MUHAMMAD TEHAAM
01-134172-071
AHMED GHANI
01-134172-006

Criminal Threat Observing System (CTOS)

Bachelor of Science in Computer Science

Supervisor: Ma'am Maryam Bibi

Department of Computer Science
Bahria University, Islamabad

April 2021

Certificate

We accept the work contained in the report titled "Criminal Threat Observing System (CTOS)", written by Mr. Muhammad Tehaam AND Mr. Ahmed Ghani as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . .:

Supervisor: Ms. Maryam Bibi ()

_____

Internal Examiner: Name of the Internal Examiner ()

_____

External Examiner: Name

_____

Project Coordinator: Ms. Zubaria Inayat ()

_____

Head of the Department: Dr. Muhammad Muzammal ()

_____

April 30th, 2021

Abstract

Security in general has been improving with time, but time and again we have seen that our systems have failed. The recent motorway case was a hit for our entire nation where a helpless woman was harmed unimaginably, and the perpetrators are still on the loose and no one even has any legitimate evidence to find the culprits. This is one of the many cases that unfortunately have been happening on a daily basis and not only that but have been increasing in numbers. The project proposed, is designed to fill those holes in our currently used systems, that the criminals have been happily getting by to fulfil their aims.

Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful. We would like to express our deepest thank and gratitude towards Ma'am Maryam Bibi who gave us the chance to work on this project, her supervision and her guidance proved very successful in making this project successful so far. She provided us with the reference material and many other project related materials that not only helped us making the project without any considerable trouble but also implementing it till so far.

MUHAMMAD TEHAAM, AHMED GHANI
Bahria University Islamabad, Pakistan

April 2021

Contents

Abstract iii

1 Introduction 1
- 1.1 Project Background/Overview . . . 1
- 1.2 Problem Description . . . 2
- 1.3 Project Objectives . . . 2
- 1.4 Project Scope . . . 2

2 Literature Review 3
- 2.1 Image Processing based Gun Detection . . . 3
- 2.2 Deep Learning based Gun Detection . . . 5
  - 2.2.1 Convolutional Neural Network . . . 5
  - 2.2.2 VGG16 . . . 6
  - 2.2.3 Single Shot Detector . . . 7
  - 2.2.4 YOLO . . . 7
  - 2.2.5 Haar Cascade Classifier . . . 8
  - 2.2.6 Transfer Learning . . . 8
- 2.3 Machine Learning vs Deep Learning . . . 9
- 2.4 Common Practices . . . 10
- 2.5 Comparison of the Discussed Techniques . . . 11

3 Requirement Specification 15
- 3.1 Proposed System . . . 15
- 3.2 Methodology . . . 16
- 3.3 Non-Functional Requirements . . . 17
- 3.4 Use Cases . . . 18
  - 3.4.1 Use Case 1: Start Desktop Application . . . 18
  - 3.4.2 Use Case 2: Select Images/Videos from Inventory . . . 19
  - 3.4.3 Use Case 3: Feed the Image/Video of Interest to the System . . . 19
  - 3.4.4 Use Case 4: Get the Results . . . 20

4 Design 21
- 4.1 System Architecture . . . 21
  - 4.1.1 System Architecture Diagram: . . . 22
  - 4.1.2 Conceptual Diagram: . . . 22
- 4.2 Design Constraints . . . 23
  - 4.2.1 Application Constraints . . . 23
  - 4.2.2 Software Requirements . . . 23

4.3 Design Methodology . . . 23
4.3.1 Process-wise Analysis . . . 24
4.4 High Level Design . . . 26
4.4.1 Component Diagram . . . 26
4.4.2 System Interaction Diagram . . . 26
4.4.3 Deployment Diagram . . . 27
4.4.4 Modules . . . 28
4.4.5 Security . . . 28
4.5 Low Level Design . . . 28
4.5.1 Data Acquisition . . . 29
4.5.2 Image Processing . . . 29
4.5.3 Train and Classification . . . 29
4.6 GUI Design . . . 30

5 System Implementation **31**
5.1 System Summary . . . 31
5.2 System Architecture . . . 31
5.2.1 Algorithmic Workflow . . . 31
5.3 System Components . . . 32
5.3.1 Convert Video to Frames . . . 33
5.3.2 Pre-processing Frames or Image . . . 33
5.3.3 Displaying the Images or Videos . . . 33
5.3.4 Passing Frames/Images to a Gun Detector Function for Parallel Check . . . 34
5.3.5 Comparing the Results of both Model and the Function . . . 34
5.3.6 Converting the Frames Back to Video (if video) . . . 34
5.3.7 Displaying the Images or Video . . . 35
5.4 Tools and Techniques . . . 35

6 System Testing and Evaluation **37**
6.1 Graphical User Interface Testing . . . 37
6.1.1 Test Case . . . 38
6.2 Usability Testing . . . 39
6.3 Software Performance Testing . . . 39
6.4 Exception Handling . . . 40
6.5 Load Testing . . . 40
6.6 Test Cases . . . 40
6.6.1 Software Startup Test Case . . . 40
6.6.2 Video/Image Load Test Case . . . 41
6.6.3 Criminal Detection Test Case . . . 41
6.6.4 Second Criminal Detection Test Case . . . 42
6.7 Datasets . . . 42
6.7.1 Custom Dataset . . . 42
6.8 Detail Analysis and Results . . . 43
6.8.1 Image Dataset . . . 44
6.8.2 Video Dataset . . . 47
6.9 Comparison . . . 49

7 Conclusion **51**
7.1 Future Work 51
7.2 Recommendation 52
7.3 Learning Outcomes 52

A User Manual **53**
A.1 Start Program 53
A.2 Pick a Video 54
A.3 Processing 55
A.4 Results 55

B Bibliography **57**

List of Figures

2.1 Harris plus FREAK descriptor of gun [10] . . . 4
2.2 (a) Referenced image; (b) Segmented image; (c) Extracted gun color part from referenced image [10] . . . 4
2.3 Categorization of the DL methods and their representative works [12] . . . 9
2.4 Block diagram of the proposed approach for classifying image pistol/gun or not [12] 10
2.5 The training process of the proposed system with negative and positive dataset images [12] . . . 10

3.1 Images from Gun Dataset . . . 16
3.2 Images from Not Gun Database . . . 16
3.3 Methodology Diagram . . . 17
3.4 Use Case Diagram . . . 18
3.5 Use Case 1 Diagram . . . 18
3.6 Use Case 2 Diagram . . . 19
3.7 Use Case 3 Diagram . . . 19
3.8 Use Case 4 Diagram . . . 20

4.1 System Architecture . . . 22
4.2 Package Diagram . . . 22
4.3 Methodology Flowchart . . . 24
4.4 Image/Frame Acquisition Flow Diagram . . . 25
4.5 Image Pre-Processing Flow Diagram . . . 25
4.6 Image/Frame Feature Extraction Flow Diagram . . . 25
4.7 Classification Process Flow Diagram . . . 26
4.8 Component Diagram . . . 26
4.9 System Interaction Diagram . . . 27
4.10 Deployment Diagram . . . 28
4.11 Class Diagram for Numpy Module . . . 29
4.12 Class Diagram for OpenCV Module . . . 29
4.13 Class Diagram for Sklearn Module . . . 30
4.14 Graphical User Interface (GUI) . . . 30

5.1 System Activity Diagram . . . 32
5.2 Video Frames . . . 33
5.3 Image Resizing . . . 33

6.1 Main Screen Layout . . . 37
6.2 Browse Window for Videos/Images . . . 38
6.3 Video Processing . . . 38

6.4 Results . . . 39
6.5 (a) Original Image, (b) Software Result . . . 41
6.6 (a) Original Image, (b) Software Result . . . 42
6.7 Gun Images . . . 43
6.8 Not Gun Images . . . 43
6.9 Videos . . . 43
6.10 Model accuracy and loss graphs for Alex-Net . . . 44
6.11 Model accuracy and loss graphs for CNN-6 Layers . . . 45
6.12 Model accuracy and loss graphs for CNN-4 Layers . . . 46
6.13 Model accuracy and loss graphs for CNN-6 Layers with tanh . . . 47
6.14 Model accuracy and loss graphs for Alex-Net . . . 48
6.15 Model accuracy and loss graphs for CNN-6 Layers . . . 48
6.16 Model accuracy and loss graphs for CNN-4 Layers . . . 49

A.1 Main Screen Layout . . . 53
A.2 Browse Window for Images . . . 54
A.3 Browse Window for Videos . . . 54
A.4 Video Processing . . . 55
A.5 Results for Videos . . . 56
A.6 Results for Images . . . 56

List of Tables

2.1 Technique Comparison Table 13

3.1 Use Case 1: Start Desktop Application Table 18
3.2 Use Case 2: Select Images/Videos from Inventory 19
3.3 Use Case 3: Feed the Image/Video of Interest to the System 19
3.4 . Use Case 4: Get the Results 20

6.1 Software Startup Test Case 40
6.2 Video/Image Load Test Case 41
6.3 Criminal Detected Test Case 41
6.4 Second Criminal Detected Test Case 42
6.5 Conventional Alex-Net Architecture with activation function Relu 44
6.6 Three Convolutional Layers and three Max-pooling Layers with activation function Relu 45
6.7 Two Convolutional Layers and two Max-pooling Layers with activation function Relu 46
6.8 Two Convolutional Layers and two Max-pooling Layers with activation function tanh 47
6.9 Conventional Alex-Net Architecture with activation function Relu 47
6.10 Three Convolutional Layers and three Max-pooling Layers with activation function Relu 48
6.11 Two Convolutional Layers and two Max-pooling Layers with activation function Relu 49
6.12 Comparison with Research Papers 49

Acronyms and Abbreviations

CNN	Convolutional Neural Network
R-CNN	Region based Convolutional Neural Network
YOLO	You Look Only Once
HOG	Histogram of Oriented Gradients
SPP-net	Spatial Pyramid Pooling-net
SSD	Single Shot Detector
GUI	Graphical User Interface
VGG	Visual Geometry Group

Chapter 1

Introduction

1.1 Project Background/Overview

Security and peace of mind is what everyone wants and strives for, but we still encounter cases in our day to day lives where this exact security that we work so hard for, can be compromised. Over the years, a lot of work has been done for improvements, and the betterment of our already functioning and deployed security systems but there are always certain loopholes that wrongdoers can get a hold of and use it against us, and by doing so they can harm us. We have seen how despite having the best possible surveillance cameras installed, top of the line sensors and entire organizations working for the sole purpose of providing security, can fail, and this is the reason why, despite of all these achievements and technology, crimes in general are still increasing [1].
In a country like Pakistan, we have cities such as Karachi that are known for their crimes and the level of sophistication that those doing the crimes have and the incompetence and failure of the authorities to not only provide justice but to even capture the culprits [2].
A normal person these days, does not feel safe being on the streets. There is no surety that no one is going to cause any harm. Mobile phones are being snatched on gunpoint by the minute and no one is ready to take the responsibility for it. Places like ATM machines have become a hotspot for such crimes where a normal visit to withdraw your salary has become a topic on which proper discussion and planning is done to avoid any possible harm that can easily take place at a location which otherwise should have been a simple touch and go [3].
This project is based on the purpose of tackling these exact issues by dealing with those loopholes to provide justice to those who have grown old but have only wished for justice to be served. It will have a desktop application which will be able to differentiate between the criminal and the victim by recognizing the actions of the actors involved in the scenario [4].

1.2 Problem Description

One of the major issues in our security systems is the lack of surveillance data regarding criminals which can later be used to capture them by identifying their facial images. This is the reason why in many of the cases, the culprit/criminal is never captured because he/she cannot be recognized. The criminals end up free without any consequences and successful in their dark motives. Our project aims to tackle this issue by recognizing and differentiating the criminal from the victim, by analyzing their actions. This will not only give us an analysis of the video/photo evidence of the criminal action but also will inform the authorities whenever a criminal action is taking place, during the scene, making it easy for the in-area security authorities to take action either at the time or if the criminal does escape.

1.3 Project Objectives

To design an application that can detect a criminal action through videos of surveillance cameras, as the scene is taking place for Criminal Threat Observing System (CTOS), that can be used by security authorities for capturing and detecting criminals.

1.4 Project Scope

This application aims to detect the criminal actions from images and identify the threat and victim. The images/frames should contain sufficient characters that will perform criminal activity and our project should be able to detect these actions. This project is widely applicable as it only uses a computer system and a camera. Initially the project will only take images as input for training as well as testing but later on we aim to use videos as well. It can be easily deployed using surveillance cameras, drone cameras or any other area that has a camera installed. This project will not only reduce the crime rate but will also help the authorities to catch them so that people can live freely without fear.

Limitations: The dataset chosen, is limited to the criminal holding a gun while the victim is unarmed. Any scenario otherwise may not be correctly recognized for now.

Chapter 2

Literature Review

Criminal Detection and primarily object-based weapon detection determining a criminal activity has seen its share of research over the years. Criminal activity and its detection have been a great priority for agencies and organizations trying to reduce and control criminal activities. This has led to an increase in researchers coming up with different ways, algorithms and techniques to detect such activities by enhancing what we now know as computer vision. Since our main focus here is object-based gun detection which ultimately determines whether the actor in the image is a criminal or not and whether the action being performed in the image is a criminal action or not therefore the major focus was on techniques that use object detection in order to determine whether a given object in an image is a gun or not. Over the years there have been several methods and techniques that have been used to detect a gun. These techniques are categorized as follows:

- Conventional Image Processing based Object Detection
- Machine Learning & Deep Learning based Object Detection
  (In combination with the above conventional method)

2.1 Image Processing based Gun Detection

Before we had any advancement in Machine Learning technology, conventional image processing methods and techniques were used in order to detect objects. Such work was done by authors Rohit Kumar et al. [10], in which they initially used preprocessing techniques such as Colour based Segmentation using K-means Clustering Algorithm. Next, they would extract the segment related the color of the gun. After which they would extract the blob. Once they had a rough image of a blob of a gun, they used Morphological Closing and Boundary Extraction to get an outline of the detected object. The authors used a specific technique known as Interest Point Feature Extraction using Haris plus FREAK Combination. This is a combination of two techniques that is used to measure the similarity score with extracted interest point features of the extracted blobs. This

similarity is later used to determine whether the detected blob is a gun or not. The authors used a self-made dataset that consisted of 65 positive images in which gun is present and 24 negative images in which gun is absent. This method achieved a true positive rate of 84.26%.

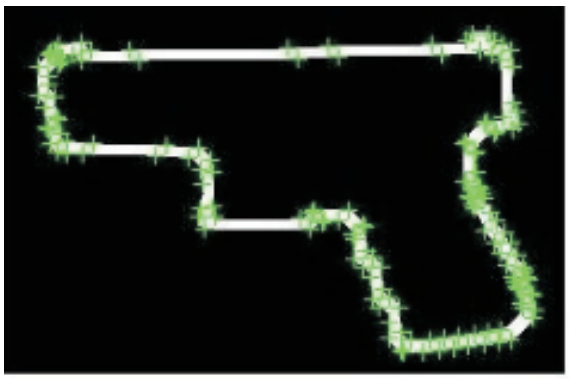

Figure 2.1: Harris plus FREAK descriptor of gun [10]

Figure 2.2: (a) Referenced image; (b) Segmented image; (c) Extracted gun color part from referenced image [10]

There have been several other techniques developed for gun detection and object detection, in general, such as the author Rohit et al [18] has proposed in which different image processing steps have been used to detect a gun in an image. These steps start off with pre-processing of the images, next the images go through Blob Extraction to get different objects from the images. Once they have the blobs, next they apply morphological closing and boundary extraction is implemented in order to get the shapes of the detected blob objects. SURF Feature extraction is a technique that the authors have used in this paper to extract features from the identified different objects and those features are then compared with the standard features of a gun which determines whether the detected object is a gun or not.
The authors used a self-created dataset comprising of 15 positive images and 10 negative images. The dataset consists of images of different types of guns. With this image processing-based technique, the authors were able to achieve a true positive rate of 86.67%.

Authors Bhavna et al [19] used a Concealed Weapon Detection (CWD) technique which uses Infrared sensor-ed images as input and uses several steps to detect any kind of a weapon in it. First the image is passed through denoising to remove noise from the image. Next, the authors have used wavelet technique to enhance the image further after noise has been removed. Since the images

are taken from multiple sensors, there are multiple images of one object hence those images once denoised and enhanced, need to be fusion-ed together as one.
The images are further decomposed which is a process that works in combination with fusion of the images by creating image pyramids. This process is also necessary for the last step which is Segmentation. This step is the last step where the objects are further extracted from the decomposed and fusion-ed images. These extracted objects are at the end classified as weapons depending upon their shape and form. The authors have not clarified the dataset but the information that is available seems to point to self-made senor images. Accuracy rate or true positive rate has also not been mentioned in the paper by the authors.
Since these techniques are based solely on image processing methods, they were not only extremely slow [10] but also acquired a huge sum of resources and not to mention were very prone to errors and mismatches.
These problems were greatly reduced when Machine Learning and then Deep Learning became mainstream, and researchers started conducting research in order to develop new techniques and algorithms to detect objects.

2.2 Deep Learning based Gun Detection

2.2.1 Convolutional Neural Network

One of the same authors Gyanendra et al. [11] from the previously discussed research paper, used a different approach to tackle the same issue. They used Deep Learning based Convolutional Neural Network (CNN) [5], primarily Faster R-CNN (Region based CNN) which uses a combination of region proposals along with convolutional neural networks to extract the features of an image based on the regions, along with labeling those extracted features and then training the entire model using this process by going through the entire training dataset.
During training, the authors performed preprocessing step by abstracting the mean value of RBG, which is computed on the training data. After that, the image is moved along the stack of convolutional layers whose filters are of the size 3×3, which have a convolutional stalk of 1 pixel. Next, they implemented spatial pooling by using, five max-pooling layers. With the help of two strides, max pooling was performed on a 2×2 - pixel window. Therefore, three fully connected layers were initiated to follow the stack of those convolutional layers, which have varied depth in varied architectures. Softmax was used in the final output layer and with the rectification non-linearity, the entire hidden layer can be provisioned. The authors used Gradient Decent Learning for weight changes in the classification network of the CNN [5] since the problem required a multi-layer architecture to implement classification. The authors used a self-made dataset which was ordered by them with 65 positive images (gun present) and 24 negative images (gun is not present) The dataset was set up such that it comprises of images of various kind of handheld gun with various scale, revolution, and orientation.
CNN [5], a Deep Learning based technique, allowed the authors to design a system that was

less resource intensive and achieved an accuracy of 93% which is remarkably better from the conventional image processing techniques they used previously.
CNN [5] is an important and one of the most effective architectures used by various authors and researchers such as the authors Gulzar Ahmad et al [27] who used an approach called Intelligent Ammunition Detection and Classification [27], IADC [27] in short in which CNN [5] was predominantly used as the main architecture for training and making use of deep learning models. Their methodology consisted of several preprocessing steps before passing the data through the CNN [5] model. The authors have not given many details regarding their dataset that they used but what is evident is that the images are gun based and are are classified in to two portions, gun and without gun. The authors were able to achieve an accuracy of 96.74% using their discussed methodology.

Another approach was implemented by the authors Jose L. Salazar Gonzalez et al [22] in which they used a model based on R-CNN-FPN [6] architecture with ResNet-50. has been trained with 14 different dataset combinations and tested with four datasets. All the experiments were executed in a GeForce GTX-1080Ti with 11 GB of memory. The datasets used during the training were augmented with a horizontal random flip, adding more data while preserving the rifles, and "1:1", since it is the ratio that usually presents the guns and in some cases the rifles. These four scales and three ratios cover all the most common possibilities in which these objects appear. Other scales and ratios were also studied; however, this configuration was the one with the best results due to the nature of the desired objects. The models were trained with a batch size of 2 and a learning rate of $10^{-}3$ in the first 40k steps, and then $10^{-}4$ up to 80k steps, as this allowed us to avoid over-fitting and obtain better results. The authors used several previously existing datasets which are images generated from mock attack videos. These were a total of 33,500 images consisting of different weapons all a combination of multiple datasets. The authors were able to achieve an average accuracy of 91.43%.

Certain authors have used the word 'Deep Learning' as their architecture itself for model training. Such as authors N.R.Sathis Kumar et al [28] have done. They have used an approach where the deep learning model has been given preprocessed data for training as well as testing. The preprocessing includes, WLS Filter and HDR Tone Mapping. Once the images are preprocessed, they are fed into the model for training or testing on the purpose of object detection. Information regarding dataset has not been provided other than the fact that the data was unstructured.

2.2.2 VGG16

The authors Justin Lai et al [25] used the deep learning techniques. They pre-processed the images by re-sizing them to 640x480 so that they can be compatible with Tensor-Box framework. They then used a script to set bounding boxes around each firearm within each image. The script output the x, width and y, height of each respective bounding box to an output script. They then reformatted the output script to be used as a json using a separate script. For their baseline they used VGG-16 classifier and tested on 200 self-collected images of various guns. It achieved an accuracy of 58% on revolver classification and 46% accuracy on rifle classification. Then they decided to implement

Over-feat through tensor-box and trained by using 20% confidence threshold and a learning rate of 0.0003. This resulted in train accuracy of 93% and test accuracy of 89%.

Similarly, we have another use of this architecture as researched by the authors Javed Iqbal et al [29]. They have implemented a system where they have used VGG16 in combination with Region Proposal Network (RPN). RPN is applied to the deep features computed by VGG16. Next, they have used Faster R-CNN for object classification. The authors have used firearm datasets collected from multiple sources. The dataset in its finished form includes 10,973 fully annotated images with 13647 firearm images. The dataset has been classified into two parts, Gun and Rifle. The authors were able to achieve an accuracy of 88.3%.

Another author Jiahao Li et al [30] have used the above discussed combination approach in which they have used two architectures that is VGG16 and FRCNN for model training. The dataset that they have used have been sourced from multiple openly available sources. It consists of a total of 200 images of guns and people holding guns. Out of a training dataset of these images, 1500 were randomly selected for training and 500 were used for testing. The authors were able to achieve an accuracy of 98.8%.

2.2.3 Single Shot Detector

We have authors JunYi Lim et al [21], who have tapped into other techniques within the deep learning field. They have used a modified version of the Single Shot Detector (SSD) [9] known as the M2Det. It integrates a multi-level feature pyramid network (MLFPN) into Mobilenet-SSD. M2Det is a single-stage object detector based on a multi-level feature pyramid network. The purpose of using M2Det is to essentially train a feature pyramid network to learn the multi-scale and multi-level spatial representation of objects at varying resolutions, object rotations and angles. Such features would be advantageous in detecting firearms as an object within the perspective of a video surveillance environment. The authors have used an existing dataset of images of guns from CCTV footages. The dataset is UCF Crime dataset [20] consisting of 3000 images of guns. The authors were able to achieve 80% average accuracy using the mentioned dataset.

2.2.4 YOLO

Arif Warsi et al [23] also used a deep learning model called Yolo V3. They created their own dataset containing guns with different position and orientation with imageNet dataset. They have used transfer learning for training YOLOv3 model for gun detection and used the weight trained on ImageNet by YOLOv3 team instead of starting from zero. YOLOv3 is an object detection algorithm widely used for real time processing. Input images were divided into $M \times M$ grids. A single object is then predicted by this grid cell. Logistic regression is used to predict an object scores for each bounding box by YOLOv3 and changes the method to compute the cost function. The detection results are examined frame by frame in the videos the highest accuracy they achieved

was 98.64%.

Object recognition is the process of predicting the real class or category of an image to which it belongs by making probability high only for that class. CNN's are used to efficiently perform this process. Many state-of-the art Classification and Detection algorithms use CNN as a backend to perform their tasks. This approach has been implemented by the authors Muhammad Tahir Bhatti et al [31]. The architectures they used for the object detection in real-time is faster R-CNN- inception ResNetV2, Yolo V3, V4 and for classification they used VGG-16, inception V3. They collected databases from different sources and used tree different datasets, where dataset 1 contains (Total 1732 images), dataset 2 contains (Total 5254 images) and dataset 3 contains (Total 8327 images). These images data was pre-processed using various image processing techniques such as RGB to Gray scale, equalizing etc. The highest precision was observed on dataset 3 by YOLO V4 and Faster R-CNN which is 93% and 86.38% respectively.

2.2.5 Haar Cascade Classifier

Aarchi Jain et al [24] also used Haar Cascade Classifier, open-cv libraries. OpenCV, the most used library for computer vision applications. Haar cascade classifier is an approach for object identification in an image, video, or live streaming. They used datasets from different sources that included both positive and negative images. The Video was converted into frames, and then each frame was checked if they contain any feature of the object that feature is detected with a rectangle and entitled with the object name. They created positive and negative images for dataset and then trained the Haar classifier with those images. For finding the model and type of the gun in the video XML files are used. After that they detected and classified guns by using created xml files. According to the results, a high accuracy rate is obtained by Submachine Gun which reaches the 95%. The next best accuracy is of Assault Rifle which shows an accuracy of 87.5%. The accuracy of the Machine gun is 85% and at last accuracy of Pistol is 80%. Moreover, in the video, the gun can come at any angle and there can be an identification problem and to solve this the images of the gun should be from different angles and should cover every feature of the gun for more accuracy.

2.2.6 Transfer Learning

Transfer learning can play an important role in improving the over all performance and accuracy of a model since this method uses the weights from a previously trained model and declares them as the starting weights for the newly to be trained model, making a good use of previous work and the datasets that were previously used. This approach has been implemented by the authors Mehmet Tevfik Ağdaş et al [26] where they have used multiple architectures such as Alex-Net, VGG16 and VGG19 in combination with transfer learning to achieve their desired results. The authors used dataset from various open access sources. The total images in their dataset were 16000 images containing 9500 knives, 3500 guns, and 3000 ordinary pictures. The authors were able to get a much higher accuracy value using transfer learning as opposed to learning from scratch. They got

the following accuracies, Alex-Net with 97.74%, VGG16 with 99.38% and VGG19 with 99.27% respectively which is substantially higher than the highest accuracy they got from Alex-Net with 84.58% when learning from scratch.

2.3 Machine Learning vs Deep Learning

Machine Learning without a doubt was a huge achievement in computer vision technology but it lacked certain abilities that made the process of building and implementing a model that much difficult. The major issue was that for any Machine Learning algorithm, the features of the images had to be extracted manually which on its own is a cumbersome process but at the same time makes the process prone to human errors the elimination of which has been the priority for making machines intelligent in the first place.
Deep Learning solved this issue by extracting the features automatically and then performing learning of these features making the entire process seamless and more efficient. This was a remarkable achievement in computer vision that paved way for so many solutions to previously, impossible to solve, problems.

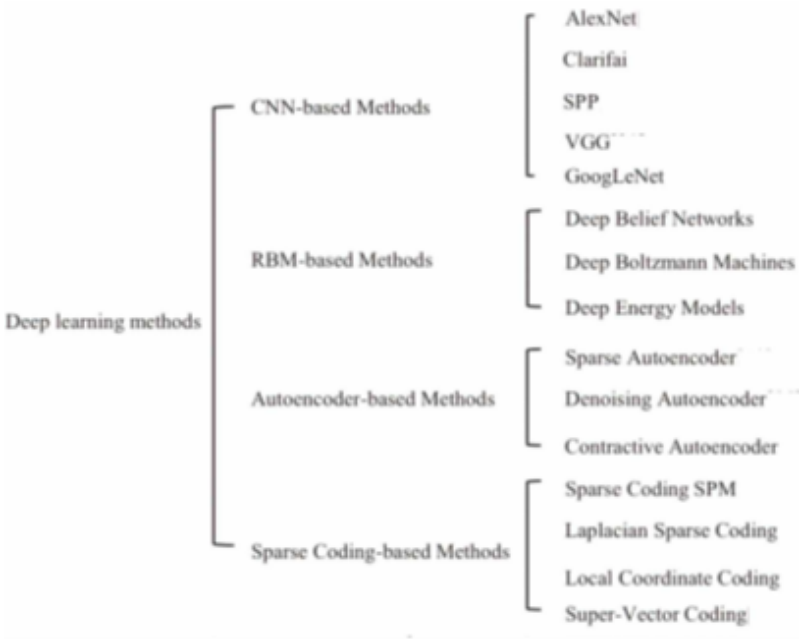

Figure 2.3: Categorization of the DL methods and their representative works [12]

The authors Mai et al. [12] conducted a research where they used Deep Learning techniques to detect guns in an image. They used a CNN [5] approach to building, training, and implementing a model. The approach that they used, consisted of first Resizing and Scaling the RGB images to a suitable requirement for CNN [5]. Next, they divided their dataset into 70% Training and 30% Testing images. Once the division was done, Transfer Learning (TL) [12] was done to gain knowledge and information from previously trained models. Then features were extracted using CNN [5] and finally classified either as a gun or not a gun.

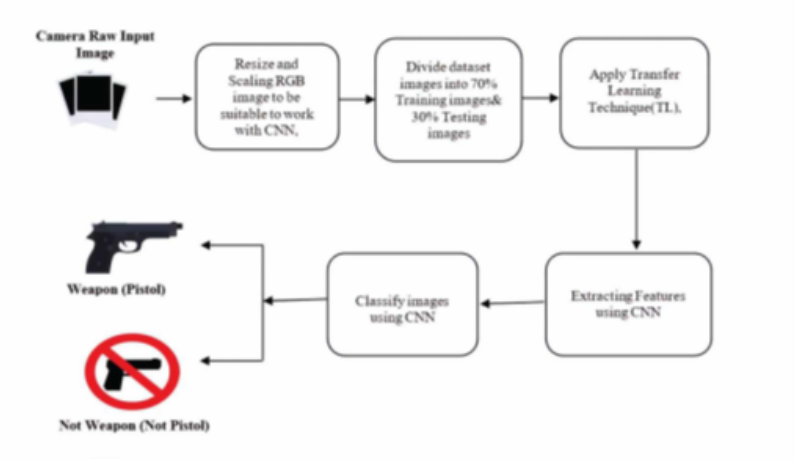

Figure 2.4: Block diagram of the proposed approach for classifying image pistol/gun or not [12]

In order to extract the features and apply classification, the authors used pre-trained networks to extract rich features and to accomplish the classification process. Max-pooling was used since it is the best because it accelerates the learning process, and it is more effective in computation. After that, they attached ReLU activation function after every convolutional and fully connected layer. The authors used a total of 13743 images in the dataset all comprising of guns.

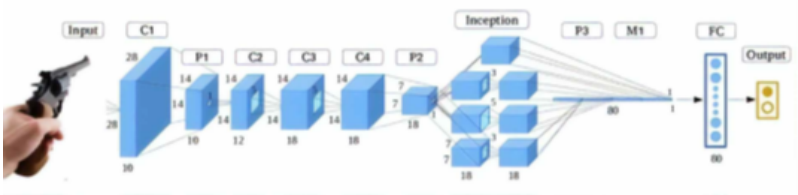

Figure 2.5: The training process of the proposed system with negative and positive dataset images [12]

Since this approach required the use of Alex-Net and GoogleNet, the authors were able to achieve a 97.9% of accuracy rate with GoogleNet and a 99.2% accuracy rate with Alex-Net.

2.4 Common Practices

In the recent years, deep learning has been the field within computer vision and machine learning that has been used the most. CNN [5], a model of deep learning has been at the front of it all. Majority of the approaches that are now used are based on CNN [5].
Different authors and researchers have started using other techniques in combination with CNN [5] such as R-CNN [6], Faster R-CNN etc but the foundation nonetheless is still Convolutional Neural Network (CNN).
This same technique is used in the case of our proposed approach as well.

2.5 Comparison of the Discussed Techniques

Category	Research Paper	Author/'s	Publication Year	Technique Used	Dataset	Accuracy (%)
Image Processing	A Computer Vision based Framework for Visual Gun Detection using Harris Interest Point Detector	Rohit Kumar Tiwari et al [10]	2015	Haris plus FREAK Combination Interest Point Detector	Self-Made Dataset (65 positive and 24 negative images)	84.26%
	A Computer Vision based Framework for Visual Gun Detection using SURF	Rohit Kumar Tiwari et al [18]	2015	Visual Gun Detection using SURF	Self-Made Dataset (15 positive and 10 negative images)	86.67%
	Concealed Weapon Detection Using Image Processing	Bhavna et al [19]	2012	CWD on Sensor Images	Not Specified	Nan
Deep Learning	A Handheld Gun Detection using Faster R-CNN Deep Learning	Gyanendra K. Verma et al [11]	2017	R-CNN Deep Learning (VGG-16/10)	Self-Made (65 positive and 24 negative images)	93%
	Automatic Gun Detection Approach for Video Surveillance	Mai Kamal el den Mohamed et al [12]	2020	CNN in combination with Alex-Net and GoogleNet	Multiple Sources (13743-images)	GoogleNet 97.9% Alex-Net 99.2%

continued on next page

	Gun Detection in Surveillance Videos using Deep. Neural Networks	JunYi Lim et al [21]	2019	M2Det	UCF Crime Dataset (3000-gun images)	80%
	Real-time gun detection in CCTV: An open problem	Jose L. Salazar Gonzalez et al [22]	2020	R-CNN-FPN	Multiple Datasets (33,500-gun images)	91.43%
	Developing a Real-Time Gun Detection Classifier	Justin Lai et al [25]	2017	VGG-16	Self-Collected Dataset (200-images)	89%
	Gun detection system using YOLOv3	Arif Warsi et al [23]	2019	YOLOv3	Different Datasets from ImageNet	98.64%
	Gun Detection with Model and Type Recognition using Haar Cascade Classifier	Aarchi Jain et al [24]	2020	Haar Cascade	Different Sources	87.5%
	Deep Neural Networks Based on Transfer Learning Approaches to Classification of Gun and Knife Images	Mehmet Tevfik Ağdaş et al [26]	2021	Transfer Learning	Different Sources (16000 images)	Alex-Net 97.74% VGG16 99.38% VGG19 99.27%

continued on next page

	Intelligent Ammunition Detection and Classification System Using Convolutional Neural Network	Gulzar Ahmad et all [27]	2021	CNN	Not Specified	96.74%
	Detection of Suspicious Activity in ATM Using Deep Learning	N.R.Sathis Kumar et al [28]	2020	Deep Learning	Not Specified	Nan
	Leveraging Orientation for Weakly Supervised Object Detection with Application to Firearm Localization	Javed Iqbal et al [29]	2021	VGG16, RFCNN, RPN	Different Sources (10,973 + 13647 images)	88.3%
	Preprocessing Method Comparisons for VGG16 Fast-RCNN Pistol Detection	Jiahao Li et al [30]	2021	VGG16, RFCNN	Different Sources (2000 images)	98.8%
	Weapon Detection in Real-Time CCTV Videos Using Deep Learning	Muhammad Tahir Bhatti et al [31]	2021	VGG16, YOLOv3, FRCNN	Different Sources (1732+ 5254+ 8327)	93%

Table 2.1: Technique Comparison Table

Chapter 3

Requirement Specification

Different systems for object detection have been developed, but anomaly detections such as criminal act detection are very rare. The basic approach used for this purpose is neural networks. Computer vision and image processing techniques such as convolution etc are essential components for this project. The idea was to develop a system using CNN (Convolutional Neural Network) [5] to detect a gun in a particular image to declare the scene as criminal act.
The above analysis shows that the work has been done on the detection of guns in a scene however, the achievements are limited as different aspects have affected the overall performance.

3.1 Proposed System

The proposed system is used to detect whether the scene in an image portrays a criminal act or not-criminal (Victim). This detection will be done on the basis of features extracted by CNN [5].
The system would be a desktop application that would be convenient to the end-user. This will be accomplished with the exceptionally intuitive Graphical User Interface (GUI). The interface will be improved for the accommodation of the end-user experience. The system will be created with the assistance of Image Processing to process the images and extract features out of it. It will be able to detect and process the image with the previously learned data and thus will provide the results using neural networks.
The data set used for the development of the application is from different sites such as Kaggle [15,16], GitHub etc. The data is composed of 12226 images, 6113 images for gun data and 6113 for not-gun comprised of different resolutions.
Following are the some of the images from our dataset.

- **From Gun Dataset**

Figure 3.1: Images from Gun Dataset

- **From Not-Gun Dataset**

Figure 3.2: Images from Not Gun Database

The system will be developed to make the process of criminal act detection automated and more practical. The data set provided contains a diverse range of images so using a deep learning model (CNN) won't be a problem and will help in generalizing the model well.

3.2 Methodology

The project will be developed using agile methodology for iterative development and testing. The agile methodology provides an incremental approach with testing of the certain iteration and is a suitable approach for development [13]. As mentioned earlier this project will be using Convolutional Neural Networks for our application. CNN [5] is a deep learning technique which is used to extract features and then classify. First images from both the datasets are converted into arrays and then normalized to gray scale by diving each image with 255. Normalization will help to remove distortions caused by lights and shadows in an image. After that the labels are stored in a list by fetching the path folder name i.e., gun, not-gun. After this step, data is divided into train and split. In order to increase the size of the training set without acquiring new images augmentation is applied, as this concept will duplicate images with some kind of variation so the model can learn from more examples.

In the end the model is trained where image data is fed to CNN [5]. In CNN [5] two layers of convolution and max pooling with 5x5 filters is applied which will eventually generate feature maps of major features. These extracted features will be used in testing phase where we will use these features to identify the object of interest. Finally, on the classification layer two neurons with activation soft-max (as we are dealing with classification problem) is used to find the probabilities of each class. The class with higher probability will be considered as the final answer/output.

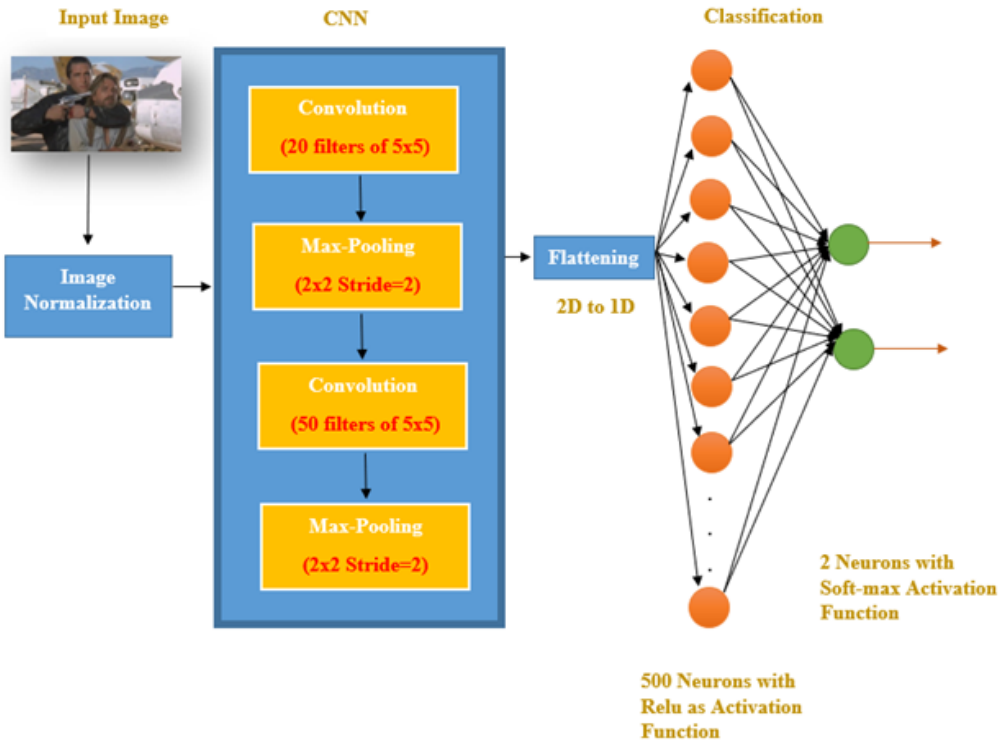

Figure 3.3: Methodology Diagram

The above Figure shows the in-depth methodology. In convolution layer, two layers of convolution with 20, 50 filter of size 5x5 (CNN [5] uses its own standard filters) are used and two layers of max pooling of 2x2 size with stride=2 is used to extract the major features.
After CNN [5], image is flattened from 2-D to 1-D so that it can be fed to neural networks for classification. 500 neurons with activation function Relu is used as all the value are max-pooled so majority of them will be positive.
For Output layer 2 neurons with soft-max activation functions are used to predict the probabilities of each class.

3.3 Non-Functional Requirements

- **Reliability:** The system should be reliable in detecting a gun from an image.
- **Maintainability:** Developers will be responsible for system maintainability.
- **Availability:** After installation of the application on computer, it will be available for use.
- **Re-usability:** Different modules of the system will able to be reusable in some other systems.

3.4 Use Cases

"Note: These use cases may change during SDLC!"

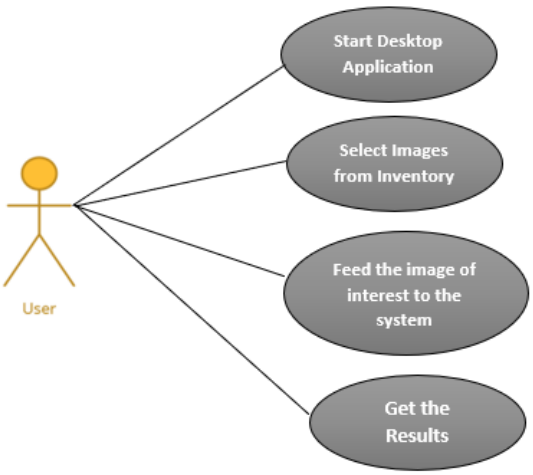

Figure 3.4: Use Case Diagram

3.4.1 Use Case 1: Start Desktop Application

Figure 3.5: Use Case 1 Diagram

Title	Start Desktop Application
Actor	User
Description	This use case represents the interaction of user with the actual application.
Main Success Factor	Application should launch upon double click.
Pre-Condition	Desktop must be running, and windows should be working.
Post-Condition	Application should show the home screen.

Table 3.1: Use Case 1: Start Desktop Application Table

3.4.2 Use Case 2: Select Images/Videos from Inventory

Figure 3.6: Use Case 2 Diagram

Title	Select Images/Videos from Inventory
Actor	User
Description	This use case specifies the selection of images/videos of interest.
Main Success Factor	Application loads the selected image/video.
Pre-Condition	Application must be running.
Post-Condition	Application should load the image/video properly.

Table 3.2: Use Case 2: Select Images/Videos from Inventory

3.4.3 Use Case 3: Feed the Image/Video of Interest to the System

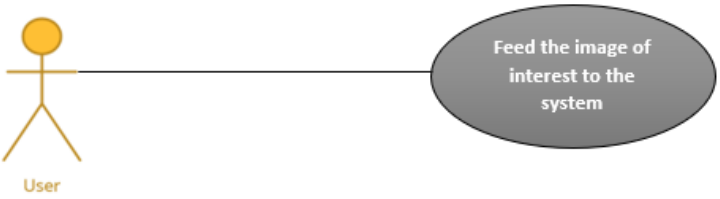

Figure 3.7: Use Case 3 Diagram

Title	Feed the Image/Video of Interest to the System
Actor	System
Description	This use case represents the interaction between front end and backend.
Main Success Factor	Front end should deliver image/video to backend properly for further operation.
Pre-Condition	Backend should be connected properly, and model should be loaded.
Post-Condition	The results from the classification should be received at the front end.

Table 3.3: Use Case 3: Feed the Image/Video of Interest to the System

3.4.4 Use Case 4: Get the Results

Figure 3.8: Use Case 4 Diagram

Title	Get the Results
Actor	User
Description	This use case represents the viewing of original result.
Main Success Factor	The correctly classified image/video is displayed.
Pre-Condition	Application is running.
Post-Condition	The classified label if displayed.

Table 3.4: . Use Case 4: Get the Results

Chapter 4

Design

System design defines the architecture, components, modules, interfaces, and data for system to satisfy specific requirements. Following represents the system design for developed application.

4.1 System Architecture

The application provides the user with an interactive user interface which takes images or video frames as input and provides the classified result. The detection is based on data provided for the training of the system. The image is processed, and the extracted feature maps are provided to the system. The processed image is analyzed by the system on the basis of extracted features and the resultant image with classified class is provided by the system. The application classifies the images using CNN (Convolutional Neural Networks) [5]. This completes the process of object detection in images and the scene in an image is classified.

The application works on the modules which include the database or the dataset module. The application works on the Architecture with the higher-level layer of the graphical user interface. The interface contains the controls which provide the basic functionality to the system known as the backend functionality. The functionality includes various processing options to segment out and extracts different features from the image. The detection is done on the basis of training provided to the system with the help of a dataset provided for training.

The developed system is a desktop application. The application has a simple architectural design comprising of following steps:

- Image/Video Frame as Input.
- Implementation of gun detection on provided input.
- Output the classified class after successful identification of the scene.

The developed system is a desktop application. The application has a simple architectural design comprising of following steps shown in the figure below:

4.1.1 System Architecture Diagram:

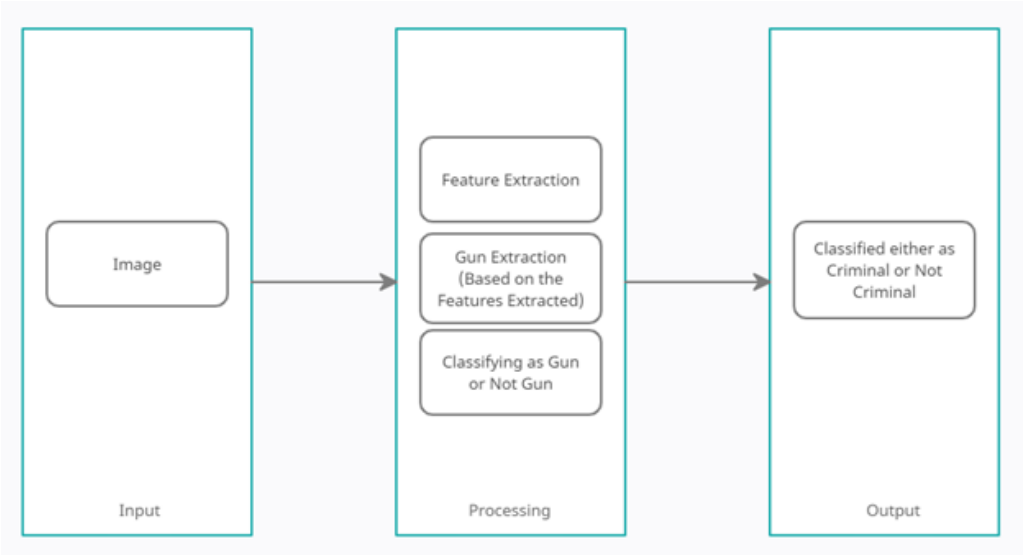

Figure 4.1: System Architecture

4.1.2 Conceptual Diagram:

Logically, the application can be divided into various packages to completely understand the main features and functionalities of the application.

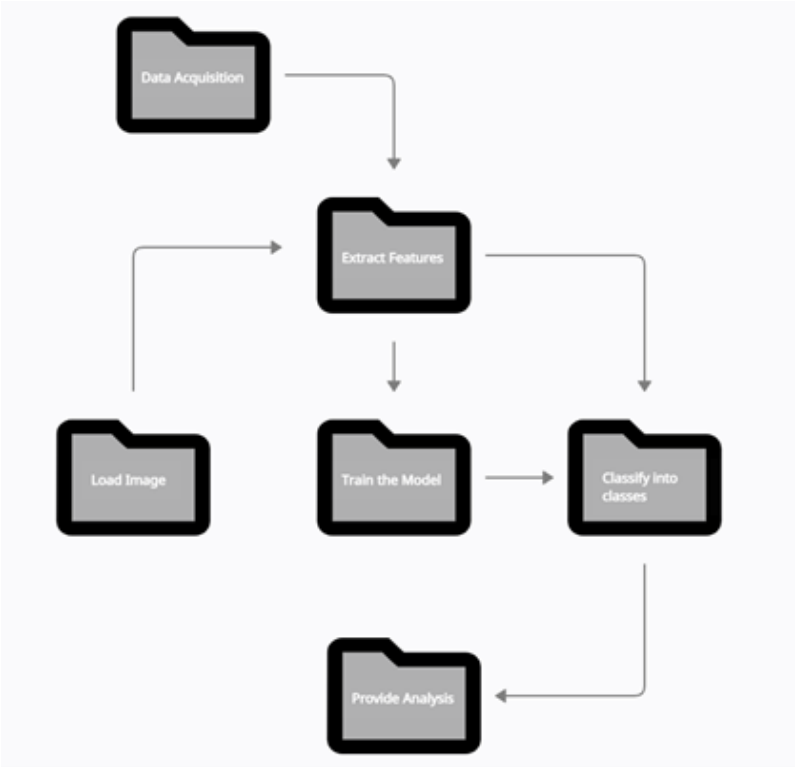

Figure 4.2: Package Diagram

4.2 Design Constraints

One major constraint imposed on the system is the use of a small dataset for its training for gun detection due to un-availability of bigger dataset for our application to train on. As deep learning models are data hungry, so generalization is a bit difficult task. Due to this limitation, our application may not be able to detect the desired object in some cases or misclassify them.
The model used to train our system only take raw images as an input, no pre-processing is done besides normalization, so there is a chance if the image is not clear or contains any noise that is making it difficult to process can cause classification error or may lead to bad generalization.
As this is an initial prototype, so this limitation is subjected to change in the future. Our application only works on windows specific environment.

4.2.1 Application Constraints

The automated system is a desktop application that will take images as input and further process the image for object detection. The application cannot be accessed across the web.

4.2.2 Software Requirements

Python is the programming language used for the development of the application. There is no integrated language required for the development of the application. Anaconda and Visual Studio is the main development environments considered for the development of the application.
The system uses various APIs and libraries for the implementation of image processing and deep learning algorithms. Some of them are,

- Keras Model [17] for CNN [5] (Image Processing and Neural Networks)
- OpenCV and CV2 for Image Processing

4.3 Design Methodology

Since the project is based on deep learning, we have implemented Convolutional Neural Networks (CNN's) [5]. The back end of the project is developed using python, so we are using TensorFlow API [14] in the backend and Keras [17] for the implementation of CNN [5] in python.
Initially we divided our dataset into two parts, one is the train data and the other is the test data. The train data is provided to the CNN [5] API which will perform the following tasks:

- Feature Extraction
- Model Training
- Classification

The training data is used to train the model, in doing so, the system extracted the different features and as per our determined labels, the system classified the required image on the basis of features to later on determine the action.

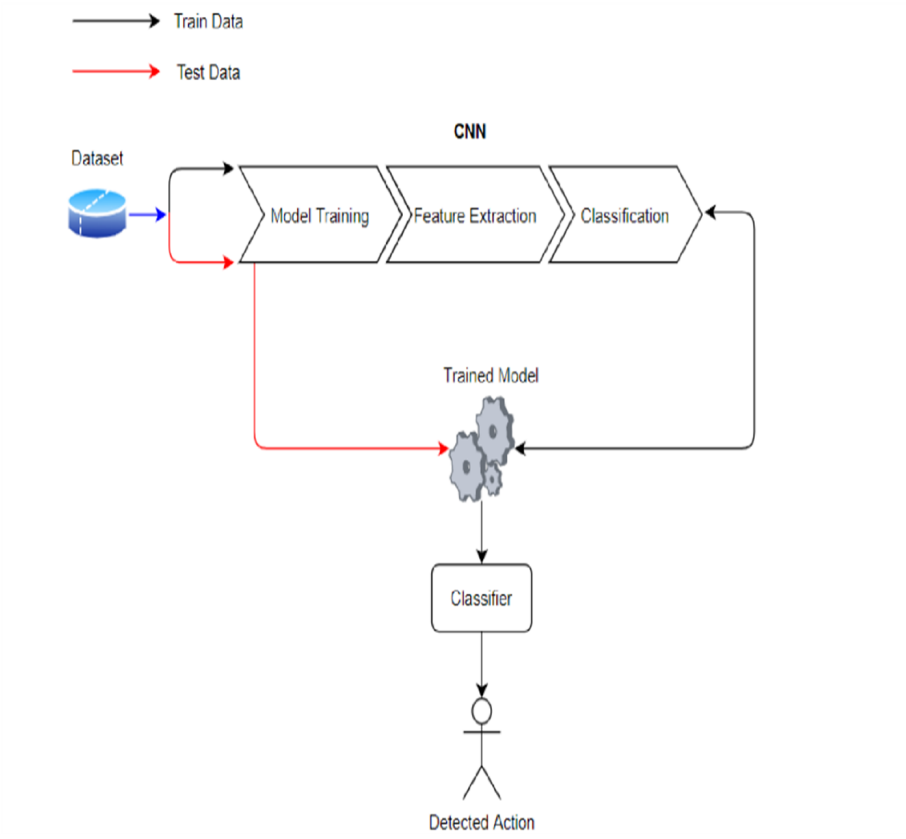

Figure 4.3: Methodology Flowchart

After training the model, the test data was used as the input to test and check the accuracy of the trained model and ultimately determine the action being performed in the respective test data.

4.3.1 Process-wise Analysis

In order to understand various processes of design methodology, we need to elaborate on all the processes and steps. Given below is the component-wise analysis of each process.

4.3.1.1 Image/Frame Acquisition Process

Initially, the image/frame is acquired using the load image control. The acquired image/frame is then provided for preprocessing.

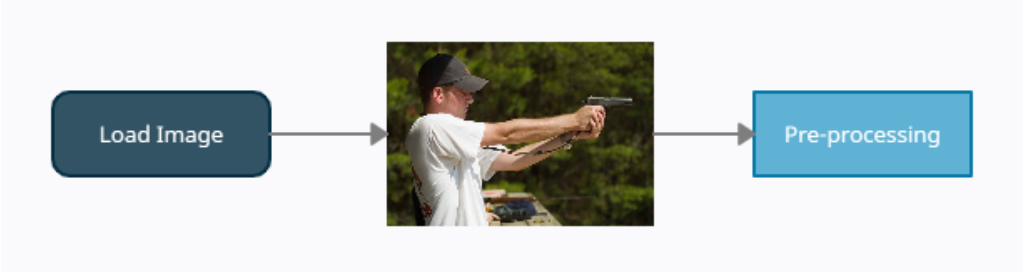

Figure 4.4: Image/Frame Acquisition Flow Diagram

4.3.1.2 Pre-Processing

After loading the image/frame, it is normalized to gray scale from RGB (0-255) in order to remove any noise.

Figure 4.5: Image Pre-Processing Flow Diagram

4.3.1.3 Feature Extraction

After normalization, image/frame is fed to CNN [5]. In CNN [5] two layers of convolution and max-pooling are applied to extract the features and create a feature map.

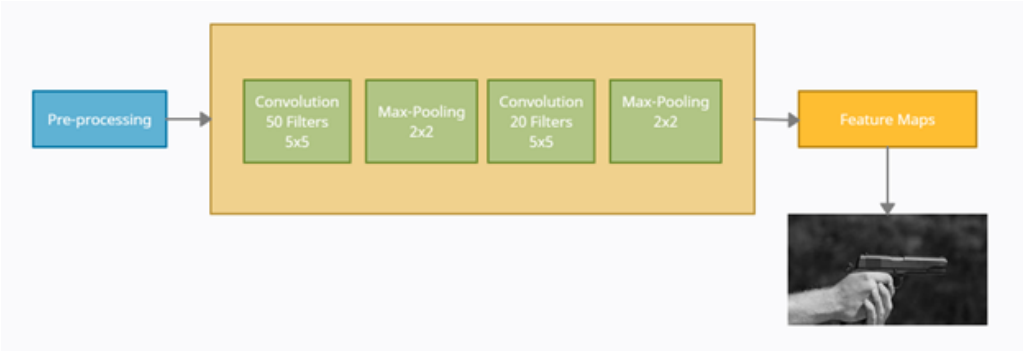

Figure 4.6: Image/Frame Feature Extraction Flow Diagram

4.3.1.4 Classification Process

Once the features are extracted from the image/frame, the classification is done. The feature set obtained from the feature extraction phase is used for further detection. The prediction is done using a fully connected neural network. The results are provided based on the predictions.

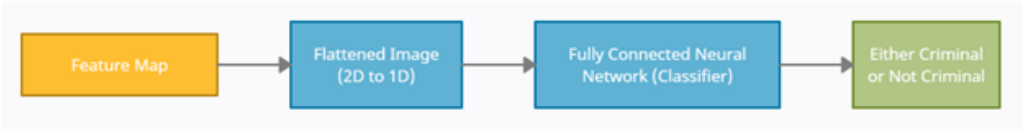

Figure 4.7: Classification Process Flow Diagram

4.4 High Level Design

4.4.1 Component Diagram

The component diagram shows the overall high-level view of the application and explains the functionalities and responsibilities of the system, and how different tasks are divided and assigned to different components. Our system can be divided into three main modules as illustrated in Figure below.

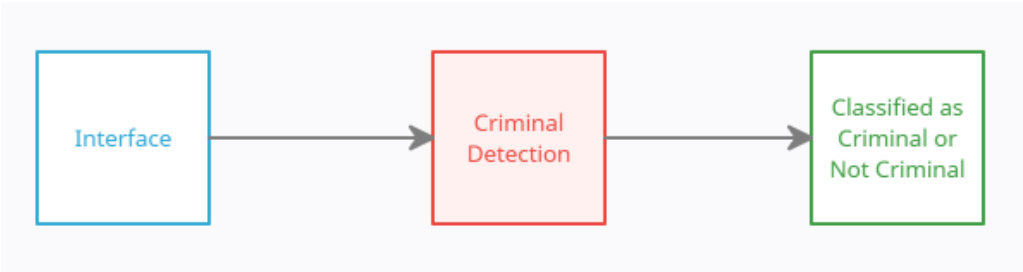

Figure 4.8: Component Diagram

The above figure represents the basic functionality of the application into separate packages. The packages are linked according to the flow of the processes. All the above packages provide the basic functionality of the system and the important function modules as well.

4.4.2 System Interaction Diagram

System interaction diagram shows how the various processes within the system interact with one another. It also shows how the messages are exchanged between the objects to carry out the functionalities of the given scenario. Figure shows the system interaction diagram.

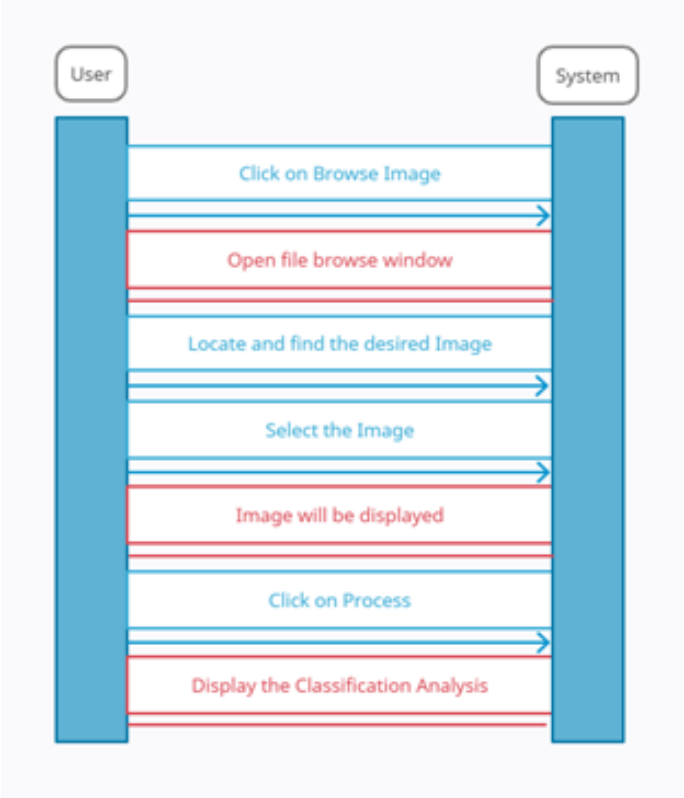

Figure 4.9: System Interaction Diagram

4.4.3 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Deployment diagram is displayed in Figure 4.10.

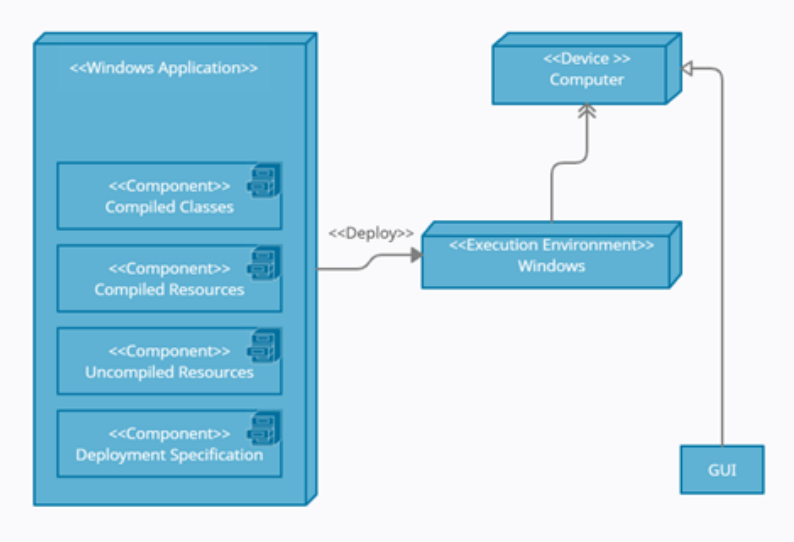

Figure 4.10: Deployment Diagram

4.4.4 Modules

The application is intended to be developed using Python as the programming language. The application will have separate modules of code written for various functionalities. Since the modules can be imported into the present working directory of any development environment for Python, the related modules of functionality will be used when required during the development procedure.
For GUI separate libraries are used as it has been developed using Tkinter in python.

- Preprocessing and training module
- Processing options for images/frames
- Classification module
- GUI for the application

4.4.5 Security

Since the criminal detection application is a desktop version, the application cannot be accessed across the web. This makes the application isolated and thus protected from any external access or malware. The application is associated with the desktop system therefore, no external modules or files can be accessed from the application.

4.5 Low Level Design

The application ideally depends on two main concepts image processing for gun detection and fully connected neural networks for classification. Since the programming language considered for the

development is Python, the modules associated with the python programming are utilized for image processing and deep learning. There are several modules used for the processing of data.

4.5.1 Data Acquisition

The dataset is converted to an array in order to process for further analysis using NumPy.

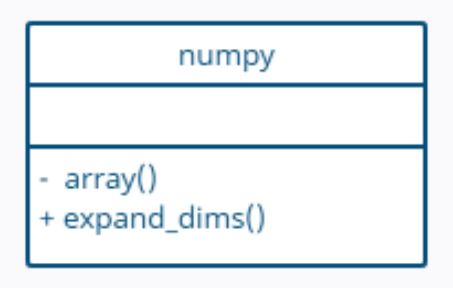

Figure 4.11: Class Diagram for Numpy Module

4.5.2 Image Processing

For various image processing and feature extraction techniques, OpenCV (CV2 modules are utilized).

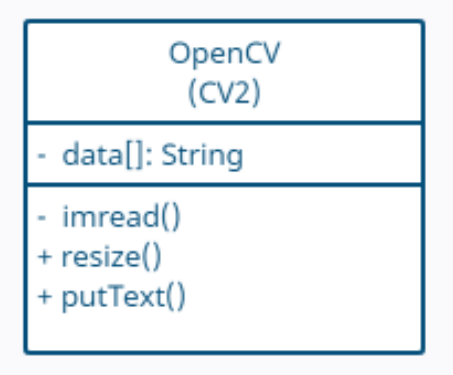

Figure 4.12: Class Diagram for OpenCV Module

4.5.3 Train and Classification

For training of the dataset and classification algorithms, the sklearn module has been used. It provides various algorithms and techniques to split the data into train and test.

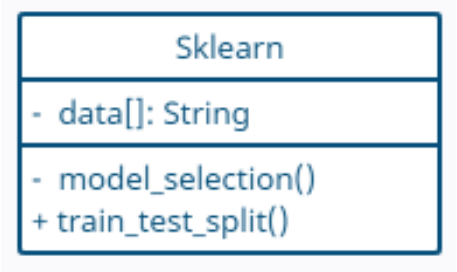

Figure 4.13: Class Diagram for Sklearn Module

4.6 GUI Design

The Graphical User Interface of the automated system is intended to be user-friendly and highly interactive for the end-user. The interface is equipped with a browse image and a browse video button along with a picture box to view the loaded input. Next, the detection system starts execution, and the results are shown in the picture box. It is a very simple GUI where the user is required to choose the image and press the process button to get the desired results.

"Note: Product design may change during SDLC!"

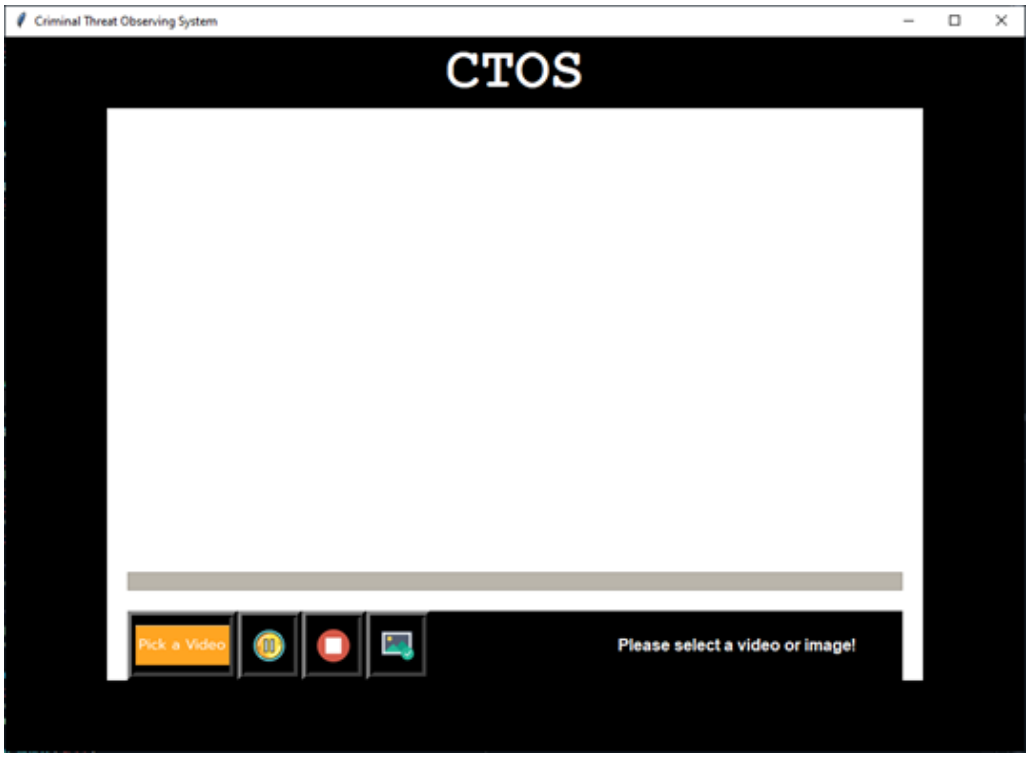

Figure 4.14: Graphical User Interface (GUI)

Chapter 5

System Implementation

5.1 System Summary

Criminal Threat Observing System is an application that provides criminal act detection in a scene using Deep Learning model. The user can identify the act by simply passing an image or a video to the system. All this process only requires the selection of image or video by the user, rest will be done by the system which makes the system quite simple and usable for the end-users.

5.2 System Architecture

The system architecture is divided into three major parts:

- **Input**
  The inputs consist of reading videos and images.

- **Processing**
  The processing is the main unit of the system which performs various tasks i.e., extracting frames from a video (IF VIDEO FILE IS SELECTED!), resizing the frames or images that suites the model and then feeding them frame by frame or the image to our trained deep learning model. After the results are obtained from the model frame is sent for a parallel checkup function to double check.

- **Output**
  After the double check, the respective label is attached to the frame or an image and finally displaying the results.

5.2.1 Algorithmic Workflow

Figure 5.1 below shows the overall workflow of the system:

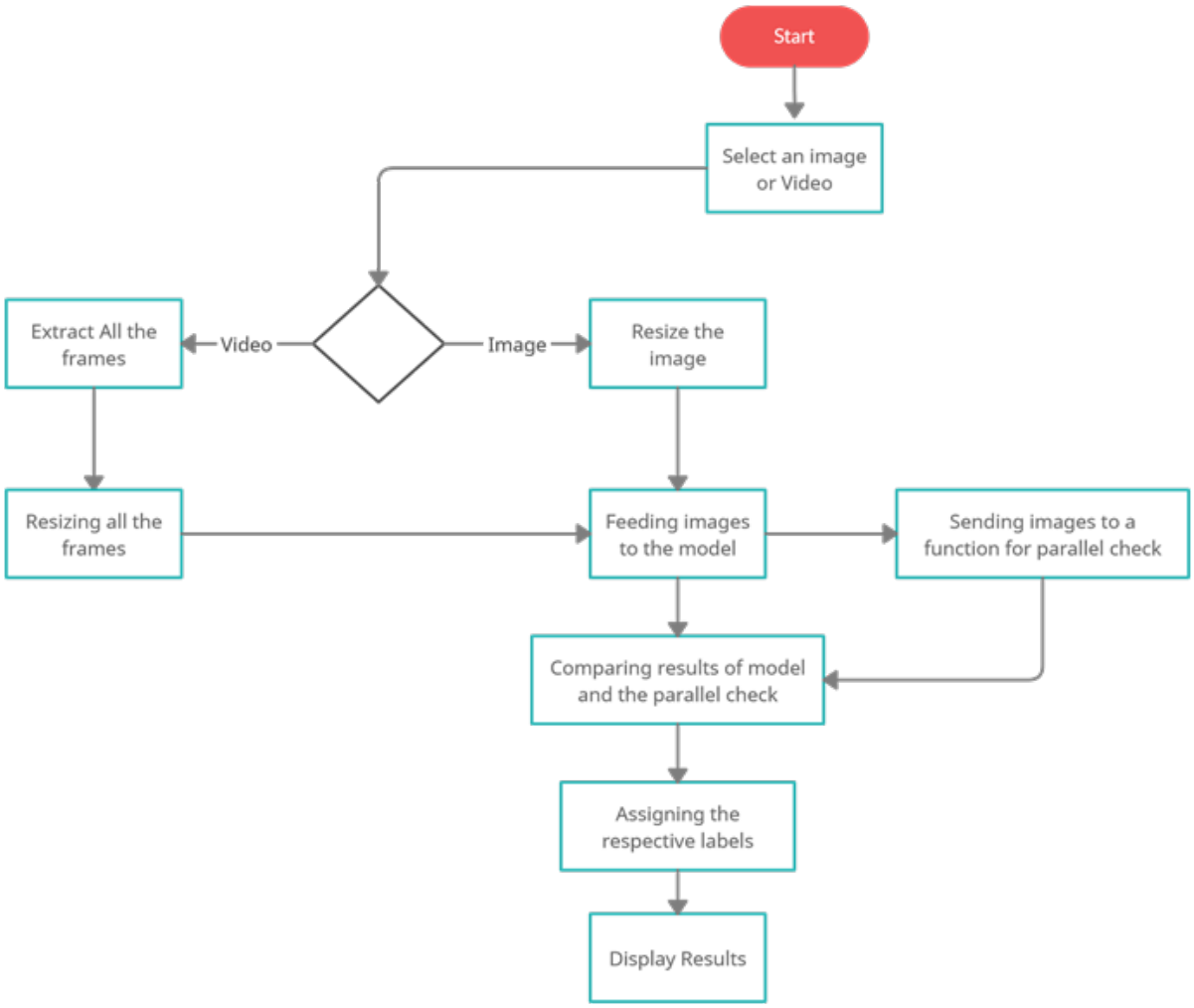

Figure 5.1: System Activity Diagram

5.3 System Components

Following is the list of the components of the system based on algorithms deployed:

- Converting video to frames (if video file is provided otherwise move to step 2).
- Pre-processing the frames or image.
- Deep Learning model for detection
- Passing frames/images to the model and gun detector function for parallel check
- Comparing the results of both model and the function and predicting class
- Converting the frames back to video (if video)
- Displaying the images or video

5.3.1 Convert Video to Frames

When user selects video as in input, system needs to extract frames from it. The number of frames in the video depends upon its duration. This process has been made a lot easier with python's OpenCV library that provides a built-in function (VideoCapture) to capture the frames in a video. After extracting these frames are stored in a temporary folder as JPEG files and are stored with a unique name to avoid any duplicate or replaced frames. From where they are picked in a sequence later for further process.

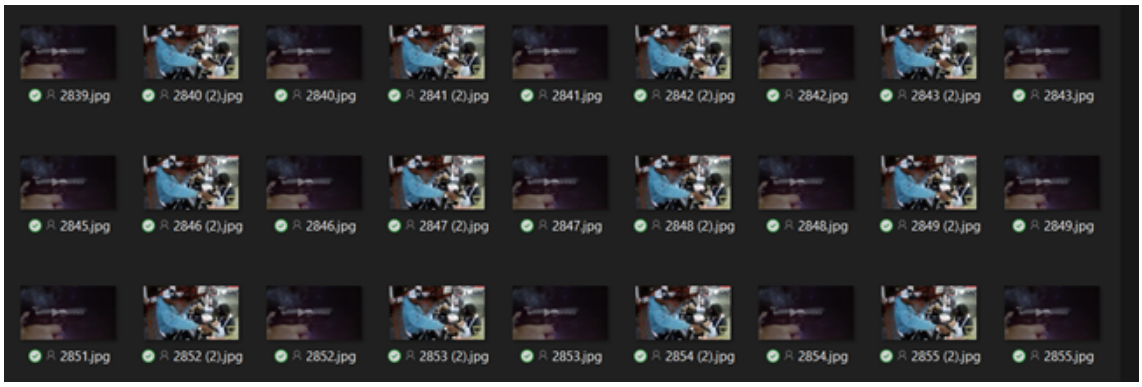

Figure 5.2: Video Frames

5.3.2 Pre-processing Frames or Image

After the extraction of frames/image, system pre-processes these frames to convert them into a format which is acceptable by deep learning model which detects what kind of scene is in the given frame/image. This process is achieved by resizing the frames/images.

Figure 5.3: Image Resizing

5.3.3 Displaying the Images or Videos

There were three different models with different parameters were tested and the one that had the highest accuracy, qualified to be integrated into the system. Tests were conducted on a simple CNN with 4 Layers, CNN with 6 Layers and Alex-Net.
Different activation functions, kernel sizes for CNNs, layers for CNNs, learning rate, epochs and batch-sizes were experimented. In some cases, over-fitting was observed. The tests are briefly explained in chapter 6. All these experiments were performed on custom collected dataset from various sites. Some models have been explained below.

5.3.3.1 CNN-6 Layers

Our first model used was CNN with 6 layers, 3 convolution layers and 3 max-pooling layers. It was used for feature extraction. Initially this model was tested on activation function (Relu) in hidden layers and on final layer (SoftMax). Many other variations were made in activation functions and other parameters and the results are discussed in chapter 6.

5.3.3.2 CNN-4 Layers

Results of this model were very promising and is selected for our final system. The best accuracy was observed after using 2 layers of convolution and 2 layers of max-pooling. Both on video and image dataset the results were impressive. This model was tested using different parameters i.e., epochs, learning rates and batch sizes.

5.3.3.3 Alex-Net

Upon using this model over fitting on image dataset was observed and the results on video dataset was low in accuracy than CNN with 4 layers.

5.3.4 Passing Frames/Images to a Gun Detector Function for Parallel Check

Finally, the pre-processed images are sent to the deep learning model to perform classification. The model chosen to be the part of the system has achieved the highest accuracy among all the tested models and configurations. As no system can alone be trusted, we have used another image processing technique as a parallel check for detecting the gun as an object and draw a bounded box around the objects in a frame/image. This a function that returns the image with bounded boxes and a flag. If the flag is true, the function has detected the gun and if false, vice versa. Now applying this parallel check gives us a double confirmation of the scene helping us to determine the results correctly.

5.3.5 Comparing the Results of both Model and the Function

After receiving the results from both the model and the function, results are compared. As our model is highly sophisticated, well trained and has achieved the highest accuracy, we decided to give our model results more dominance. We made 3 checks, where first the model results are checked then the system checks the probability of that class, if it is greater than 80%, the label detected by the model is assigned, if less than 80%, it double checks the results obtained from the parallel check function and decides accordingly. In this way a lot of faulty readings have been improved the overall results are correct.

5.3.6 Converting the Frames Back to Video (if video)

After the labels are assigned, these frames are stored in another temporary folder. These frames are then retrieved and stored in an array via loop and converted back to the video. For this process,

OpenCv has a built-in function called (VideoWriter) that combines all those frames at a given frame rate and make the given file format which is stored later to the result folder.

5.3.7 Displaying the Images or Video

If the above processes are successful, the resultant video/image with the detected action label will be displayed for the user.

5.4 Tools and Techniques

This whole application is developed using Python and Anaconda Spyder as the Integrated Development Environment (IDE). The application requires a lot of processing on frames/images so for that python libraries such as OpenCV are used to load and process frames/images. For optimization the frames/images after processing and generating the results are deleted from the temporary folders so that the extra memory is not used, and the system doesn't over-load the hardware. The front-end is also developed in python using the standard GUI library called Tkinter. It provides a fast and easy way to create GUI applications providing a powerful object-oriented interface to the TK GUI toolkit.

Chapter 6

System Testing and Evaluation

6.1 Graphical User Interface Testing

The Graphical User Interface is the only interaction between a user and a system. So, keeping this in mind GUI is designed after a detailed research and as a result the user can interact with our system easily without any additional training or knowledge. It consists of only two buttons for the main process, one for image selection and the other for video selection. User just needs to press any of these buttons and select whatever he requires (video/image), and the results will pop-up on the display in front of him. Two additional buttons are added to play/pause the resultant video. This GUI has been tested using different subjects with very little knowledge of computers and were easily able to use it.

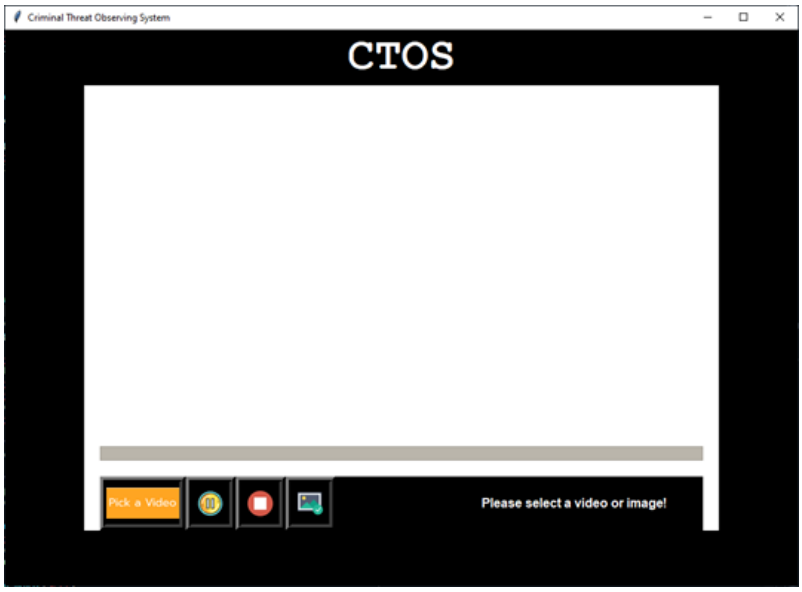

Figure 6.1: Main Screen Layout

6.1.1 Test Case

6.1.1.1 Input Image/Video

The image or video file is to be selected from the browse window. As seen in Figure 6.2.

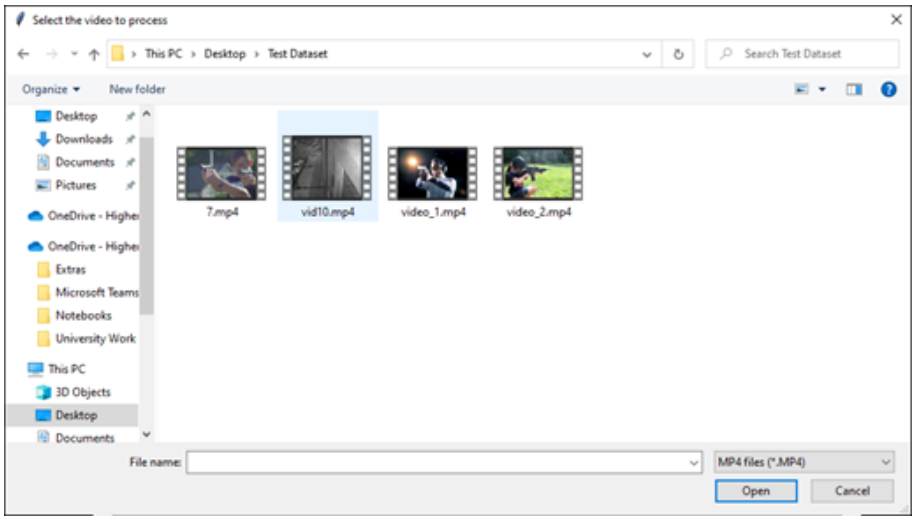

Figure 6.2: Browse Window for Videos/Images

6.1.1.2 Task Execution

Wait for the system to generate results. As seen in Figure 6.3.

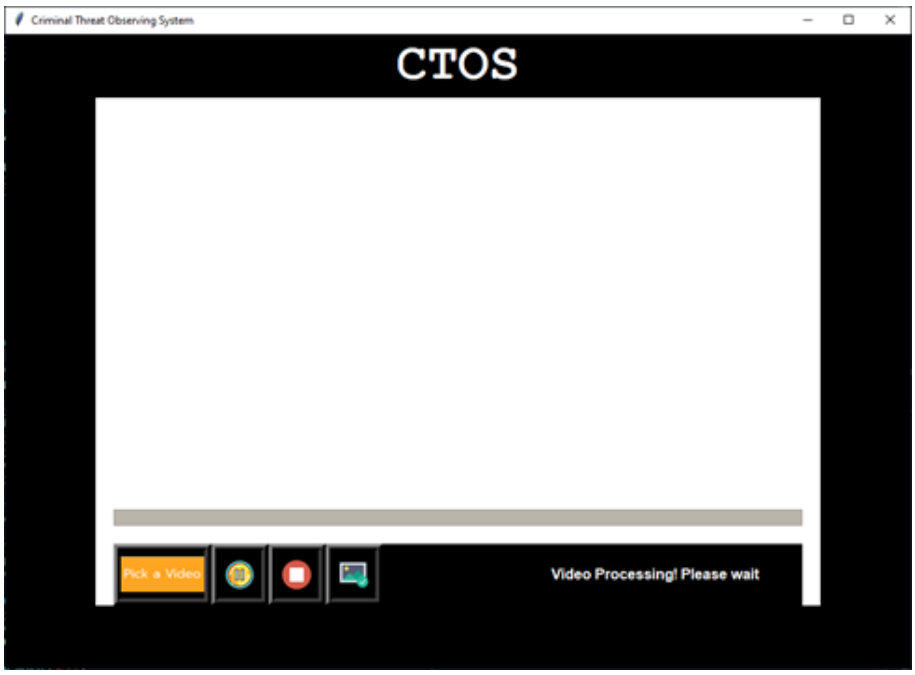

Figure 6.3: Video Processing

6.1.1.3 Results

After the execution, results will be displayed. As seen in Figure 6.4.

Figure 6.4: Results

6.2 Usability Testing

Usability testing was carried out deeply and was made sure that users are having a smooth and issueless interaction with our system. As mentioned earlier in the GUI section, GUI was designed by keeping all these usability scenarios in mind. Our application only requires the selection of images/videos and rest of the operations will be performed by the system. So, after being tested by different individuals a positive feedback was received.

6.3 Software Performance Testing

In general, system performance testing is a testing practice performed to determine how the system will perform in terms of responsiveness, accuracy, and stability. After running various tests under different situations, our system was responsive, smooth, and accurate every single time. In the case of images, result generation and processing were snappy, but in case of video, depending on the system power and the number of frames, performance may be affected accordingly.

6.4 Exception Handling

Exception Handling is an important step to make sure that the end user does not get any unexplainable error while they use the software. We have implemented this in two major cases. First, we have set a fixed size for each image. This means that any image or frame that is given to the system as an input will be fixed to a given size before passing it on to the model. The second exception that has been handled is that when frames of a particular video are created and stored, they are deleted after they have been passed on to the system. Similarly, the frames that are created as result are also deleted after their conversion back to video thus eliminating the chance of excessive memory fill-up.

6.5 Load Testing

In load testing, our model was tested under stress and extensive load. The model performed very well and handled every possible load i.e., in the case of videos, some of them had long duration and hence had a large number of frames, similarly, in the case of images we used images of various different resolution and sizes. Our model was able to handle them smoothly, obviously it took a bit of time to process them as the processing on video requires many additional steps other than just passing images/frames to the model and receiving the results. Thus, our model passed our load test upon various tests.

6.6 Test Cases

Following are the test cases implemented on our software.

6.6.1 Software Startup Test Case

Test Case ID	**Test Req ID**	**Description**	**Applicable for**	**Initial Conditions**
TC01	TR01	Testing Software Startup	All Desktop Windows Systems	None
Step	**Task & Expected Results**			**Status (Pass/Fail)**
1	Open Software Program			Pass
2	Verify Software Startup on Multiple Systems			Pass
3	Verify the Software is Displaying Main UI Properly			Pass

Table 6.1: Software Startup Test Case

6.6.2 Video/Image Load Test Case

The following Table 6.2 shows the entire verification and testing of the software, step by step while loading an image or a video.

Test Case ID	Test Req ID	Description	Applicable for	Initial Conditions
TC02	TR02	Testing Video Image Load	All Desktop Windows Systems	Software Startup Successful
Step	**Task & Expected Results**			**Status (Pass/Fail)**
1	Select either the video or image options (buttons)			Pass
2	Verify the options let you browse for an image or video			Pass
3	Verify status displaying 'Processing Video/Image'			Pass

Table 6.2: Video/Image Load Test Case

6.6.3 Criminal Detection Test Case

The following Table 6.3 shows the entire verification and testing of the software, step by step while criminal action is being detected. Test results have also been shown in Figure 6.5.

Test Case ID	Test Req ID	Description	Applicable for	Initial Conditions
TC03	TR03	Testing Criminal Detection	All Desktop Windows Systems	Image/Video Selected Successfully
Step	**Task & Expected Results**			**Status (Pass/Fail)**
1	Verify status displaying 'Detecting Criminal'			Pass
2	Verify Result being Displayed on the Screen			Pass
3	Verify Criminal Activity is being Detected			Pass

Table 6.3: Criminal Detection Test Case

Figure 6.5: (a) Original Image, (b) Software Result

6.6.4 Second Criminal Detection Test Case

The following Table 6.4 shows further testing which was done to check the causes due to dataset limitation. Test results have also been shown in Figure 6.6.

Test Case ID	Test Req ID	Description	Applicable for	Initial Conditions
TC04	TR04	Testing Criminal Detection	All Desktop Windows Systems	Image/Video Selected Successfully
Step	**Task & Expected Results**			**Status (Pass/Fail)**
1	Verify Label of The Detected Activity			Pass
2	Verify Result being Displayed on the Screen			Pass
3	Verify Criminal Activity is being Detected			fail

Table 6.4: Second Criminal Detection Test Case

Figure 6.6: (a) Original Image, (b) Software Result

6.7 Datasets

Dataset was a major challenge for this project. As our project was based on a deep learning model so, it was data hungry and needed as much data as possible to perform better. Deep learning requires a lot of training data because of the huge number of parameters needed to be tuned by a learning algorithm. [33].

6.7.1 Custom Dataset

The problem was that the data suited for our system was not for free, and the free data was not sufficient. So, data was collected from various sources [32-36] and made our own custom collected dataset. The total dataset that was collected for images was 12,226 out of which there are 6113 Gun images, as shown in Figure 6.7, and 6113 Not Gun images, as shown in Figure 6.8. For videos, 68 videos were collected that contained frames for both Gun and Not Gun, as shown in Figure 6.9.

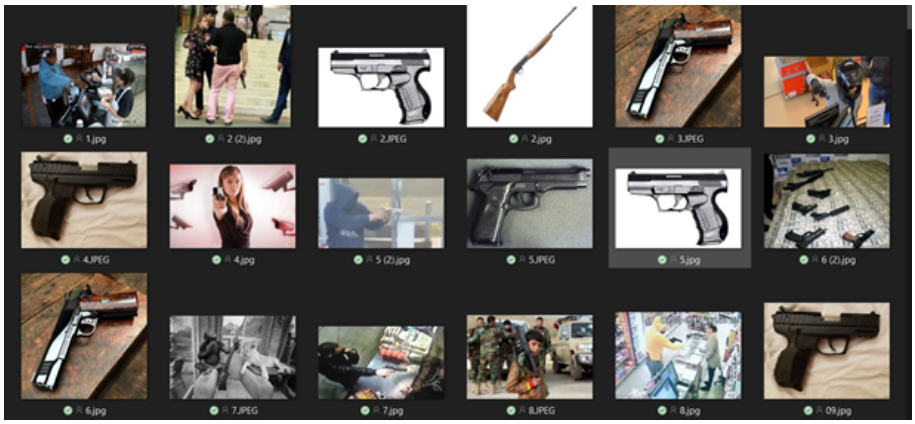

Figure 6.7: Gun Images

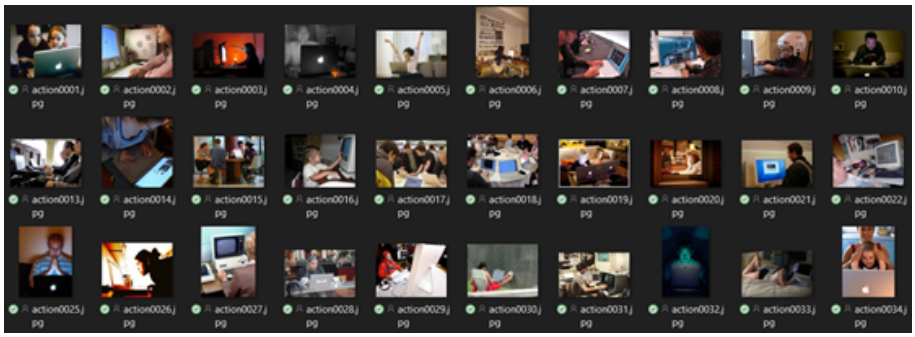

Figure 6.8: Not Gun Images

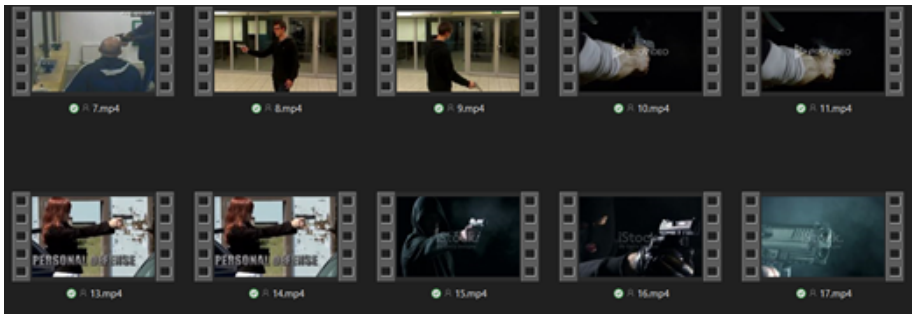

Figure 6.9: Videos

6.8 Detail Analysis and Results

Our detailed evaluation metric is comprised of accuracies calculated through various models used during the development and testing phase of our project. The model with highest acceptable accuracy is selected as a base model and was further tested using different parameters in-order to improve it more. All these tests can be seen in our evaluation metric. As mentioned earlier, a custom dataset collected from different sources was used. There are two types of datasets, one with images and the other one with videos. All the model's testing was done on both datasets. In case

of images, they were simply pre-processed (resized, augmented etc.) and split into train (70%) test (30%). After that they were subjected over to the model for training. Whereas in the case of videos, first their frames were extracted, then each frame was pre-processed. After frames were pre-processed, they were split into test train and subjected to the model for training. The models used for training were CNN-4 layers, CNN-6 Layers and Alex-Net. All these datasets were tested on all these models with different parameters and their results are shown below in a tabular form.

6.8.1 Image Dataset

Image dataset was tested on all the models, the best model was further tested using different parameters given in a tabular form.

6.8.1.1 Alex-Net

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	99%	99%
10	28	0.001	99%	99%
20	32	0.01	99%	99%
5	**68**	**0.000000001**	**92.20%**	**91.35%**
10	68	0.0000001	99%	99%

Table 6.5: Conventional Alex-Net Architecture with activation function Relu

As the results above clearly show, that Alex-Net is over fitting at any given parameter except for one, which is having a learning rate of 10e-9. These parameters are not ideal for any model. So, we decided to move to another simpler architecture. Following is the graphical representation of Accuracy and Loss for the ideal model:

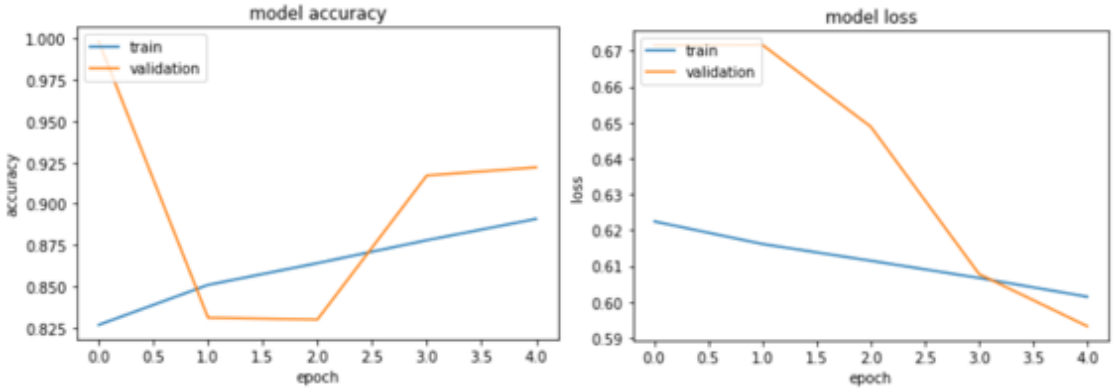

Figure 6.10: Model accuracy and loss graphs for Alex-Net

6.8.1.2 CNN-6 Layers

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	99%	99%
10	28	0.0001	99%	99%
20	32	0.001	99%	99%

Table 6.6: Three Convolutional Layers and three Max-pooling Layers with activation function Relu

The results of CNN model with 6 layers are also over fitting and gives a 99% accuracy which is by no means acceptable. It is clear that increased layers of CNN are causing the over fitting. So, it is time to move to a simpler CNN model. Following is the graphical representation of Accuracy and Loss for the model:

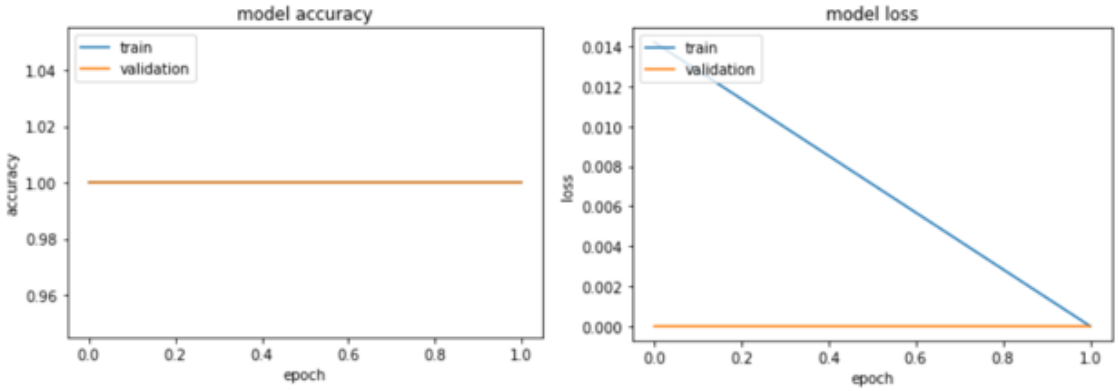

Figure 6.11: Model accuracy and loss graphs for CNN-6 Layers

6.8.1.3 CNN-4 Layers

Learning Rate	Epoch	Batch Size	Test Accuracy	Train Accuracy
0.001	10	18	86.03%	85.01%
		28	88.52%	87.15%
		38	88.52%	85.75%
	30	18	89.89%	88.03%
		28	89.60%	87.77%
		38	90.91%	89.01%
	50	18	90.55%	88.97%
		28	90.91%	89.01%
		38	89.50%	88.19%
		18	89.92%	88.57%
	70	**28**	**91.43%**	**90.52%**
		38	90.84%	90.01%
	90	18	90.06%	89.32%
		28	89.99%	89.12%
		38	91.10%	90.43%
	110	18	90.58%	89.21%
		28	90.01%	88.33%
		38	91.10%	90.32%

Table 6.7: Two Convolutional Layers and two Max-pooling Layers with activation function Relu

The above Table 6.6 shows that reducing the layers and complexity of CNN was a good idea. As the model was not over fitting or under fitting, we tested the model on various parameters to achieve the highest possible accuracy which is 91.43%. Following is the graphical representation of Accuracy and Loss for the ideal model:

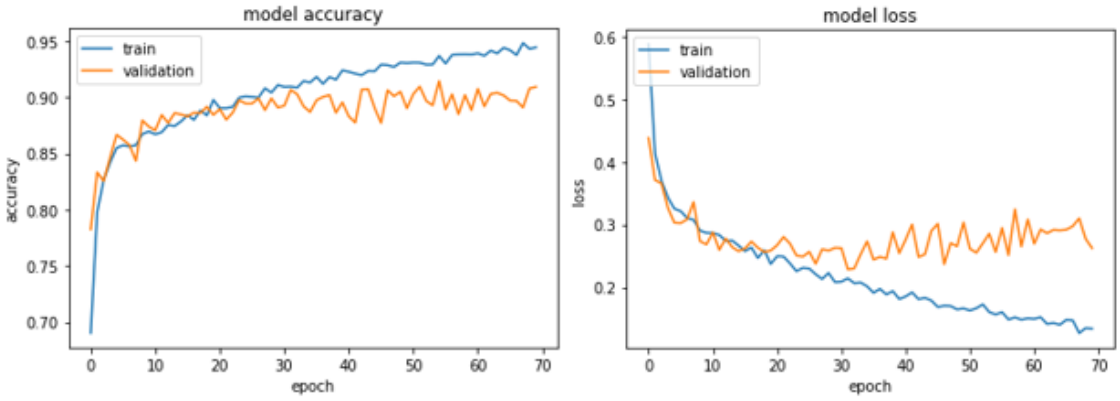

Figure 6.12: Model accuracy and loss graphs for CNN-4 Layers

The results of this model were very promising, so we decided to change some parameters inside the CNN architecture, i.e., changing the activation function to "Hyperbolic Tangent Sigmoid (tanh)". As it is preferable to use differentiable activation functions just as Hyperbolic Tangent Sigmoid Function.

6.8.1.4 CNN-4 Layers with tanh

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	99%	99%
10	28	0.001	99%	99%
20	32	0.00001	99%	99%

Table 6.8: Two Convolutional Layers and two Max-pooling Layers with activation function tanh

The results of the CNN model with activation function (tanh) are clearly over fitting again. So, it was a better idea not to change the activation function from Relu as the results on that were perfect. Following is the graphical representation of Accuracy and Loss for the model:

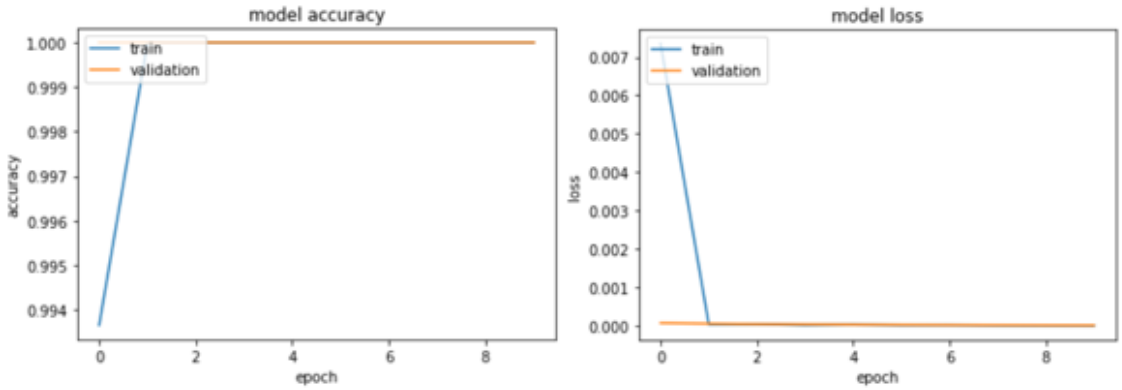

Figure 6.13: Model accuracy and loss graphs for CNN-6 Layers with tanh

6.8.2 Video Dataset

Video dataset [37-39] was also tested on all the models, the best model was further tested using different parameters given below in a tabular form.

6.8.2.1 Alex-Net

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	96.93%	97.07
10	**28**	**0.0001**	**98.24%**	**98.20%**

Table 6.9: Conventional Alex-Net Architecture with activation function Relu

Alex-Net worked very well for our video's dataset set, but we wanted to test the exact same circle we did for images data. So, every model with same parameters was tested on videos dataset. Following is the graphical representation of Accuracy and Loss for the ideal model:

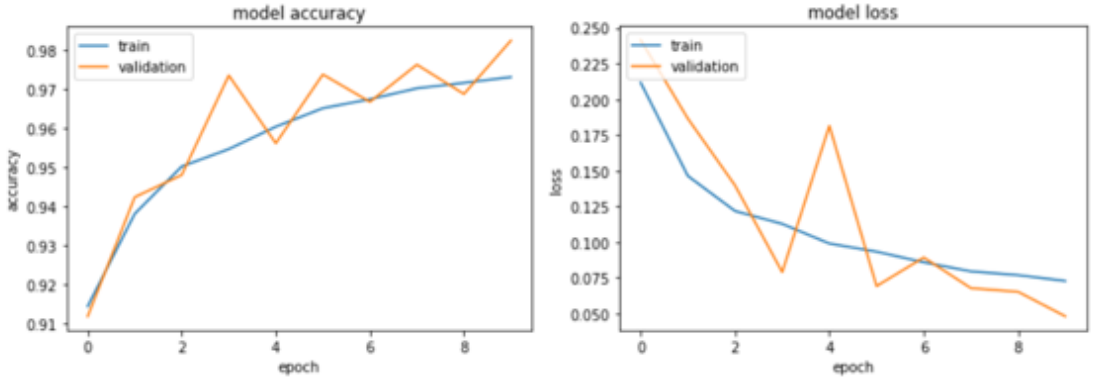

Figure 6.14: Model accuracy and loss graphs for Alex-Net

6.8.2.2 CNN-6 Layers

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	95.01%	94.75%
10	**28**	**0.0001**	**96.84%**	**96.79%**

Table 6.10: Three Convolutional Layers and three Max-pooling Layers with activation function Relu

CNN with 6 layers also worked very well for our video's dataset set. It seems that all the models are working better with the video dataset. It is time to move on. Following is the graphical representation of Accuracy and Loss for the ideal model:

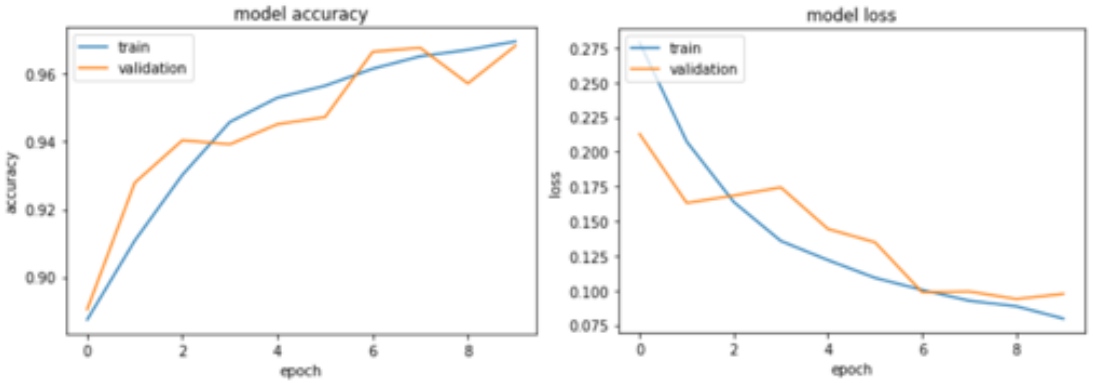

Figure 6.15: Model accuracy and loss graphs for CNN-6 Layers

6.8.2.3 CNN-4 Layers

Epoch	Batch Size	Learning Rate	Test Accuracy	Train Accuracy
5	28	0.0001	94.97%	95.10%
10	**28**	**0.0001**	**97.26%**	**97.32%**

Table 6.11: Two Convolutional Layers and two Max-pooling Layers with activation function Relu

Further testing for video dataset was stopped because it was clear that whatever model is used, accuracy will be very promising. Following is the graphical representation of Accuracy and Loss for the ideal model:

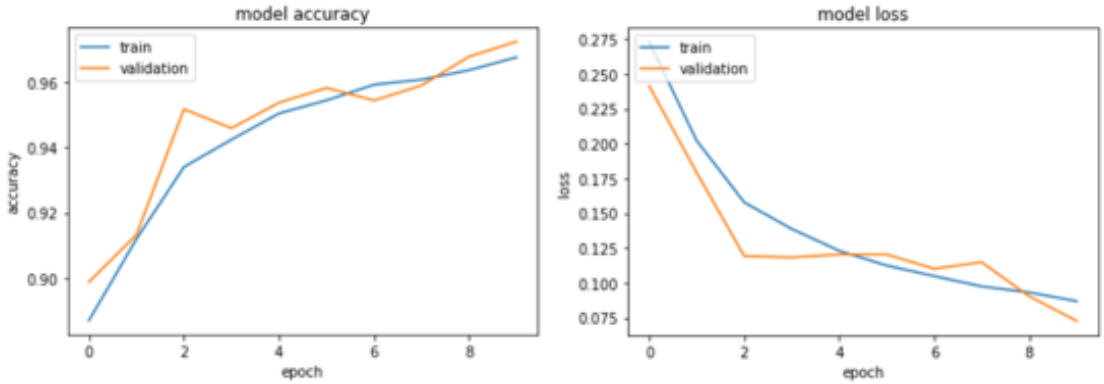

Figure 6.16: Model accuracy and loss graphs for CNN-4 Layers

Hence, our testing and evaluation came to an end with two successful models with promising accuracies for both images and videos.

6.9 Comparison

After detail testing and evaluation, it is time to compare our results with some of the research papers in Table 2.1 that mentioned in Chapter 2 Literature Review. Some papers have used our methodology and some papers have used some of our dataset. So, we have tried to compare our results with them. As no direct comparison can be done due to some modifications but, it will help us to compare our work with the work done by professional researchers.

Research Paper Model	Result of Research Paper	Our Model	Our Result
R-CNN Deep Learning (VGG-16/10)	93%	CNN- 6 Layers	99%
CNN in combination with Alex-Net	99.2%	Alex-Net	92.20%
R-CNN-FPN	91.43%	CNN- 4 Layers	91.43%
CNN	96.74%	CNN- 4 Layers	91.43%
VGG16, RFCNN	98.8%	CNN- 4 Layers	91.43%
VGG16, YOLOv3, FRCNN	93%	CNN- 4 Layers	91.43%

Table 6.12: Comparison with Research Papers

The comparison above shows that our accuracy is approaching their accuracies which is a huge achievement of this project.

Chapter 7

Conclusion

Security and a sense of safety is a prime requirement for all of us. In a day and age where our lives have become so vulnerable, it is our utmost duty to not only stay protected but also provide others with a means to keep themselves protected. This project was intended with these exact thoughts in mind to provide an easy and an interactive software for the security authorities to detect a criminal activity and respond well in time, enough for them to act upon and save the day.
This software is designed to allow authorities to not waste time in manually checking and realizing when and if a criminal activity occurs, also reducing the aspect of human error of neglect and lack of attentiveness. The software can take footage, that can be both, an image, or a video, in the form of an input, process it and give results by determining whether there was any criminal activity detected in the that footage or not based on the presence of a weapon such as a gun. In the case if a criminal activity is detected, the software will give a status in red color stating that *'Criminal Threat Detected'*, by also creating a bounding box around the weapon and potentially the criminal as well. The software was tested under different circumstances and scenarios and was tested with different architectures as well, but Convolutional Neural Network (CNN) proved to work the best by providing the performance that we needed. The performance of the software and its accuracy of detection is also dependent upon the quality of the image or the video and its clarity, for which preprocessing techniques have been implemented, along with the system it is being used on since deep learning algorithms tend to require more power.

7.1 Future Work

The project idea in itself, is incredibly vast with an unlimited potential of further work and features that can be implemented but couldn't be done in the limited time and resources we had. We intend upon providing more features in the future such as being able to detect a criminal activity based on other weapons such as a knife or a taser etc. These sorts of implementations require a huge dataset since deep learning algorithms are data hungry. Similarly, we also intend on making the software

capable enough to detect criminals in real time CCTV footage live as it is happening which requires a lot more resources and processing power to achieve.

7.2 Recommendation

One of the features that we originally had planned on implementing was for our system to differentiate between a criminal and a victim and actually detect and notify who the criminal is and who the victim is. At the moment, we are able to detect a criminal activity in an image or a video, but the system lacks the ability to separate the victim from the criminal which is a top priority task that we will most definitely implement in the future.

7.3 Learning Outcomes

This project made us learn and appreciate the importance of machine and more specifically deep learning techniques and how important they are for us. We got to work on and learn in utmost detail how a deep learning algorithm works and what outcomes we get on tweaking certain aspects of it. This project also gave us a sense of importance of time management and being able to work with what we have. This most certainly will be immensely beneficial for us in the future as we enter the professional world.

Appendix A

User Manual

A user manual helps guide a user to get the basic idea of how to operate a software. Following is the user manual for our software:

A.1 Start Program

The following Figure A.1 shows the main screen layout of the software.

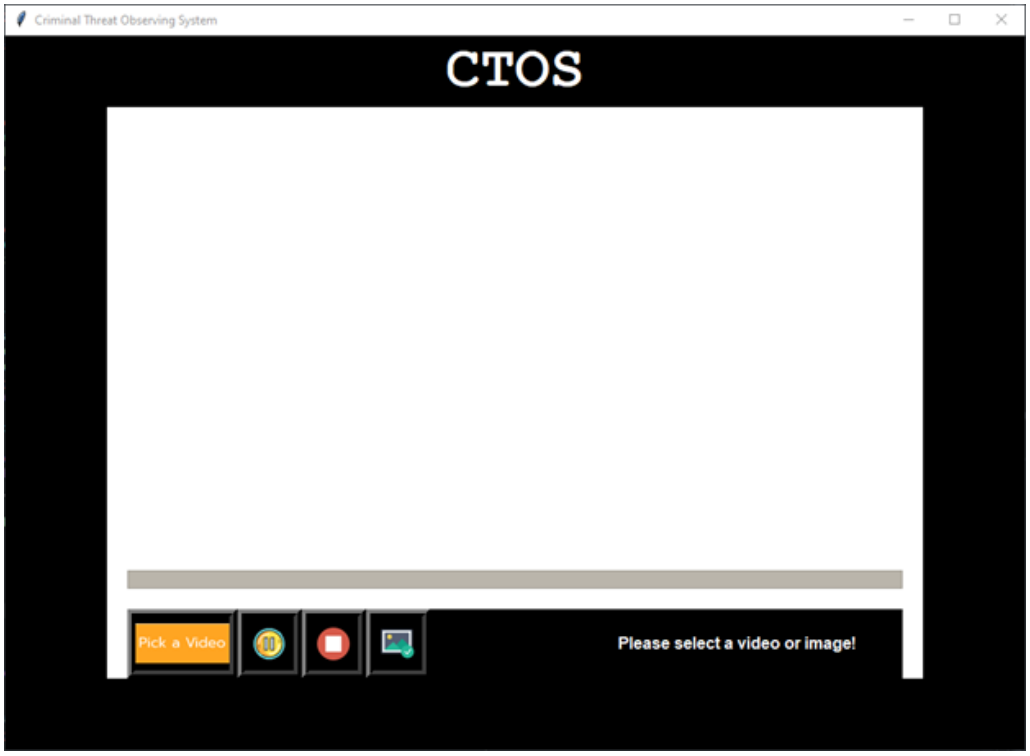

Figure A.1: Main Screen Layout

A.2 Pick a Video

The following Figure A.2.1 shows the browse window that appears for the user to browse and pick a video once the 'Pick a Video' button has been clicked while Figure A.2.2 shows the browse window that appears for the user to browse and pick an image once the button containing the image icon has been clicked.

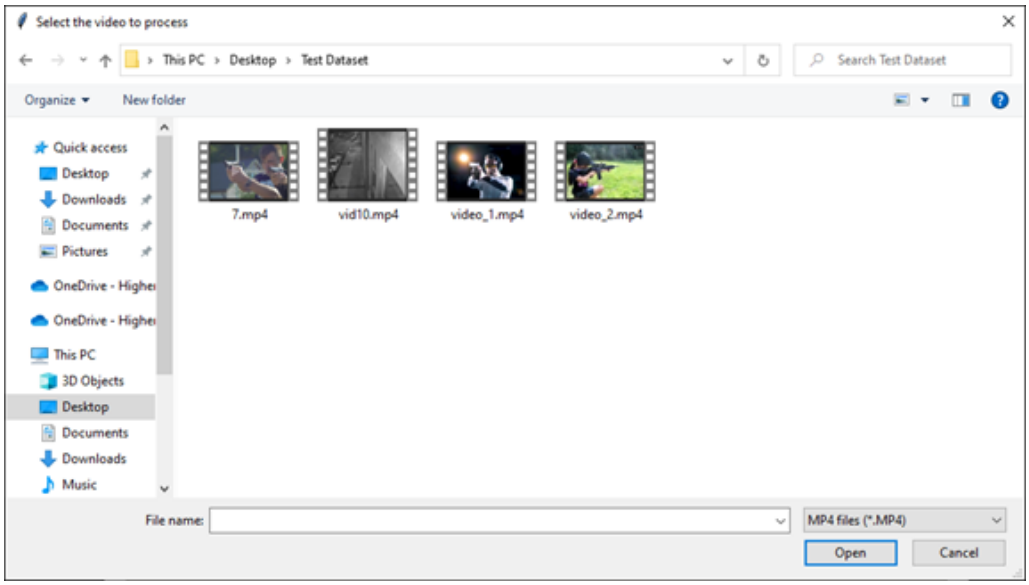

Figure A.2: Browse Window for Images

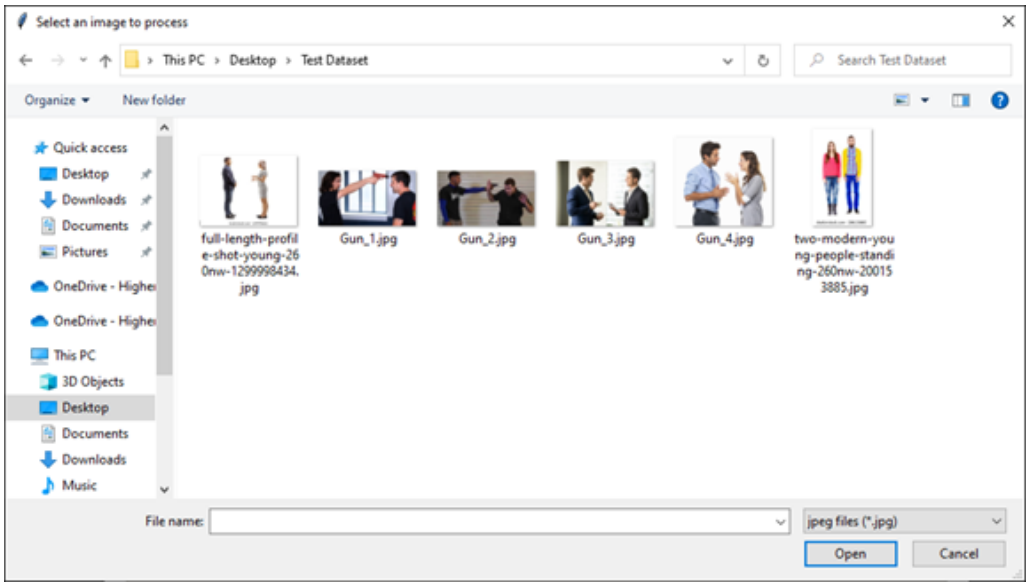

Figure A.3: Browse Window for Videos

A.3 Processing

The following Figure A.3 shows status of the software changing to *'Video Processing'* determining that the selected video is being processed and analyzed by the trained model, frame by frame.

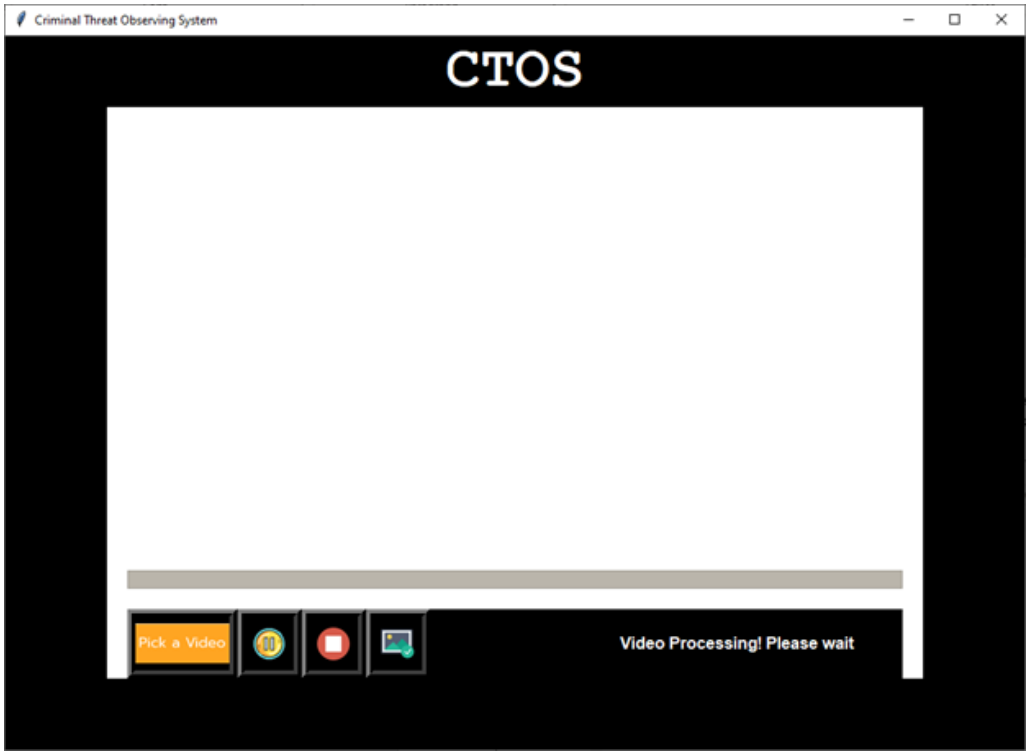

Figure A.4: Video Processing

A.4 Results

The following Figure A.4.1 shows status of the software changing to *'Detecting Criminal'* determining that the video has been processed and the results are being displayed while in the case of an image, the resultant processed image along with the detected results are displayed as shown in Figure A.4.2.

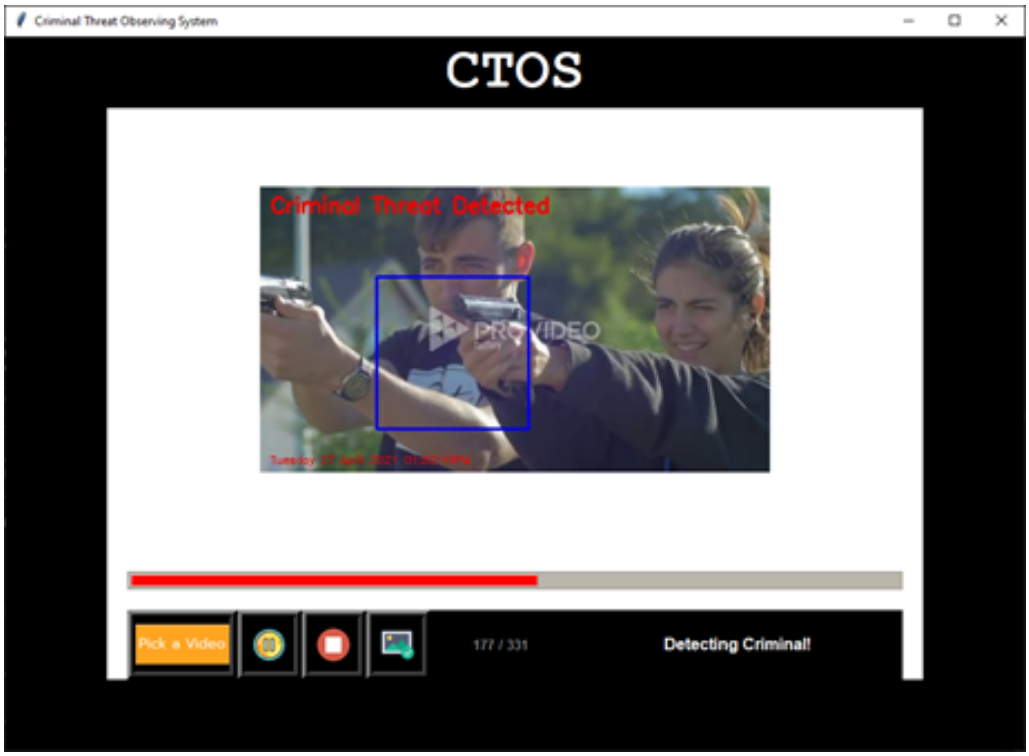

Figure A.5: Results for Videos

Figure A.6: Results for Images

Bibliography

[1] - Internet Web Page: "Surge In Crimes". The Nation. October 15th, 2020. https://nation.com.pk/29-May-2020/surge-in-crimes

[2] - Internet Web Page: "Street crimes in Karachi on the rise as above 9,000 phones, 700 cars stolen in a year". July 15th, 2020. International The News. October 15th, 2020. https://www.thenews.com.pk/latest/687112-street-crimes-in-karachi-on-the-as-above-9000-phones-700-cars-stolen-in-a-year

[3] - Internet Web Page: "Karachi man gets robbed at ATM after leaving door unlocked". February 26th, 2020. Samaa. October 15th, 2020. https://www.samaa.tv/video/2020/02/karachi-man-gets-robbed-at-atm-after-leaving-door-unlocked/

[4] - Research Paper: Rupesh Mandal and Nupur Choudhury. (2016). "Automatic Video surveillance for theft detection in ATM machines: An enhanced approach". New Delhi, India.

[5] - Research Paper: Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). "Understanding of a convolutional neural network". 2017 International Conference on Engineering and Technology (ICET).

[6] - Internet Web Page: "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms". Towards data science. July 10th, 2018. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[7] - Rybski, P. E., Huber, D., Morris, D. D., & Hoffman, R. (2010). Visual classification of coarse vehicle orientation using Histogram of Oriented Gradients features. 2010 IEEE Intelligent Vehicles Symposium.

[8] - He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9), 1904–1916.

[9] - Internet Web Page: "Review: SSD — Single Shot Detector (Object Detection)". Towards

data science. November 3rd, 2018. https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

[10] – Research Paper: Tiwari, R. K., & Verma, G. K. (2015). A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector. Procedia Computer Science, 54, 703–712.

[11] – Research Paper: Verma, G. K., & Dhillon, A. (2017). A Handheld Gun Detection using Faster R-CNN Deep Learning. Proceedings of the 7th International Conference on Computer and Communication Technology - ICCCT-2017.

[12] - Research Paper: El den Mohamed, M. K., Taha, A., & Zayed, H. H. (2020). Automatic Gun Detection Approach for Video Surveillance. International Journal of Sociotechnology and Knowledge Development, 12(1), 49–66.

[14] - Internet Web Page: "Basic classification: Classify images of clothing". October 15th, 2020. TensorFlow. October 15th, 2020. https://www.tensorflow.org/tutorials/keras/classification

[15] - Internet Web Page: Parth Mehta. "Fire-Gun". March 18th, 2020. Kaggle. October 15th, 2020. https://www.kaggle.com/parthmehta15/fire-gun

[16] - Internet Web Page: Anton Krylov. "Gun-Videos". October 21st, 2019. Kaggle. October 15th, 2020. https://www.kaggle.com/akrylov/gun-videos Real-Time Action Detection in Video Surveillance using Sub-Action Descriptor with Multi-CNN

[17] - Internet Web Page: "Keras". Keras. https://keras.io/

[18] - Research Paper: Tiwari, R. K., & Verma, G. K. (2015). A computer vision based framework for visual gun detection using SURF. 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO).

[19] - Research Paper: Bhavna Khajone & Prof. V.K. Shandilya. (2012). Concealed Weapon Detection Using Image Processing. 2012 International Journal of Scientific & Engineering Research, Volume 3, Issue 6.

[20] – Research Paper: W. Sultani, C. Chen, and M. Shah, "Real-world Anomaly Detection in Surveillance Videos," tech. rep., 2018.

[21] – Research Paper: Lim, J., Al Jobayer, M. I., Baskaran, V. M., Lim, J. M., Wong, K., & See, J. (2019). Gun Detection in Surveillance Videos using Deep Neural Networks. 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC).

[22] – Research Paper: Salazar González, J. L., Zaccaro, C., Álvarez-García, J. A., Soria Morillo, L. M., & Sancho Caparrini, F. (2020). Real-time gun detection in CCTV: An open problem. Neural Networks, 132, 297–308. doi:10.1016/j.neunet.2020.09.013

[23] – Research Paper: Warsi, A., Abdullah, M., Husen, M. N., Yahya, M., Khan, S., & Jawaid, N. (2019). Gun Detection System Using Yolov3. 2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA).

[24] – Research Paper: Jain, A., Aishwarya, & Garg, G. (2020). Gun Detection with Model and Type Recognition using Haar Cascade Classifier. 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT).

[25] – Research Paper: Justin Lai & Sydney Maples. (2017). Developing a Real-Time Gun Detection Classifier. Stanford University (2017).

[26] – Research Paper: Mehmet Tevfik Ağdaş, Muammer Türkoğlu and Sevinç Gülseçen. (2021). Deep Neural Networks Based on Transfer Learning Approaches to Classification of Gun and Knife Images (2021).

[27] – Research Paper: Gulzar Ahmad, Saad Alanazi, Madallah Alruwaili, Fahad Ahmad, Muhammad Adnan Khan, Sagheer Abbas and Nadia Tabassum (2021). Intelligent Ammunition Detection and Classification System Using Convolutional Neural Network (2021).

[28] – Research Paper: N.R.Sathis Kumar and T.Nirmalraj (2020). Detection of Suspicious Activity in ATM Using Deep Learning (2020).

[29] – Research Paper: Javed Iqbal, Muhammad Akhtar Munir, Arif Mahmood, Afsheen Rafaqat Ali, Mohsen Ali (2021). Leveraging Orientation for Weakly Supervised Object Detection with Application to Firearm Localization (2021).

[30] – Research Paper: Jiahao Li, Charles Ablan, Rui Wu, Shanyue Guan, and Jason Yao (2021). Preprocessing Method Comparisons for VGG16 Fast-RCNN Pistol Detection (2021).

[31] – Research Paper: Muhammad Tahir Bhatti, Muhammad Ghufran Khan, Masood Aslam, and Muhammad Junaid Fiaz (2021). Weapon Detection in Real-Time CCTV Videos Using Deep Learning (2021).

[32] - Internet Web Page: Atulya Kumar. "Gun Detection Dataset". 2020. Kaggle. April 28th, 2020. https://www.kaggle.com/atulyakumar98/gundetection

[33] - Internet Web Page: "Why does deep learning require a lot of data". Quora. April 28th, 2020. https://www.quora.com/Why-does-deep-learning-require-a-lot-of-data

[34] - Internet Web Page: Sasank Yadati. "Gun-Dataset". GitHub. April 28th, 2020. https://github.com/SasankYadati/Gun-Dataset

[35] - Internet Web Page: Sai Sasank. "Guns Object Detection". 2019. Kaggle. April 28th, 2020. https://www.kaggle.com/issaisasank/guns-object-detection

[36] - Internet Web Page: Larxel. "Handgun Detection". 2020. Kaggle. April 28th, 2020.

https://www.kaggle.com/andrewmvd/handgun-detection

[37] - Internet Web Page: “Gun Firing”. 2021. Pro Video Factory. April 28th, 2020. https://provideofactory.com/videos/gun-
firing-two

[38] - Internet Web Page: “Criminal Holding Gun”. 2021. iStock. April 28th, 2020. https://www.istockphoto.com/search/2/

[39] - Internet Web Page: “Criminal Holding Gun”. 2021. Videvo. April 28th, 2020. https://www.videvo.net/search/crimina