels/feature maps are created, one from each convolutional kernel.

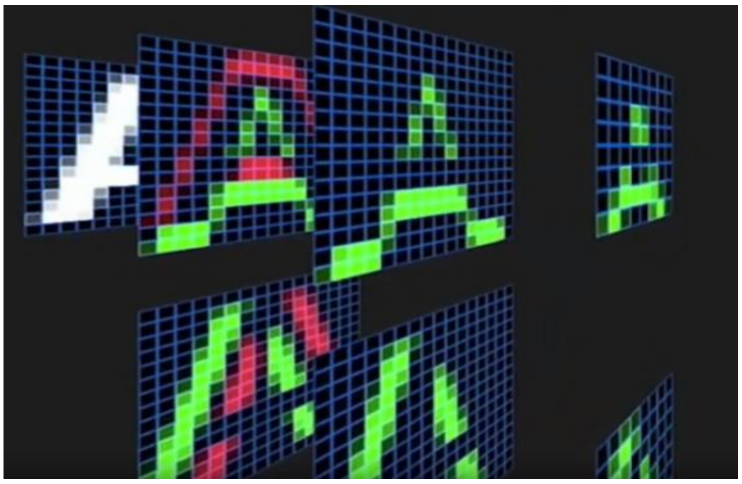


Fig. 3.3: Visualisation of feature maps created from a layer of convolutional kernels.

3.3.1.7 RGB 3 channel input:

Most image processing needs to operate on RGB images with three channels. A RGB image is a three-dimensional array of numbers otherwise known as a rank three tensor.

When processing a three channel RGB image, a convolutional kernel hat is a three-dimensional array/rank 3 tensor of numbers would normally be used. It is very common for the convolutional kernel to be of size 3x3x3 —the convolutional kernel being like a cube.

Usually there is at least three convolutional kernels in order that each can act as a different filter to gain insight from each colour channel.

The convolution kernels as a group make a four-dimensional array, otherwise known as a rank four tensor. It is difficult, if not impossible, to visualise dimensions when they are higher than three. In this case imagine it as a list of three-dimensional cubes.

The filter moves across the input data in the same way, sliding or taking strides across the rows then moving down the columns and striding across the rows until it reaches the bottom right corner:

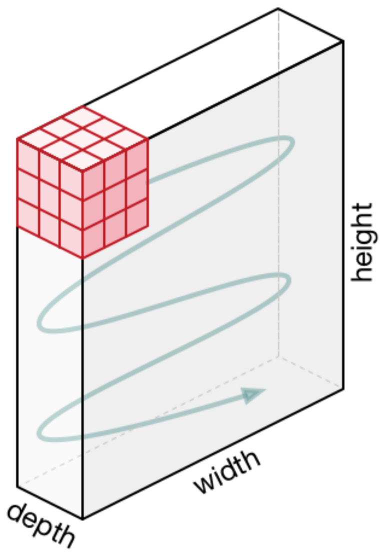


Fig. 3.4: 3x3x3 convolutional kernel acting on a 3-channel input.

With padding and a stride of one, the output from an input of width x, height y and depth 3 would be width x, height y and depth 1, as the (Fig-3.4) shows a cube produces a single summed output value from each stride. For example, with an input of 3x64x64 (say a 64x64 RGB three channel image) then one kernel taking strides of one with padding the edge pixels would output a channel/feature map of 64x64 (one channel).

3.3.1.8 Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to 1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

With padding, the output from an input of width w, height h and depth 3 would be the ceiling of width w/2, height h/2 and depth 1, as the kernel outputs a single summed output from each stride.

For example, with an input of 3x64x64 (say a 64x64 RGB three channel image), one kernel taking strides of two with padding the edge pixels, would produce a channel/feature map of 32x32.

3.3.1.9 Many kernels:

In CNN models there are often there are many more than three convolutional kernels, 16 kernels or even 64 kernels in a convolutional layer are common.

These different convolution kernels each act as a different filter creating a channel/feature map representing something different. For example, kernels could be filtering top edges, bottom edges, diagonal lines and so on. In much deeper networks these kernels could be filtering to animal features such as eyes or bird wings.

Having a higher number of convolutional kernels creates a higher number of channels/feature maps and a growing amount of data and this uses more memory. The stride 2 convolution, as per the above example, helps to reduce the memory usage as the output channel of the stride 2 convolution has half the width and height of the input. This assumes reflection padding is being used otherwise it could be slightly smaller.

3.3.1.10 An example of several convolutional layers of stride 2:

With a 64-pixel square input with three channels and 16 3x3x3 kernels our convolutional layer would have:

Input:64x64x3

Convolutional kernels: 16x3x3x3 (a four-dimensional tensor)

Output/activations of the convolutional kernels: 16x32x32 (16 channels/feature maps of 32x32)

The network could then apply batch normalisation to decrease learning time and reduce overfitting, more details below. In addition, a non-linear activation function, such as RELU is usually applied to allow the network to approximate better, more details below.

Often there are several layers of stride 2 convolutions, creating an increasing number of channels/feature maps. Taking the example above a layer deeper:

Input: 16x32x32

Convolutional kernels: 64x3x3x3

Output/activations of the convolutional kernels: 64x16x16 (64 channels/feature maps of 16x16)

Then after applying ReLU and batch normalisation (see below), another stride 2 convolution is applied:

Input: 64x16x16

Convolutional kernels: 128x3x3x3

Output/activations of the convolutional kernels: 128x8x8 (128 channels/feature maps of 8x8).

3.3.1.11 Classification:

If, for example, an image belongs to one of 42 categories and the network's goal is to predict which category the image belongs to.

Following on from the above example with an output of 128x8x8, first the average pool of the rank 3 tensor is taken. The average pool is the mean average of each channel, in this example each 8x8 matrix is averaged into a single number, with 128 channels/feature maps. This creates 128 numbers, a vector of size 1x128.

The next layer is a matrix or rank 2 tensors of 128x42 weights. The input 1x128 matrix is (dot product) multiplied by the 128x42 matrix producing a 1x42 vector. How activated each of the 42 grid cells/vector elements are, is how much the prediction matches that classification represented by that vector element. SoftMax is applied as an activation function and then argmax to select the element highest value.

3.3.1.12 Normalisation:

Normalisation is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.

It is common to normalize the input features, standardising the data by removing the mean and scaling to unit variance. It is often important the input features are centred around zero and have variance in the same order.

With some data, such as images the data is scaled so that its range is between 0 and 1, most simply dividing the pixel values by 255.

This also allows the training process to find the optimal parameters quicker.

- **Batch normalisation:**

  Batch normalisation has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularisation and speeds up training by an order of magnitude.

  Batch normalisation is the process of carrying normalisation within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batch's activations.

  This is necessary as even after normalizing the input as some activations can be higher, which can cause the subsequent layers to act abnormally and makes the network less stable.

  As batch normalisation has scaled and shifted the activation outputs, the weights in the next layer will no longer be optimal. Stochastic gradient descent (SGD) would undo the normalisation, as it would minimise the loss function.

  To prevent this effect two trainable parameters can be added to each layer to allow SGD.

  To de-normalise the output. These parameters are a mean parameter "beta" and a standard deviation parameter "gamma". VGG-16 Network Architecture is the complete structural formatof processing shown in (Fig-3.5)Batch normalisation sets these two weights for each activation output to allow the normalisation to be reversed to get the raw input, this avoids affecting the stability of the network by avoiding having to update the other weights.

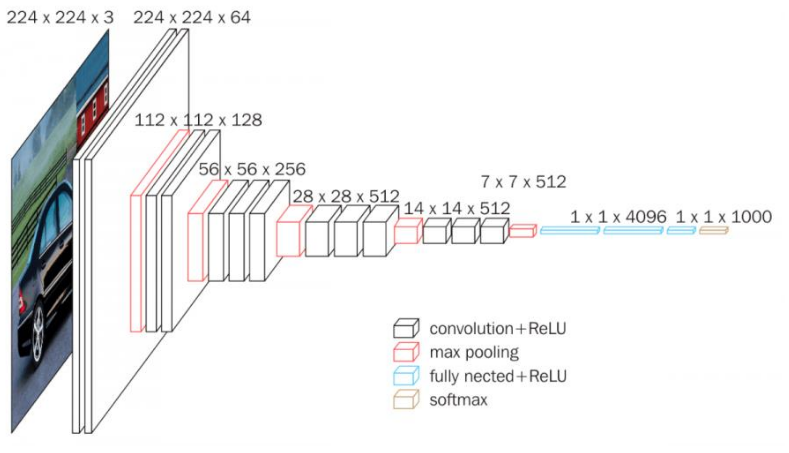


Fig. 3.5: VGG-16 Network Architecture

3.3.1.13 Design Requirements:

YOLOv3 is an object detection algorithm based on convolutional neural network and uses the concept of bounding box regression. Instead of processing region of interest (ROI) to detect the target object, it directly predicts bounding boxes and its classes for the image with a single stage architecture.

YOLOv3 components consists of feature extractor, prediction layer and grid cells with anchor boxes. Given an image, the feature extractor compute the salient feature and it is passed to the prediction layer to produce the predicted bounding boxes.

3.3.1.14 Feature Extractor:

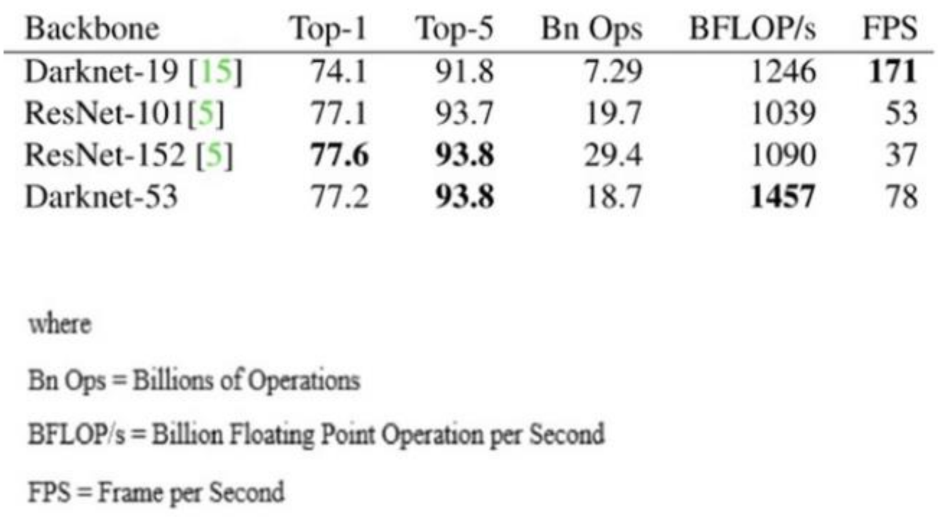
The Darknet-53 is a feature extractor designed specifically for YOLOv3 and it consists of 53 convolutional layers as shown in Fig. 3.5. The Darknet-53 feature extractor or backbone is chosen mainly due to its high accuracy when compared to other models. Darknet-19 is a smaller model with 19 convolutional layers. Table 3.1, shows the performance of Darknet- 53 is higher when compared to Darknet-19 and ResNet-101. However, when compared with Darknet-19 as backbone, the detection speed of YOLOv3 with Darknet-53 is slower in terms of detection speed as measured in FPS (Frame Rate per Second).

- **Bounding Box Regression:**

  The bounding box is defined as the rectangular spatial box that defines the predicted object location in the image. Every bounding box has width (b_w), height (b_h), class (knife or pistol) represented by c and bounding box centre (b_x, b_y). Figure 3.7 shows the bounding box definition. During image classification and localisation, the model will predict bounding box, class of the object, centre of bounding box and probability of object in bounding box.

- **Grid Cells and Anchor Boxes:**

  The YOLOv3 algorithm works by dividing the image into N grid cells with equal size (S × S). Each grid cell has a set of predefined anchor boxes with certain height and width.



Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

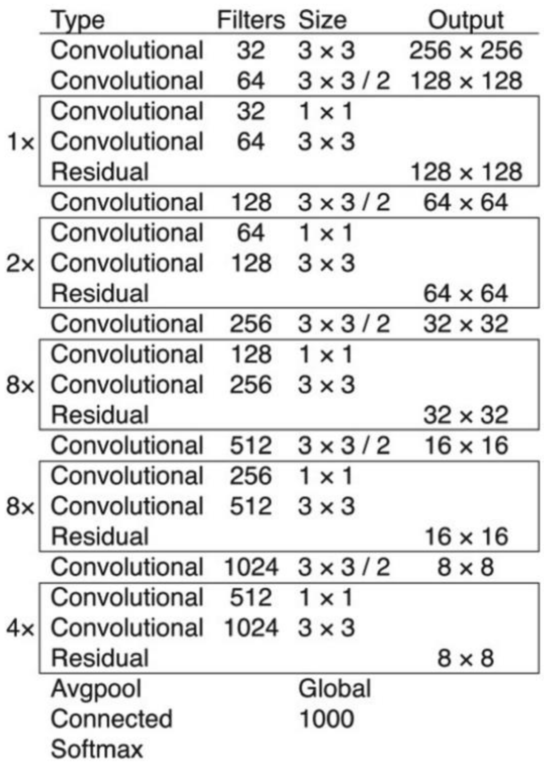
where

Bn Ops = Billions of Operations

BFLOP/s = Billion Floating Point Operation per Second

FPS = Frame per Second

Tab. 3.1: Comparison of Backbone with different frameworks.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3.6: Architecture of Darknet-53

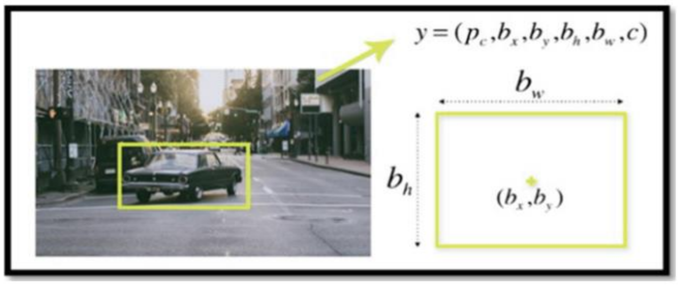


Fig. 3.7: Bounding the detected object

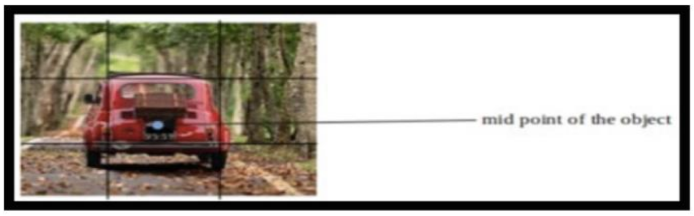


Fig. 3.8: Example of grid cells of size (3 × 3)

By Considering of (Fig-3.6)it shows the process of ongoing comparisons in different frameworks and it Each anchor box predicts the object label. Shows in (Fig-3.7, 3.8), probability of the object presents in the cell within the anchor box and the bounding box coordinate (w, h, x, y). For instance, in Fig. 3.9, the image is divided into 9 (3 × 3) grid cells. Each grid cell with anchor boxes, predicts the presence of the object in the cell. The size of the grid cell will influence the ability of the model to make prediction for small object. Therefore, smaller grid cell is used for detecting smaller object as shown in Fig. 3.9.

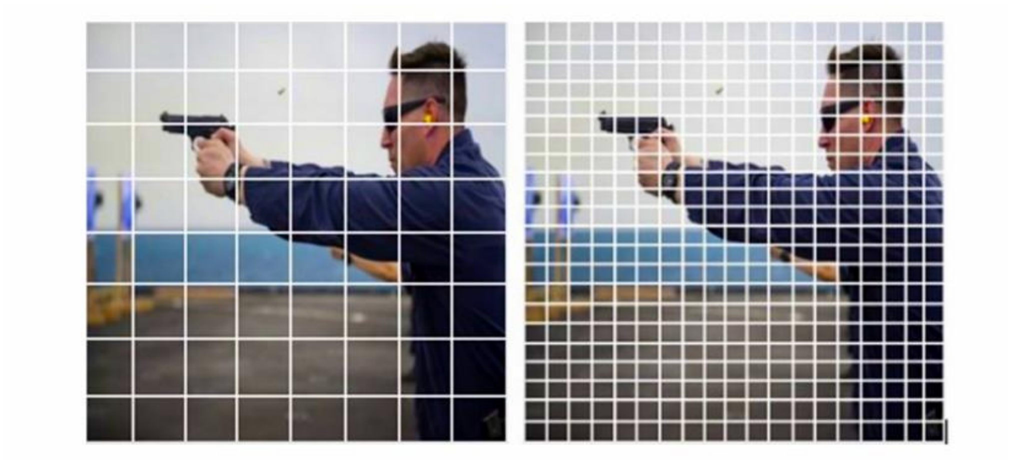


Fig. 3.9: Comparison between large grid cells and small grid cells

3.3.2 OPTIMISATION OF YOLO-V3 DARKNET-53:

Modification on YOLOv3 model has been made to improve the performance of the model for detecting gun and pistol. The modification is made by using custom anchor boxes, dataset expansion and adding extra prediction layer.

3.3.2.1 Custom Anchor Boxes:

By default, YOLOv3 is designed for object detection in COCO (Common Objects in Context) dataset for 80 object categories. Hence, the model needs to be modified to accommodate the detection of object from two classes namely knife and pistol. In addition, anchor boxes need to be customised to detect the small knife and pistol object in the Sohas weapon detection dataset.

The custom anchor boxes are generated by using K-Means clustering on the Sohas training dataset. K-means clustering is an unsupervised algorithm to group similar data points together in order to discover the underlying pattern. This can be achieved by grouping the data into a selected number of clusters. A total of 4014 data points has been used to obtain anchor boxes dimension that cover the pistol and knife objects in the training image set. Next, the K-means clustering is used to group them into predetermined number of clusters.

Experiment on the use of different number of clusters and its impact on the coverage of the ground truth object is shown in Fig. 3.10. The higher the number of clusters, the performance as measured by Average IOU (Intersection Over Union) becomes higher. The number of cluster N is chosen to be 9 since the curve is flattened at N=9. Therefore, 9 anchor boxes is defined for each grid cell and the anchor box parameters are obtained from the cluster's centroid.

3.3.2.2 Dataset:

In order to evaluate the performance of the detection model, the Sohas weapon detection dataset is used. The dataset contains 4014 images with weapons. The images are categorised based on the type of handheld weapon objects used, namely pistol and knife. A total of 3250 images is used for training and 764 images are used for testing the trained model. The number of images for the two classes is approximately balanced. A total of 1425 images from the pistol category and 1825 images from the knife category is used for training. The test set consist of 374 pistol images and 390 knife images.

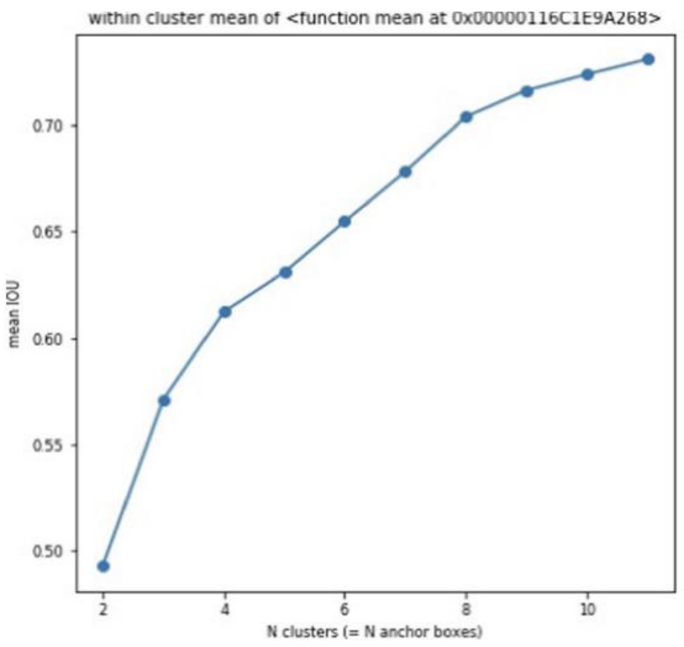
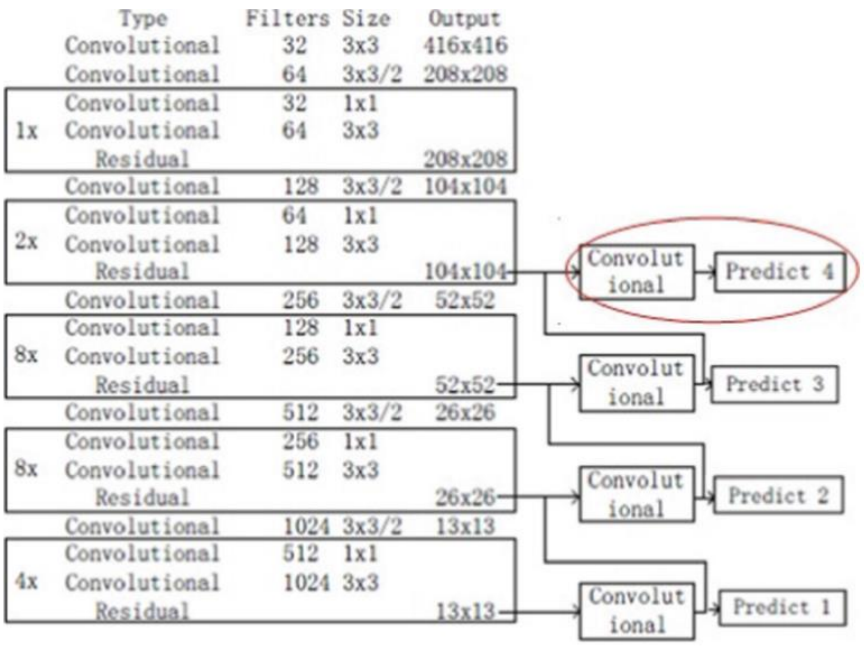


Fig. 3.10: Graph of test set consisting of 374 pistol images and 390 knife images.

The Sohas dataset lacks blurry image sample typically found in surveillance video frame. Therefore, the training and test samples is expanded with blurry images from surveillance video to obtain a larger and more diverse dataset. This can help the model to generalise better with images obtained from surveillance videos. The additional images are obtained by extracting the images from the YouTube video. Five surveillance videos obtained from Closed-Circuit Television (CCTV) camera are downloaded from YouTube. One frame is extracted every ten seconds of the video and the image that contained weapons are used. This dataset is named as Dataset 1 with a total of 479 images. A total of 339 images is allocated for model training and 140 images for testing.

3.3.2.3 Addition of Prediction Layer:

As the knife and pistol objects are seen smaller in the video, sometimes it can be undetected by the YOLOv3 model.We propose to add another prediction layer with smaller scale to the YOLOv3 backbone. The default YOLOv3 model consists only of three prediction layers. The prediction layer Predict 1 with grid cells size 13 × 13 is used for detecting large object. The Predict 2 layer is used for detecting the medium size object and it has the grid cells of size 26 × 26. Predict 3 layer has the resolution 52 × 52 and the scale is still not suitable for detecting small object like knife and gun. We add another prediction layer (Predict 4) at grid cells size 104 × 104 to cater for the detection of smaller object. The given figure 3.11 the Predict 4 layer added to the original YOLOv3 network.



	Type	Filters	Size	Output
	Convolutional	32	3x3	416x416
	Convolutional	64	3x3/2	208x208
1x	Convolutional	32	1x1	
	Convolutional	64	3x3	
	Residual			208x208
	Convolutional	128	3x3/2	104x104
2x	Convolutional	64	1x1	
	Convolutional	128	3x3	
	Residual			104x104
	Convolutional	256	3x3/2	52x52
8x	Convolutional	128	1x1	
	Convolutional	256	3x3	
	Residual			52x52
	Convolutional	512	3x3/2	26x26
8x	Convolutional	256	1x1	
	Convolutional	512	3x3	
	Residual			26x26
	Convolutional	1024	3x3/2	13x13
4x	Convolutional	512	1x1	
	Convolutional	1024	3x3	
	Residual			13x13

Fig. 3.11: Predictions made at different layers in CNN

Dataset	Results	
	Mean Average Precision (mAP)	Frame Rate Per Second (FPS)
Sohas Dataset (YOLOv3 Darknet-53)	88.97%	10.25
Sohas Dataset + Dataset 1 (YOLOv3 Darknet-53)	89.65%	10.27

Tab. 3.2: Comparison of performance for YOLOv3 with different dataset.

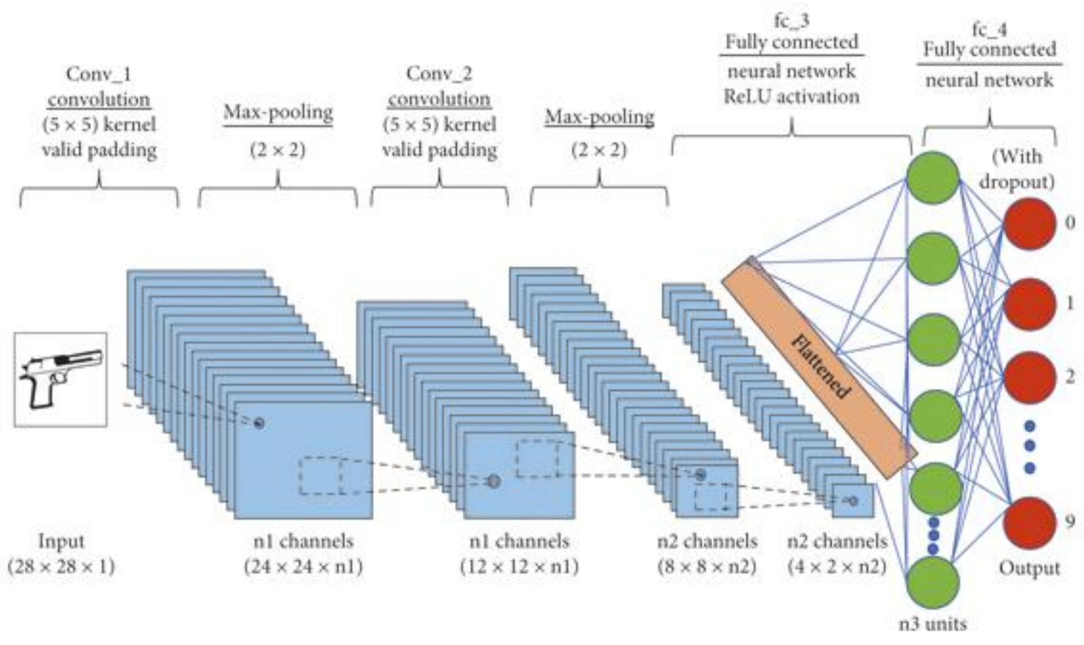


Fig. 3.12: CNN Architecture layout

Object detection is primarily related to computer vision that includes distinguishing objects in computerized images. Object detection is a domain that has benefited immensely from the recent advancements in the realm of deep learning. YOLO is basically a pretrained object detector. It is a CNN model. A CNN is a deep learning algorithm which can take in a raw input image and assign learnable weights and biases to various aspects/objects in the image. A convolutional layer in CNN model is responsible of extracting the high-level features such as edges, from the input image. This works by applying k x k filter known as kernel repeatedly over raw image. This further results in activation maps or feature maps. These feature maps are the presence of detected features from the given input. Thus, the pre-processing required is much lower as compared to other classification algorithms, whereas in standard approach, filters are hand-engineered and in CNN these are learned through a number of iterations and training. Figure 3.12 indicates a basic CNN architecture as classification model for 10 different weapons. Subsequently, the next layer is Max-Pooling or Subsampling layer, which is responsible for reducing the spatial size of the convolved features. This is to decrease the computational power required to process the data through dimensionality reduction. ReLU is a rectified linear unit activation expressed in which is related to the feature of no saturating activation. It eliminates undesirable values from an activation map effectively by setting them to nil. Finally, the last layers are fully connected layers transforming the data into a one-dimensional array. To create a particular long feature vector, the flattened output is fed to a feedforward neural network and backpropagation is applied to every iteration of training. These layers are liable to learn nonlinear combinations of the high-level features as represented by the output of the convolutional layer.

As mentioned earlier that YOLO is a pretrained object detector, a pretrained model simply means that another dataset has been trained on it. It is extremely time consuming to train a model from scratch; it can take weeks or a month to complete the training step. A pretrained model has already seen tons of objects and knows how each of them must be classified. The weights in the abovementioned pretrained model have been obtained by training the network on COCO and ImageNet dataset. Thus, it can only detect objects belonging to the classes present in the dataset used to train the network. It uses Darknet-53 as the backbone network for feature extraction and uses three scale predictions. The DarkNet-53 is again convolutional neural network that has 53 layers as elucidated in Figure 3.12. DarkNet-53 is a fully convolutional neural network. Pooling layer is replaced with a convolution operation with stride 2. Furthermore, residual units are applied to avoid the gradient dispersion.

Initially, CNN architectures were quite linear. Recently, numerous variations are introduced, for example, middle blocks, skip connections, and aggregations of data between layers.

These network models have already acquired rich feature representations by getting trained over a wide range of images. Thus, selecting a pretrained network and using it as a starting point to learn a new task is a concept behind transfer learning. In order to recognize the weapons, we took the weights of a pretrained model and trained another YOLO V3 model.

YOLO V3 is designed to be a multi scaled detector rather than image classifier. Therefore, for object detection, classification head is replaced by appending a detection head to this architecture. Henceforth, the output is vector with the bounding box coordinates and probability classes.

YOLO V3 inherits Darknet-53 as its backbone, a framework to train neural networks with 53 layers as indicated in Figure 3.12. Moreover, for object detection task additional 53 layers are stacked over it, accumulating to a total of a 106-layer fully convolutional architecture. Due to its multiscale feature fusion layers, YOLO V3 uses 3 feature maps of different scales for target detection.

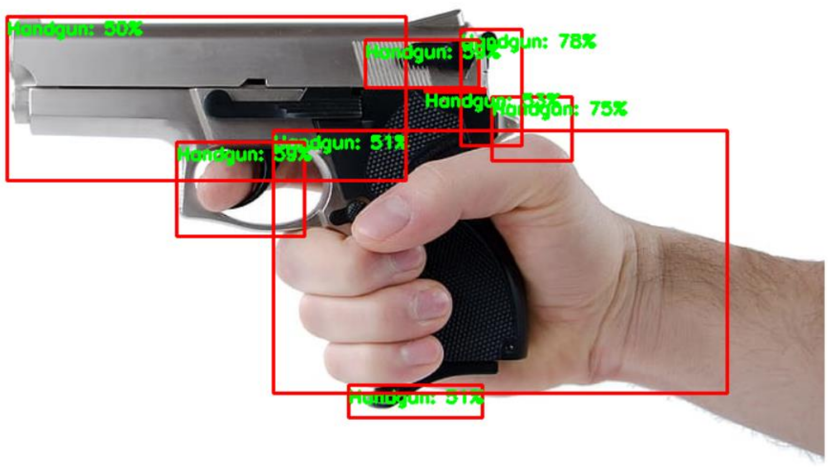


Fig. 3.13: A system that can identify a Weapon within a given image or frame.

In most projects related to weapon classification, I was only able to find a dataset of 100–200 images maximum. This posed an issue because, from my experience, it is hard to get a working model with so little images. To gather images, I rigged my raspberry pi to scrape IMFDB.com- a website where gun enthusiasts post pictures where a model gun is featured in a frame or clip from a movie. If you visit the website, this will be clearer. To access the images that I used, you can visit my Google Drive. In this zip file, you will find all the images that were used in this project and the corresponding .xml files for the bounding boxes. If you are in need of bounding boxes for a large dataset, I highly recommend ScaleOps.AI, a company that specializes in data labelling for machine learning algorithms. Now, let's get to the logic. The architecture of this project follows the logic shown on this website. Although we implement the logic here, there are many areas for which it is different so that it can be useful for our specific problem detecting weapons. The project uses 6 basic steps:

1. Build a dataset using OpenCV Selective search segmentation
2. Build a CNN for detecting the objects you wish to classify (in our case this will be 0 = No Weapon, 1 = Handgun, and 2 = Rifle)
3. Train the model on the images built from the selective search segmentation

4. When creating a bounding box for a new image, run the image through the selective search segmentation, then grab every piece of the picture.
5. Run each piece of an image through the algorithm, and whenever the algorithm predicts the object, you are looking for mark the locations with a bounding box
6. If multiple bounding boxes are marked, apply non-Maxima suppression to include only the box with the high confidence/region of interest (this part I am still figuring out… you will see my issue below)

   Below is a gif showing how the algorithm works. For a given image, each square will be fed into the neural network. If a square is predicted as positive (handgun or rifle), we will mark the area that we fed onto the original image.

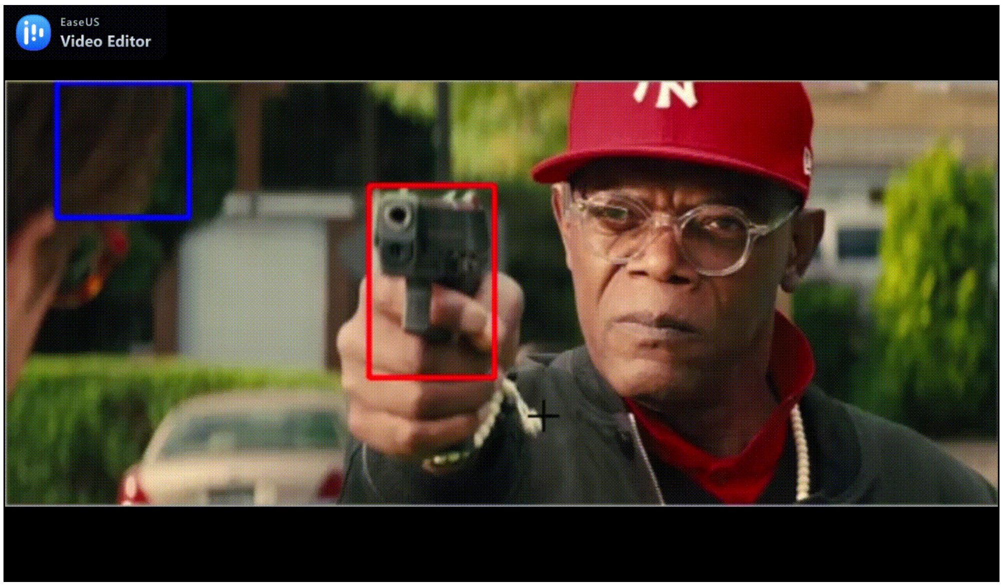


Fig. 3.14: Sliding Window Approach: Object Detection

The data I linked above contains a lot of folders that I need to explain in order to understand what’s going on. After unzipping the folder, these are the files & folders that are important for the project: AR, Final

Images, Labels, Pistol, Stock_AR, and Stock, Pistol, and PATHS.csv. Inside the folders, you will find the corresponding images pertaining to the folder name. So, for the AR folder, you will find images of Assault rifles inside. Inside the Labels folder, you will see the .xml labels for all the images inside the class folders. Lastly, the PATHS.csv will point to every single image that will be used in the algorithm. For the purpose of this tutorial these are the only folders/files you need to worry about:

1. Final Images / No Weapon
2. Final Images / Pistol
3. Final Images / Rifle

The way the images within these folders were made is the following.

- For every image with a bounding box, extract the bounding box and put it into its corresponding class folder. So, for an image where a person is holding a pistol, the bounding box around the pistol will become positive, while every part outside the bounding box will become the negative (no weapon)
- In the image below, imagine a bounding box around the image on the left. After extracting the pixels inside the bounding box (image on the right), we place that image to another folder (Final Images/Pistol), while we place all the white space around the bounding box in the No Weapons folder.
- Although the image on the right looks like a resized version of the one on the left, it is really a segmented image. Picture a bounding box around the gun on the left. The image on the right is *just* the bounding box and nothing else (removing everything outside the box). This technique is called region of interest (ROI).

Fig. 3.15: ROI Extraction

After gathering the dataset (which can be found inside Separated/Final Images), we need to use these files for our algorithm, we need to prepare it in such a way where we have a list of RGB values and the corresponding label (0= No Weapon, 1 = Pistol, 2 = Rifle).

CHAPTER 4
SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

4.1 UML DIAGRAMS:

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts. The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

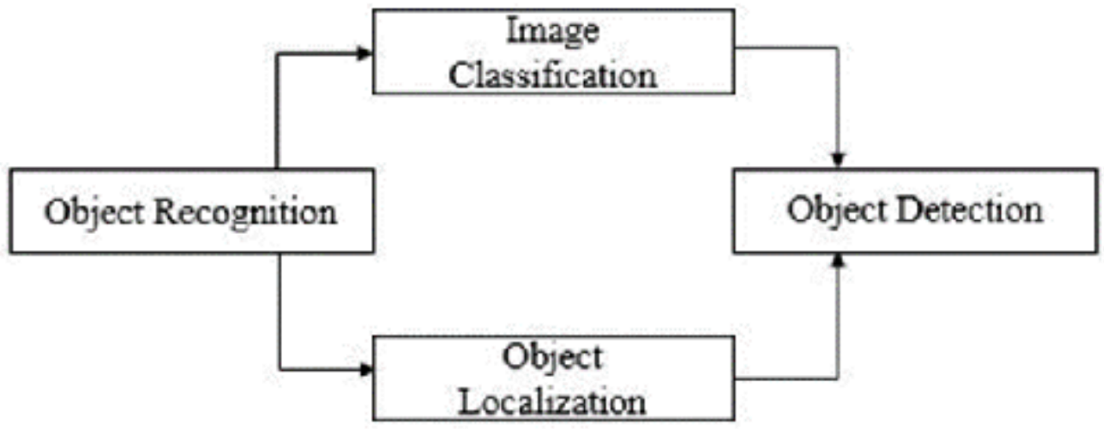


Fig. 4.1: UML Diagram For Arm Detection

UML specification does not preclude mixing of different kinds of diagrams, e.g.: to combine structural and behavioural elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram. UML specification defines two major kinds of UML diagram: structure diagrams and behaviour diagrams. Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. Behaviour diagrams show the dynamic behaviour of the objects in a system, which can be described as a series of changes to the system over time.

4.2 DATA FLOW DIAGRAM:

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analyzing existing as well as proposed system.

It provides an overview of

- What data is system processes.
- What transformation are performed.
- What data are stored.
- What results are produced, etc.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

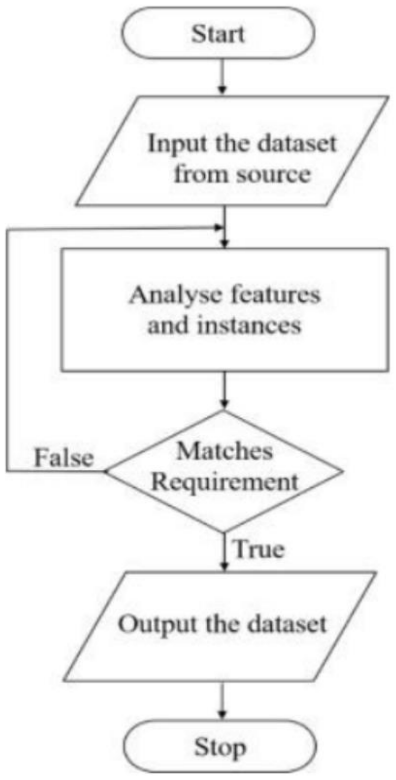


Fig. 4.2: Flowchart For Arm Detection

4.3 USE CASE DIAGRAM:

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent: Scenarios in which your system or application interacts with people, organizations, or external systems. Goals that your system or application helps those entities (known as actors) achieve.

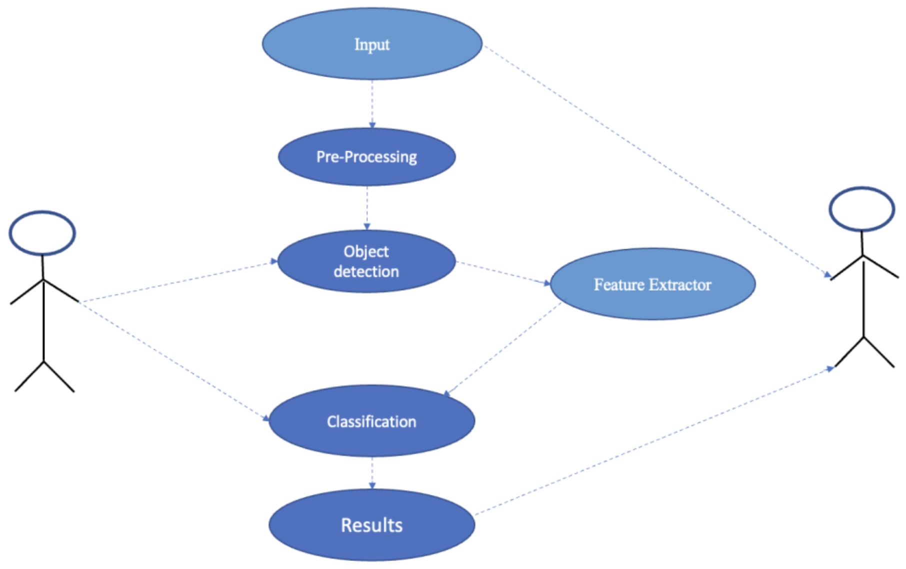


Fig. 4.3: Use Case Diagram For Arm Detection

4.4 SEQUENCE DIAGRAM:

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

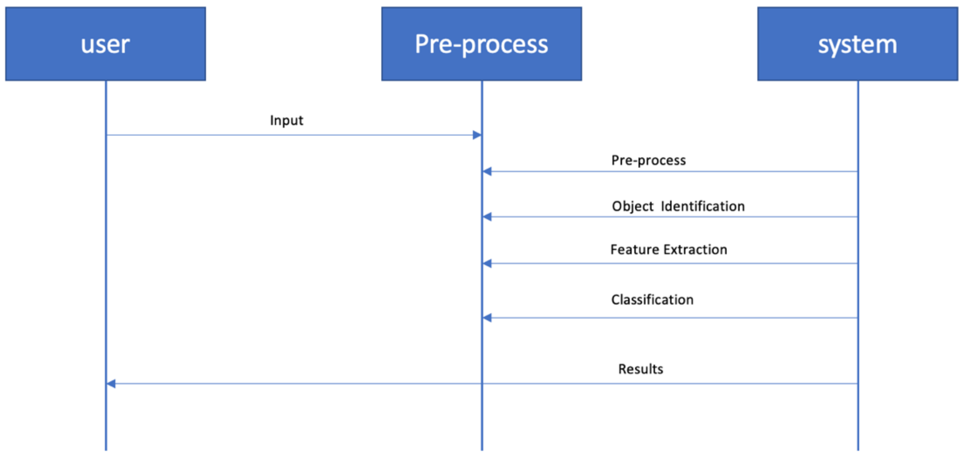


Fig. 4.4: Sequence Diagram For Arm Detection

4.5 IMPLEMENTATION:

4.5.1 Data Collection and Pre-Processing:

Source of data: Security cameras, surveillance systems, or other sources of feeds.

Format of data: image frames in a standard format such as JPEG or PNG.

Labelling: Each frame should be labelled as containing a weapon or not.

4.5.2 Object Detection and Tracking:

Pre-trained models: Popular object detection models can be used for detecting and localising objects in each frame.

Training data: The pre-processed video or image frames with labels Indicating the presence or absence of weapons can be used to train the object detection model.

Tracking data: Once a weapon is detected, its trajectory needs to be tracked across frames to determine if it is being carried or used.

4.5.3 Feature Extraction:

Pre-trained models: pre-trained deep learning models such as ResNet or VGG can be used to extract features from the detected object.

Training data: The pre-processed video or image frames with labels indicating the presence or absence of weapons can be used to train the feature extraction model.

4.5.4 Classification:

Pre-trained models: pre-trained deep learning models such as a Convolutional Neural Network (CNN) can be used for classifying the extracted features as weapons or non-weapons.

Training data: The pre-processed video or image frames with labels indicating the presence or absence of weapons can be used to train the classification model.

4.6 ALGORITHMS:

4.6.1 CONVOLUTIONAL NEURAL NETWORK (CNN):

CNN is a learning method inspired by biological processes. It is believed that the main computation element of living creatures is neuron. The connected network of neurons forms the basis of all the decisions made based on the information gathered. CNN is a feed-forward network with three key mechanisms: local receptive fields, weight sharing, and sub-sampling. The network is trained like a normal neural network using back-propagation algorithm. The common architecture of a CNN is structured by four main layers. Convolutional layer has main function for applying a filter mask. The filter is applied on a specified area of the image. The area that has been filtered is called the receptive field. The operation of this layer involves computing the weighted summation of the values covered by the receptive field using the weights of the filter mask. Then, the weighted summation is added with a bias and passes through an activation function. Once this value is computed, receptive field moves on the feature map by a number of strides to cover new area for computation of next value using the same filter mask. The purpose of filter mask is to learn basic pattern of object in the image. The weights of filter mask are shared across a feature map in the model.

4.6.2 DECISION TREE CLASSIFIERS:

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision-making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labelled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi, we build a subsidiary decision tree by invoking the same procedure recursively on the set Si**.**

4.6.2.1 Gradient boosting:

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

4.6.3 K-NEAREST NEIGHBORS (KNN):

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not "learn" until the test example is given
- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Examples:

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset.

4.6.4 LOGISTIC REGRESSION CLASSIFIERS:

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analysing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modelling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cut-off point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

.

4.6.5 NAIVE BAYES:

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature.

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they do not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

4.6.6 RANDOM FOREST:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "Blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

4.6.7 SVM:

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* (*iid*) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point *x* and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms* (*GAs*) or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

4.7 CONCLUSION:

- The goal of this project was to create an algorithm that can integrate itself into traditional surveillance systems and prevent a bad situation faster than a person would (considering the unfortunate circumstances in today's society).
- Although this was cool, the hardware in my computer is not yet there. To segment an image and process each portion of the image takes about 10–45 seconds, which is too slow for live video.
- The video demonstration I showed above was a 30-second clip, and that took about 20 minutes to process.
- However, although live video is not feasible with an RX 580, using the new Nvidia GPU (3000 series) might have better results.
- Also, this technique can be used for retroactive examination of an event such as body cam footage or protests.

CHAPTER 5
SYSTEM TESTING

CHAPTER 5
SYSTEM TESTING

5.1 TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

5.2 UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural-testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.

- The entry screen, messages and responses must not be delayed.

Features to be tested:

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

5.3 INTEGRATION TESTING:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of

high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.
The following are the types of Integration Testing:

1. **Top-Down Integration**

   This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

   In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. **Bottom-up Integration**

   This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom - up integration strategy may be implemented with the following steps:

   a. The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
   b. A driver (i.e.) the control program for testing is written to coordinate test case input and output.
   c. The cluster is tested.
   d. Drivers are removed and clusters are combined moving upward in the program structure.
   e. The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

5.4 FUNCTIONAL TESTING:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.5 SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.6 WHITE BOX TESTING:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.7 BLACK BOX TESTING:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

5.8 ACCEPTANCE TESTING:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.9 OTHER TESTING METHODOLOGIES:

5.9.1 USER ACCEPTANCE TESTING:

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

5.9.2 OUTPUT TESTING:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

5.9.3 VALIDATION CHECKING:

Validation checks are performed on the following fields.

- **Text Field:**

  The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

- **Numeric Field:**

  The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

  Successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

5.10 PREPARATION OF TEST DATA:

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

5.10.1 USING LIVE TEST DATA:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

5.10.2 USING ARTIFICIAL TEST DATA:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

5.11 USER TRAINING:

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

5.12 MAINTAINENCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

5.13 TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

5.13.1 SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g., Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

5.13.2 UNIT TESTING:

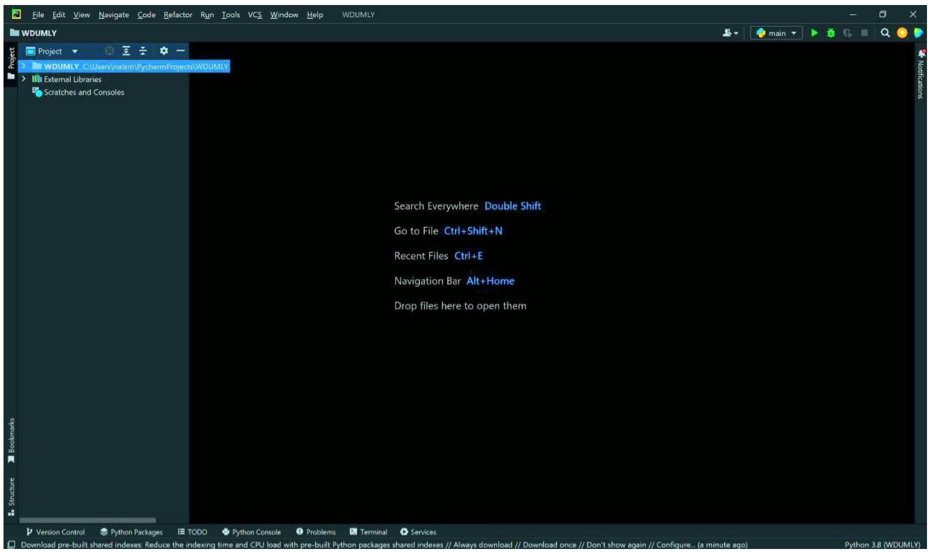
In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

CHAPTER 6
RESULTS

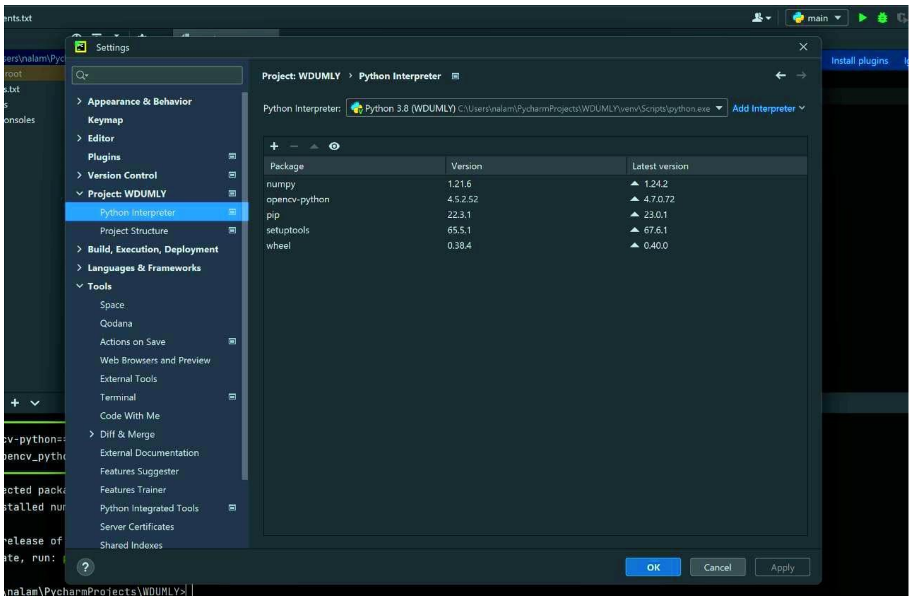
CHAPTER 6
RESULTS

6.1 SCREENSHOTS:


File Edit View Navigate Code Refactor Run Tools VCS Window Help WDUMLY
WDUMLY
Project
WDUMLY C:\Users\nalam\PycharmProjects\WDUMLY
External Libraries
Scratches and Consoles
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
Version Control Python Packages TODO Python Console Problems Terminal Services
Python 3.8 (WDUMLY)

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago)

Fig. 6.1: Screenshot of project folder in PyCharm



Project: WDUMLY › Python Interpreter

Python Interpreter: Python 3.8 (WDUMLY) C:\Users\nalam\PycharmProjects\WDUMLY\venv\Scripts\python.exe

Package	Version	Latest version
numpy	1.21.6	▲ 1.24.2
opencv-python	4.5.2.52	▲ 4.7.0.72
pip	22.3.1	▲ 23.0.1
setuptools	65.5.1	▲ 67.6.1
wheel	0.38.4	▲ 0.40.0

Fig. 6.2: Screenshot including the details of the modules used.

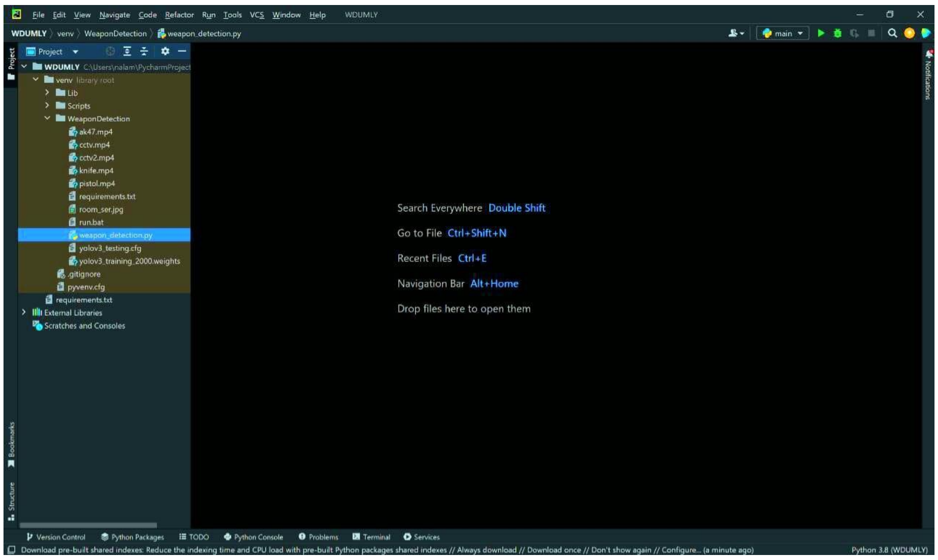
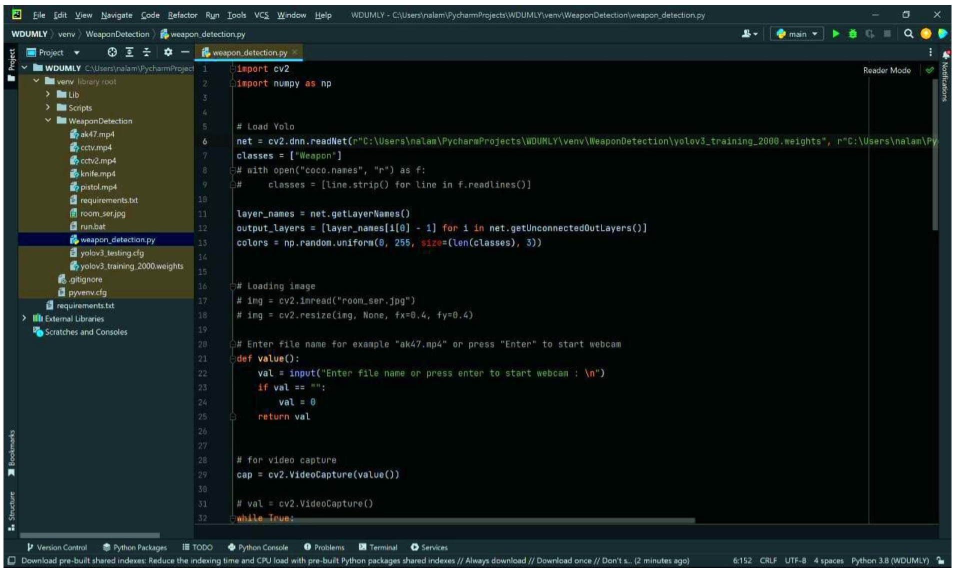
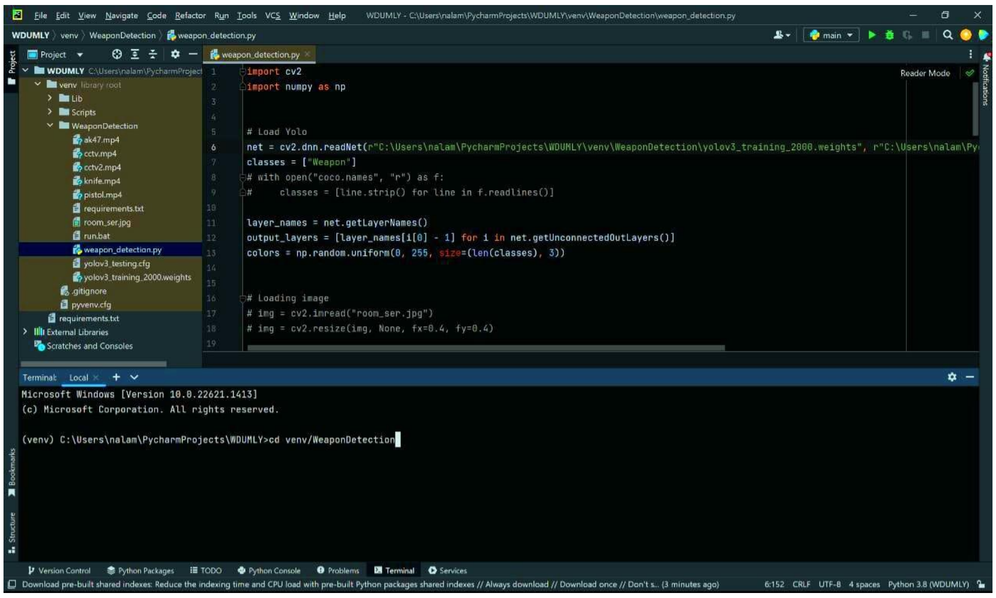


Fig. 6.3: Screenshot of multiple files included in the project file.



```python
import cv2
import numpy as np


# Load Yolo
net = cv2.dnn.readNet(r"C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection\yolov3_training_2000.weights", r"C:\Users\nalam\Py
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))


# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)

# Enter file name for example "ak47.mp4" or press "Enter" to start webcam
def value():
    val = input("Enter file name or press enter to start webcam : \n")
    if val == "":
        val = 0
    return val


# for video capture
cap = cv2.VideoCapture(value())

# val = cv2.VideoCapture()
while True:
```

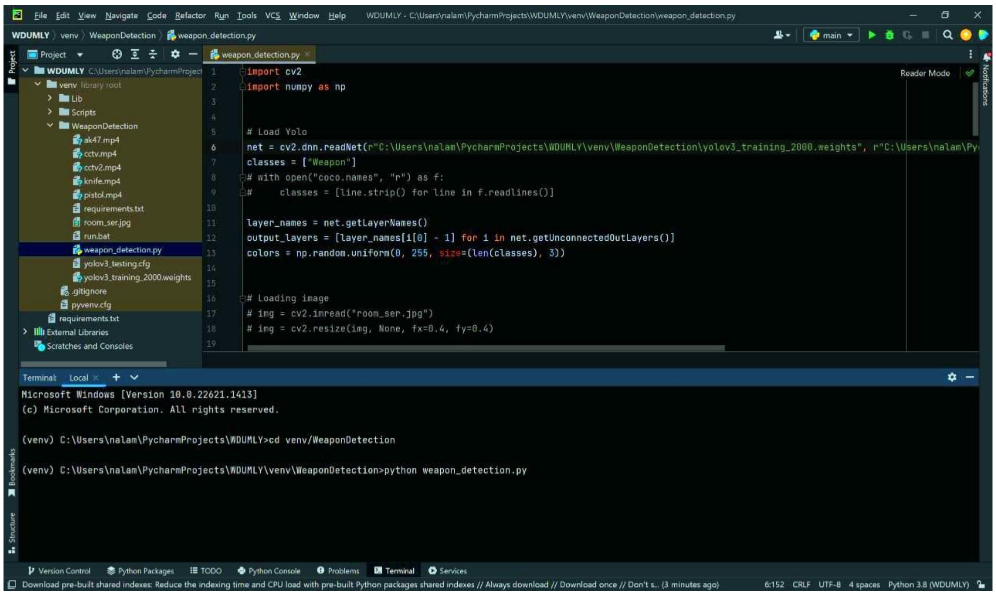
Fig. 6.4: Screenshot regarding code of the project.



```python
import cv2
import numpy as np


# Load Yolo
net = cv2.dnn.readNet(r"C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection\yolov3_training_2000.weights", r"C:\Users\nalam\Py
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))


# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)
```

```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\nalam\PycharmProjects\WDUMLY>cd venv/WeaponDetection
```

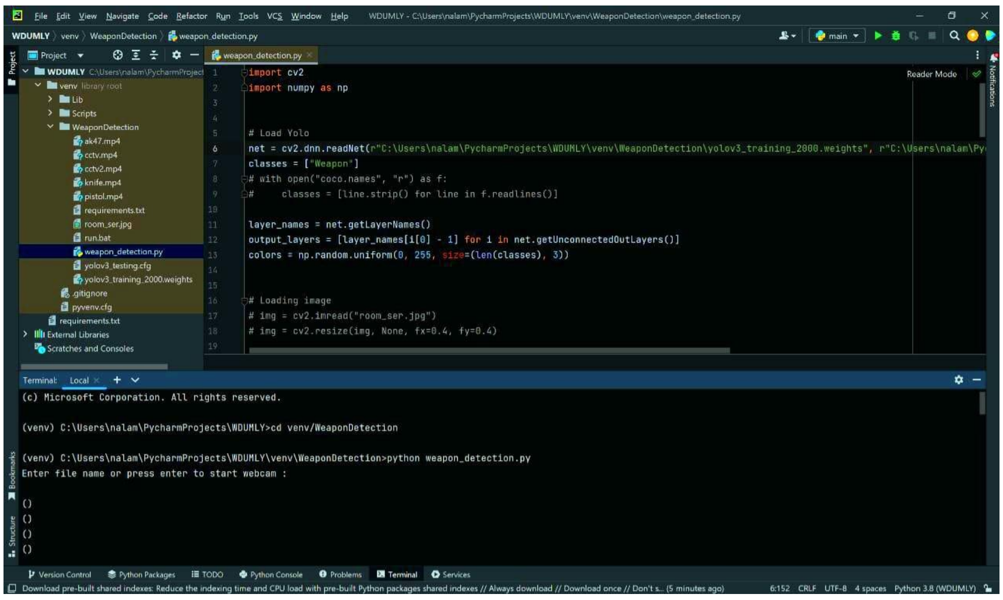
Fig. 6.5: Screenshot of terminal and path setting.



```python
import cv2
import numpy as np


# Load Yolo
net = cv2.dnn.readNet(r"C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection\yolov3_training_2000.weights", r"C:\Users\nalam\Py
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))


# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)
```

```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\nalam\PycharmProjects\WDUMLY>cd venv/WeaponDetection

(venv) C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection>python weapon_detection.py
```

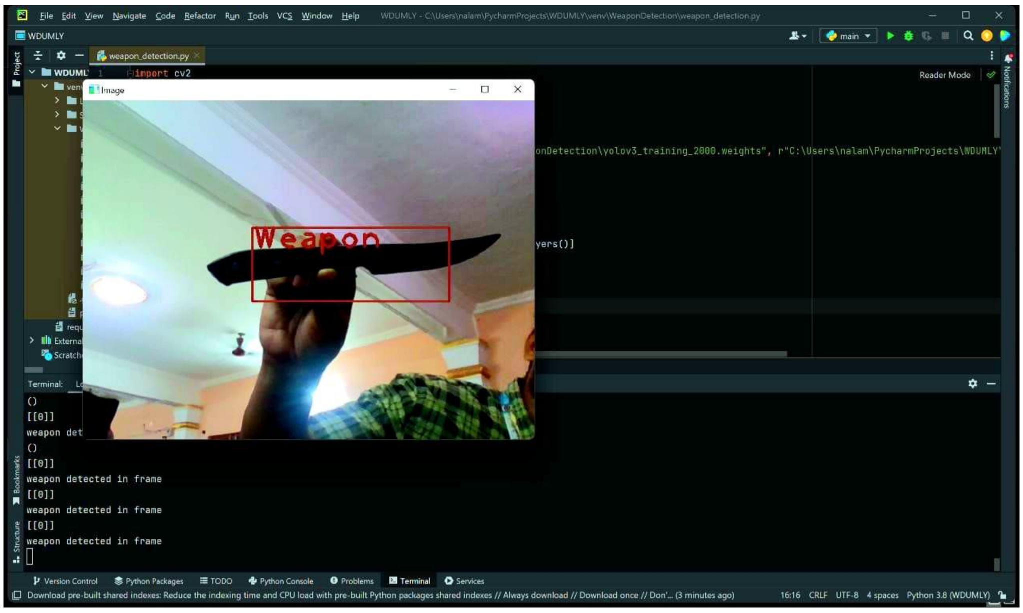
Fig. 6.6: Screenshot regarding the commands for the execution of the code.



```python
import cv2
import numpy as np


# Load Yolo
net = cv2.dnn.readNet(r"C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection\yolov3_training_2000.weights", r"C:\Users\nalam\Py
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))


# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)
```

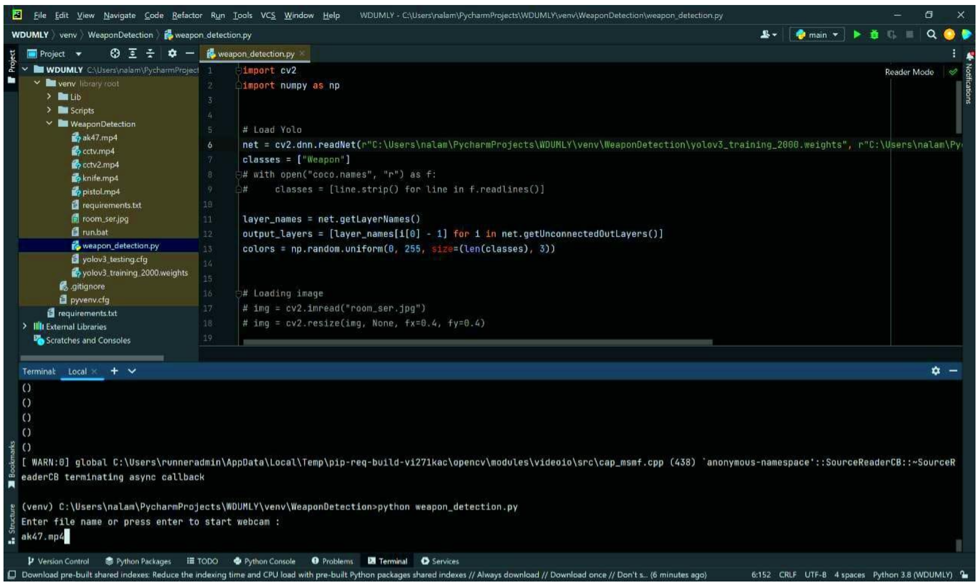
```
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\nalam\PycharmProjects\WDUMLY>cd venv/WeaponDetection

(venv) C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection>python weapon_detection.py
Enter file name or press enter to start webcam :

()
()
()
()
```

Fig. 6.7: Screenshot during the execution of code accessing the device camera.



```
()
[[0]]
weapon det
()
[[0]]
weapon detected in frame
[[0]]
weapon detected in frame
[[0]]
weapon detected in frame
```

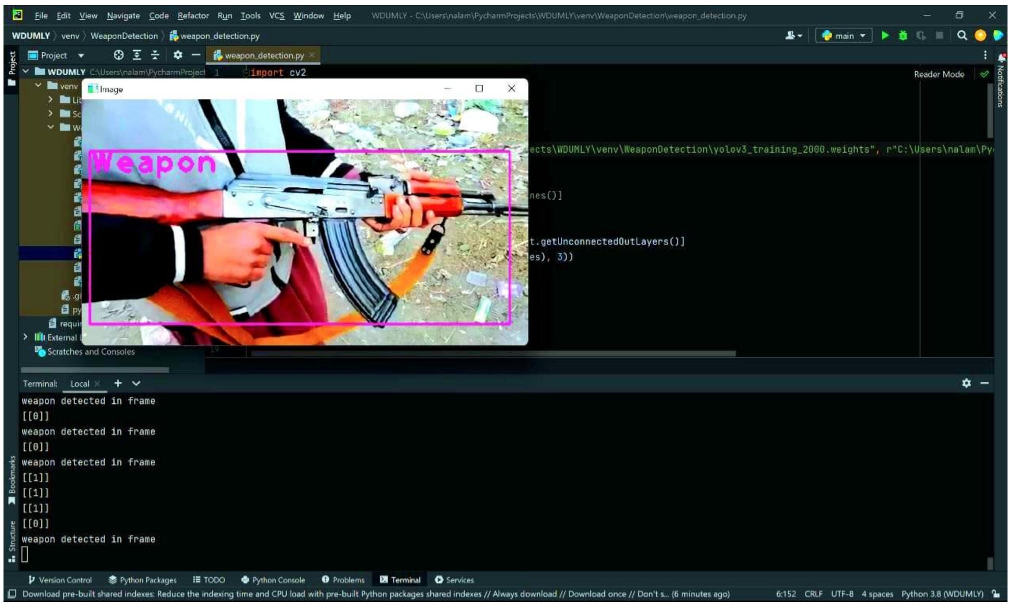
Fig. 6.8: Screenshot of recognition of weapon.



```
import cv2
import numpy as np


# Load Yolo
net = cv2.dnn.readNet(r"C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection\yolov3_training_2000.weights", r"C:\Users\nalam\Py
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))


# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)
```

```
()
()
()
()
()
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-vi271kac\opencv\modules\videoio\src\cap_msmf.cpp (438) `anonymous-namespace'::SourceReaderCB::~SourceR
eaderCB terminating async callback

(venv) C:\Users\nalam\PycharmProjects\WDUMLY\venv\WeaponDetection>python weapon_detection.py
Enter file name or press enter to start webcam :
ak47.mp4
```

Fig. 6.9: Screenshot during the execution of code accessing recorded video file (mp4).



```
weapon detected in frame
[[0]]
weapon detected in frame
[[0]]
weapon detected in frame
[[1]]
[[1]]
[[1]]
[[0]]
weapon detected in frame
```

Fig. 6.10: Screenshot of recognition of weapon in a video file (mp4).

CHAPTER 7 CONCLUSION AND FUTURE WORK

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSION:

SSD and quicker RCNN algorithms are simulated for pre labelled and self-created image dataset for weapon (gun) detection. each the algorithms are economical and provides sensible results however their application in real time is predicated on a exchange between speed and accuracy. In terms of speed, SSD algorithmic program provides higher speed with zero.736 s/frame. Whereas quicker RCNN provides speed one.606s/frame, that is poor compared to SSD. With relevancy accuracy, quicker RCNN provides higher accuracy of eighty-four.6%. Whereas SSD provides AN accuracy of seventy-three. 8%, that is poor compared to quicker RCNN.SSD provided real time detection thanks to quicker speed however quicker RCNN provided superior accuracy.

7.2 FUTURE WORK:

Future research in the field of deep learning-based weapon detection is an exciting area with many promising directions. By experimenting with new architectures, optimization strategies, and data augmentation techniques, researchers can concentrate on enhancing the accuracy of current deep learning models. Real-time weapon detection is another important area of research that can improve public safety. Furthermore, combining various modalities, including audio, thermal imaging, and images, can increase the overall accuracy of weapon detection. Another technique that can be investigated to improve the accuracy of weapon detection models is transfer learning. Detecting hidden weapons and developing models that are resistant to adversarial attacks are also important areas of future research in deep learning weapon detection.

REFERENCES

[1] (2019). Christchurch Mosque Shootzings. Accessed: Jul. 10, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Christchurch_mosque_shootings

[2] (2019). Global Study on Homicide. Accessed: Jul. 10, 2019. [Online]. Available: https://www.unodc.org/unodc/en/data-and-analysis/globalstudy- on-homicide.html

[3] W. Deisman, ``CCTV: Literature review and bibliography," in Research and Evaluation Branch, Community, Contract and Aboriginal Policing Services Directorate. Ottawa, ON, Canada: Royal Canadian Mounted, 2003.

[4] J. Ratcliffe, ``Video surveillance of public places," US Dept. Justice, Office Community Oriented Policing Services, Washington, DC, USA, Tech. Rep. 4, 2006.

[5] M. Grega, A. Matiolaski, P. Guzik, and M. Leszczuk, ``Automated detection of _rearms and knives in a CCTV image," Sensors, vol. 16, no. 1, p. 47, Jan. 2016.

[6] TechCrunch. (2019). China's CCTV Surveillance Network Took Just 7 Minutes to Capture BBC Reporter. Accessed: Jul. 15, 2019. [Online]. Available: https://techcrunch.com/2017/12/13/china-cctv-bbc-reporter/

[7] N. Cohen, J. Gattuso, and K. MacLennan-Brown. CCTV Operational Requirements Manual 2009. St Albans, U.K.: Home Office Scientific Development Branch, 2009.

[8] G. Flitton, T. P. Breckon, and N. Megherbi, ``A comparison of 3D interest point descriptors with application to airport baggage object detection in complex CT imagery," Pattern Recognit., vol. 46, no. 9, pp. 2420_2436, Sep. 2013.

[9] R. Gesick, C. Saritac, and C.-C. Hung, ``Automatic image analysis process for the detection of concealed weapons," in Proc. 5th Annu. Workshop Cyber Secur. Inf. Intell. Res. Cyber Secur. Inf. Intell. Challenges Strategies (CSIIRW), 2009, p. 20.