Real-Time Crime Detection Using Convolutional LSTM and YOLOv7

by

Cyrus Sakiba
19101512
Syeda Maisha Tarannum
19101178
Farzana Nur
19101480
Fahad Faisal Arpan
22341070
Ahnaf Ahmed Anzum
22341086

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
May 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Cyrus

Cyrus Sakiba
19101512

Maisha

Syeda Maisha Tarannum
19101178

Farzana

Farzana Nur
19101480

Arpan

Fahad Faisal Arpan
22341070

[illegible]

Ahnaf Ahmed Anzum
22341086

Approval

The thesis/project titled "Real-Time Crime Detection Using Convolutional LSTM and YOLOv7" submitted by

1. Cyrus Sakiba (19101512)
2. Syeda Maisha Tarannum (19101178)
3. Farzana Nur (19101480)
4. Fahad Faisal Arpan (22341070)
5. Ahnaf Ahmed Anzum (22341086)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 22, 2023.

Examining Committee:

Supervisor:
(Member)

Md. Khalilur Rhaman

Dr. Md. Khalilur Rahman
Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The principal goal of this study is to create a crime detection system in real-time that can effectively handle closed-circuit television (CCTV) video feeds and evaluate them for possible criminal occurrences. The system's goal is to improve public safety by offering an advanced approach that makes use of ConvLSTM's expertise in modeling temporal dynamics and YOLO v7's expertise in object recognition. We suggest a posture and weapon recognition system that can be applied to real-time videos. The first method proposes the utilization of ConvLSTM for the detection of violent postures. The Conv part is derived from MobileNet v2, while a bi-directional LSTM technique is used. MobileNet v2 was chosen for its superior accuracy and efficiency as a result of its lightweight architecture. The model will be trained to recognize illegal behavior by being exposed to annotated datasets of surveillance videos that depict different types of crime. The output of the system distinguishes between violent and non-violent postures in real-time videos. The system identifies violent postures as kicking, collar grabbing, choking, hair pulling, punching, slapping, etc., while identifying non-violent postures as hugging, handshaking, touching shoulders, walking, etc. We used the real-time violence and non-violence dataset from Kaggle. The second method uses YOLO v7 to detect weapons in three categories, e.g., sticks, guns, and sharp objects. The YOLO v4 was also employed for the aforementioned objective; however, the YOLO v7 yielded superior outcomes, hence it was chosen for further implementation. We customized the weapons dataset to enable our model to accurately detect local Asian weapons like machetes and sticks. The system's intended use is to prevent illegal acts using two distinct machine learning models in a seamless way.

Keywords: Machine Learning, Deep Learning, Bidirectional LSTM, YOLOv7, YOLOv4, MobilenetV2, Violence Prediction, Realtime

Acknowledgement

First and foremost, praise be to the Almighty Allah for His blessings and guidance throughout the completion of this thesis. We would also like to thank our supervisor, Dr. Md. Khalilur Rahman sir, for his guidance and patience. We are grateful for his support. We are grateful to our mentor Sayantan Roy Arko for being understanding and guiding us. Finally, we would like to thank our parents for their love and support. They have always been there for us, and we are grateful for their encouragement. We could not have done this without them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The absence of preventive measures to apprehend perpetrators during an increase in crime rates represents a problematic scenario. Cities and major metropolitan regions typically have surveillance systems in place that collect data on an ongoing basis. The abundance of surveillance information increases the likelihood of criminal behavior. These tasks are far too complicated and resource-intensive for artificial intelligence to handle; hence, they require human monitoring for detection. It is possible to automate formerly complicated processes by breaking them down into smaller, more manageable ones and looking for any underlying patterns that would indicate the presence of criminal intent. Our models attempt to identify two broad categories of criminal behavior. Our system utilizes real-time video and image analysis to identify instances of criminal activity by using ConvLSTM, a recurrent neural network that employs convolutional architectures in both the input-to-state and state-to-state transitions for spatiotemporal prediction. The ConvLSTM algorithm incorporates the inputs and states of adjacent cells in order to accurately predict the condition of a given grid cell. This system is capable of detecting violent postures and weapons, enabling it to accurately identify potential threats. The proposed system uses the deep learning mode YOLO v7 to identify potentially lethal objects, such as a gun or knife, held by a person who is threatening another. The YOLOv7 object identification model is a single-stage approach that has attained outstanding performance on several object detection benchmarks. In addition to being a vast improvement over prior YOLO models, its speed makes it a viable option for use in real-time object identification software. We are able to identify crime scenes where weapons have been used by using data from a wide variety of weapons. This data allows us to identify crime scenes where weapons have been used. The approach also places an emphasis on identifying anomalous conduct, which includes behavior that can be interpreted as aggressive behavior. Rather than relying solely on human crime identification and resource allocation processes, we suggest a real-time crime-detecting technology to aid law enforcement and boost public safety.

1.2 Problem Statement

Closed-circuit television (CCTV) systems are commonly used to monitor areas for signs of criminal activity. Crime rates have not decreased despite the widespread use of closed-circuit television cameras to monitor the area. This is because surveillance cameras necessitate human oversight, which can lead to human error such as the omission of significant criminal events by people while monitoring numerous CCTV displays at once. There are a wide variety of algorithms and models available for criminal identification. However, the study of recognizing violent actions is still in its early stages. Not all of them are done together for the common goal of catching real-world criminals with the aid of surveillance cameras.

Using CONV LSTM and MobileNet v2, we have improved the accuracy of our posture recognition results. The Deep Learning YOLO method has been widely recognized as the most effective way for detecting objects. Its primary function is to deter criminal activity by identifying individuals who might be carrying weapons. Deep learning YOLO ideas have been widely accepted because of the benefits they provide in terms of saving time, memory, and other resources like the CPU and processors. Additionally, YOLO framework models can deliver more accurate results than Transfer Learning models that are explicitly built from scratch. Our primary research area is posture detection, which can identify potentially dangerous body language and actions. Some existing approaches include the predict-update method, where information is retrieved from every body part and the segmentation process has to be precise. Some give false results or cannot identify the crime properly. We propose a system that combines weapon identification and posture recognition to greatly improve accuracy. It will also enhance accurate detection and decrease false detection to a great extent.

Chapter 2

Related Work

2.1 Violent Posture Detection

This section thoroughly analyzes some previous research papers in the domain we are working on and gives vast ideas about the domain. In [2], photographs were collected from drone surveillance systems and used to identify violent individuals by using the feature pyramid network (FPN), the ScatterNet hybrid deep learning network, and the support vector machine (SVM) to estimate the individuals' violent behavior. As part of the study, a few datasets for the detection of violent acts in public gatherings (Aerial Violent Individual (AVI) DSS framework, Common Object in Context (COCO)) are presented. Stick figures in various colors are used to symbolize each individual after they have been identified in order to make it easier to determine who is to blame for what. Non-violent and violent actions are identified using green and red borders in CNN picture categorization.

The study [6] proposes the implementation of e-police systems based on artificial intelligence. This paper presents a system that consists of two main components: a learning-based automated surveillance notification system and a machine learning-based crime prediction system. The video surveillance monitoring systems contain an abnormal behavior identifier and an unidentified human identifier. A classification process for behavior involves a series of steps. This paper focuses on the extraction of frame-level features. The extraction of the output from the final pooling layer, which retains important features while reducing dimensionality, is achieved through the utilization of pre-trained, cutting-edge models like VGG16, InceptionV3, and ResNet-50. The final behavior is classified through the utilization of an LSTM network, which employs a set of extracted high-level feature maps. This thesis presents the development of thick CNN models for suspicious human recognition through the use of variable numbers of layers. These models utilize techniques such as feature extraction from resources, blurring, sharpening, edge detection, noise reduction, and learning specific characteristics of an image. The models were developed from scratch and demonstrate promising results in the field of suspicious human recognition. The model utilizes frame level feature extraction for the purpose of identifying the presence of abnormal expression. The utilization of both regression and classification techniques is common practice. This paper discusses various techniques for data classification, namely support vector machines (SVMs), division trees (DTs), random forests (RFs), and logistic regression (LR). The system utilizes collected data to predict criminal behavior and map out high-risk areas.

A neural network solution using the Hybrid Deep Learning (HDL) algorithm, Deep Convolutional Neural Networks (DCNN), and Recurrent Neural Networks (RNN) is suggested in this paper [3]. A video set can be used in HDL to extract frame data that can be used to model various behavioral patterns, such as crowd movement and facial features. RNNs are used to teach computers how to recognize objects and people. Research aims to find a system with a low computational cost and a high accuracy rate. High-performance features can be extracted from each frame of video using HDL, a facial recognition technology. When dealing with nonlinear problems, such as tracking and detecting objects in crowds, DNN is the method of choice. DCNN uses the multiplayer perception model for face recognition. RNN is for extracting the temporal actions of a person, which may then be checked against other obtained data.

This paper [12] suggested a new approach named Deep Crime based on spatiotemporal CNNs. Malicious activity in video sequences will be detected with this tool. Movement in restricted areas, throwing objects, hiking, and walking in the opposite direction are some of the abnormalities that can be found on public sidewalks. There have also been fights, battles, shootings, assaults, vandalism, thefts, robberies, and arrests. New York City subway and U.S. National Railways (UMN) data sets are used for abnormal activity detection. The strongest dataset for detecting abnormalities in the current world comes from UCF. There are a total of 13 recordings about actual anomalies in this collection. For violence detection, handcrafted feature extraction and the Hierarchical Hidden Markov Model (HHMM) are used. In addition to HOG, HOMO, LSTM, and BD-LSTM, STIP, SFIT, and MOSFIT were used for brute force detection. For the anomalies described, multi instance learning is applied. Deep neural network training in this scenario is limited to 32 blocks with 16 frames. When a single malicious packet frame is identified in a block, the entire block is flagged as malicious. It has a classification accuracy of only 28 percent. Furthermore, because of the slow rate of picture processing, it is impractical for real-time applications.

According to the paper [8], surveillance cameras can be used to both suspect and trace any organization's criminal activity. A centralized computer collects video data, scans the data for single frames, processes the frames to generate single images, and compares the resulting images to a trained and stored dataset. By comparing footage from various cameras, suspicious activities are suspected simultaneously. Classified photos of criminal activity are stored in a crime database that serves as a repository. (CNN provides the classification.) Frame extracting software is used to extract data. The algorithms employed in the process are stated below: 1. Extraction of the video frames from the camera's video file 2. A database of crime photos and their classifications can be stored. 3. Suspicious activity can be seen in the photographs that have been taken of the scene. 4. A Centralized System for Crime Prevention and Control In the study, the proposed model was not used, but the researchers were able to show that the system could be used and that it was important.

This paper's [22] major purpose is to employ gait speed monitoring to study people's behavior through video surveillance. The MM track method is applied in this situation to complete the job. Using video data, this technology is typically used to track pedestrians. Here, the spatial coordinate module and the fixed coordinate system module are applied to extract the gait characteristics. Based on the speed profile, spatial analysis is applied to measure the suspicious actions of pedestrians. A fixed coordinate is also applied to measure the suspicious behaviors of pedestrians based on their list of components and axis. For research objectives, walk ratio and acceleration autocorrelation (AAC) are also used. This proposed module's effectiveness was examined by comparison with the RealBoost approach and the Deformable Part Model (DPM). This proposed module beats the DPM and real boost techniques in both gait speed detection time (PDT) and true positive rate. Another purpose of this study is stationary crowd analysis. According to the walk ratio (WR), pedestrians were separated into two categories: those who walked more slowly than the threshold value and those who crossed it. We borrowed the principle of identifying people in crowded places who want to commit crimes by evaluating their posture and gestures using the media pipe structure from this paper.

The application of the C3D Model (3D convolutional neural network) for crime detection in surveillance films is the dominant issue of this paper [11]. The C3D model was able to successfully recognize some of the events, but at one point it did raise a false alarm since it couldn't capture the context that was continuous from frame to frame. The proposed model also made use of the UFC-Anomaly Detection Dataset and was constructed using a semi-supervised learning strategy. Here, two-stage techniques were applied for research reasons. In the first stage, video features were collected and then used as inputs in the second model, a conventional, fully connected neural network. The AUC-ROC Curve was utilized to analyze the model's performance, and the AUC-ROC Curve was used to evaluate the model's performance, with ROC conducting the probability assessment and AUC performing the separability evaluation.

This paper [18] focuses on automating the process of feature extraction from CCTV images by utilizing the process of feature extraction from CCTV images. The created architecture of this research is able to manage issues including cluttered scenes, changes in illumination, shadows, and reflection, as well as variations in appearance and partial occlusions. The framework is capable of generating detection and tracking results at a rate of four frames per second. A state-of-the-art object recognition framework called Faster R-CNN is utilized to distinguish pedestrians in each frame of CCTV photos. The proposed framework points out issues such as partial occlusion, variations in illumination, changes in stance, form, and scale of pedestrians, crowded backgrounds; and total occlusions for short times. The framework is not able to manage entire occlusions of long durations and fails to address the problem of having similar people in the same frame. A new algorithm has been created for making the connection between the detections over numerous frames. Variations in illumination, busy backdrops, partial occlusions, and changes in scale are all issues that the detector can overcome. The system can follow pedestrians with 71.13 percent accuracy and addresses the issue of changes in appearance (position and shape) as well as total occlusions for short periods of time.

The aim of this study is to propose a method for classifying people's posture by integrating and enhancing various evaluated techniques of human posture analysis through a machine learning phase to model the posture features. The utilization of certain features is implemented in a statistical classifier with the aim of forecasting human posture. The chosen method for supervising the tracking and classification process and generating alarms, is through the use of a finite state chart. This paper presents a method for extracting and tracking human bodies in a movie using the Sakbot system, which is a statistical knowledge-based object detection system capable of recognizing and tracking MVOs. The present study employs PMFC. The pre-classifying step is utilized to differentiate between tracks of individuals and non-individuals after each MVO has been tracked. This thesis explores the process of identifying and analyzing the posture of individuals through the use of MVOs designated as "People Tracks" and the Human Posture Monitoring System (HPMS). The aim is to recognize and prevent unsafe conditions that may arise.

The paper [1] uses the action recognition Bag of Words (BoW) approach of STIP and MoSIFT. The game dataset is used and categorized as fight and non-fight. Here, the histogram intersection kernel (HIK) has been used beside STIP, which resulted in 91.7 percent accuracy compared to the other approaches.

The primary focus of this paper [10] is to extract motion and posture features and subsequently merge them into a comprehensive set of features. This paper presents a novel approach to human body moving target pose detection in the Internet of Things environment using semi-supervised learning. The proposed method utilizes a large amount of unlabeled data and introduces three-layer restriction conditions of time domain, space domain, and data to enable efficient training. The results demonstrate the effectiveness of the approach in accurately detecting human body poses. The improvement of algorithm detection accuracy is achieved through the use of a classifier. The study demonstrates that the proposed approach is highly accurate in both feature extraction and multi-feature fusion. This study proposes a novel approach that achieves a high correct classification rate of up to 95 percent and a low average running time of 1.1 s. Compared to existing literature techniques, the suggested approach demonstrates superior efficiency and accuracy.

This [9] is a unique model and a theorized system to handle violence utilizing deep learning. The model uses CCTV video streams as input and, via inference, determines whether a violent movement is occurring. And the proposed architecture focuses on probability-driven video feed computation, which decreases the overhead of naively calculating for each CCTV video stream. It is a pseudo real time violence detection system that takes a video, whether with audio or without, and somehow alerts when violent activities are detected. This solely focuses on taking inferences from whatever data can be extracted from the video feeds coming from the CCTV networks to one workstation (or, in the case of parallelism, a cluster). The violent detection challenge is tackled using two novel approaches: CNN and LSTM. In further cases, different models of CNN are tested to see which one provides the most accuracy.

2.2 Weapon Detection

Three aspects of crime detection are the topic of this study [20]. The first involves more accurate face identification from CCTV footage using the deep learning library YOLO (you only look once). This document also implies that if a weapon—such as a gun, knife, or other cutting implement—is found in a location under video surveillance, an alarm will activate. However, no detection rates were stated in this article to give readers a comprehensive understanding of the subject. The third component of this study uses CNN to recognize fires in order to prevent accidents.

Gun-based crimes and abandoned luggage on frames are the focus of this paper [5]. Images were analyzed using object detection and feature extraction so that guns could be identified. The detection of weapons using x-ray and infrared imaging is being researched. In the segmented photos, we used color-based segmentation to segregate items, as well as the Harris interest point detector and FREAK to detect weapons. Objects can be accurately identified with the application of image processing and video analysis. Some people use transfer learning to avoid having to re-teach a complex network from scratch. Background subtraction, in which moving items are viewed as the foreground and non-moving objects as the background, is utilized as the first step in the process of detecting abandoned luggage. Several studies, however, have combined the Blob Tracker and the Human Tracker in an effort to reduce the number of false positives. In some cases, finite-state automata miss two occurrences, such as one that is caused by a shadow and another that is caused by a failure in object monitoring. Guler Farrow utilizes a new stationary algorithm to detect moving objects. Both the tracker and the stationary item detector are part of the system, which allows it to quickly discover abandoned objects. Based on a TensorFlow implementation of Faster RCNN and the Inception v2 network for feature extraction, this detection model is applied in this scenario.

This study [7] analyzes the detection of firearms, specifically guns, using a CNN-based algorithm (SSD and faster RCNN) and two unique datasets: pre-leveled photographs and self-created images. Even if the algorithms are competent, their real-time implementations collide in terms of accuracy and speed. The SSD algorithm delivers a faster speed of 0.736 s/frame. In comparison, faster RCNN has a low performance of 1.606 s/frame when compared to SSD. Faster RCNN delivers greater accuracy, with a score of 84.6 percent. While RCNN is faster, SSD only delivers an accuracy of 73.8 percent, which is mediocre. This research [13] uses a Deep Learning Algorithm to analyze CCTV footage in real-time, recognizing criminal faces and objects using the YOLO algorithm and processing photos in real-time at a pace of 45 frames per second. The YOLO technique is employed in this real-time criminal detection system with a mean average precision of 78.3 percent and a final average loss of 0.6 s. For the dataset, 100 high-resolution pictures of criminals and weapons—such as knives, guns, and pistons—were added. The photos were aligned using the YBAT annotation tool. However, due to the disparities in posture, it was challenging for them to identify the offender. And in order to detect any crime scene in addition to object or weapon detection, our system concentrates on posture and gesture recognition rather than the criminal's face. Additionally, MediaPipe is deployed to gain exact findings for posture and gesture detection.

This study [4] addresses the detection of criminality using real-time video or image analysis and notifies the neighboring human supervisor to take further action. The pre-trained deep learning model VGGNet-19, which recognizes the presence of a knife or firearm in the hand of a person aiming at another person, was utilized to develop the system. The VGG19 model was picked as the best match by comparing two distinct pre-trained models, such as GoogleNet InceptionV3, in training. VGGNet19 provides better training accuracy results for crime datasets than GoogleNet's initial model because it leverages both the FRCNN and R-CNN algorithms to localize objects in photos, whereas GoogleNet solely uses the FRCNN technique to categorize entities. The videos and photographs for the dataset were gathered from YouTube and Google, and the acquired dataset was related to robbery, murder, and some illicit behaviors like carrying firearms in locations like ATMs and banks where the use of weapons is strictly outlawed. In order to give the closest authorities specific information about the criminal scenario, their recommended system tries to distinguish firearms as well as the posture and sound effects linked to the crime scene.

The purpose of this study [19] is to integrate two machine learning models that are both operational. Transfer learning on the VGG16 model is used for criminal/suspect face recognition, while the darknet framework and YOLOv4 algorithm are to be employed for weapon identification. About 800 images altogether, including relatives and acquaintances, have been changed to form the data set. A computer vision development program called Roboflow was applied to automatically label the training files for the custom dataset of weaponry, specifically guns. The validation accuracy from the model's training was 98.44 percent, and the training accuracy was 97.66 percent.

Another study [15] compares and contrasts YOLOv5, the current popular version, with YOLOv7, the most recent release. To determine which YOLO version yields superior results in terms of precision, recall, mAP@0.5, and mAP@0.5:0.95, a custom model was trained using both YOLOv5 and YOLOv7. The experimental dataset was created specifically for Remote Weapon Station, and it included 9,779 photos annotated with 21,561 annotations from three different sources: Google's Open Images Dataset, Roboflow's Public Dataset, and a locally gathered dataset. The four categories are as follows: handguns, rifles, knives, and people. When compared to YOLOv5, which achieved a precision score of 62.6%, a recall value of 53.4%, mAP@0.5 of 55.3%, and mAP@0.5:0.95 of 34.2% in its experiments, YOLOv7 achieved a precision score of 52.8%. It was shown that during testing, YOLOv7 had a greater recall value than YOLOv5, but that YOLOv5 performed better in terms of precision, mAP@0.5, and mAP@0.5:0.95. When compared to YOLOv7, YOLOv5 is 4.0% more precise. In order to better detect tiny objects, the HPS-YOLOv7 method is proposed in this research [16]. To address the issue of depth model convergence worsening over time, they presented a tweaked version of a high-efficiency layer aggregation network for feature extraction and introduced a lightweight Bottleneck-structured model processing method. To fully integrate shallow object semantic information, they suggested C-recursively gated convolution, which increases the model's capacity. To compensate for the data loss caused by small objects, a shallow feature fusion network (SFN) was added to the deep convolutional network. The detection head was increased in size from 2020 to 160160. Model training employed mosaic data enhancement and an a priori anchor adaptive adjustment technique to boost detection effectiveness and efficiency. Both the VisDrone2019 and Tinyperson datasets were used in the experimental evaluation. When comparing mAP to YOLOv7 at IoU=0.5, the results showed a 3.0% and 13.29% rise, respectively. The theoretical and practical utility of HPS-YOLOv7's benefits in tiny object detection were demonstrated in this research. Search and rescue operations at sea rely heavily on object detection algorithms; however, the SeaDronesee dataset offers difficulties due to its small targets and considerable interference. To deal with these problems, a new detection system called YOLOv7-sea was suggested in a research paper [17]. They built YOLOv7 [14] by including a prediction head that can spot people and things at the microscopic scale. They also incorporated the Simple, Parameter-Free Attention Module (SimAM) to identify focal points in the environment. They presented additional helpful methodologies, including data augmentation, Test time augmentation (TTA), and bundled box fusion (WBF), that can be used to further enhance their proposed YOLOv7-sea. The AP result of YOLOv7-sea is 59.00% on the ODv2 challenge dataset, which is almost 7% higher than the baseline model (YOLOv7).

The YOLO v4 and v7 algorithms have been utilized in our system to detect various types of weapons, such as guns, knives, and other sharp metal objects, which could potentially be utilized as tools for criminal activities. The utilization of ConvLSTM technology enables the monitoring of hostile gestures and postures exhibited by individuals, in addition to weapon tracking. The ability to identify distortions in various conditions enables the detection of crime scenes and subsequent reporting to authorities.

Chapter 3

Research Methodology

3.1 Proposed Model

Combining the recurrent neural network (RNN) method Bidirectional LSTM with the convolutional neural network (CNN) model MobileNetV2, we were able to develop a real-time violent action detection model. The method is based on the ConvLSTM architecture. We proposed an additional methodology for weapon-based aggressive behavior identification using YOLOv7 to identify and categorize all three types of weaponry from our customized dataset.

The process of identifying violent content in a video involves segmenting the video into individual frames to extract data at the frame level. Subsequently, the detection of human motion within these frames occurs. Afterwards, spatial characteristics are obtained through either manual or automated machine learning techniques. Finally, classification algorithms are employed to categorize the extracted features. The area of computer vision has experienced significant expansion owing to the swift emergence of deep learning algorithms and the accessibility of extensive data and computational capabilities. This paper presents a method for detecting violence in real-time video using the MobileNet v2 architecture.

For proposed methodology for weapon detection is we used YOLOv7. For YOLOv7 model, after annotation, we needed to upload this annotation file to Google Drive, load the dataset on Google Colab, install the YOLOv7 environment and weight file. We chose three classes from among the 80 available; gun, sharp-object, and stick. After that we trained the desired model and tested the inferences.

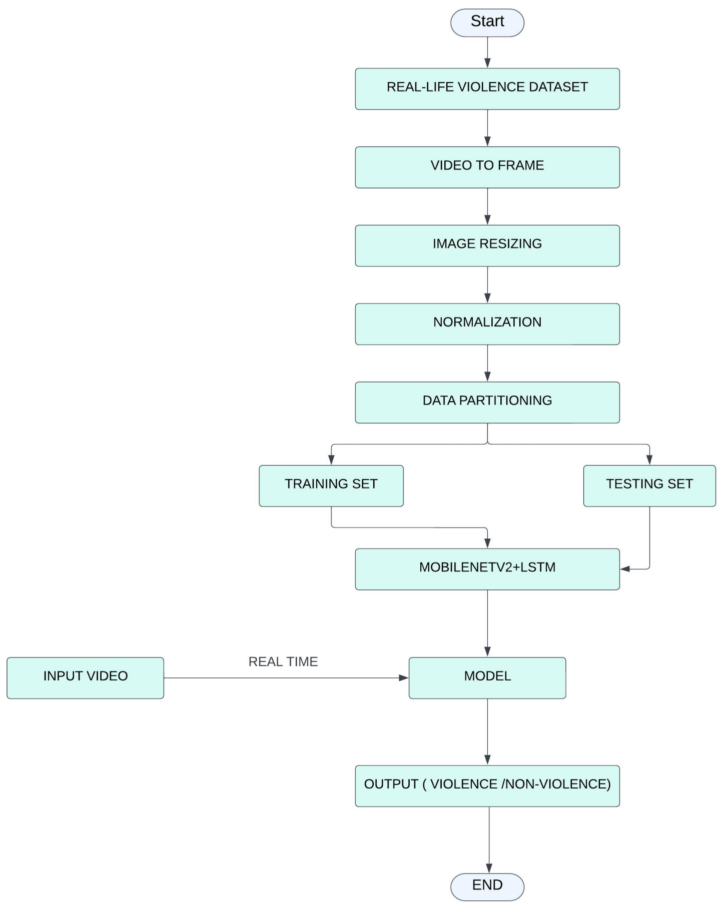


Figure 3.1: Proposed ConvLSTM architecture

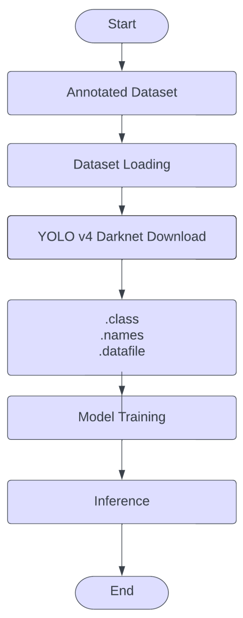


Figure 3.2: YOLOv4 Workflow

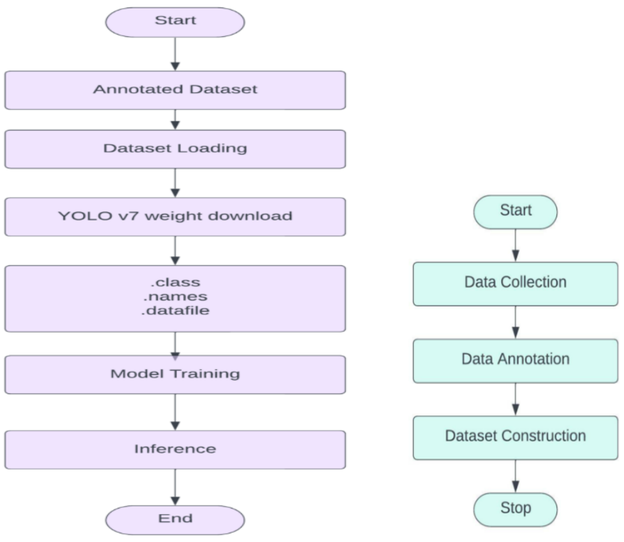


Figure 3.3: YOLOv7 Workflow

3.2 Data Assembling and Pre-processing

3.2.1 Dataset collection

The initial and essential phase of our proposed work, which is real-time crime detection, was to gather and organize data for training and testing the model. We had to collect data from multiple sources because we have concentrated on detecting crime through comparative analysis of local and international weapons. The free data science communities like Kaggle and others were not enough for us; we had to look further. The datasets we used for our research were split into two categories:

- Posture dataset collection
- Weapon dataset collection

3.2.2 Description of Dataset

3.2.2.1 Posture Dataset

We collected our dataset for violence posture detection from Kaggle under the name ”Real Life Violence Situations Dataset” [21], which is a combination of 2000 videos, 1000 of which are categorized as non-violent while the remaining 1000 are violent.

3.2.2.2 Weapon Dataset

We divided our data collection procedure for weapon detection into two subcategories :

- Guns
- Local weapons

  - Sticks
  - Sharp Objects

In order to get this data, we had to carry out a search on Kaggle; however, because we did not have access to any local weapons (such as sticks or sharp things such as machetes (ramda), knives, etc.), we were compelled to acquire it through Google. A total of 1109 images were gathered for our weapon dataset, combined with the gun images obtained from Kaggle under the name ”Guns Object Detection.”

3.2.3 Data Pre-processing

3.2.3.1 Posture Dataset Preprocessing

To train and test our posture detection model, we had to prepare our dataset by going through a few procedures, which included:

1. **Frame Extraction**

   The dataset we obtained from Kaggle consisted of 2000 videos divided into two main classes: violence and nonviolence. Upon importing the requisite libraries in Google-Colab and mounting the video dataset from the drive, the file path was designated for the extraction of frames. Utilizing the openCV video capture module, the video file was read, then we calculated the total number of frames in the video. Subsequently, the interval between the two frames was calculated.

   Next, we checked if it matches our sequence length or not (by sequence length, we meant how many segments we want to extract from each video, and in our case, we set it to be 16 segments/video). If it matches our sequence length, next we set the frame position in the video, and if it's not successful, the loop breaks.
   We proceeded with image/frame resizing after successfully extracting the frames.

2. **Frame Resizing**

   In this step, we fixed our frame dimension to 64 (which means image height = 64 and image width = 64). Resizing the images, especially the decreasing dimension of the actual image, helps in dealing with less noise, and eventually this helps in faster and more precise image processing algorithms. After resizing the frames, we shift to normalizing them, as it is an essential step in CNN.

3. **Normalization**

   In this step, we divide our resized frames by the maximum pixel value, which is 255 for 8 bit images, and this process is known as normalization.
   Normalization of input data is essential because the input values influence the speed at which a network converges during training; if its inputs are adjusted to have a range between 0-1, it converges faster and produces the least amount of error. It contributes to the network's unbiasedness by keeping the pixel value of each frame within the same range.

4. **Array listing**

   After normalizing each frame, we append them into an array with the help of the Numpy library.

5. **Split**

   We split our final dataset array into 80:20 ratios for training and testing data by using train_test_split from the sklearn.model_selection package.

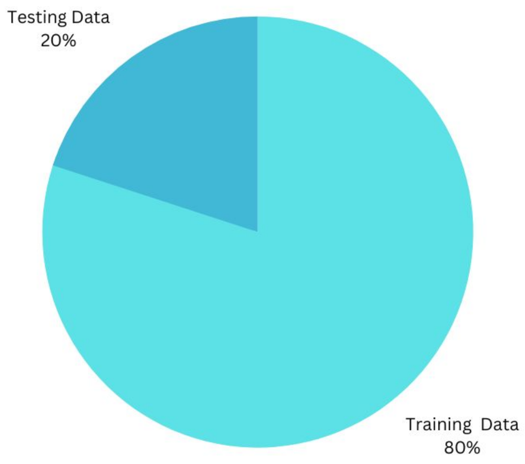


Figure 3.4: Piechart of posture dataset splitting

Table 3.1 shows the tabular format of data pre-processing details for posture data.

Total Data	Classification of Dataset	Freame extraction/ Video	Frame Dimension	Train Dataset	Test Dataset
2000 videos	2 (violence & non-violence)	16	64	80% of total data = 1600	20% of total data = 400

Table 3.1: Posture data pre-processing

3.2.3.2 Weapon Dataset Preprocessing

We had to create our dataset from scratch in order to train and test our weapon detection model, which included:

1. **Data Collection**

   Guns and other local weapons like knives, sticks, and machetes are included in the images we gathered for our weapon detection model. The images of the

guns were taken from Kaggle,we contributed a few of the knife photos, and Google was used for the rest.
The search was discontinued upon collecting approximately 1109 images due to a significant number of redundancies within the dataset. Subsequently, we transitioned to the practice of annotating.

2. **Data Annotation**

   In order to train the model with the same file format prior to annotation, we manually converted our collected images to JPEG.
   The process of annotation is considered necessary for tasks involving supervised machine learning. The learning process of the model is based on the analysis of examples, which subsequently determines its efficacy.
   The Labellmg annotation tool was employed to label our image dataset using the bounding box concept, which is a widely used technique in object detection tasks. A bounding box is a rectangular box that specifies the location of target objects.
   Since our model for detecting objects is based on the YOLO technique, we employed the YOLO labeling format, which generates a .txt file in the same directory as each image that describes the object's class/label, coordinates, and dimensions.
   The image dataset was partitioned into three distinct categories and subsequently classified into three classes, namely guns (labeled as 0), sticks (labeled as 1), and sharp objects (labeled as 2). The knives and machetes in our data set were stored in the "sharp objects" category.

3. **Allocation for YOLOv4 and YOLOv7 Models**

   After annotation, we kept the label files and images in the same directory for utilizing the data in YOLO v4, but for v7, we had to keep them in a different directory.

4. **Split**

   In the weapon detection model, we split our dataset by a ratio of 70:10:20 for training:validation:testing.

   Here, we used 10% validation data to check the dataset's accuracy by removing overfitting-related data errors.

Table 3.2 shows the tabular format of data pre-processing details for weapon data.

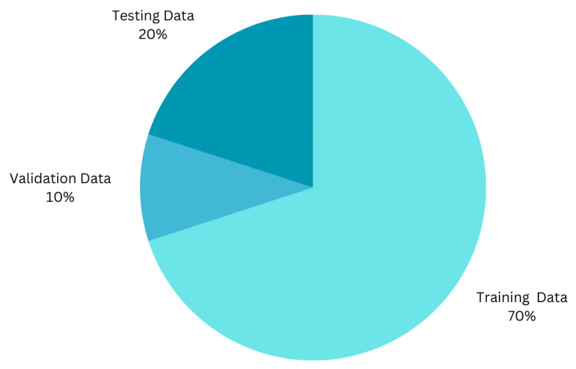


Figure 3.5: Piechart of weapon dataset splitting

Total Data	Classification of Dataset	Train Dataset	Validation Dataset	Test Dataset
1109 images	3 (gun, sharp object & stick)	70% of total data = 776	10% of total data = 111	20% of total data = 222

Table 3.2: Weapon data pre-processing

3.3 Violent and Non-violent Posture Detection using ConvLSTM

3.3.1 CNN Architecture

Among Deep Learning models, Convolutional Neural Networks (CNN) is the most powerful, particularly for pattern recognition. It has transformed computer vision and is widely utilized for many different purposes, such as the identification of real-life violent acts in video data.

Deep neural networks, known as CNNs, were created primarily to interpret visual input, such as images or videos. They are made up of several layers, each of which is in charge of extracting distinct features from the input data and learning them. Convolutional layers, pooling layers and fully connected layers make up the main parts of a fundamental CNN architecture.

3.3.1.1 Convolutional Layers

The foundation of CNNs is convolutional layers. They apply a collection of teachable filters to the input data to conduct convolutional operations. These filters capture various visual patterns at various spatial locations within the picture or video frame, including edges, textures, and shapes. Feature maps that highlight significant features while preserving spatial information are the outcome or output of this layer.

3.3.1.2 Pooling Layers

By reducing the spatial dimensions of the feature maps, pooling layers can capture the most important information while requiring less processing. Max pooling, which takes the highest value available within a given range, is the most widely used pooling operation. This phase of downsampling aids in the extraction of significant features resistant to minute spatial variations.

3.3.1.3 Fully Connected Layers

Fully connected layers use extracted characteristics to learn high-level representations and make predictions. These layers enable complicated relationships and classifying decisions by joining all neurons from one layer to the next.

3.3.1.4 Activation Functions

By introducing non-linearity to the network, activation functions allow the model to recognize intricate, irregular patterns in the input. ReLU (Rectified Linear Unit), a function that sets negative values to zero while maintaining positive values, and softmax, which transforms the network output into probabilities for various classes, are examples of common activation functions.

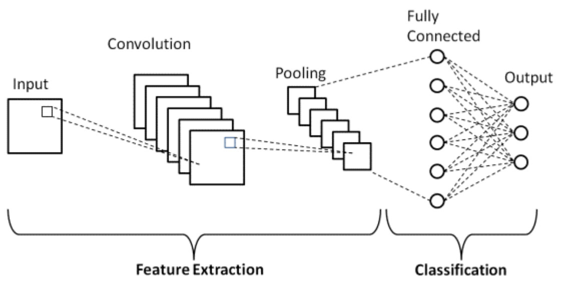


Figure 3.6: CNN Architecture

3.3.1.5 CNN for Detecting Violent Activities in Video Data

Analyzing temporal patterns and motion data is necessary to find violent activity in video data. Preprocessed data should be run via the convolution layer in order to apply CNN here. The CNN then collects features for each frame or frame sequence by running them through the convolutional layers. These layers record temporal dependencies, visual patterns, and motion data. Additional methods can be used to record temporal data. To describe the temporal dependencies between frames, long short-term memory (LSTM) is frequently used. These models sequentially process the retrieved characteristics while taking the temporal order of the frames into account. Fully connected layers are then given the output from the preceding layers, which represents the learned features. These layers classify the input into predetermined categories (such as violence vs. non-violence) or forecast the likelihood of violent activities based on the extracted attributes. Video data with annotations indicating the presence or absence of violent activity is generally used to train the CNN model. Through the use of optimization techniques like stochastic gradient descent (SGD), training entails minimizing a loss function, such as categorical cross-entropy. To enhance the predictions and reduce the loss, the model iteratively modifies its weights.

CNN has demonstrated success in detecting actual violent acts in video data. CNN architectures may examine video frames or sequences and predict the existence of violence by utilizing their capacity to extract visual features, record motion data, and model temporal dependencies. This technology has the capacity to improve security and safety in a number of contexts, helping to identify and stop violent crimes. Mobilenetv2, a sophisticated CNN architecture-based network, was created based on this architecture.

3.3.2 Mobilenetv2 Architecture

A convolutional neural network (CNN) architecture called MobileNetV2 was created to enable quick and easy image classification on mobile and embedded devices. It advances MobileNetV1 by using cutting-edge methods that increase computational effectiveness and accuracy. In situations where computing resources are constrained, such as with mobile phones, IoT devices, and real-time applications, MobileNetV2 is particularly beneficial. Here, we've provided a description of the MobileNetV2 architecture's operation. MobilenetV2 can be used in classification, object detection and semantic segmentation.

A series of layers gradually change the input image in MobileNetV2's "bottleneck" architecture in order to learn higher-level representations. It primarily consists of three essential elements: depthwise separable convolutions, inverted residuals, and linear bottlenecks.

3.3.2.1 Depthwise separable convolutions

Depthwise separable convolutions have two parts. It divides the typical convolution into a depthwise convolution and a pointwise convolution. This is the key component of MobileNetV2's significant use of convolutions and its uniqueness. Depthwise convolution considerably lowers the computational cost by applying a single convolutional filter to each input channel separately. The output channels of the depthwise convolution are combined using the pointwise convolution, also referred to as the 1x1 convolution. The number of parameters and computing complexity can be decreased because of this separation without compromising accuracy.

The computational cost of a standard convolutional layer is $h_i.w_i.d_i.d_j$.k.k. Standard convolutional layers can be swapped out for depthwise separable convolutions. In terms of empirical performance, they are nearly as effective as standard convolutions, but they only cost:

$$hi \cdot wi \cdot di(k^2 + d_j) \tag{3.1}$$

which is the total of the depthwise and 1x1 pointwise convolutions. This layer decreases processing by almost a factor of $k^2, k^2dj/(k^2 + dj)$ to be specific when compared to standard layers. With only a slight accuracy loss, MobileNetV2's computational cost is 8 to 9 times lower than that of ordinary convolutions because of its use of 3 x 3 depthwise separable convolutions.

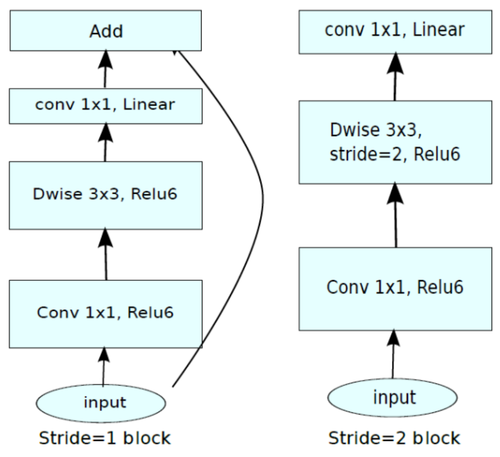


Figure 3.7: MobilenetV2 Architecture

3.3.2.2 Inverted Residuals

Inverted residuals have been added to MobileNetV2 to improve the network's capacity for representation. Expansion, depthwise convolution, and projection are a few of the lightweight operations that make up inverted residuals. The expansion stage boosts the input's channel count, enabling it to record a wider range of information. The extended features are subjected to separable convolutions during the depthwise convolution. The projection stage then brings the channel dimensions back down to the desired level and produces the intended result.

3.3.2.3 Linear Bottlenecks

MobileNetV2 uses linear bottlenecks, which include limiting the number of channels prior to expensive activities and then expanding them later, to further optimize the architecture. This strategy aids in preserving the network's expressive power while lowering the computational cost of expensive operations.

The following stages can be used to summarize how MobileNetV2 operates:
MobileNetV2 accepts either an input image or a feature map for its initial convolution. The initial convolutional layer uses a conventional 3x3 convolution to transform the input into a set of feature maps. MobileNetV2 is made up of several building blocks, each of which is made up of a series of layers. These building components include linear bottlenecks, inverted residuals, and depthwise separable convolutions. These stacked construction blocks process the input as it travels through them, gradually extracting higher-level information from the input image.

Downsampling and Upsampling: To decrease the spatial dimensions of the feature maps while increasing the number of channels, downsampling is carried out via strided convolutions or pooling procedures. This downsampling process aids in gathering more abstract and comprehensive data. Transposed convolutions are frequently used in upsampling to reduce the number of channels while increasing the spatial dimensions, which makes it easier to recover finer details.

The classification layer is the last component of MobileNetV2, often consisting of a fully connected layer with an activation after a global average pooling layer. This layer effectively summarizes the learned features by reducing the spatial dimensions of the feature maps to a vector representation. Utilizing the fully connected layer, we can map this vector to the necessary number of classes and produce the classification probabilities.

3.3.3 Bi-directional LSTM architecture

The long-short-term memory network is a recurrent neural network used primarily for sequential data. To address the issue of vanishing and exploding gradients, LSTM plays a vital role. This network is also better at understanding the connection between values at the start and end of a sequence. LSTM networks share RNN-like structures; however, the LSTM for the memory or repeating module is different. Memory blocks are recurrently linked to form an LSTM layer. Each includes three units, which are the input, output, and forget gates, along with recurrently connected memory cells.

Bidirectional LSTM is better than the usual LSTM in terms of sequence data. Unlike regular LSTM, as it can use data from both sides and input flows from both direction, it is a powerful method for modeling the sequential relationships between data. By adding one extra LSTM layer, BiLSTM reverses the direction of information flow. As a result,the input sequence flows backward in the new layer. After that, outputs from the two LSTM layers are combined in multiple ways.

Each training sequence is presented both forward and backward to two separate LSTM networks that are both connected to the same output layer in order for bidirectional LSTM networks to work. This indicates that the Bi-LSTM has detailed sequential information about all points prior to and following each point in a specific sequence. So, the outputs from both the forward and the backward LSTM are concatenated at each time step rather than just encoding the sequence in the forward direction. Due to the sequence of images in video data, to extract temporal features, bidirectional LSTM will be a better option than regular LSTM in this situation.

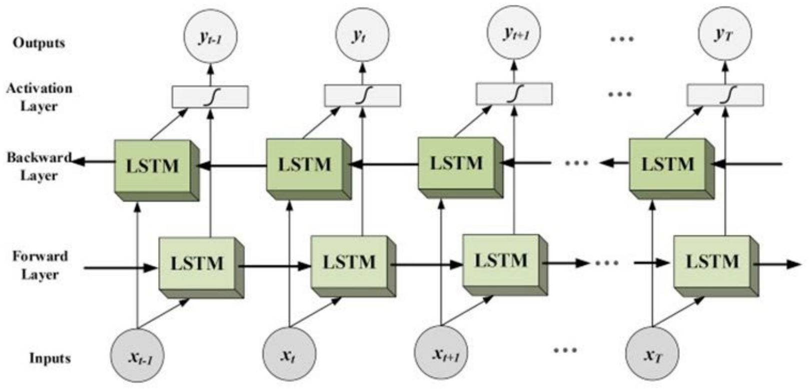


Figure 3.8: Bidirectional LSTM Architecture

3.3.4 Proposed ConvLSTM architecture

In this research, we propose a violence detection model with the combination of MobilenetV2 and bi-directional LSTM. We provide preprocessed video input data that includes both violent and nonviolent actions. In the first part of the model, we used the convolutional layers of MobilenetV2 to extract the spatial information from the dataset. As it is a lightweight model, real-time detection will be faster and cheaper. Further layers with the proper settings, such as the Time Distributed layer, Dropout, Time Distributed Flatten layer, LSTM, Dense layer, etc., are included as the network's remaining layers once MobilenetV2 is added to the topmost layer. We used bi-directional LSTM to capture temporal information and deal with variable dependencies. Finally, the flattened and combined feature maps are passed to fully connected layers for classification. The sizes of the layers are 256, 128, 64, and 32, and between each fully connected dense layer we used the ReLU activation function. In the final binary output dense layer, we used the activation function softmax. We used the SGD optimizer. For the loss function, we used categorical cross entropy to calculate loss from the output layer. The ratio of 80:20 data is selected as training and testing data. Lastly, the model is tested using video data and predicts violence and non-violence actions, showing the detection in real-time.

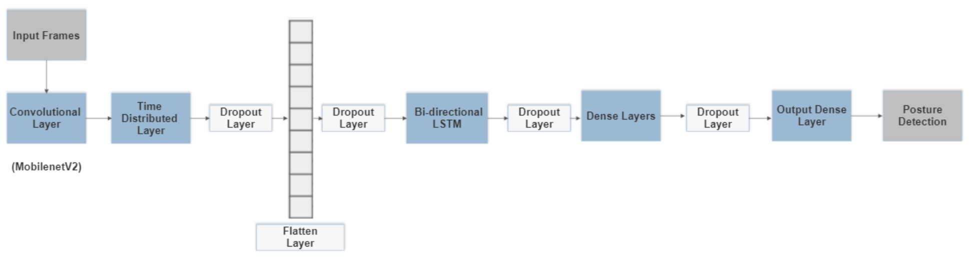


Figure 3.9: Proposed ConvLSTM model Architecture

3.3.5 Posture Detection method

A collection of preprocessed violent and nonviolent action data is trained in this model to successfully predict the actions in real time. The model learns through transfer learning. We use the first few layers of the pretrained MobilenetV2 model to extract the spatial features from which our model learns about patterns and LSTM to extract temporal data. After that, the rest of the layers of our model learn to classify the actions into two categories. As a result, after proper training of the model, if we pass a video containing violent activities, we get the predicted class name on top of the frames.

3.4 Weapon Detection and Classification

3.4.1 YOLOv4 Architecture

3.4.1.1 YOLOv4 in Object Detection

The components of the architecture can be divided into the following groups: The GPU processes the input, which arrives first and essentially represents the collection of training images that will be supplied to the network, in simultaneous batches. The following are the backbone and neck, which carry out feature extraction and aggregation. The Detection Head and Detection Neck are referred to as the Object Detector together. The detection and prediction are then carried out by the head. The crown is primarily in charge of detection (including localization and categorization).

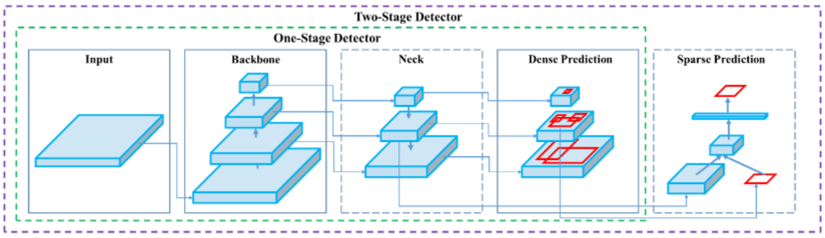


Figure 3.10: Object Detector of YoloV4

The YOLO detector, also known as dense detection, performs both of these simultaneously because it is a one-stage detector. A two-stage detector, however, does each individually and then combines the findings (sparse detection).YOLOv4 investigates various data augmentation techniques and backbone networks.

Backbone Network

What acts as the foundation? The majority of the layers in this deep neural network are convolution layers. Because the backbone's main objective is to extract the essential information, choosing the right backbone is an essential step that will improve object identification performance. Frequently, the neural network that trains the backbone has already been trained. The backbone networks were initially anticipated to be CSPResNext50, CSPDarknet53, and EfficientNet-B3. They ultimately settled on CSPDarknet53 CNN after conducting rigorous testing and analyzing the results. The CSPDarkNet53 design is built on the DenseNet platform. It concatenates the previous inputs with the present input before entering the dense layers; this is referred to as the dense connection pattern. There are two blocks in CSP-DarkNet53: 1. Convolutional Base Block Cross Stage Partial (CSP) for Layer 2. Over Stage The base layer's feature map is split in half and merged using the partial technique, which addresses the dreaded ”vanishing gradient” problem by allowing more gradients to flow across the layers.

The convolutional base layer is composed of the entire input feature map. The CSP block, which is stacked next to the Convolutional Base layer, divides the input into two parts; one is sent through the dense block, while the other is routed directly to the next step without any processing, as was previously indicated. CSP promotes the reuse of network features, safeguards fine-grained features for better forwarding, and minimizes the number of network parameters. Only the last convolutional block in the backbone network, which can extract more semantic data, is dense because more tightly coupled convolutional layers may slow down detection.

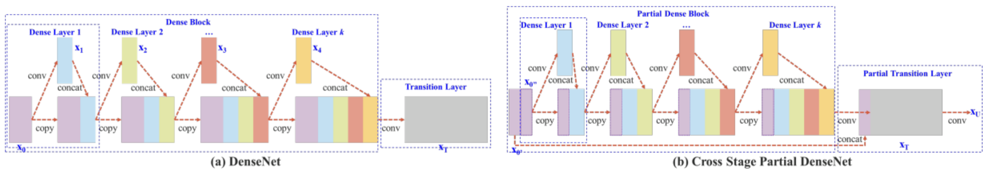


Figure 3.11: CSP

Neck

The neck is where features come together. The feature maps are gathered from the multiple backbone stages and blended and merged to prepare them for the next stage. A neck often consists of both a number of top-down and bottom-up routes. Typically, only the last few levels of the convolutional network's neck's connections move between layers.

SPP

The feature aggregator network (PANet) and the CSPDarkNet53 backbone are connected by an additional block called SPP (Spatial Pyramid Pooling). This has little to no effect on network operation speed and serves to increase the receptive field and segregate the most crucial context characteristics. It is connected to the strongly coupled final CSPDarkNet convolutional layers.

The receptive field is the area of the image that is exposed to one kernel or filter at a time. It grows linearly as more convolutional layers are added, but when dilated convolutions are introduced, which causes non-linearity, it expands exponentially.

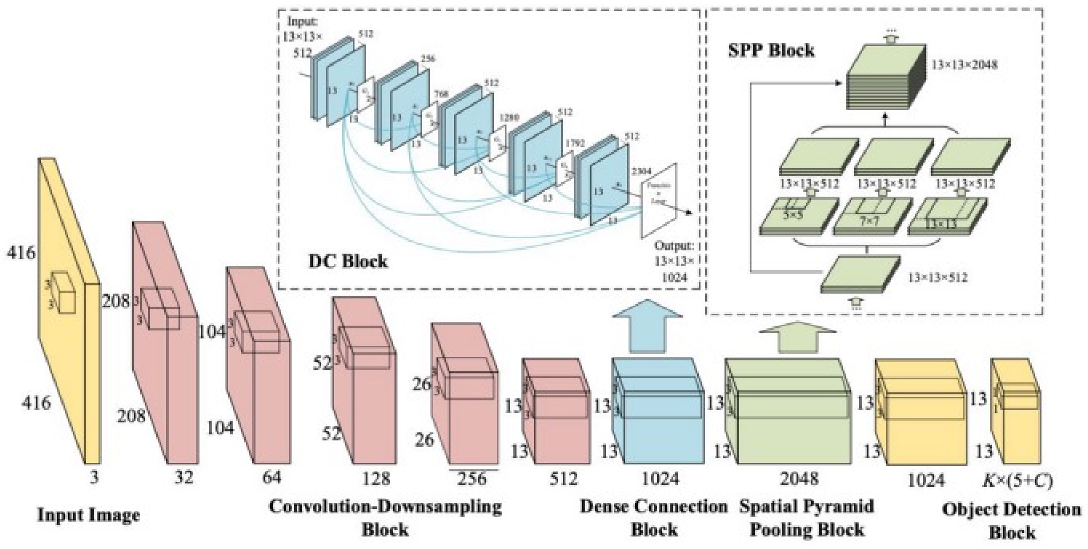
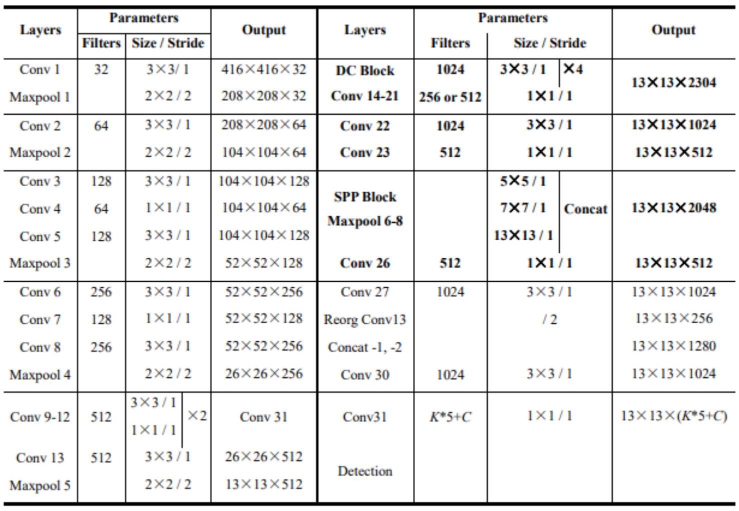


Figure 3.12: SPP

Here in Figure 3.13 is a summary of the actions taken:



Layers	Parameters		Output	Layers	Parameters		Output
	Filters	Size / Stride			Filters	Size / Stride	
Conv 1	32	3×3/ 1	416×416×32	**DC Block**	**1024**	**3×3 / 1 ×4**	**13×13×2304**
Maxpool 1		2×2 / 2	208×208×32	**Conv 14-21**	**256 or 512**	**1×1 / 1**	
Conv 2	64	3×3 / 1	208×208×64	**Conv 22**	**1024**	**3×3 / 1**	**13×13×1024**
Maxpool 2		2×2 / 2	104×104×64	**Conv 23**	**512**	**1×1 / 1**	**13×13×512**
Conv 3	128	3×3 / 1	104×104×128	**SPP Block**		**5×5 / 1**	
Conv 4	64	1×1 / 1	104×104×64	**Maxpool 6-8**		**7×7 / 1 Concat**	**13×13×2048**
Conv 5	128	3×3 / 1	104×104×128			**13×13 / 1**	
Maxpool 3		2×2 / 2	52×52×128	**Conv 26**	**512**	**1×1 / 1**	**13×13×512**
Conv 6	256	3×3 / 1	52×52×256	Conv 27	1024	3×3 / 1	13×13×1024
Conv 7	128	1×1 / 1	52×52×128	Reorg Conv13		/ 2	13×13×256
Conv 8	256	3×3 / 1	52×52×256	Concat -1, -2			13×13×1280
Maxpool 4		2×2 / 2	26×26×256	Conv 30	1024	3×3 / 1	13×13×1024
Conv 9-12	512	3×3 / 1 ×2 1×1 / 1	Conv 31	Conv31	*K**5+*C*	1×1 / 1	13×13×(*K**5+*C*)
Conv 13	512	3×3 / 1	26×26×512	Detection			
Maxpool 5		2×2 / 2	13×13×512				

Figure 3.13: Summary of Operations

PANet

YOLOv4 principally uses a modified path aggregation network as a design improvement to make it more suitable for training on a single GPU. By maintaining spatial information, PANet's main purpose is to improve the instance segmentation process, which in turn helps with precise pixel localization for mask prediction. Bottom-up path augmentation, adaptive feature pooling, and Fully-Connected Fusion are essential features that contribute to their high level of mask prediction accuracy. The modified PANet concatenates neighbouring layers rather than adding them while using adaptive feature pooling.

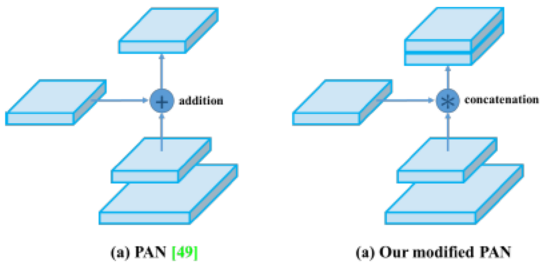


Figure 3.14: PANet

Head

The main duties in this situation are finding bounding boxes and doing categorization. The bounding box coordinates (x, y, height, and width) are discovered along with the scores. The b-box's center is indicated here by the x and y coordinates in reference to the boundary of the grid cell. Forecasts for height and breadth take into account the complete image.

Additionals of YoloV4

The terms "Bag of Freebies" (BoF) and "Bag of Specials" (BoS) were created by the authors.

1. **Bag of Freebies (BoF)**

   Most of these data augmentation techniques improve network efficiency without lengthening inference times. The ability to create different versions of a single image is made feasible through data augmentation, which improves the network's prediction abilities. The two main techniques included in this architecture are self-adversarial training (SAT) and mosaic data augmentation.

2. **BoF for the backbone**

   Class label smoothing, DropBlock regularization, CutMix, and mosaic data augmentation.

3. **BoF for detector**

   When employing many anchors for a single ground truth, CIoU-loss, CmBN, DropBlock regularization, mosaic data augmentation, Self-Adversarial training, grid sensitivity be gone, cosine annealing scheduler, ideal hyperparameters, and random training shapes are employed. The primary and crucial components of BoF are explained below.

   The model can be trained to find smaller items and pay less attention to the surroundings by combining four training photographs into a mosaic. Self-adversarial Training (SAT), another technique, drives the network to recognize new features by concealing the region of the image it relies on the most. To enable training on just one GPU, Cross mini-Batch normalization is introduced. since the majority of batch normalization solutions use multiple GPUs. A regularization method to address over-fitting is called DropBlock. A block of pixels is dropped. In contrast to pixel dropout, which is ineffective on convolution layers, it does so. A regularizer that lowers the prediction's goal upper bound is class label smoothing. This is an additional measure implemented to solve the over-fitting problem. Its formula is:

   $$y_{lb} = (1 - \alpha) * y_{hot} + \alpha/C \tag{3.2}$$

   C = how many label classes there are
   α = Smoothing Hyper-Parameters

4. **Bag of Specials (BoS)**

   Although they somewhat lengthen inference times, BoS considerably improves performance. The backbone's BoS is composed of Mish activation, Cross-stage partial connections (CSP), and Multiinput weighted residual connections (MiWRC). The detector's BoS includes Mish activation, SPP-block, SAM-block, PAN path-aggregation block, and DIoU-NMS.

5. **Mish Activation**

   Using this activation function, signals can be pushed to the left and right, which is not possible with ReLU-style activation functions. Mish performs better than ReLU, Swish, and Leaky ReLU in terms of empirical results. Furthermore, when Mish is applied, various data augmentation techniques react consistently.

6. **Multi-input weighted residual connections (MiWRC)**

   EfficientDet's weighted bidirectional feature network-based tailored compound scaling method, which improves accuracy and efficiency, uses MiWRCs to construct a crucial part.

7. **Non-Maximum Suppression (NMS)**

   The DIoU-NMS method only keeps the box with the highest confidence score after filtering out all other boxes. Distance IoU (DIoU) is used to achieve this, which considers the IoU values and the distance between the center points of two bounding boxes while suppressing the redundant boxes. In situations where there is occlusion, this is ideal.

3.4.1.2 Loss Function

The YOLOv4 loss function combines a number of elements in order to maximize the model's performance in reliably recognizing and localizing objects. The three main elements of the overall loss function in YOLOv4 are the localization loss, the confidence loss, and the class loss.

1. **Localization Loss (Lcoord)**

   The localization loss (Lcoord) gauges how well bounding box predictions are made. The model is penalized based on the discrepancy between the predicted and real bounding boxes. YOLOv4 use the sum of squared differences (SSD) loss to calculate the Euclidean distance between the predicted box coordinates and the ground truth box coordinates.

2. **Confidence Loss (Lobj)**

   The confidence loss measures how well the model can distinguish between object and background regions. It assesses the confidence of the model's predictions. In order to compare the anticipated objectness score—which denotes the presence of an object—with the ground truth label, YOLOv4 uses binary cross-entropy loss.

3. **Class Loss (Lcls)**

   The class loss gauges the object classification's precision. To determine the difference between the anticipated class probabilities and the actual class labels, YOLOv4 uses categorical cross-entropy loss.

   The weighted sum of the aforementioned elements—which includes factors to balance the effects of each loss term—represents the overall loss (Ltotal) for YOLOv4. During training, the relative importance of each component can be changed to highlight particular features, for as emphasizing correct localization or enhancing classification accuracy. With the help of backpropagation, YOLOv4 tries to jointly optimize these loss components in order to reduce total loss and enhance the model's performance in object detection tasks.

3.4.2 YOLOv7 Architecture

3.4.2.1 YOLOv7 in Object Detection

To gain a thorough understanding of the image or video, we should estimate the positions of items in each image and focus on classifying the various images. The term "object detection" refers to this action, which frequently entails a number of smaller ones like "face detection," "vehicle detection," "person detection," and so forth. Object detection techniques are used in many commercial applications, including picture classification, face recognition, autonomous driving, and the analysis of human behavior. They are one of the fundamental computer vision issues that can offer crucial data for comprehending the semantics of both images and videos. Meanwhile, advancements in these fields have had a big impact on the field of computer vision and its ability to cross numerous boundaries. These abilities are inherited from the neural network and related learning system families.

Among the many various object detection models that are efficient for particular use cases, the recently released YOLOv7 stands out. It asserts that it outperforms all known object detectors in both speed and accuracy and has the highest accuracy (56.8% AP) among all known real-time object detectors. The proposed YOLOv7 version E6 outperformed transformer-based detectors like the SWINL Cascade-Mask R-CNNR-CNN in terms of accuracy and speed. Scaled-YOLOv4, Scaled-YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, and Vit-Adapter-B have all underperformed YOLOv7.

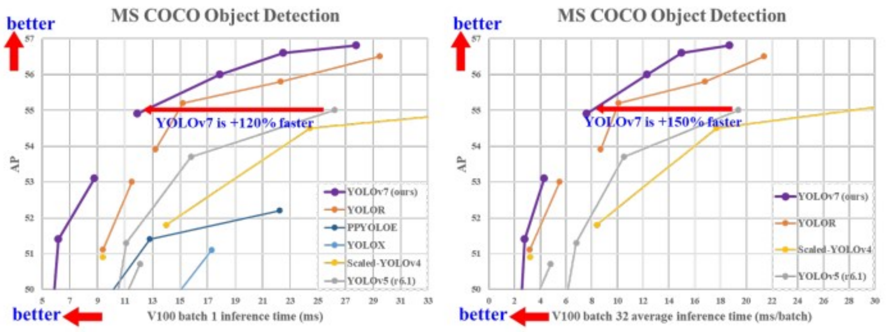


Figure 3.15: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

3.4.2.2 Model Re-parameterization

By integrating numerous computational modules into one during the inference step, model re-parameterization aims to shorten inference time. Model-level ensembles and module-level ensembles are the two types of model re-parameterization techniques. In the first instance, training several identical models using various training sets results in the averaging of their weights. In the latter, a weighted average of model weights over various iteration counts is calculated. Module-level reparameterization has gained favor recently. Additionally, some re-parameterization techniques are architecture-specific, which limits their applicability to a small number of architectures. A new kind of re-parameterization is introduced in YOLO v7 to solve the drawbacks of earlier techniques.

3.4.2.3 Model Scaling

A method for scaling an existing model to accommodate any computer platform is called "model scaling." The architecture is very flexible as a result. Stage scaling (number of feature pyramids), depth scaling (number of layers), width scaling (number of channels), and resolution scaling (size of the input image) are a few of the several scaling methods. The trade-off between accuracy and speed is revealed via model scaling. Techniques for model scaling, such as network architecture search (NAS), are frequently employed. Without setting unduly complicated rules, NAS may automatically explore the search space for suitable scaling factors. The drawback of using NAS is that finding model scaling factors needs expensive compute. The main problem with NAS is the independent study of the scaling factors. To deal with the problem of independent assessments, YOLO v7 provides a new scale method.

3.4.2.4 Extended efficient layer aggregation networks

The size, quantity, and processing density of a model are the main factors to be taken into account while creating an efficient architecture. The VovNet model goes one step further by examining how the input/output channel ratio, the quantity of architectural branches, and element-wise operation affect network inference speed. The next significant advancement in architecture search is ELAN, which YOLOv7 expanded into E-ELAN. The ELAN article claims that if the shortest and longest gradient pathways are controlled, a deeper network can successfully train and converge.

No matter how many processing blocks are stacked or how long the gradient path is, large-scale ELAN has attained stability. This stable condition can be lost, and the rate of parameter utilisation will decrease, if there are infinitely more processing blocks added to the stack. While maintaining the initial gradient path, E-ELAN uses expanding, shuffling, and merging cardinality to continuously enhance the network's learning capacity. E-ELAN solely modifies the computer block's architecture; the transition layer's architecture is unaffected. To expand the channel and cardinality of computing blocks, the E-ELAN approach uses group convolution. It gives each computational block in a computational layer the same group parameter and channel multiplier. The generated feature maps from each computation block are then split into g groups and concatenated using the specified group parameter g. There will be the same number of channels in each group of feature maps as there were in the original architecture. Merge cardinality by including g feature map groupings lastly. While preserving the original ELAN design architecture, E-ELAN can instruct other collections of computational blocks to pick up extra, more varied functionality.

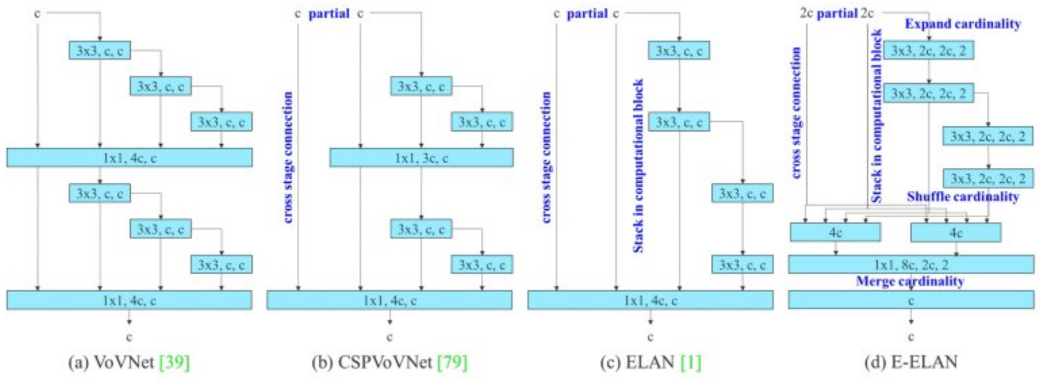


Figure 3.16: E-ELAN architecture

Model scaling for concatenation-based models

Model scaling is typically used to alter specific model characteristics and produce models at various scales to account for various inference speeds. They scale width, depth, and resolution in the well-known Google architectural design, EfficeintNet. Later, scientists made an effort to ascertain the impact of scale-running groups and vanilla convolution on the quantity of parameters and computation.

Concatenation-based architecture is incompatible with EfficientNet's methodology because it affects the degree of the translation layer that comes after a concatenation-based computation block when scaling up or down in depth. For instance, scaling-up depth will alter the proportion of a transition layer's input channel to its output channel, potentially lowering the hardware requirements of the model. The suggested method should also identify how a computational block's output channel changes as its depth factor changes. After making the same amount of adjustment to the width factor scaling for the transition layers, the outcome is displayed in the image below. The ideal structure and the model's original characteristics can both be preserved using the YOLOv7 compound scaling method.

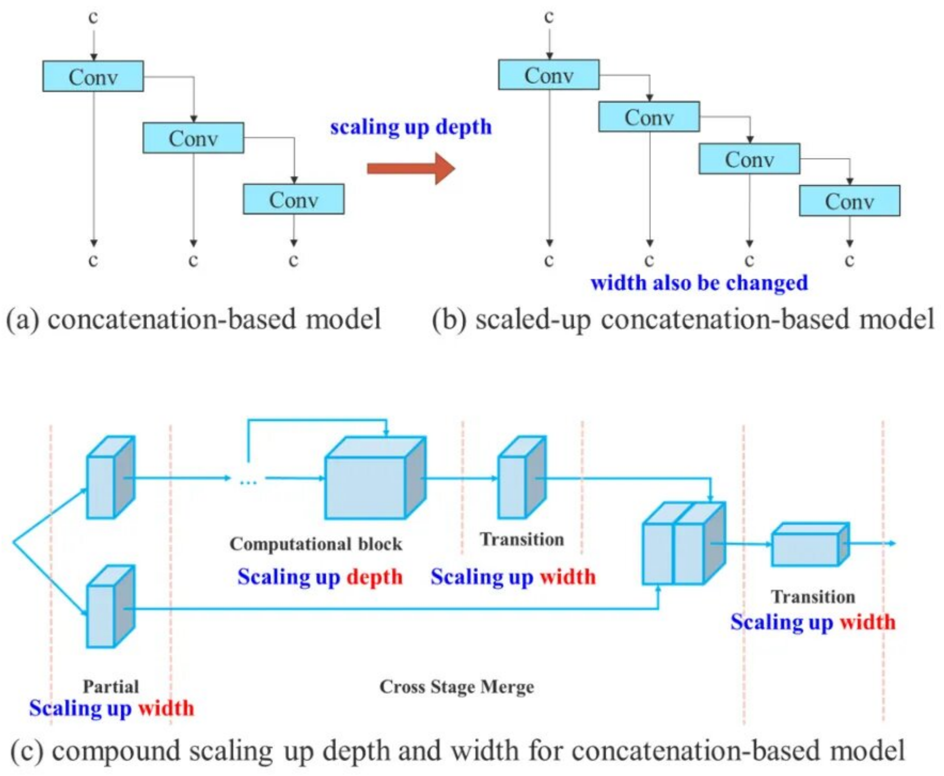


Figure 3.17: Model Scaling

Trainable bag-of-freebies

1. **Planned re-parameterized convolution**

   Before we go into more detail about RepConv, another type of convolutional block, let's take a closer look at it. RepConv, like Resnet, has one identification connection as well as a second connection with a 1x1 filter in between.

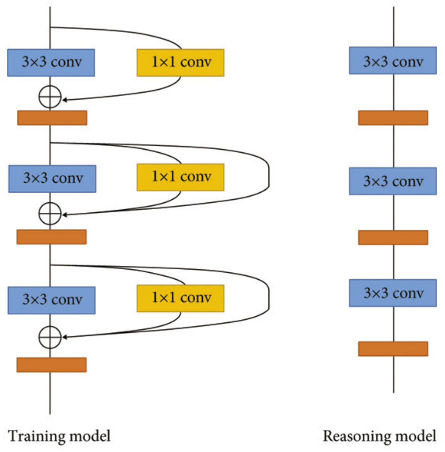


Figure 3.18: RepConv being used in VGG

   Before we go into more detail about this, on RepConv, a different form of convolutional block. RepConv, like Resnet, has a 1x1 filter sandwiched between a first identifying connection and a second connection.

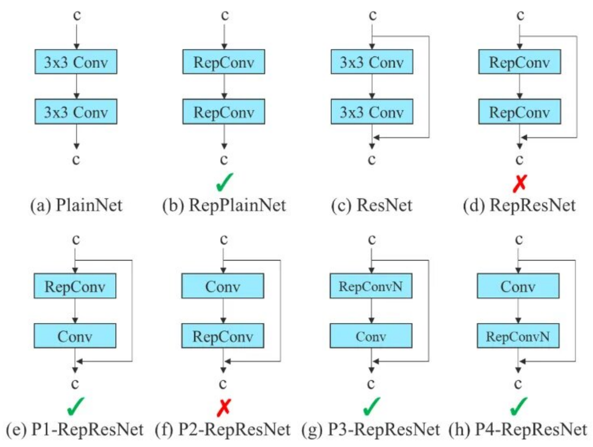


Figure 3.19: Planned re-parameterized model

2. **Coarse for auxiliary and fine for lead loss**

   When training deep neural networks, deep supervision is a common strategy. Increasing the number of auxiliary heads at the middle level of the network is the primary objective of the assistant loss-guided shallow network weights. Deep supervision, even for architectures that commonly converge effectively, like ResNet and DenseNet, can greatly boost the model's performance on numerous tasks. Below is an illustration of the object detector architecture in both the ”without” and ”with” deep supervision stages.In the YOLOv7 architecture, the lead head is in charge of producing output, while the auxiliary head is in charge of assisting in training.

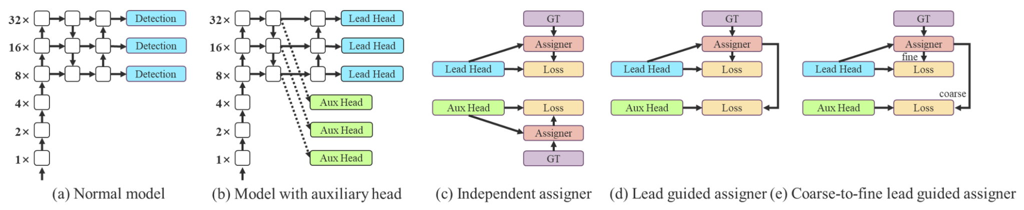


Figure 3.20: Causes of auxiliary and fine for lead head label assigner

Label assignment during deep network training directly references ground truth and produces complex labels per the prescribed rules.

3. **Assigned lead head directed label**

   The lead head can focus on learning the remaining information that is yet undiscovered by having the shallower auxiliary head directly absorb the knowledge the lead head has learned.

4. **Assigned coarse-to-fine lead head guiding label**

   Because of this approach,the fine label’s optimizable upper bound is always more important than the coarse label’s, allowing for dynamic adjustment of the relative relevance of fine and coarse tags during the learning process.

3.4.2.5 Loss Function

Confidence loss(L(obj), classification loss(L(cls), and localization loss(L(box) make up the three components of the YOLOv7 model’s loss function. The sum of the three losses with varying weights is the overall loss. Formula 3.3 is used to express the total loss function.

$$LOSS = a \times L_{obj} + b \times L_{cls} + c \times L_{box} \tag{3.3}$$

Chapter 4

Implementation

4.1 Posture Detection

The implementation process for violence detection will be described here. Our model will work using a CCTV camera for real-time inferences or predictions. After extracting spatial and temporal features and training, our model can accurately predict violence in real time. For this, our system has combined two models for a better result, as mentioned before. Firstly, the models check to see if there are any activities or sequential actions of violence in the frames of the video. If there are any, the class name 'violence' is shown in those video frames. The step by step processes are explained further.

4.1.1 Splitting

We divided our preprocessed dataset into train, test, and validation. We had 1000 videos of violence and nonviolence each. After enough preprocessing, we have a usable custom dataset.

Data Segments	Percentage	Total Dataset
Training	80% of total data	1600
Testing	20% of total data	400

Table 4.1: Posture data splitting

4.1.2 Model Training

Mobilenetv2 is a pre-trained deep learning network that is trained on millions of images from the ImageNet dataset. First of all, we import the weights from the ImageNet dataset to use in our custom dataset. During the training process, the model learns parameters known as weights. The purpose of using them is to convert input data into a more beneficial format for the purpose of classification or prediction. Imagenet has 1000 classes. Since there are only two classes in our dataset, we will just employ this model's convolution layers to obtain feature maps so that we can train it on our dataset. By removing the last few dense layers from the pre-trained model Mobilenetv2, we can get a feature vector instead of the predicted class.

4.1.2.1 Fine tuning and Hyperparameter Selection

The final study applies a fine-tuning method to the model by using the training set from the dataset in order to obtain further improvements to the proposed model. The proposed model's hyper-parameters are adjusted, such as by decreasing the learning rate and increasing the number of epochs. The accuracy results obtained from testing the refined model with the test set of the dataset are on par with the state of the art.

Mobilenetv2 is 53 layers deep. In the first few layers, the model learns simple or generic features. The higher the layers go, the features become more specific. To stop that and train the model solely for our dataset, we need to freeze the layers and modify the rest. Here, we modified the last 40 layers for our purpose. For that, all the layers except the last 40 should be set to untrainable.

The subsequent section will explicate the model's structure. After splitting the preprocessed data in an 80:20 ratio for the train and test sets, we import the MobilenetV2 model. We used the weights of the ImageNet dataset to utilize the model. Then we froze the first 13 layers and started training the rest of the 40 layers. We used the sequential function of Keras to create the model layer by layer. Here, we used the top layers of convolution from the pretrained model.

Firstly, for the input dimension, we passed sequence length, image height, image weight, and parameter 3, which represents the RGB factor of the image. Here, the sequence length represents a batch of 16 successive frames. The dimension of the image is 64x64. In the input layer, the difference between each pair of adjacent frames is calculated.

As mobilenet uses conv3D, it expects the input to have a shape that is three dimensional in size. As our input has 4 parameters, we have to use a time distributed layer to handle that and pass the sequence length. So we passed mobilenet to the time distributed layer. Then we added a time-distributed flatten layer, which converts the feature maps into a 1D vector or array so that they can be passed to dense layers. Feature maps are the output that we get from convolution layers.

After that, we used bidirectional LSTM with units 32 for both forward and backward. Bidirectional LSTM is better than basic LSTM for sequence data. Also, it can learn from both the past and future contexts of a sequence. We include a regular dropout layer after every other dense layer and LSTM to deal with the overfitting. The dropout layers drop 25% of the information each time. Finally, we added a chain of dense layers, or normal ann layers. The sizes of the layers are 256, 128, 64, and 32. Between each fully connected dense layer, we use the ReLU activation function. In the final binary output dense layer, we used the activation function softmax. We used categorical cross entropy as the loss function to calculate loss from the output layer and the stochastic gradient descent (SGD) optimizer. The default learning rate is kept at 0.001. We used only 20 epochs and an early stopping point of 10. The ratio of 80:20 data is selected for training and testing data. We used dropout layers regularizer: early stopping to prevent overfitting. Lastly, we get two neurons as we have two classes.

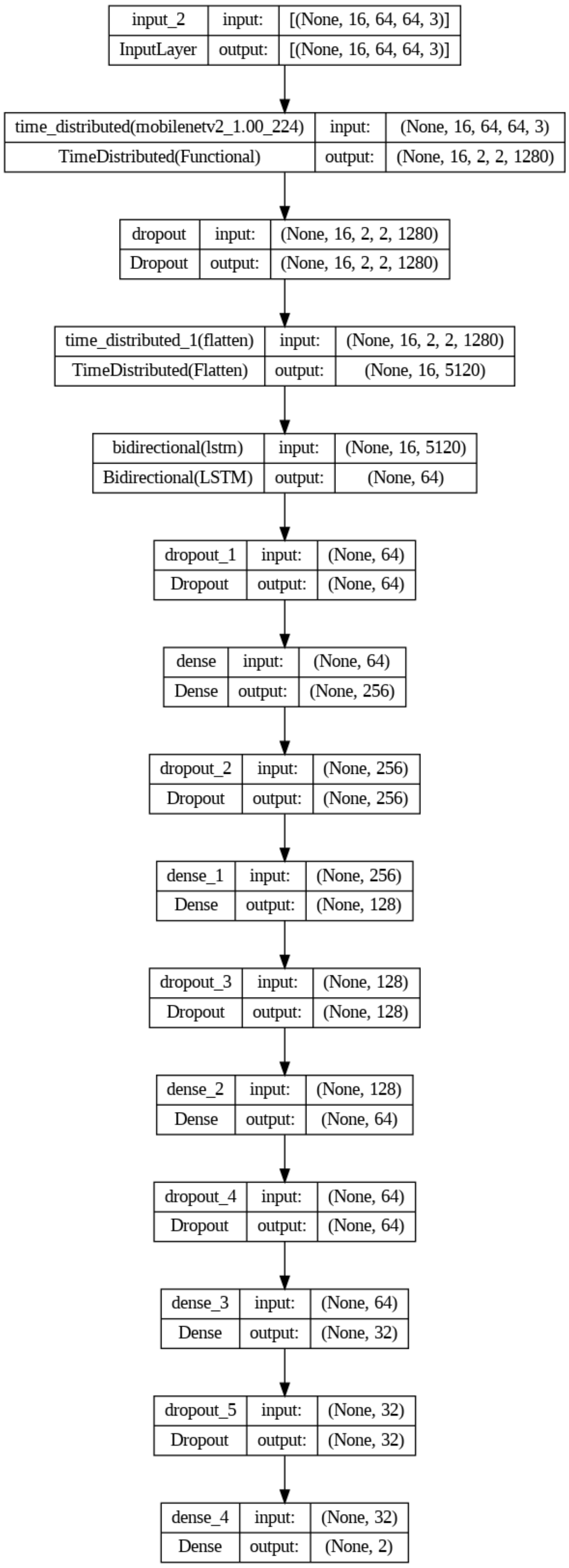

input_2 | input: | [(None, 16, 64, 64, 3)]
InputLayer | output: | [(None, 16, 64, 64, 3)]
time_distributed(mobilenetv2_1.00_224) | input: | (None, 16, 64, 64, 3)
TimeDistributed(Functional) | output: | (None, 16, 2, 2, 1280)
dropout | input: | (None, 16, 2, 2, 1280)
Dropout | output: | (None, 16, 2, 2, 1280)
time_distributed_1(flatten) | input: | (None, 16, 2, 2, 1280)
TimeDistributed(Flatten) | output: | (None, 16, 5120)
bidirectional(lstm) | input: | (None, 16, 5120)
Bidirectional(LSTM) | output: | (None, 64)
dropout_1 | input: | (None, 64)
Dropout | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 256)
dropout_2 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 128)
dropout_3 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 64)
dropout_4 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_3 | input: | (None, 64)
Dense | output: | (None, 32)
dropout_5 | input: | (None, 32)
Dropout | output: | (None, 32)
dense_4 | input: | (None, 32)
Dense | output: | (None, 2)

Figure 4.1: Model Architecture

4.1.3 Model Testing

Following training, model testing is done. A test dataset is used to evaluate the model after each iteration is complete. Testing is carried out after training. A test dataset is used to evaluate the model after each iteration is complete. Finally, future predictions are made using the model. Multiple classifiers are evaluated by utilizing a testing set. After testing the model using video data, it predicts the violence and non-violence actions shown by the label. Here, we tested our model in two ways. Firstly, we tested it in real time using the webcam on our PC. If there are any actions like slaps, kicks, punches, or any other violent movements engaging multiple people, the model detects them and classifies them as violence. If violence is absent in a frame, then the classification name becomes "non-violence. The classification name is shown at the top of the frame on the screen.

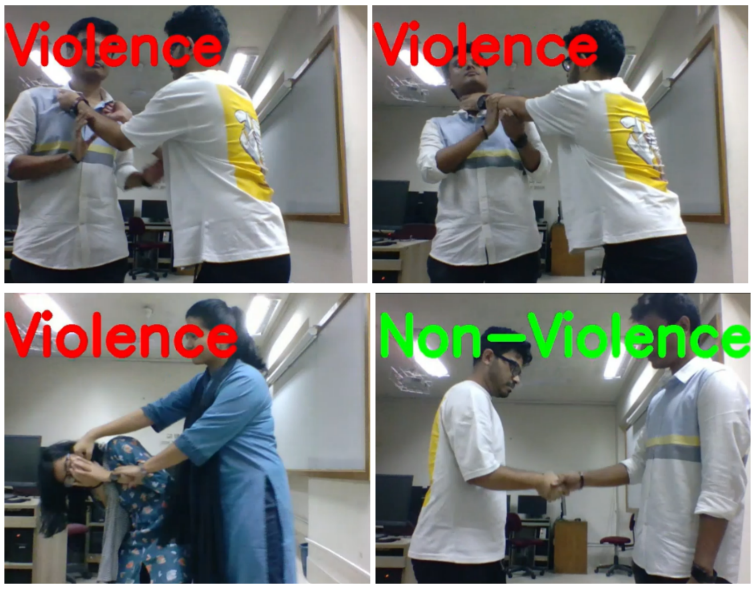


Figure 4.2: Real-time Violence and Non-violence Detection

The second method of testing involves giving video data as input. It will again be converted into frames, and the model will predict the violence.

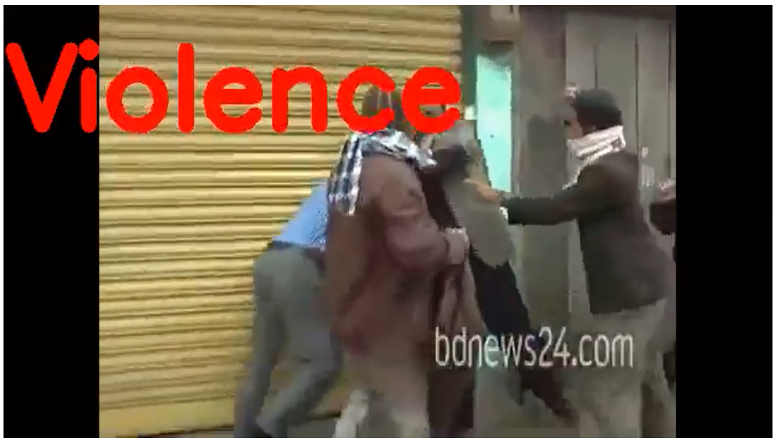


Figure 4.3: Violence and Non-violence Detection from video input

4.2 Weapon Detection and Classification using YOLOv4

In this section how yolov4 is implemented for weapon detection will be discussed Yolov4 is excellent for detecting multiple objects in a single frame. Here we will be focusing on mainly the detection of weapons. We used our own custom dataset for the training method. Three groups are made from the dataset. The first group of datasets is guns. This dataset only consists of guns, specifically handguns and rifles. The second group of the dataset is Sharp Objects. Here for this dataset, we specifically collected sharp objects that are widely used in Bangladesh. So that we can create an accurate dataset that is viable for the training of yolov4 so that it can detect the sharp objects used in Bangladesh easily. Lastly, the final group of the dataset is sticks. This entire category only consists of sticks that can be used as weapons in a fight.

4.2.1 Data Pre-processing

In order to preprocess our data, we need to split our dataset into some specific segments. Our dataset here is a custom dataset, and it has to be split into three specific segments. These are Train, Test and Valid. For this purpose, we used Roboflow. By offering better methods for data collection, preprocessing, and model training, users may more rapidly and accurately develop computer vision models thanks to the Roboflow computer vision platform. To build our own dataset, we utilized the YOLOV4 model and the model-expected image size. Our final, acceptable dataset was adjusted as a result of these procedures. We also used Labellimg to annotate the entire custom dataset. We took three classes for annotation in Labelimg. We labeled guns as 0 , sharp objects as 1, and sticks as 2.

4.2.2 Model Implementation

Two of the key responsibilities of this methodology are to identify and categorize weapons. In the following procedure, a specially trained YOLOV4 classifier makes our proposed model weapon-compatible.

1. **Setup**

   In order to speed up processing, we must first build a new Colab notebook and link it to a GPU runtime. Next, we must copy the YOLOv4 implementation from the Darknet repository. The Makefile must then be modified in order to configure the darknet for YOLOv4. Set "GPU=1" to enable GPU acceleration and "OPENCV=1" to support OpenCV. After completing these steps, compile Darknet by executing the "!make" command to create the executable for Darknet. We also have to download the weights of the pre-trained YoloV4. We must download a file from a specified URL using the wget command. Yolov4.conv.137, a pre-trained weight file for the YOLOv4 object detection model, is the file that is now being downloaded. When this command is run, the file yolov4.conv.137 will be downloaded from the provided URL and saved in the current directory.

Next, we must execute a particular command. Using the Darknet framework, this command starts the training of an object detection model. The data file, model setup, weights that have already been trained, and other choices for the training process are all specified. Once more, we must use two distinct commands. When it comes to computer vision tasks like object identification and image processing, cv2 is employed, and matplotlib.pyplot is used to create visualizations like plots and graphs. The studies of computer vision, image analysis, and data visualization make extensive use of these libraries. The darknet detection test will be the next phase. 0.3 threshold: This is a command-line operation carried out in Colab or a Jupyter Notebook using the! syntax. It applies the Darknet framework to an image to perform object detection.

2. **Real Time Implementation**

   Using a specific piece of code, the Colab notebook that we are using will be able to take a picture using the webcam, identify objects in the picture, and display the result. The *take_photo*() function takes a picture from the webcam using JavaScript and Colab-specific tools. When the "Capture" button is hit, it produces a JavaScript function that manages the photo-taking process, including showing the webcam's video stream and taking a picture. A base64-encoded JPEG image is then created using the acquired image.

3. **Model Testing**

   Here, the Yolov4 is able to detect weapons from an image and also detected sharp objects and sticks from different images. This pictures illustrates how the suggested algorithm successfully detects and classifies firearms by constructing a perfect bounding box.

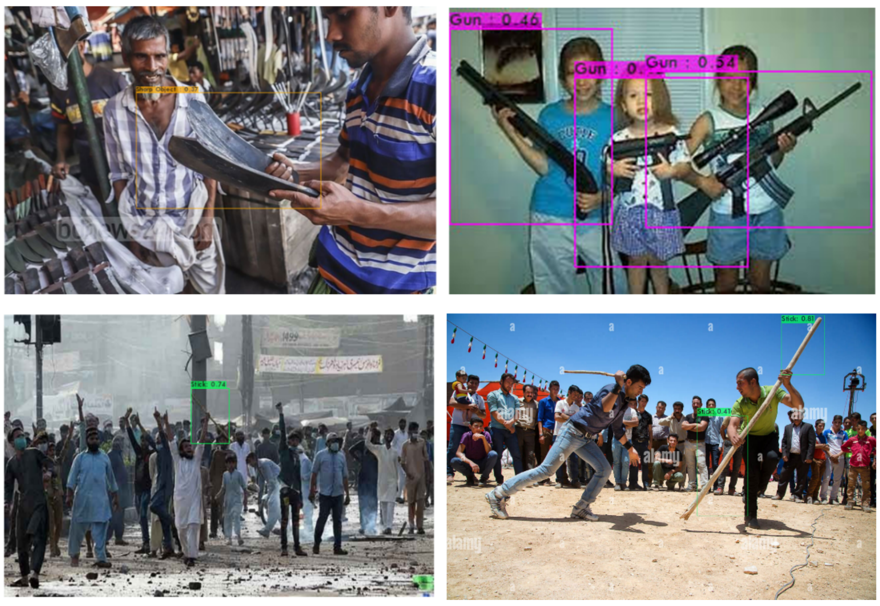


Figure 4.4: Detecting Sticks, sharp Object and gun

4.3 Weapon Detection and Classification using YOLOv7

The implementation portion of Yolov7 will be covered in this chapter. A variety of items in pictures and videos can be found using the effective object detection algorithm YOLOv7. It is a quick and precise algorithm that has a number of uses, including robotics, surveillance, and security. We are going to focus on how we can detect the weapon by using Yolov7. To train our Yolov7 model, we gathered three different types of datasets. The first type is a gun, the second type is sharp objects, with a focus on Bangladeshi sharp objects like knives and machetes, and the third type is sticks used as weapons. Since Yolov7 is a single-stage object detection algorithm, it can identify objects in just one pass through the image.

4.3.1 Data Pre-processing

Additionally, we used "Roboflow" to divide our custom dataset into train, test, and valid segments. We used the YOLOv7 model and the model-expected image size to create our own dataset. Following these processes, our final usable dataset was modified. Before dividing our dataset, we used Labellimg to annotate the entire custom dataset. We took 3 classes for annotation in Labeling. We labeled the gun as 0 , the sharp object as 1, and the stick as 2.

4.3.2 YOLOv7 Training Process

Identification and classification of weapons are two of this methodology's main responsibilities. Through the process described below, a specially trained YOLOv7 classifier is utilized to make our suggested model compatible with the weapon. The

Data Segments	Percentage	Total Dataset
Training	70% of total data	776
Validation	10% of total data	111
Testing	20% of total data	222

Table 4.2: Weapon data splitting

experiment on the detection models was conducted using Google Colab, a platform that provides free coding notebooks, cloud virtual machines with storage, a GPU, and tensor processing units (TPU) for doing time-consuming and difficult computations. Google Colab, which doesn't require configuration and offers free access to potent GPUs, was used to train the model. We applied YOLOv7-based pre-trained COCO weights from a Roboflow.ai notebook. The Google Colab drive module is first imported using the code from the google.colab import drive file. The drive module gives Google Colab notebook users access to Google Drive files. Google Drive is mounted to the current working directory.

4.3.3 Real-time Implementation

We are proposing code for taking a picture with a camera in a Google colab setting, then using a trained YOLOv7 model to identify objects in the picture. The usage The (take_photo()) function in JavaScript is used to take a webcam picture. The process initiates the activation of the camera feed, creates a video element, and remains in a state of readiness until the user initiates the "Capture" function. When the button is pressed, a video frame is recorded, the stream is stopped, and the picture is returned as a base64-encoded string. Using the b64decode function and the open block, the taken image is then saved to a file with the name "photo.jpg." Additionally, the try-except block makes an attempt to take the picture and uses the display(Image) function to show it. Errors that happen throughout the capture process are detected and shown as error messages. Lastly, the script (detect.py) uses the recorded photo as input. The input image is resized to 224x224 pixels, the confidence threshold is set to 0.5, and object detection is carried out on the image using the loaded weights of a trained YOLOv7 model (best.pt).The (–no-trace) option specifies that no tracing data should be shown throughout the detection process.

4.3.4 Model Testing

Firstly, we've gathered pictures of weapons. Then we labeled every picture with boundary boxes that surrounded the weapon. We have developed a configuration file that contains the video path, model architecture, training parameters, and classification labels in order to instruct our model on how to label weapons. We also specified how many classes we wanted to identify and categorize. The results of visualizing the three different types of weapons are shown in the figure. By creating a perfect bounding box, this figure shows how the suggested algorithm accurately detects and identifies weapons. Our method exhibits the ability to effectively identify and differentiate various occurrences, such as a civil protest, a solitary gun, or an individual carrying a knife. Furthermore, upon playing a video on our model, it was able to accurately detect the presence of weapons. As can be seen in Fig.**??** and Fig.4.6, our model successfully detected both the sharp object and the stick that was used in the strike.

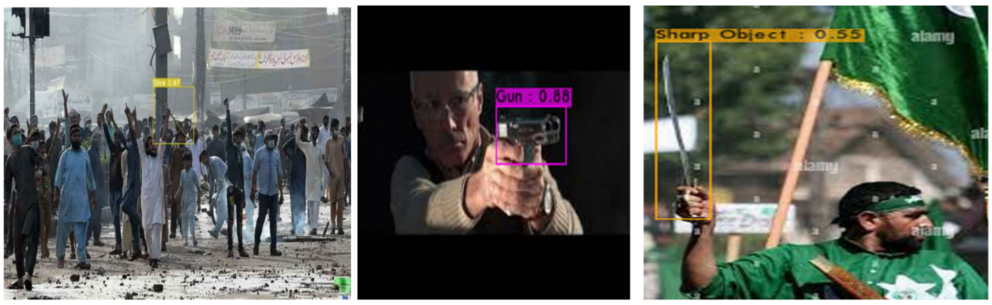


Figure 4.5: Detecting stick, sharp object and gun

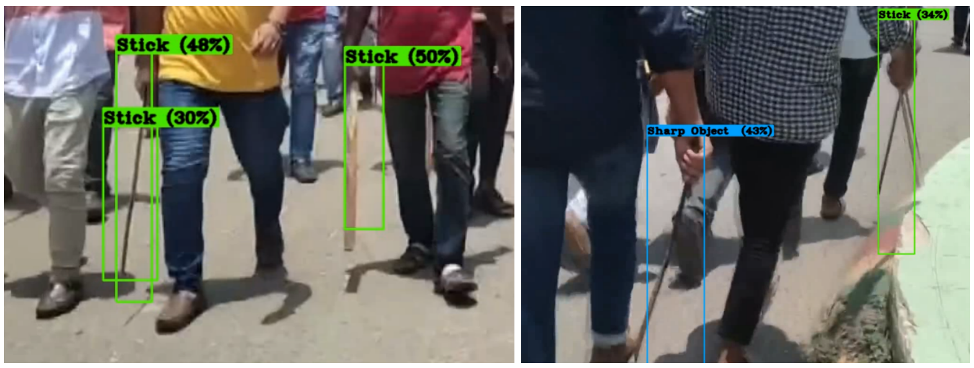


Figure 4.6: Detecting weapon from video

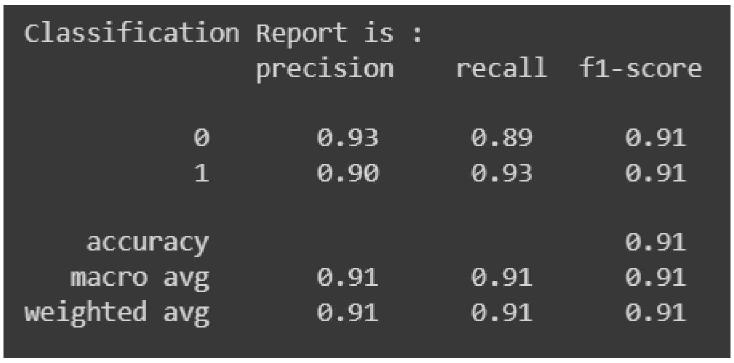
Chapter 5

Result Analysis

5.1 Result of ConvLSTM Posture Detection Model

5.1.1 Classification Report

We assessed the model's effectiveness using a number of factors, including classification accuracy, recall, F1 score, precision, etc. The accuracy metric is considered to be the main evaluation metric for detection and classification models. 1000s violent and non violent videos of average duration of 6 seconds were given as input data. From the classification report we can see that the accuracy of the overall prediction is 91% which is for 20 epochs. Here, we can see that The f1-score is similar to the accuracy where precision and recall for the categories are 93%, 89% and 90%, 93%. The results are remarkable along with low detection time compared to the other existing detection methods considering the number of epochs.



```
Classification Report is :
              precision    recall  f1-score

           0       0.93      0.89      0.91
           1       0.90      0.93      0.91

    accuracy                           0.91
   macro avg       0.91      0.91      0.91
weighted avg       0.91      0.91      0.91
```

Figure 5.1: Classification report

5.1.2 Evaluation metrics

The model was trained using 20 epochs with a batch Size of 16.. The training and validation accuracy of the model is 0.97 and 0.91 respectively which surpasses many existing models. At the same time, the training loss converged at 0.1 and the validation loss was 0.25 though it had some fluctuations over the training period.

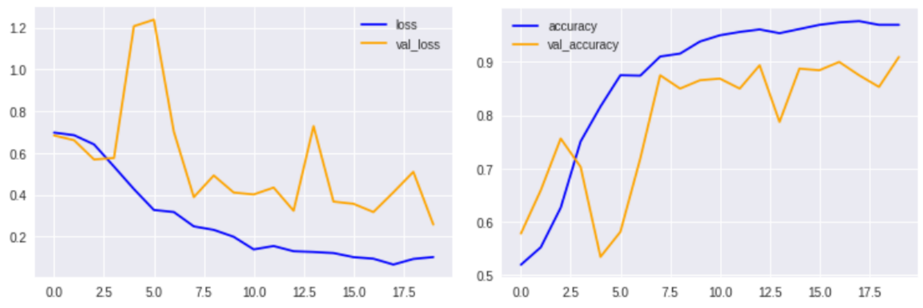


Figure 5.2: Total loss vs validation loss graph and Total accuracy vs validation accuracy graph

The confusion matrix shows the evaluation findings with the true labels and the predicted labels. We have come upon a collection of results that can illuminate the precision of our classification. We have reached a true positive count of 178 for non-violence detection, which means that 178 non-violent incidents were accurately recognized and classified. It is crucial to remember that our classification model has occasionally made mistakes, leading to a total of 21 false positives. These false positives show that we misclassified 21 incidents, labeling them as non-violent when they actually involved violence. On the other hand, we have discovered a genuine negative count of 187 while seeking to identify instances of violence. These ”true negatives” show situations where our model correctly classified 187 incidents as violent. But there have also been 14 false negatives that we have encountered. These ”false negatives” represent occasions where our model mistook violence for something else and labeled them as such.

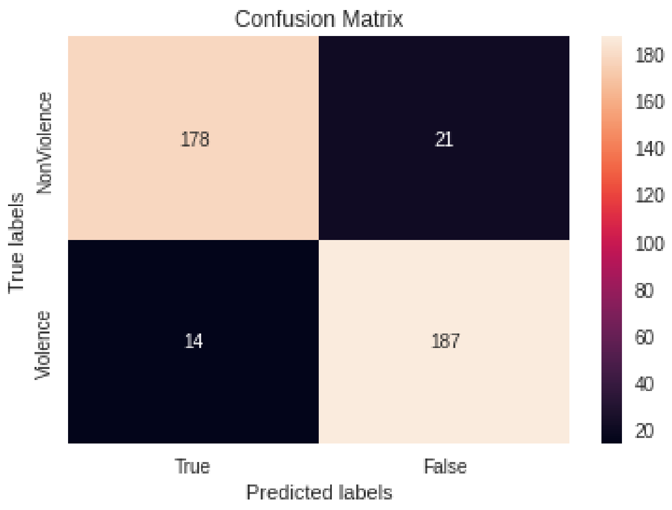


Figure 5.3: Confusion matrix of violent posture detection

5.2 Result of YOLOv4

We trained our model using 2000 iterations because we have 3 classes of weapons. The entire training process used the GPU provided by Google Colaboratory and took 3 hours. The loss score converges, as shown in the figure, after around 2000 iterations, and the current average loss is recorded as 0.674020. We also used a variety of test data to assess the model's performance.

5.2.1 mAP@0.5

mAP@.5 measures the overall accuracy of the model in YOLO v4. The mean average precision at a threshold of 0.5 is used to calculate it. Our mAP@.5 is 0.67, which means that the model achieves an average precision of 67% when the detection threshold is set to 0.5. mAP@.5 is a well-liked metric for evaluating the performance of object identification models. It is a trustworthy sign of the model's accuracy and completeness. A precise and thorough object detection is indicated by a high mAP@.5 value. A mAP@.5 of 0.67 is a good result for an object detection model. It demonstrates how precise and detailed the model's object identification is.

5.3 Result of YOLOv7

5.3.1 Graph Analysis

In this part,we are going to explain the overall performance of YOLO v7 by analyzing the performance graph. Box loss, objectness loss, and categorization loss are three different types of loss depicted in Figure 1.The box loss measures the precision with which a bounding box and precise center can be determined for an object by an algorithm. The term "objectness" describes how likely it is that an object will turn up somewhere in particular. According to high objectivity, an object is likely to be present in the visible area of an image. The classification loss is a measure of how accurately an algorithm determines an object's class. The model's performance was consistently improved as the epochs count went from 100 to 500.

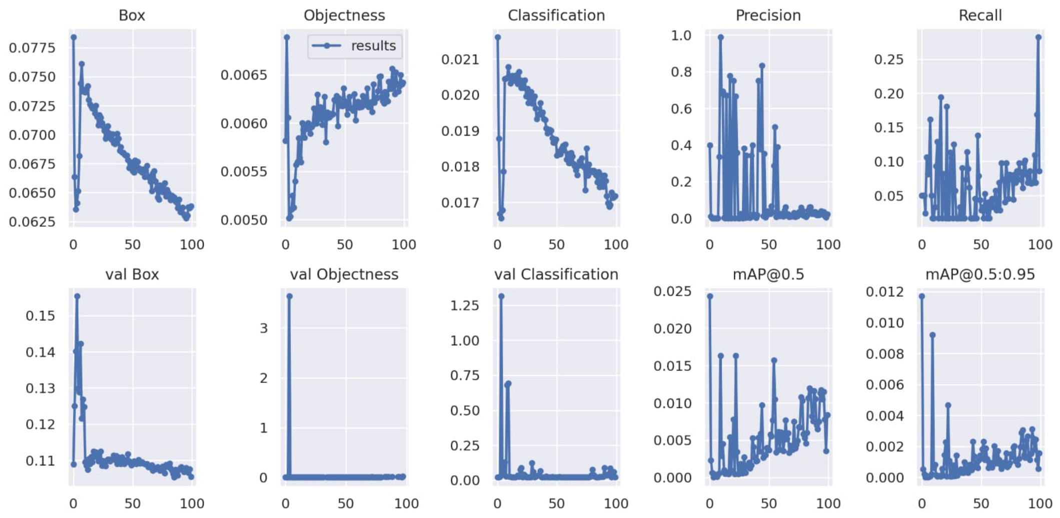


Figure 5.4: Analyzing model evaluation indicators visually (during training) for the proposed YOLOv7's

5.3.1.1 Precision

Precision in YOLOv7 refers to the model's sensitivity to object detection. It is calculated by dividing the total number of objects detected by the total number of correctly identified objects. Our precision rate of 0.964 indicates that the model correctly identified 96.4% of the objects in the image. We are all aware that precision is a crucial parameter to take into account when assessing object detection models, as it can help to guarantee that the model is not only detecting a high number of items but also accurately identifying the ones that are present.

5.3.1.2 Recall

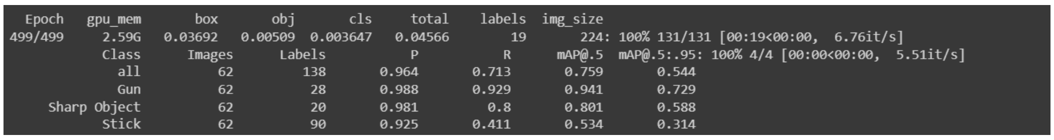
Recall in YOLOv7 refers to the proportion of objects that the model correctly identified from the image. It is determined by dividing the total number of objects in the image by the number of things that were successfully detected. Our r recall rate of 0.713 indicates that the model correctly identified 71.3% of the image's objects. When assessing object detection models, recall is a crucial measure to take into account because it can help to guarantee that the model is not just detecting a lot of items but also the actual objects that are present.

5.3.1.3 mAP@0.5

In YOLOv7, mAP@.5 serves as a gauge of the model's general accuracy. The mean average precision at a threshold of 0.5 is used to calculate it. Our mAP@.5 is 0.759, which indicates that when the detection threshold is set to 0.5, the model achieves an average precision of 75.9%. A popular statistic for assessing the effectiveness of object identification models is mAP@.5. It is a reliable indicator of the model's correctness and comprehensiveness. A high mAP@.5 value means that the model's object detection is accurate and thorough. An excellent outcome for an object detection model is a mAP@.5 of 0.759. It shows that the model is accurate and thorough in identifying things.

5.3.1.4 mAP@0.5:0.95

The YOLO v7, mAP@.5:.95 serves as a gauge of the model's general accuracy. It is calculated using the mean average precision at a threshold of 0.5 and an IoU threshold of 0.95. Our mAP@.5:.95 value is 0.544, which indicates that when the detection threshold and IoU threshold are both set to 0.5, the model achieves an average precision of 54.4%. The metric mAP@.5:.95 is more demanding than mAP@.5, since it calls for accurate localization and detection of objects. A high mAP@.5:.95 value means that the model's object detection is accurate and thorough. Another successful outcome for an object detection model is a mAP@.5:.95 of 0.544. It shows that the model detects things accurately and completely, even when they are obscured or partially obscured.



```
  Epoch   gpu_mem       box       obj       cls     total    labels  img_size
499/499     2.59G   0.03692   0.00509  0.003647   0.04566        19       224: 100% 131/131 [00:19<00:00,  6.76it/s]
            Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 4/4 [00:00<00:00,  5.51it/s]
              all        62       138     0.964     0.713     0.759     0.544
              Gun        62        28     0.988     0.929     0.941     0.729
     Sharp Object        62        20     0.981       0.8     0.801     0.588
            Stick        62        90     0.925     0.411     0.534     0.314
```

Figure 5.5: Overall result of YOLOv7 model

5.4 Comparison between YOLOv4 Model and YOLOv7 Model

5.4.1 Dataset Analysis

We used the same custom dataset we created to compare the performance of the YOLOv4 and YOLOv7 models. The custom dataset is divided into three categories. These categories are Sharp Objects, Sticks and automatic weapons. The total data in the custom dataset is 1109. The percentage of splitting the data segments is equal for both YOLOv4 and YOLOv7 models. The percentages of splitting the data segments are 70 percent for training, 10 percent for validation, and lastly, 20 percent for testing. We annotated the dataset only once for both of the models, the reason for doing this is that we used the same custom dataset.

5.4.2 Comparison in Detection

Here in the prediction, we can see that YOLOv7's prediction is much better than YOLOv4. In terms of stick objects, YOLOv4's prediction for this picture is 0.74; on the other hand, for YOLOv7, it is 0.97. Lastly, the other stick result of this model for YOLOv7 is 0.76, but for YOLOv4, it is 0.41, 0.81. YOLOv7 provided an overall prediction on the same stick object at the same time while YOLOV4 provided two predictions in only one stick due to a poor bounding box. So we can see that for object detection, YOLOv7's prediction is much better than YOLOv4's in those pictures.

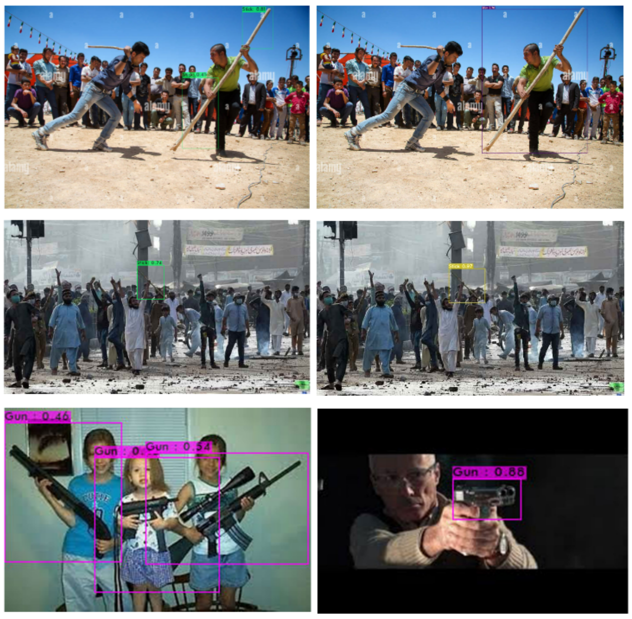


Figure 5.6: Comparison between YOLOv4 and YOLOv7 prediction

5.4.3 Comparison in terms of mAP@.5

Model	mAP@.5
YOLOv4	0.67
YOLOv7	0.759

Table 5.1: Yolo v4 vs Yolov7 compare in terms of mAP@.5

The Yolov7 model was trained over a period of 500 epochs, indicating that the dataset was utilized for training purposes 500 times in total. The model achieved a mAP@.5 score of 0.759 following 500 epochs. The value of 0.759 indicates that the model attains an average precision of 75.9%. Conversely, the YOLOv4 model attains a mAP@.5 of 0.67, indicating that it obtains a mean average precision of 67% at a detection threshold of 0.5. The metric mAP@.5 is commonly utilized in the assessment of object detection models. Therefore, it is evident that our model, YOLOv7, performs significantly better than YOLOv4.

Chapter 6

Conclusion and Future Work

6.1 Future Scope

In the future, we will try to aggregate the two models with ensemble learning so that they can provide better accuracy. In this scenario, where two models with different inferences are being combined, we must first train each model independently before combining their predictions to arrive at a single prediction.
Since we want to execute real-time processing, we will need to decrease the computational complexity for further deployment. To achieve this goal, we will attempt to train our model using a significantly larger dataset, which will increase the inference speed and make the model deployment possible.
We also have a plan to include an alert mechanism in the system that would notify a human supervisor or local police station of potential trouble and prompt them to take appropriate action. Our intention is to create an online police system that alerts authorities to impending and ongoing criminal activity. The police can use this information to better anticipate and avoid such situations in the future. In order to mitigate the risk of the model misidentifying patrolling officers as criminals, we will incorporate images of uniformed officers into the training data.

6.2 Conclusion

Even in this day of advanced technology, the number of reported criminal offenses continues to climb steadily. These days, investigators utilize a wide variety of tools to study crime scenes and follow suspects. CCTV camera data can be used in a number of different ways to determine the nature of a crime and the identity of any suspects present. There are a variety of approaches, but there is always a chance that the wrong people will be caught. Our studies aim to improve the efficiency with which CCTV cameras can identify criminal activity. Our goal is to improve upon pre-existing frameworks and algorithms for machine learning in order to produce more reliable results in our training procedures. In this regard, we are striving to make an effort to improve the reliability of the system in order to make people's lives safer.

Bibliography

[1] E. Bermejo, O. Deniz, G. Bueno, and R. Sukthankar, "Violence detection in video using computer vision techniques," *Computer Analysis of Images and Patterns*, pp. 332–339, Jan. 2011.

[2] K. Divya and S. Srinivasan, "International research journal of engineering and technology (irjet)," 2015.

[3] S. Chackravarthy, S. Schmitt, and L. Yang, "Intelligent crime anomaly detection in smart cities using deep learning," in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, 2018, pp. 399–404. DOI: 10.1109/CIC.2018.00060.

[4] U. Navalgund and P. K, "Crime intention detection system using deep learning," Dec. 2018, pp. 1–6. DOI: 10.1109/ICCSDET.2018.8821168.

[5] S. Loganathan, G. Kariyawasam, and P. Sumathipala, "Suspicious activity detection in surveillance footage," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2019, pp. 1–4. DOI: 10.1109/ICECTA48151.2019.8959600.

[6] C. Rajapakshe, S. Balasooriya, H. Dayarathna, N. Ranaweera, N. Walgampaya, and N. Pemadasa, "Using cnns rnns and machine learning algorithms for real-time crime prediction," in *2019 International Conference on Advancements in Computing (ICAC)*, 2019, pp. 310–316. DOI: 10.1109/ICAC49085.2019.9103425.

[7] H. Jain, A. Vikram, Mohana, A. Kashyap, and A. Jain, "Weapon detection using artificial intelligence and deep learning for security applications," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020, pp. 193–198. DOI: 10.1109/ICESC48915.2020.9155832.

[8] S. Tanjila Naurin, A. Saha, K. Akter, and S. Ahmed, "A proposed architecture to suspect and trace criminal activity using surveillance cameras," in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 431–435. DOI: 10.1109/TENSYMP50017.2020.9230901.

[9] T. S. Apon, M. I. Chowdhury, M. Z. Reza, A. Datta, S. T. Hasan, and M. G. R. Alam, "Real time action recognition from video footage," in *2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2021, pp. 1–6. DOI: 10.1109/STI53101.2021.9732601.

[10] L. Chen and S. Li, "Human motion target posture detection algorithm using semi-supervised learning in internet of things," *IEEE Access*, vol. 9, pp. 90 529–90 538, 2021. DOI: 10.1109/ACCESS.2021.3091430.

[11] D. Harshavardhan and D. Swamy, "an Efficient Criminal Segregation," pp. 636–641, 2021.

[12] A. Jan and G. M. Khan, "Malicious activity detection in safe city environment," in *2021 International Conference on Artificial Intelligence (ICAI)*, 2021, pp. 170–174. DOI: 10.1109/ICAI52203.2021.9445254.

[13] P. Sivakumar, J. V, R. R, and K. S, "Real time crime detection using deep learning algorithm," in *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2021, pp. 1–5. DOI: 10.1109/ICSCAN53069.2021.9526393.

[14] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," no. July, 2022. DOI: 10.48550/arXiv.2207.02696. arXiv: 2207.02696. [Online]. Available: http://arxiv.org/abs/2207.02696.

[15] "A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms," *Journal of Computing and Social Informatics*, vol. 2, no. 1, pp. 1–12, 2023. DOI: 10.33736/jcsi.5070.2023.

[16] H. Chen and H. Lou, "HPS-YOLOv7 : A High Precision Small Object Detection Algorithm," 2023.

[17] H. Zhao, H. Zhang, and Y. Zhao, "Yolov7-sea: Object detection of maritime uav images based on improved yolov7," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2023, pp. 233–238. DOI: 10.1109/WACVW58289.2023.00029.

[18] W. Acharya Khoshelham, *Ceur-ws.org/vol-1913/rl17_paper_1.pdf*, http://ceur-ws.org/Vol-1913/RL17_paper_1.pdf, (Accessed on 09/18/2022).

[19] *Ijrti*, https://www.ijcrt.org/papers/IJCRT2106136.pdf, (Accessed on 09/18/2022).

[20] D. R Manjula, *Irjet-volume7 issue5*, https://www.irjet.net/volume7-issue5, (Accessed on 09/18/2022).

[21] *Real life violence situations dataset — kaggle*, https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset, (Accessed on 05/23/2023).

[22] H. Sivalingan and N. Anandakrishnan, *Ijivp_vol_12_iss_1_paper_2_2502_2507.pdf*, https://ictactjournals.in/paper/IJIVP_Vol_12_Iss_1_Paper_2_2502_2507.pdf, (Accessed on 09/18/2022).