

Detecting Threats from Live Videos using Deep Learning Algorithms

Rawan Aamir Mushabab AlShehri¹, Abdul Khader Jilani Saudagar²
Information Systems Department, College of Computer and Information Sciences,
Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

Abstract—Threat detection is an important area of research, particularly in security and surveillance applications. The research is focused on developing a threat detection system using DL techniques. The system aims to detect potential threats in real-time video streams, enabling early identification and timely response to potential security risks. The study uses two state-of-the-art DL models, MobileNet and YOLOv5, to train the object detection system. The TensorFlow object detection API is employed for training and evaluating the models. The results of the study indicate that MobileNet outperforms YOLOv5 in terms of detection accuracy, speed, and overall performance. The justification for selecting MobileNet over YOLOv5 is based on several factors. First, MobileNet has a lightweight architecture, making it suitable for real-time applications where processing speed is critical. Second, it is efficient in terms of memory usage, enabling it to operate effectively on low-resource devices. Third, MobileNet provides high accuracy in detecting objects of different sizes and shapes. The study evaluated the performance of the threat detection system using various evaluation metrics, including mean average recall (mAR), mean average precision (mAP) and Intersection over union (IoU). The results show that the system achieved high accuracy in detecting threats, with an overall mAP (mean average precision) of 0.9125, mAR (mean average recall) of 0.9565 and Intersection over union (IoU) of 0.9045. In this study, researchers present a highly efficient and successful method for identifying threats through the utilization of deep learning methods. The research demonstrates the superiority of MobileNet over YOLOv5 in terms of performance, and the results obtained validate the effectiveness of the proposed system in detecting potential threats in real-time video streams.

Keywords— *Deep learning; machine learning; object detection; threat detection*

I. INTRODUCTION

Nowadays, technologies are experiencing unprecedented growth and advancement, particularly in the field of Data Sciences (DS). DS encompasses a wide range of disciplines, including Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL). AI involves the external creation of intelligence for various systems, enabling them to make decisions based on insights derived from available data. ML empowers machines to process data and generate knowledge and intelligence by integrating applied statistics and optimization theory. DL is a subfield of ML that specifically concentrates on constructing, training, and deploying extensive and intricate neural networks [1]. By leveraging data in parallel, these neural networks carry out their operations with the aim of accomplishing their assigned tasks. To develop an intelligent system for object detection, it is necessary to

identify and categorize moving objects. Object detection entails the ability of computers and software systems to identify the presence and location of various objects within an image, such as vehicles, animals, humans, and more. Deep convolutional neural networks have gained significant prominence in tasks like image classification, object classification, localization, and object detection [2]. DL technologies have been increasingly applied in image classification, target tracking, object detection, image segmentation. These technologies have helped in different social life events and cases. The focus in this research is to use these technologies to raise safety and security. Considering the escalating global crime and terrorism rates, the demand for automated video surveillance has surged. The combination of surveillance and detection has become critically significant. While human detection and tracking are desirable, the unpredictable nature of human movement presents significant challenges in effectively tracking and categorizing suspicious activities. The focus of this research is to identify and detect potentially threatening objects captured by Close Circuit Television Cameras (CCTV) [3]. This idea comes out of the great need for detecting threatening objects to help evaluate situations or prevent any further crimes. Finally, the idea behind this research is extracted and built based on the previous studies in the field of DL techniques. Also, among the search between the previous studies, there was no use of the MobileNet model in the field of recognizing and detecting objects in videos especially threatening objects.

A. Object Detecting Technique

Object detection is a computer vision technique that empowers software systems to identify, locate, and track objects within images or videos. One of its notable features is the ability to classify objects (such as people, tables, chairs, etc.) and precisely determine their coordinates within the image. This is achieved by drawing a bounding box around the object, although the accuracy of the bounding box may vary. The effectiveness of the detection algorithm is measured by its capacity to accurately locate objects within the image. An example of object detection is facing detection.

Object detection algorithms can either be pre-trained or trained from scratch, with pre-trained weights from existing models often utilized and fine-tuned to suit specific requirements or use cases. Object detection is a crucial area of research in computer vision and finds widespread applications in various fields, including surveillance, robotics, autonomous vehicles, and image and video search engines. The objective of object detection is to enable machines to comprehend and

interpret the visual world in a similar manner to humans. Deep learning techniques, such as convolutional neural networks (CNNs), are commonly employed in object detection algorithms to analyze images or videos and identify objects within them. These algorithms are trained on large datasets comprising labeled images, where the object's location and class are annotated. The training process involves adjusting the weights and biases of the neural network to minimize the error between predicted and actual object locations and classes. Once trained, object detection algorithms can be applied to detect objects in new images or videos. The algorithm typically outputs a set of bounding boxes corresponding to the detected objects, accompanied by their class labels and confidence scores. Post-processing techniques can be further employed to refine the bounding boxes and enhance the accuracy of the object detection process.

In recent years, there has been significant progress in object detection, driven by advances in DL and the availability of large datasets. State-of-the-art object detection algorithms can achieve high accuracy and real-time performance on a wide range of object types and scenarios. In addition to the traditional object detection methods, there are also several advanced techniques that have emerged in recent years. Below are a few examples:

1) One-shot object detection: Traditional object detection algorithms require a large amount of labeled data during the training phase. One-shot object detection, on the other hand, is a technique that can detect objects with only one or a few examples of each object class during training.

2) Instance segmentation: Instance segmentation is an extension of object detection that not only detects the objects in an image but also segments each object instance from the background. This technique is useful in scenarios where precise object boundaries are necessary, such as in medical imaging or autonomous driving.

3) 3D object detection: While traditional object detection works on 2D images, 3D object detection can detect objects in 3D space. This is important for applications such as robotics and autonomous vehicles, where the detection of objects in 3D is necessary for navigation and obstacle avoidance.

4) Few-shot object detection: Similar to one-shot object detection, few-shot object detection is a technique that can detect objects with only a few examples of each object class during training. However, few-shot object detection is more challenging than one-shot object detection as it requires the algorithm to generalize to unseen object classes.

B. Object Detecting from Image and Video

Object detection is an important and prominent area of research that combines deep learning (DL), computer vision, and image processing. Its primary objective is to identify specific semantic objects, such as people or animals, in digital images and videos. Within the field of object detection, there are well-established areas of study, such as pedestrian detection and face detection. However, object detection has broader applications in various DL fields, including image restoration and video surveillance [4]. To differentiate between different

objects, object detection employs a multi-label classifier, although it does not determine the specific identity of each object. This is where Image Localization comes into play, as it precisely determines the object's location within the image by providing a bounding box around it. Image Localization technology has practical applications in image retrieval, with facial detection being a widely used example. Additionally, it can assist in pedestrian detection at traffic lights to enhance traffic flow and aid visually impaired individuals. It also facilitates Sign Language Detection (SLD) to support communication with the deaf and mute community. The process of object detection involves providing an image or video frame, using algorithm-based models to search for targeted objects, and assigning specific categories to each identified target [5]. Object detection algorithms commonly employ machine learning (ML) and DL techniques to achieve meaningful results. Furthermore, the ultimate goal of object detection models is to emulate the rapid comprehension and recognition of objects by the human brain when observing images or videos. Training DL algorithms and models to match the intelligence of the human brain using computer technology is a challenging yet crucial task, particularly in the context of detecting potentially threatening objects.

C. Problem Statement

The research focuses on the detection and recognition of objects using ML and DL techniques, which is a prominent area of study for DL enthusiasts. The aim is to enable machines to learn autonomously by simulating the functioning of neurons in the human brain. While previous works have explored object detection for various social issues, there is a lack of research specifically addressing the detection of threatening objects in videos and evaluating the MobileNet model for this purpose. This research aims to enhance community safety and security by improving the detection of threatening objects, as incidents and accidents often result from inadequate security measures or delayed responses to immediate threats. The research seeks to answer two main questions: is there an efficient way to detect threats from live videos using DL algorithms, and how accurate and efficient is the MobileNet model in detecting objects in live videos? The objectives of the study are to detect threats in live videos to prevent the criminal use of threatening objects and to assess the efficiency and accuracy of the MobileNet model in detecting threatening objects. The research utilizes DL techniques to identify and detect threats in live video clips or surveillance videos. The idea for this research originated from the urgent need for video analysis on the internet, and it builds upon a review of previous studies in DL and its techniques. The researchers identified a gap in utilizing the MobileNet model for identifying and detecting potential shapes and threats in videos. The MobileNet model will be employed in this research to detect threatening objects, and the results will be analyzed in terms of accuracy and quality.

II. LITERATURE REVIEW

The DL fields are full and rich by the research in this domain. The related work to this contribution was different in the model that they used, or they study the same model to different dataset, or they wanted to detect images instead of

video. There are very limited kinds of literature that have adopted detecting threats by using DL or ML algorithms. This section presents most of the notable research worked on DL, ML, and object detection.

A. Object Detection Algorithms

Object Detection is a very trending domain among researchers. It was thoroughly studied from the beginning of 2010 until the current time [6]. That shows how much this domain is important and rich. Fig. 1 shows the number of publications with the keywords of “Object Detection” during the past 10 years.

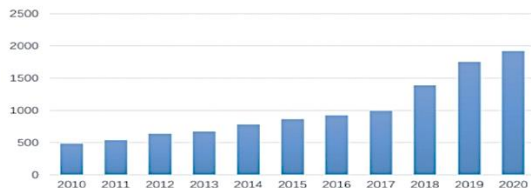


Fig. 1. The growing numbers of publications in object detection algorithms.

Object detection plays a crucial role in computer vision applications, enabling the automatic identification and localization of objects within images or videos. The researcher Shaukat Hayat. el at [7], studied a DL framework for object recognition purposes. The proposed model was further tuned, and the recognition performance was improved. They use nine different object classes taken from wide varied image dataset caltech101 and deploy five layers CNN model. They compared the proposed model's performance with different classical bag-of-words (BOW) approaches trained on a novel. This algorithm achieved an accuracy level of 90.12% is much better than different classical BOW approaches. While Kanimozhi S. el at [8], tried to detect and track the object in the sports field to make the computer learn Deeply, which is none other than the application of DL. The proposed method increases the accuracy level in identifying real-time household objects. Aniruddha Srinivas Joshi, el at [9], adopted the idea of detecting face masks from video footage. A highly effective face detection model is applied for obtaining facial images and cues. A distinct facial classifier is built using DL to determine the presence of a face mask in the facial images detected. The proposed method has shown its good effectiveness in identifying facial masks by achieving high precision, recall, and accuracy. As for using ML algorithms to detect threats included in the social media post, Shatha Alajlan. el at [10] has published research in that domain. She utilized the CNN model on the TensorFlow platform to classify Instagram content (images and Arabic comments) for threat detection. The results of this research showed that the accuracy of the developed model is 96% for image classification and 99% for comment classification.

1) *Using object detection algorithms:* Muhammad Shakeel. el at [11] have developed a passenger security screening system that employs DL techniques to detect potential security threats by rotating the image of a person's body. However, this method has limitations in identifying the specific type of threat and precisely locating its position within the body. Shaoqing Ren. el at [12] have conducted a

study on object classification and detection performance, achieving a remarkable 5-7 frame rate per second and 73.2% mean average precision (mAP). Their approach involves categorizing objects and identifying their precise location, allowing for the development of an efficient security screening algorithm that accurately detects threats at specific locations using an enhanced faster R-CNN model. A comprehensive analysis of cargo X-ray image analysis automation was carried out by Jaccard, Nicolas et al. [13]. The review emphasized the importance of employing image pre-processing techniques, including image quality enhancement, manipulation, material discrimination, and segmentation. These techniques play a crucial role in improving the accuracy of automated image understanding algorithms and rectifying errors that may arise during image acquisition.

Moreover, Jaccard's paper proposes an automated threat detection method to further improve the accuracy in the analysis of cargo X-ray images. Nicolas Jaccard el at. [13] have developed a CNN model specifically designed for detecting threats in X-ray images. Their model was trained using an augmented dataset that included real threat images, resulting in a high detection rate of 90% and a low false alarm rate of only 0.8%. The effectiveness of image manipulation and quality improvement techniques in enhancing the proposed solution is highlighted in their study. In their study, Akcay, Samet et al. [14] investigated the potential of convolutional neural networks (CNNs) for object classification in X-ray baggage images. Their research focused on utilizing CNNs to improve the accuracy of object classification in this domain.

On the other hand, Riffo, Vladimir and Flores Sebastian [15], proposed an innovative automated approach for object detection in X-ray images, specifically for baggage screening purposes. Their solution involved the utilization of an adapted implicit shape model (ASIM), an enhanced version of the implicit shape model introduced in the research conducted by Leibe, Bastian et al. [16]. The ASIM approach employed SIFT descriptors to describe objects using multiple X-ray images from different perspectives. The visual vocabulary of object parts was then used to characterize the object, and targets were detected by searching for similar visual words and spatial distributions. Although the object detector incorporated pose estimation and Q-learning computer vision techniques, it may not be ideal for region-based threat detection.

Furthermore, Nurhopipah, Ade et al. [17] conducted a study that delved into various aspects of motion detection, face detection, data training, and face identification. Their research aimed to explore the complexities associated with these areas in the context of threat detection. Their utilization of the Accumulative Differences Images (ADI) approach for motion segmentation proved successful, with motion detection reaching a high success rate of 92.655%. The Haar cascade classifier was employed for face detection, with a success rate of 76%, and face identification reached 60%. Meanwhile, Busarin Eamthanakul. el at [18] implemented the background subtraction method and median filter to compute data and analyze traffic conditions. Their system was able to detect the

number of objects or cars on the road, providing useful data for traffic management.

Nipunjita Bordoloi. et al [19] developed a security system that effectively tracks object movement and detects anomaly motion in real-time. Background subtraction was utilized to track objects, and the system achieved success in detecting suspicious activity. Alavudeen Basha. et al [20] also delved into suspicious activity detection, using a CNN-DBNN algorithm to detect human activity. The technique of foundation subtraction was used for human detection, with larger bounding boxes to enclose individuals. A Discriminative Deep Belief Network (DDBN) was implemented for activity classification, with an impressive accuracy rate of 90%. The research in computer vision systems continues to advance, providing new and innovative ways to detect and analyze data.

2) *Using mobilenet model:* MobileNet models offer an efficient solution for on-device intelligence across various recognition tasks. Developed specifically for TensorFlow, MobileNets are a family of computer vision models designed with a mobile-first approach. These models prioritize accuracy while considering the limited resources available for on-device or embedded applications. MobileNets fall under the category of lightweight deep convolutional neural networks, significantly smaller in size and faster in performance compared to many other popular models. Their small footprint, low latency, and low power consumption make them well-suited to meet the resource constraints of diverse use cases. MobileNet models can be leveraged for classification, detection, embeddings, and segmentation tasks, providing a versatile framework for on-device intelligent applications.

3) *The Gap on the Literatures:* This research aims to complete what other researchers have done by using DL algorithms and the MobileNet model to detect objects in images and videos. All the studied and reviewed literature were specifying different domain of study or different type of purposes. However, this research focus on Using the MobileNet model to detect the threatening objects on videos. According to previous studies, various researchers have reported high AP values for this model's ability to detect different classes such as cars, persons, and chairs. Some research findings suggest that the AP reaches as high as 99.76%, while others claim to achieve a slightly lower value of 97.76% [21]. Researchers want to approve if the same percentage would be detected with the same purpose they aim to study. This action has not been previously investigated by researchers in the field of object detection by using DL algorithms.

III. METHODOLOGY

This research aims to investigate the effectiveness of the MobileNet model in detecting threatening objects. Object detections a crucial task in computer vision, and it has numerous applications in various domains, such as security, surveillance, and autonomous driving. The research is focused on detecting threatening objects in public places, and the results could have significant implications for improving public

safety as its none of the research objectives. The research approach adopted is a qualitative exploratory approach, which is a suitable method for gaining an in-depth understanding of a phenomenon.

A. Research Design

A research design serves as a structured framework or strategy for collecting, measuring, and analyzing data with the purpose of addressing specific research inquiries [22]. Fig. 2 shows the research scenario that the researcher follows to answer the research question and fulfill its objectives.

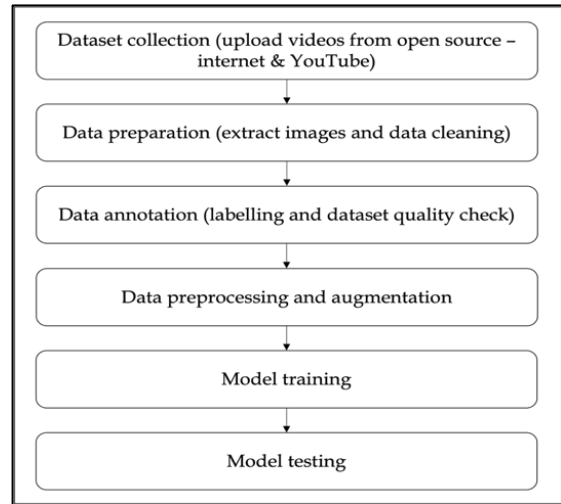


Fig. 2. Research scenario.

B. Data Collection

The dataset used in this study was curated for the purposes of the study. It was generated and collected by the researcher from various sources. It consists of five classes of objects that are considered potentially dangerous or threatening, including: fire, guns, knives, arrows, and swords. These objects were chosen due to their prevalence in public safety incidents and their potential to cause harm. To obtain the data, various sources were utilized, including YouTube videos that contained CCTV footage, educational videos for learning fighting skills, and demo videos. The videos were processed by extracting frames at a rate of 1 frame per second (1 fps) to capture the necessary images for the dataset. The use of CCTV footage is particularly useful as it allows for the collection of authentic data from public places where security cameras are commonly used. To annotate the dataset, the RoboFlow tool was utilized, which is a popular image annotation tool used for object detection tasks. Annotations provide additional information about the images, indicating the location and class of the object within the image. These annotations are essential for training ML models to accurately detect objects. To increase the diversity of the dataset, augmentations such as rotation and contrast difference were applied to the images. These augmentations help to create variations of the images, which can improve the performance of ML models by exposing them to a wider range of data. The purpose behind curing such dataset is essential for training ML models to accurately detect these objects within the five mentioned classes and improve public safety.

C. Data Preparation

Object detection from images is a fundamental task in DL that requires a significant amount of labeled data for training. The MobileNet SSD model is a popular DL model for object detection, but it requires a large, labeled dataset for effective training. In this study, a sufficient amount of data was obtained by collecting threatening videos and CCTV footage from online resources such as YouTube. The videos collected for this study mainly contained real-life activities, such as ATM robberies and police operations, to ensure that the dataset reflects real-world scenarios. Additionally, training and rehearsal-based videos were included to increase the number of images for each object in the dataset. This was an important step because having a larger dataset with a balanced distribution of classes can improve the performance of the model. To create the labeled dataset, frames were extracted from the videos at a rate of 30 frames per second (30 FPS) using the RoboFlow tool. Each frame was manually labeled for the presence of five different objects, including guns, knives, swords, arrows, and fire. Manual labeling is a crucial step in creating a high-quality labeled dataset because it ensures that the labels are accurate and consistent. The use of real-life video footage and manual labeling ensures that the dataset is of high quality and accurately reflects the threatening objects. This can improve the accuracy and reliability of the model when detecting threatening objects in real-world scenarios.

1) *Data cleaning*: After extracting all the images from videos uninformative and vague frames are discarded for data cleaning and maintaining data quality. Only frames with clear object visibility are remained after cleaning that are labelled manually for five object classes. Data classes are sampled in such a way that create a balanced number of images per object in training, validation, and testing.

D. Data Annotation

In this phase, the researcher used the RoboFlow annotation tool that enabled drawing bounding boxes around the objects of interest. RoboFlow enables the uploading of videos and extraction of images with varying FPS rates. Each image is labeled for its particular class name and bounding box. The RoboFlow tool allows saving the bounding box values in different formats, as required by the model. In this case, the labeling is saved as a CSV file to comply with the MobileNet SSD file format. The minimum and maximum values for the bounding box x and y sides are saved to draw the bounding box rectangle. The RoboFlow interface for image labeling is depicted in the Fig. 3. When labeling the images using the RoboFlow tool, a bounding box is drawn around the object of interest in the image. The bounding box is represented by a rectangular box with four values: the x-coordinate and y-coordinate of the top-left corner of the box, and the width and height of the box. To save the bounding box values in the CSV file, the RoboFlow tool records the minimum and maximum values for the x and y coordinates of the top-left corner of the box, as well as the width and height of the box. These values are saved in separate columns in the CSV file, along with the class name of the object in the image. For example, there is an image containing a gun, and the bounding box around the gun has a top-left corner coordinate of (100, 150), a width of 50

pixels, and a height of 100 pixels. The RoboFlow tool would save the following values in the CSV file for this object as follow:

- Class Name: Gun
- Minimum x-coordinate: 100
- Maximum x-coordinate: 150
- Minimum y-coordinate: 150
- Maximum y-coordinate: 250

To start the annotation step, the researcher opened each image using the chosen annotation tool. For every image, the researcher carefully drew bounding boxes around the instances of the objects that aimed to be detected. For each bounding box, the researcher assigned the correct class label from the five mentioned classes which are: fire, gun, knives, arrows, or swords. This step was done manually to ensure accurate placements and labels, as these annotations would serve as the ground truth for the models training phase. With the annotated dataset in hand, the next step was to integrate it with the respective training frameworks. The researcher utilized the annotations they had created to train the object detection models.

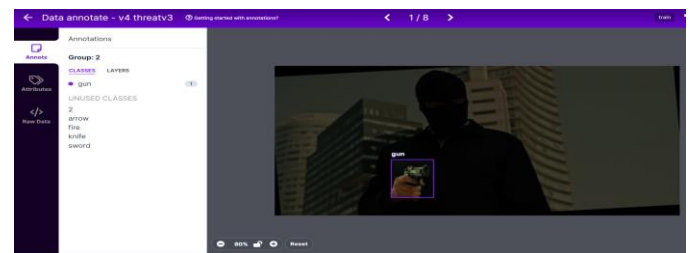


Fig. 3. RoboFlow interface.

1) *Dataset quality check*: Ensuring data quality is a critical step before building models for object detection. Poor data quality can lead to inaccurate and unreliable models. Below the used methods taken to ensure data quality in object detection:

- Annotation Quality Control
- Data Cleaning and Preprocessing
- Balanced Class Distribution
- Data Augmentation
- Validation and Anomaly Detection
- Consistent Image Quality
- Real-World Scenario Simulation
- Review Annotations for Ambiguity
- Class Label Consistency
- Cross-Validation
- Continuous Monitoring
- Use External Data Sparingly

- Expert Review
- Feedback Loop

Once reached the confident in the quality of the annotations, we proceeded to export the annotated dataset from RoboFlow. The platform typically provided options to export the data in formats that were compatible with various deep learning frameworks. Given this scenario, the researchers ensured that the exported format aligned with the chosen model architecture, whether it was YOLO or MobileNet SSD.

E. Data preparation and Augmentation

Data preprocessing is applied in order to make it suitable for effective model training. Images are resized to one scale 416x416 for MobileNet SSD model. Data preprocessing enhances data quality and decreases training time. Images are scaled, cropped, and resized to make them in same format before model training. Data augmentation is also applied to increase veracity of data so model can learn better with more data as addressed before. Images are rotated, blurred, and adjusted contrast and orientation for data augmentation purposes.

F. Data Limitations and Issues

DL models require a large amount of data to train effectively, and the more data feed into the model, the better its performance would be. In this study, the researchers have around 2000 images for each object after augmentations. However, the number of images per sample is relatively low, and the diversity of real scenarios is limited, which are some of the limitations of this dataset. To improve the performance of the model, more data can be collected from real-time scenarios. Furthermore, due to privacy concerns and issues, many establishments and markets are hesitant to share their camera recordings. Therefore, more collaboration and support are needed to collect a diverse range of datasets to improve the model's performance and deliverables.

G. Model Building

This research focus on two DL models. The YOLOv5 (You Only Look Once version five) model, which has undergone thorough scrutiny by researchers, stands out as the initial model that has gained recognition for its effectiveness in object detection. This model showcases highly promising accuracy outcomes based on two key metrics: mAP (mean average precision) and FPS (frames per second). In a study conducted by S. Murthy et al. [23], the application of YOLOv5 was investigated, and it demonstrated superior speed and a 95% accuracy rate compared to other object detection algorithms examined in the comparative analysis. Additionally, it achieved an average precision ranging between 67 and 70, along with a frames per second rate ranging between 65 and 124. A thorough comparison of the YOLO model versions in the below sections. In reference to the Debojit Biswas et al. work that has been done [24] MobileNet SSD model achieved 92.97% average detection accuracy in the experiment. Sanjay Kumar et al. confirm in his work that the SSD on MobileNet has the highest mAP among the models targeted for real-time processing [25]. That was promising to start the investigation upon this case.

1) *YOLOv1*: First object detection network that combines the problem of identifying class labels and determining bounding boxes for a set amount of classes, making it a one-stage detector (rather than two-stage detectors which first detects the regions of interest, and then classify that region as a specific class based on given input during training). This is possible by fully connecting the two important steps of bounding box prediction and classification of labels to an end-to-end differentiable network [26].

2) *YOLOv2*: From its iteration of version one of YOLO, works have been done too dramatically improve the performance of the accuracy through the addition of BatchNorm, improved resolutions, and the use of anchor boxes [27].

3) *YOLOv3*: Improvements made from the previous model included the use of more connections in its backbone network layers as well as adding a new network that aids in the model's ability to identify smaller objects better (with the use of feature pyramid network (FPN) that allows the model to learn objects of different sizes simultaneously). Added an objectness score for the model's bounding box predictions, which helps determine the bounding box to take for all the bounding boxes overlapping a specific ground truth object within an image [26][27].

4) *YOLOv4*: Additional improvements were introduced into the YOLO series in YOLOv4 through the introduction of:

- Feature Aggregation which combines the features extracted from previous layers
- Bag of Freebies - several methodologies and functions added to improve its performance without affecting the model's inference during production. The main additions are related to data augmentation such as rotation, flip, crop, hue, saturation, mosaic, MixUp, Blur, etc.
- Self-Adversarial Training which allows the model to find the region of the image that its network relies most on and subsequently editing the image to remove this reliance to enable generalisation of the model.
- CIoU loss as the loss function which not only observes the overlap of bounding boxes with the ground truth (which is already done for IoU), but also how close the box was to the ground truth box in terms of the pixel distances within the image, which is an additional part of the loss function that is trained so that it enables the network to pull the predicted bounding box closer to the ground truth box.
- Using Mish activation as the activation function instead of ReLU which improves the performance of the model due to its ability to push the features created by the model towards its optimal [27][28].

5) *YOLOv5*: YOLOv5 represents the most recent iteration of the YOLO (You Only Look Once) series of object detection models, originally introduced in 2016. Developed by Ultralytics, a reputable computer vision research company,

YOLOv5 offers notable advancements in both accuracy and speed when compared to its predecessors. It attains state-of-the-art performance across multiple benchmark datasets while preserving real-time inference speeds on modern GPUs. This algorithm employs a single convolutional neural network (CNN) to predict object classes and bounding boxes by dividing the input image into a grid and making predictions based on each grid cell. This approach enables faster inference times and improved accuracy compared to region-based CNNs used in other object detection models. YOLOv5 has gained significant traction within the computer vision community and finds applications in various domains, including autonomous vehicles, robotics, security, and more [26]. YOLOv5 contains three layers as an object detection model: Backbone as the feature extractor, the Neck which combines and mixes different features extracted from the Backbone, and the Head which takes the outputs from the Neck and predicts bounding boxes and classification. The Backbone that mentioned in Fig. 4 was CSPDarknet. CSPDarknet is a neural network that contains a set of convolutional layers which are useful for consolidating images and extracting useful features which can be learnt from the model. As shown in Fig. 4, the Backbone consists of a set of BottleNeckCSP (Cross Stage Partial) blocks and a Spatial Pyramid Pooling (SPP) block. The bottleneck part of BottleNeckCSP helps reduce the number of feature maps, which in turn reduces the model size and computation. The cross stage partial part of BottleNeckCSP also aids in reducing the model size and computation required by reducing the amount of gradient information during optimization within the network while maintaining good accuracy for the model. It does this by splitting the base layer into two parts (one as the base layer, the other is partitioned into multiple blocks) and merging them back together again through a cross-stage hierarchy strategy. The SPP block performs pooling of the features from the previous CSP block to generate fixed-length outputs. This avoids the need to do any cropping, warping or preprocessing at the start of the input to the neck and is done through pooling which is an information aggregation function [30].

After the Backbone, the feature mixing and combining model used was Path Aggregation Network (PANet). PANet is a network architecture with a bottoms-up approach, where there is a feature hierarchy which aggregates and passes the information of multiple convolutional layers at different stages, enhancing the signals between lower layers and upper layers. Linking different feature levels together to allow the model to accurately detect both larger and small objects when performing object detection.

As shown in Fig. 4, there are several concatenation blocks which combine the lower and higher-level features together to be fed into the final head layers. The final layer is a set of 1x1 convolutional layers that takes in the input of the Neck (PANet) to pass into the regression that detects the bounding boxes and classifies them, and these are then used for training and inference/prediction [28]. During training, YOLOv5 will see the images inputted from the training dataset, use the

Backbone (CSPDarknet) to extract out relevant features, thereafter, utilizing the PANet to concatenate lower and higher-level features together, and these are finally passed to output the bounding boxes and classes for different region of the image as predictions. This will be trained using the training dataset and can be used for inference after enough training is done for the model [29].

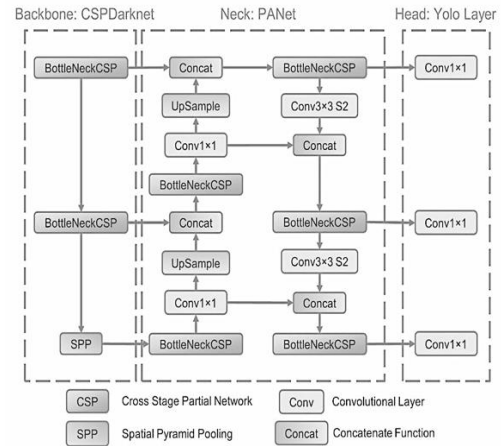


Fig. 4. YOLOv5 model architecture.

6) *MobileNet Single Shot MultiBox Detector (SSD):* The MobileNet Single Shot MultiBox Detector (SSD) is an object detection algorithm that combines the Single Shot Detector (SSD) framework with the MobileNet architecture. This algorithm was developed by Wei Liu et al. in 2016 [31]. MobileNet is specifically designed for mobile and embedded devices, offering a lightweight convolutional neural network architecture that utilizes depth-wise separable convolutions. These convolutions help reduce computational requirements while maintaining high accuracy. On the other hand, the SSD framework is a popular approach for object detection that utilizes a single convolutional neural network to predict object classes and bounding boxes [31]. MobileNet SSD enables real-time object detection on devices with limited computational resources, such as mobile devices and robotics. It is particularly beneficial for applications that require real-time object detection, including autonomous vehicles and surveillance. Despite its efficiency, MobileNet SSD achieves high accuracy on benchmark datasets like PASCAL VOC and COCO, all while maintaining fast inference times. As a result, it has gained significant adoption within the computer vision community and finds applications in various domains such as security, surveillance, and robotics [15][24]. MobileNet SSD comprises two key layers: a backbone model used to extract relevant features (in this case, VGG-16 was employed as the feature extractor), and the detector head, which outputs crucial information for object detection.

The VGG-16 model, proposed by researchers at the University of Oxford in 2014, serves as the backbone for many computer vision tasks. It is a convolutional neural network architecture comprising 16 layers, including 13 convolutional layers, 5 max pooling layers, and 3 fully connected layers. The

primary purpose of the convolutional layers is to extract meaningful features from the input image, enabling the model to capture relevant patterns and structures. On the other hand, the pooling layers play a crucial role in reducing the spatial dimensionality of the extracted features. By down sampling the feature maps, these pooling layers enhance computational efficiency during subsequent processing stages. Together, the combination of convolutional and pooling layers in the VGG-16 architecture enables effective feature extraction and representation for a wide range of computer vision applications. Finally, the fully connected layers learn to classify the extracted features into their corresponding categories. The detector head consists of several convolutional blocks that link to the detection block, as well as a post-processing step called Non-Maximum Suppression (NMS) as shown in Fig. 5. The purpose of the convolutional blocks that are linked at different levels to the detection head is to extract features at multiple levels, which enables the model to detect both small and large objects by extracting features for them. Subsequently, the NMS block will take the bounding boxes that are outputted from the model and pick out the bounding box that is closest to the ground truth bounding box using Intersection-over-Union (IoU) as the metric. NMS is only used during training. As an overview for the methodology of MobileNet SSD, the model will be fed the images from the training dataset, which goes through a feature extraction process in VGG-16 as the backbone. These feature extractors are then further convoluted, and features at different levels of convolutions are passed to the detection head for bounding box and classification prediction. For training, there is an additional process of NMS which choose the most prominent bounding box for each ground truth box. The pretrained SSD MobileNet v1 FPN with dimension of 640x640 were used for detecting the objects. it is an object detection model based on a single-shot detection (SSD) architecture with a feature pyramid network (FPN) and uses the MobileNet V1 neural network as a base feature extractor. This model is designed to detect objects in images of size 640x640 pixels. The SSD architecture is a popular object detection approach that predicts object categories and bounding boxes in a single forward pass through the neural network. The SSD MobileNet v1 FPN 640x640 model consists of a base network, feature pyramid network, and detection network.

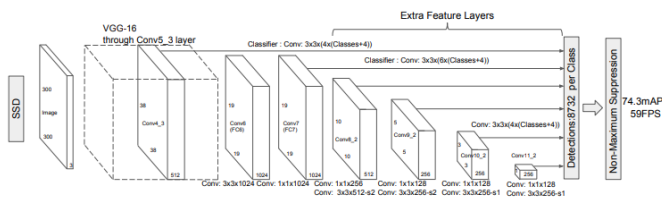


Fig. 5. MobileNet Single Shot Multi box Detector (SSD).

- **Base Network:** The base network of the model is the MobileNet V1 neural network. It is a lightweight deep neural network architecture that uses depth wise separable convolutions to reduce the number of parameters and improve computational efficiency. The MobileNet V1 architecture consists of a sequence of depth wise separable convolutional layers followed by

standard convolutional layers, which are used to extract feature maps from the input image.

- **Feature Pyramid Network:** The feature pyramid network (FPN) is used to combine feature maps from different levels of the MobileNet V1 base network. The FPN is a top-down architecture that aggregates high-resolution feature maps from the lower levels of the base network with lower-resolution feature maps from the higher levels of the network. This creates a pyramid of feature maps with rich semantic information at multiple scales, which is useful for detecting objects of varying sizes in the input image.
- **Detection Network:** The detection network is used to predict the bounding boxes and object categories in the input image. The detection network consists of a set of convolutional layers that process the feature maps generated by the FPN. These layers are used to predict the locations and class scores of the objects in the input image. The SSD MobileNet V1 FPN 640x640 model uses a set of default anchor boxes at different scales and aspect ratios to generate object proposals. These proposals are then refined by the detection network to improve the accuracy of the final object detection results.

The SSD MobileNet V1 FPN 640x640 is a powerful object detection model that combines the strengths of the MobileNet v1 architecture, the feature pyramid network, and the single shot detection approach to achieve high accuracy and computational efficiency.

a) Parameters of the MobileNet SSD Model

The parameters for this model are as follows:

- **Backbone architecture:** This model uses a MobileNet V1 architecture as the backbone. MobileNet is a lightweight convolutional neural network architecture designed for mobile devices, which makes it suitable for real-time object detection on low-power devices.
- **Feature Pyramid Network (FPN):** This model uses a Feature Pyramid Network (FPN) to generate a multi-scale feature map. FPN is a technique used to extract features from images at different scales, which helps improve the accuracy of object detection.
- **Input size:** The input size for this model is 640x640 pixels. This means that the model can detect objects in images up to 640x640 pixels in size.
- **Batch size:** The batch size is the number of images that are processed simultaneously. The batch size of 8 was used for training the model.
- **Learning rate:** The learning rate is a hyperparameter that controls how much the model adjusts its parameters during training.
- **Number of classes:** The number of classes is the number of object categories that the model can detect. In this case it was five classes.

- **Anchor boxes:** Anchor boxes are a set of predefined bounding boxes of different sizes and aspect ratios that the model uses to detect objects.
- **NMS threshold:** The Non-Maximum Suppression (NMS) threshold is a parameter that controls how much overlapping bounding boxes are merged into a single detection. The default NMS threshold for this model is 0.6, but it can be changed depending on the specific use case.

b) Model Architecture

The details of model architecture with parameter setting for threat detection is given as:

- **Number of classes:** This specifies the number of classes or object categories that the model will detect. In this case, the model is trained to detect five classes.
- **Image resizer:** This defines how the input image is resized to fit the input size of the model. Here, a fixed shape resizer is used with a height and width of 640 pixels.
- **Feature extractor:** This defines the feature extraction backbone of the model. In this case, the SSD MobileNet V1 FPN Keras architecture is used. The depth multiplier is set to 1.0, and the minimum depth of the network is set to 16. The conv hyperparams section specifies the hyperparameters used for the convolutional layers of the feature extractor, including the L2 regularization weight, random normal weight initializer, and batch normalization parameters.
- **Override base feature extractor hyperparams:** This indicates that the hyperparameters specified in this pipeline config file will be used to override the default hyperparameters of the base feature extractor.
- **FPN:** This specifies the Feature Pyramid Network used for multi-scale feature extraction. The minimum and maximum levels of the feature pyramid are set to 3 and 7, respectively.
- **Box Coder:** In object detection, the box coder is used to encode and decode the predicted boxes, which is necessary because the predicted boxes are in a relative format and need to be converted back to the absolute coordinates of the image. The box coder section specifies the method used to encode and decode boxes. In this particular case, the box coder is using the faster R-CNN box coder method, which encodes boxes using their center coordinates, width, and height. The y scale and x scale values specify the scaling factors for the center coordinates, while the height scale and width scale values specify the scaling factors for the height and width. These scaling factors are used to normalize the box coordinates to a similar range.
- **Matcher:** The matcher section specifies the method used to match predicted boxes to ground truth boxes. In this case, the argmax matcher method is used, which matches predicted boxes to ground truth boxes based on their maximum intersection-over-union (IoU) overlap.

The matched threshold value specifies the minimum IoU overlap required for a predicted box to be considered a match, while the unmatched threshold value specifies the maximum IoU overlap allowed for a predicted box to be considered unmatched.

- **Similarity Calculator:** It specifies the method used to calculate the similarity between predicted boxes and ground truth boxes. In this case, the IoU similarity method is used, which calculates the IoU overlap between two boxes.
- **Box Predictor:** In the SSD MobileNet V1 FPN 640x640 model, the box predictor is responsible for predicting the bounding boxes for the detected objects. The weight shared convolutional box predictor is used as the box predictor, which shares weights between the class prediction and box prediction layers. This helps to reduce the number of parameters in the model. The depth parameter specifies the number of filters in each convolutional layer of the box predictor. In this model, it is set to 256.
- **Number of layers before Predictor:** this parameter specifies the number of convolutional layers before the predictor layers. In this model, four convolutional layers are used before the predictor.
- **Kernel Size:** It specifies the size of the convolutional kernel used in the predictor layers. In this model, a kernel size of 3 is used.
- **Class prediction Bias init:** It initializes the bias for the class prediction layer. In this model, it is initialized to -4.599999904632568.
- **Convolutional Hyperparameters:** It specifies the hyperparameters for the convolutional layers in the box predictor. It includes the regularizer, initializer, activation function, and batch normalization parameters.
- **L2 Regularizer:** It applies L2 regularization to the convolutional layers to prevent overfitting. The weight value provided is 3.9999998989515007e-05.
- **Random Normal Initializer:** This parameter initializes the weights of the convolutional layers using a normal distribution with a mean of 0 and a standard deviation of 0.009999999776482582.
- **Activation:** The activation parameter specifies the activation function used in the convolutional layers. In this model, the RELU_6 activation function is used.
- **Batch Normalization:** It applies batch normalization to the convolutional layers to improve the training process. It includes the decay rate, scale, and epsilon values. In this model, the decay rate is set to 0.996999979019165, the scale is set to true, and the epsilon is set to 0.0010000000474974513.
- **Anchor boxes:** In object detection, anchor boxes are pre-defined bounding boxes of various sizes and aspect ratios that are used to identify objects in an image.

- Anchor Generator: It specifies how these anchor boxes should be generated. In the SSD MobileNet V1 FPN 640x640 model, the anchor generator uses the multiscale anchor generator which generates anchors at multiple scales and aspect ratios. The values provided in the multiscale anchor generator section is the following:
- Min level and max level: These specify the minimum and maximum levels of feature maps in the FPN.

In this case, the feature maps are generated at levels 3 to 7.

- Anchor Scale: This specifies the base size of the anchor boxes. The size of the anchor boxes is proportional to the square root of the area of the feature map.
- Aspect Ratios: These specify the aspect ratios of the anchor boxes. In this case, three aspect ratios are used: 1.0, 2.0, and 0.5.
- Scales per Octave: This specifies the number of scales to be used per octave. In this case, two scales are used per octave.
- Score Threshold: minimum confidence score for detections to be considered. Value is 9.9999993922529e-09.
- IoU Threshold: intersection over union (IoU) threshold used for non-maximum suppression. Value is 0.600000238418579.
- Max Detections per Class: maximum number of detections to keep per class after non-maximum suppression. Value is 100.
- Maximum total Detections: maximum number of detections to keep over all classes after non-maximum suppression. Value is 100.
- Use Static Shapes: whether to use static shapes for the output tensor shapes. Value is false.
- Score Converter: method for converting scores. Value is SIGMOID.
- Normalize loss by number of Matches: whether to normalize the total loss by the number of matched ground truth boxes. Value is true.
- Localization Loss: `weighted_smooth_l1`: the localization loss function. No values provided, uses default parameters.
- Freeze Batch norm: whether to freeze the batch normalization parameters during training. Value is false.
- Batch Size: The number of images that are fed into the network at once during training. In this case, the batch size is set to 8.
- Data Augmentation Options: A list of data augmentation options to apply to the input images during training. In this case, two types of data

augmentation are used: random horizontal flips and random crops.

- Sync Replicas: A Boolean variable that controls whether to use synchronous gradient updates during training. When set to true, the gradients are computed and averaged across all replicas before the weights are updated. This can lead to better convergence but requires more memory and communication.
- Optimizer: Specifies the optimizer used during training. In this case, the momentum optimizer is used with a cosine learning rate schedule.
- Learning Rate: The learning rate schedule used during training. The learning rate is decreased according to a cosine schedule that decreases the learning rate from a base value of 0.04 to a final value of 0 over 25,000 steps. The learning rate is also gradually increased from a warmup value of 0.0133 over 2,000 steps.
- Momentum Optimizer Value: The momentum value used by the optimizer. In this case, the momentum is set to 0.9.
- Use Moving Average: A Boolean variable that controls whether to use a moving average of the model weights during training. When set to false, the raw weights are used. When set to true, the moving average of the weights is used instead, which can improve the robustness of the model.

7) *Python*: Model implementation, training and evaluation is done in python programming using several libraries as listed below:

- PyTorch: PyTorch is a python library, a deep learning framework for building and training neural networks, widely used for research and production in machine learning programming language and the Torch library. Torch is an open-source ML library used for creating deep neural networks and is written in the Lua scripting language. It's one of the preferred platforms for deep learning research. Outcome of PyTorch is a model file that can be loaded in mobile device and can be used for prediction. The researchers used PyTorch for MobileNet and yolov5 implementation, training, and evaluation.
- TensorFlow: An open-source machine learning framework for developing and deploying machine learning models, including deep learning models. Both yolov5 and MobileNet SSD models can be implemented in PyTorch and TensorFlow these are just two standard libraries for implementing neural networks. Researchers tried TensorFlow for implementation, but PyTorch was more user friendly, so PyTorch was adopted.
- OpenCV: An open-source computer vision library offering tools for image and video processing, including object detection and analysis processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and

videos to identify objects, faces, or even the handwriting of a human. Researchers have used OpenCV for reading images and applying preprocessing steps like scaling and normalization of images.

- RoboFlow: A platform for managing, annotating, and preprocessing data for computer vision projects, assisting in training machine learning models.
- Matplotlib: A Python plotting library used for creating static, interactive, and animated visualizations in data analysis and model output visualization. Matplotlib is also used for evaluation and analysis of results like building confusion metrics after prediction is done through this library. Outcome of matplotlib plots and charts generated as images that can be used for visualization of training loss and accuracy with each epoch.
- Seaborn: A statistical data visualization library built on top of Matplotlib, designed to generate informative and attractive statistical graphics. Seaborn is also used for plotting various analysis charts of training loss and accuracy values.

H. Model Evaluation

1) *Model evaluation metrics:* Mean Average Precision (MAP) is the universal standard metric used to compare performance between object detection models created by different authors [32]. This metric is specifically derived from Average Precision (AP). Since classification is performed during object detection for different bounding boxes along with the provision of the ground truths, the fundamentals of the confusion matrix apply. The matrix enables computations to be made with accuracy, precision, and recall. It consists of the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values [32]. In the context of object detection:

- True Positive: Detection made correctly by the model.
- True Negative: Background region correctly detected by the model (where there are no objects)
- False Positive: Wrongly detected regions made by the model.
- False Negative: Regions where the ground truths are missed by model.

Intersection-over-Union (IoU) is the next metric used to determine whether bounding boxes are TP, TN, FP or FN. IoU is defined as the area of overlap between the bounding box predictions and the ground truth, divided by the area of union between them [33]. An IoU of 1 means the bounding boxes predicted match exactly the ground truth boxes, whereas an IoU of 0 depicts no overlap between the two bounding boxes [34]. Fig. 6 shows the IoU equation.

A threshold is a hyperparameter predetermined to decide between TP, TN, FP or FN. For example, with a threshold of 0.5 for a ground truth bounding box, if the predicted bounding

box has an IoU greater than 0.5 for that particular ground truth, it is considered a TP. Whereas an IoU lower than 0.5 means the predicted bounding box is a FP. Through the IoU, we are able to determine the confusion matrix (TP, TN, FP, FN) for every bounding box, and subsequently calculate the precision and recall.

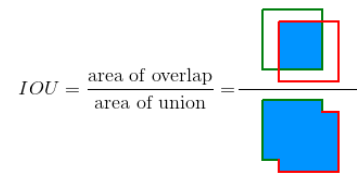

$$IOU = \frac{\text{area of overlap}}{\text{area of union}}$$

Fig. 6. Intersection over union equation.

- Precision: of the positive classes that are correctly detected, how many are actually positive? This follows the following Eq. (1) [33]:

$$\text{Precision} = TP / TP + FP \quad (1)$$

- Recall: of all positive classes, how much can we predict the class correctly? It is preferred that this measure is as high as possible. It follows the following Eq. (2) [33]:

$$\text{Recall} = TP / TP + FN \quad (2)$$

We also, calculate the accuracy that measure considers the correct classification out of all classes, where a high value of accuracy is preferred. This factor is explained in the following Eq. (4) [33]:

$$\text{Accuracy} = TP + TN / TP + FP + TN + FN \quad (3)$$

AP uses precision and recall creating an Area Under the Curve graph (AUC-PR) for the model. For every threshold, there is a different precision since the object detector will output a confidence score, which is then determined by the threshold on whether each bounding box is TP, TN, FP or FN. AP will take the precision and recall at every threshold, graph out a precision-recall plot, and thereafter take the area under the curve. The closer AUC is to 1, the better the model, and vice versa. It is better to have a high AUC as the model is good at predicting and distinguishing between classes it calculated based on the TPR (y-axis) versus the FPR (x-axis). AP is done separately for each class within the object detection model. Also, F1-measure that measure mainly computes the harmonic mean of precision and recall measuring them at the same time [33]. It has the below Eq. (4):

$$F1\text{-measure} = 2 \times \text{Recall} \times \text{Precision} / \text{Recall} + \text{Precision} \quad (4)$$

To consolidate all these scores into one metric, mAP was introduced. mAP will take all the AP of each class (will have n number of scores for n number of classes) and take the mean of those AP to obtain one score. This score is the metric used to determine the overall performance of the object detection model.

2) *Model validation:* Model validation refers to the procedures and actions conducted to verify that a model is functioning as intended and aligns with its objectives and intended business applications. Typically, the validation

process involves the assessment performed by individuals who are not the model developers or owners, as their impartial perspective is valuable due to their non-technical background [31]. In the context of machine learning, validation may involve ML experts evaluating the labeling process to ensure its accuracy and reliability.

IV. RESULTS AND DISCUSSION

This section aims particularly to answer the research questions which is: is there any efficient way to detect threats from live videos using DL algorithms? The other question was how accurate and efficient is using the MobileNet model to detect objects from live videos? The analysis has been conducted according to detailed steps that will be mentioned on the analysis Section 4.3. Comparison is done based on mean average precision (mAP) and frames per second (FPS) on the datasets collected as a part of research findings.

A. Data Preparation

1) *Dataset collection*: For the purpose of this study, a carefully curated dataset was created to support the research objectives. The dataset was generated and collected by the researcher from various sources, specifically chosen to include objects that are considered potentially dangerous or threatening. The dataset comprises five classes of objects: fire, guns, knives, arrows, and swords. These object classes were selected due to their relevance in public safety incidents and their potential to cause harm.

To obtain the necessary data, a range of sources was utilized. This included gathering footage from YouTube videos that contained CCTV recordings, educational videos demonstrating fighting skills, and demo videos. By extracting frames from these videos at a rate of 1 frame per second (1fps), the required images for the dataset were captured. The inclusion of CCTV footage is particularly valuable as it provides authentic data from public places where security cameras are commonly employed.

To annotate the dataset with the necessary information for object detection, the researcher employed the RoboFlow tool. RoboFlow is a widely used image annotation tool specifically designed for object detection tasks. Annotations provide crucial additional details about the images, such as the precise location and class of the object within each image. These annotations are vital for training machine learning models to accurately detect and classify objects. To enhance the diversity of the dataset and improve the performance of the machine learning models, various augmentations were applied to the images. Techniques such as rotation and contrast adjustment were employed to create variations of the original images.

By introducing these augmentations, the models were exposed to a wider range of data, enabling them to better handle different image conditions and variations. The careful curation of this dataset, encompassing the five specified object classes, serves as a crucial foundation for training machine learning models to accurately detect these objects and contribute to public safety improvements.

2) *Data cleaning*: In this research, the process of data cleaning and maintaining data quality played a crucial role in preparing the dataset for object detection models. After extracting images from videos, uninformative and vague frames were carefully discarded to ensure that only relevant and clear frames were included in the dataset. This step aimed to eliminate any noise or ambiguity that could hinder the performance of the models. The remaining frames with clear object visibility were then subjected to manual labeling for five object classes. Manual labeling involves human annotators carefully marking the objects of interest in each frame, providing accurate ground truth annotations. To ensure a balanced distribution of images per object class in the training, validation, and testing sets, the data classes were sampled strategically. This sampling process helps prevent bias towards specific object classes and ensures that the models are exposed to a diverse range of objects during training and evaluation. By performing data cleaning, manual labeling, and strategic sampling, the researchers improved the overall quality and representativeness of the dataset. This, in turn, enhances the reliability and generalizability of the object detection models, allowing them to effectively detect and classify objects in various real-world scenarios.

3) *Data preprocessing*: To ensure effective model training, data preprocessing techniques were applied to the dataset. One of the key preprocessing steps involved resizing the images to a standardized scale of 416x416 pixels, which is suitable for the MobileNet SSD model input. This resizing step helps to ensure consistency in the input size across all images, facilitating efficient model training. Data preprocessing serves to enhance the quality of the data and reduce training time. In addition to resizing, other preprocessing operations were applied to make the images compatible with the model requirements. These operations included scaling, cropping, and further resizing to bring all images into a consistent format prior to model training. By standardizing the images, the model can effectively process and analyze them. Data augmentation techniques were also employed to increase the diversity and veracity of the data, thereby enabling the model to learn better. Data augmentation involves applying various transformations to the images to create additional training samples. These transformations include rotation, blurring, and adjustments in contrast and orientation. By introducing such variations, the model becomes more robust and capable of handling different image conditions and variations that may be encountered in real-world scenarios. The combination of data preprocessing, including resizing and standardizing the images, along with data augmentation techniques, enhances the quality, variety, and quantity of the training data. This, in turn, contributes to the overall performance and generalization capabilities of the object detection model during training and subsequent inference tasks.

4) *Data annotation*: During this phase, the researcher utilized the RoboFlow annotation tool to facilitate the

annotation process for object detection. This tool allowed for the drawing of bounding boxes around the objects of interest in the images. By uploading videos into RoboFlow, images with varying frames per second (FPS) rates were extracted. Each image was then labeled with its corresponding class name and bounding box. For each image, bounding boxes were manually drawn around the instances of the objects to be detected. The researcher carefully assigned the correct class label from the predefined set of five classes: fire, gun, knives, arrows, or swords. This manual annotation process ensured accurate placement of the bounding boxes and correct labeling, as these annotations served as the ground truth for the subsequent model training phase. Once the dataset was annotated, the next step involved integrating it with the respective training frameworks. The annotations created by the researcher were utilized as the training data for the object detection models. These annotations, combined with the corresponding images, formed a labeled dataset that could be used to train the models and enable them to detect and classify objects accurately.

a) Data Quality Check

Below is the explanation of the used methods and steps taken to ensure data quality in object detection:

- **Annotation Quality Control:** Ensure accurate and consistent annotation of bounding boxes and class labels in the dataset. Annotators should follow clear guidelines and have a solid understanding of the objects of interest. This research has five object types (guns, swords, knives, arrows, and fire) in the dataset. Researchers have annotated each object carefully using RoboFlow. Bounding boxes were created with extreme care.
- **Data Cleaning and Preprocessing:** researchers removed duplicate images after annotations to build a good model. Also, we have eliminated corrupted images or annotations that might negatively impact training.
- **Balanced Class Distribution:** researchers ensured that the dataset has a balanced distribution of objects across classes to prevent bias towards dominant classes and improves the model's ability to detect all classes accurately. Normal videos are recorded at 30 fps, means 30 frames per second can be extracted and in this case, 1 frame/sec was extracted. There were some images that do not have any object, such images were deleted, and remaining images were annotated accurately. Exact number of samples before augmentation for all objects is provided in Table I below:
- **Data Augmentation:** researchers have applied data augmentation techniques such as random rotation between -15 degree to + 15 degree to increase the number of data samples.

TABLE I. CLASSES SAMPLES

Classes	Train	Test	Valid
Arrow	1286	186	360
Gun	1362	209	379
Sword	1370	221	394
Fire	1297	197	353
Knife	1312	198	369

- **Validation and Anomaly Detection:** researchers have eliminated images that have no object or object is not clearly visible, etc.
- **Consistent Image Quality:** researchers have ensured that images are of consistent quality and resolution. They have applied resizing for all images to be 416x416.
- **Real-World Scenario Simulation:** researchers have collected videos from demos, CCTV videos and social media to collect diverse and real-world scenarios to train a good model.
- **Review Annotations for Ambiguity:** researchers have reviewed annotations that are ambiguous or challenging for the model to detect, such as partially occluded objects or objects in cluttered scenes.
- **Class Label Consistency:** researchers have verified that class labels are consistent across annotations. With every annotation object name was specified with that annotation to make sure that each object has its correct name.
- **Cross-Validation:** Divide the dataset into training, validation, and test sets. Cross-validation can help assess how well the model generalizes by training on one subset and testing on another.
- **Continuous Monitoring:** Continuously monitor and update the dataset as needed. Over time, as the model's requirements change or new challenges arise, the dataset should evolve accordingly.
- **Use External Data Sparingly:** When using external data sources like stock images or online datasets, ensure that they are relevant and high-quality. External data should complement the dataset without introducing noise.
- **Expert Review:** researchers engaged experts' volunteers to review and validate the quality of the dataset, ensuring that the annotations and data align with the real-world scenarios.
- **Feedback Loop:** Establish a feedback loop with annotators to address questions, provide clarification on guidelines, and continually improve annotation quality.

5) Model evaluation

a) Results of Performance Metrics Comparison

In Table II researchers indicate the difference across different aspects. These results are achieved after training both models on same dataset. A dataset is divided into three parts which are: training, validation, and testing. For each class 70%, 20% and 10% images are used for train, valid test, respectively.

- Train set is provided to the model during training so model can learn pattern from this data.
- Validation set is used to evaluate model during training, this is unseen for model but during training results are analyzed through this unseen data. If model is not learning correctly then we tune the parameters of model to see if it is performing good on train data and validation data.
- Test set is totally unseen that is used after correct training of model, it depicts the real-world testing of model on unseen data. If a model performs as good on testing data as it is for training, then model is considered to be reliable for that task.

TABLE II. PERFORMANCE METRICS RESULTS

Performance Metrics	MobileNet SSD	YOLOv5
mAR (Mean Average Recall)	0.9565	0.8450
mAP (Mean Average Precision)	0.9125	0.7549
IoU (Intersection over Union)	0.9045	0.8020
False Positive Rate	0.053	0.078
False Negative Rate	0.053	0.078
Inference Speed	~ 18ms/image (CPU)	~ 41ms/image (CPU)
Memory Usage	1.3 GB	7 GB
Model Size	6.6 mbs	15 mbs
Class-wise Performance	90%	83%
F1 score	0.92	0.81

To evaluate and measure the model performance AUC method were used. This method is used to check the ability of the model to detect accurately among the different classes. The higher the AUC, the better the performance of the model is. Fig. 7 illustrates the AUCs of the MobileNet model.

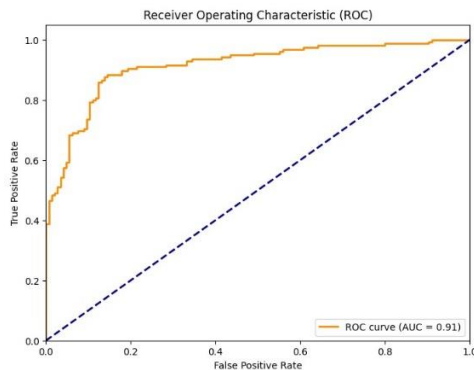


Fig. 7. ROC AUC curve.

The graph in Fig. 7 provides the following results:

- Since AUC = 0.91, the model is able to distinguish perfectly between all positive and negative class points.

The graph in Fig. 8 and 9 illustrate the results of the confusion matrix. The confusion matrix plot shows the predicted vs. true labels, and the values in each cell represent the percentage of the correct and incorrect classifications. Diagonal values are high as they show values for correct classification and off diagonal values are incorrect classification. As per the graphs the MobileNet model prove that it detects the classes efficiently over the Yolov5 model.



Fig. 8. Confusion matrix for mobilenet SSD model.



Fig. 9. Confusion matrix for YOLO model.

6) *Model validation:* In this research, the model validation was a meticulous process that contributed significantly to the success of the object detection models. The researcher dedicated time and efforts to ensure the annotations were accurate and comprehensive, as this foundation would directly impact the performance of the models in real-world scenarios. This validation step is carried out by ML experts who as their impartiality is crucial as the context of the labeling process, a ML expert is often involved in performing this step to validate the accuracy and quality of the labels assigned to the data [31]. There were experts in ML who volunteering to examine the validity of the model. The feedback from the experts were that all the dataset was accurately labeled since it was at first a manual step that takes a lot of time and efforts.

For the validity of the model, Table II examines the success rates, encompassing accuracy and F1-measure outcomes. The MobileNet SSD model exhibits an impressive success score of 0.92. Moreover, it achieves an accuracy rating of 96.5%. These findings collectively validate the superior performance of the

MobileNet model over the YOLO model in detecting threatening objects.

7) *Discussion*: In conclusion, MobileNetSSD performs better than YOLOv5 in the scenarios of detecting threatening objects due to its fast inference speed, memory efficiency, optimization for small objects, and training strategy as shown in Table II. Below are additional reasons why MobileNet model preferable over YOLO5 model. Certainly, here's the list of scenarios tailored to the context of the problem, considering the involvement of detection for five classes including: fire, gun, knives, arrows, and swords with a dataset of 2000 images per and a deployment on a mobile platform:

a) *Mobile-Optimized Inference Speed*: MobileNet SSD's fast inference speed is crucial for mobile deployments. It ensures that objects are detected rapidly on the mobile device, enhancing real-time detection capabilities.

b) *Memory Efficiency for Mobile Devices*: Deploying on mobile devices demands efficient memory usage. MobileNet SSD's architecture reduces memory requirements, allowing smooth operation on resource-constrained mobile platforms.

c) *Real-Time Threat Detection on Mobile*: MobileNet SSD's quick detection of objects like guns and knives is vital for real-time threat detection scenarios on mobile devices, such as identifying potential weapons in public spaces.

d) *Streaming Video Analysis on Mobile*: MobileNet SSD's fast inference speed aligns well with streaming video analysis on mobile devices. This is valuable for continuous monitoring using mobile cameras.

e) *Optimization for Small Objects on Mobile*: MobileNet SSD's specialization in detecting small objects, like arrows or knives, is advantageous for accurate detection on mobile screens, where these objects might appear relatively small.

f) *Responsive Fire Detection on Mobile*: Fast detection of fire instances using MobileNet SSD on mobile devices is critical for timely response to fire incidents, aiding firefighting efforts and safety protocols.

g) *Edge Computing for Mobile*: Deploying MobileNet SSD on mobile platforms extends the benefits of edge computing. The model's lightweight architecture is suitable for processing data on the device, reducing latency.

h) *Mobile Surveillance Solutions*: MobileNet SSD's deployment on mobile devices allows for portable surveillance solutions. Users can leverage their mobile phones for security monitoring, quickly detecting threats like fires or intruders.

i) *Accurate Object Detection on Mobile*: MobileNet SSD's optimization for small object detection ensures accurate identification of objects like arrows or swords on mobile screens, where details matter.

j) *Reduced Data Transmission*: MobileNet SSD's on-device detection reduces the need for transmitting sensitive data to remote servers, maintaining user privacy and potentially reducing data costs.

k) *User-Friendly Mobile Applications*: The combination of fast detection and accuracy makes MobileNet SSD suitable for developing user-friendly mobile apps that offer intuitive and effective object detection functionalities.

l) *Cost-Effective Mobile Deployments*: MobileNet SSD's low computational demands align with mobile platforms, making it a cost-effective choice for deploying object detection capabilities on mobile devices

V. RECOMMENDATIONS FOR FUTURE WORK

Several recommendations can be made for future study that will extend the present study's findings. Below are some possible recommendations for future studies:

1) *Expand* the scope of the study: The current study may have focused on a specific aspect or application of the topic. Future studies could expand the scope of the research to include other related areas, applications, or datasets.

2) *Improve* the performance of the model: The current study may have achieved good results with the model used, but, there may be other models or techniques that could improve performance further. Future studies could explore alternative models or techniques for the task and compare their performance.

3) *The current study* may have some limitations due to the dataset used. Future studies could address these limitations by using different datasets, models, or evaluation metrics.

4) *Explore* ethical considerations: The current study may not have explicitly addressed the ethical implications of the research. Future studies could explore the ethical considerations of the research and investigate ways to ensure that the technology is used ethically and responsibly.

ACKNOWLEDGMENT

My deepest gratitude and sincere appreciation go to all those who have contributed to the completion of this thesis. My supervisor, my family, and my friends. Their unwavering support, guidance, and encouragement have been invaluable throughout this academic journey. This thesis stands as a testament to the collective efforts and unwavering support from each and every one of you. Thank you for being a part of this remarkable journey and for helping me reach this significant milestone in my academic career. This research stands as a testament to the collective efforts and unwavering support from each and every one of you. Thank you for being a part of this remarkable journey and for helping me reach this significant milestone in my academic career.

REFERENCES

- [1] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. neural networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019.
- [2] Suharto, A. P. Widodo, and E. A. Sarwoko, "The use of mobilenet v1 for identifying various types of freshwater fish," in *Journal of Physics: Conference Series*, 2020, vol. 1524, no. 1, p. 12105.
- [3] Z. Lin and W. Guo, "Cotton stand counting from unmanned aerial system imagery using mobilenet and centernet deep learning models," *Remote Sens.*, vol. 13, no. 14, p. 2822, 2021.

- [4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.
- [5] N. Jaccard, T. W. Rogers, E. J. Morton, and L. D. Griffin, "Detection of concealed cars in complex cargo X-ray imagery using deep learning," *J. Xray. Sci. Technol.*, vol. 25, no. 3, pp. 323–339, 2017.
- [6] Zeng, K. et al. (2022) 'FPGA-based accelerator for object detection: A comprehensive survey', *The Journal of Supercomputing*, 78(12), pp. 14096–14136. doi:10.1007/s11227-022-04415-5.
- [7] Hayat, S. et al. (2018) 'A deep learning framework using convolutional neural network for multi-class object recognition', 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC) [Preprint]. doi:10.1109/icivc.2018.8492777.
- [8] Kanimozhi, S. et al. (2021) 'Key Object Classification for action recognition in tennis using cognitive mask RCNN', *Proceedings of International Conference on Data Science and Applications*, pp. 121–128. doi:10.1007/978-981-16-5348-3_9.
- [9] Joshi, A.S. et al. (2020) 'Deep Learning Framework to detect face masks from video footage', 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN) [Preprint]. doi:10.1109/cicn49253.2020.9242625.
- [10] AlAjlan, S.A. and Saudagar, A.K. (2020) 'Machine Learning Approach for threat detection on social media posts containing Arabic text', *Evolutionary Intelligence*, 14(2), pp. 811–822. doi:10.1007/s12065-020-00458-w.
- [11] M. F. Shakeel, N. A. Bajwa, A. M. Anwaar, A. Sohail, and A. Khan, "Detecting driver drowsiness in real time through deep learning based object detection," in *Advances in Computational Intelligence: 15th International Work-Conference on Artificial Neural Networks, IWANN 2019, Gran Canaria, Spain, June 12-14, 2019, Proceedings, Part I 15, 2019*, pp. 283–296.
- [12] Ren, S. et al. (2017) 'Faster R-CNN: Towards real-time object detection with region proposal networks', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), pp. 1137–1149. doi:10.1109/tpami.2016.2577031.
- [13] N. Jaccard, T. W. Rogers, E. J. Morton, and L. D. Griffin, "Detection of concealed cars in complex cargo X-ray imagery using deep learning," *J. Xray. Sci. Technol.*, vol. 25, no. 3, pp. 323–339, 2017.
- [14] d S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, "Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery," *IEEE Trans. Inf. forensics Secur.*, vol. 13, no. 9, pp. 2203–2215, 2018.
- [15] V. Rizzo and D. Mery, "Automated detection of threat objects using adapted implicit shape model," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 46, no. 4, pp. 472–482, 2015.
- [16] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *Int. J. Comput. Vis.*, vol. 77, pp. 259–289, 2008.
- [17] A. Nurhopipah and A. Harjoko, "Motion Detection and Face Recognition for CCTV Surveillance System," *IJCCS (Indonesian J. Comput. Cybern. Syst.)*, vol. 12, no. 2, pp. 107–118, 2018.
- [18] B. Eamthanakul, M. Ketcham, and N. Chumuang, "The traffic congestion investigating system by image processing from CCTV camera," in *2017 International Conference on Digital Arts, Media and Technology (ICDAMT)*, 2017, pp. 240–245.
- [19] N. Bordoloi, A. K. Talukdar, and K. K. Sarma, "Suspicious Activity Detection from Videos using YOLOv3," in *2020 IEEE 17th India Council International Conference (INDICON)*, 2020, pp. 1–5.
- [20] A., A.B., P., P. and S., V. (2019) 'Detection of suspicious human activity based on CNN-DBNN algorithm for Video Surveillance Applications', 2019 Innovations in Power and Advanced Computing Technologies (i-PACT) [Preprint]. doi:10.1109/i-pact44901.2019.8960085.
- [21] A. Younis, L. Shixin, S. Jn, and Z. Hai, "Real-time object detection using pre-trained deep learning models MobileNet-SSD," in *Proceedings of 2020 the 6th international conference on computing and data engineering*, 2020, pp. 44–48.
- [22] Kumar, C.R. (2012) *Research methodology*. New Delhi: APH Publishing Corporation. page 366.
- [23] Murthy, J.S. et al. (2022) 'ObjectDetect: A real-time object detection framework for advanced driver assistant systems using yolov5', *Wireless Communications and Mobile Computing*, 2022, pp. 1–10. doi:10.1155/2022/9444360.
- [24] D. Biswas, H. Su, C. Wang, A. Stevanovic, and W. Wang, "An automatic traffic density estimation using Single Shot Detection (SSD) and MobileNet-SSD," *Phys. Chem. Earth, Parts A/B/C*, vol. 110, pp. 176–184, 2019.
- [25] Sanjay Kumar, K.K. et al. (2020) 'A mobile-based framework for detecting objects using SSD-mobilenet in indoor environment', *Intelligence in Big Data Technologies—Beyond the Hype*, pp. 65–76. doi:10.1007/978-981-15-5285-4_6.
- [26] Atik et al., (2022). Comparison of YOLO Versions for Object Detection from Aerial Images, *International Journal of Environment and Geoinformatics (IJEGEO)*, 9(2):087-093 doi. 10.30897/ijegno.1010741
- [27] Diwan, T., Anirudh, G. and Tembhrne, J.V. (2022) 'Object detection using yolo: Challenges, architectural successors, datasets and applications', *Multimedia Tools and Applications*, 82(6), pp. 9243–9275. doi:10.1007/s11042-022-13644-y.
- [28] Jiang, P. et al. (2022) 'A review of Yolo algorithm developments', *Procedia Computer Science*, 199, pp. 1066–1073. doi:10.1016/j.procs.2022.01.135.
- [29] Yar, H. et al. (2023) 'A modified Yolov5 architecture for efficient fire detection in smart cities', *Expert Systems with Applications*, 231, p. 120465. doi:10.1016/j.eswa.2023.120465.
- [30] Thuan, D. (2021) Evolution of YOLO Algorithm and Yolov5: The State-Of-The-Art Object Detection Algorithm.
- [31] Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-0_2.
- [32] Khurana, Y., 2019. Difference between Model Validation and Model Evaluation? [WWW Document]. Medium. URL <https://medium.com/yogesh-khuranas-blogs/difference-between-model-validation-and-model-evaluation-1a931d908240>
- [33] Narkhede, S., 2021a. Understanding Confusion Matrix [WWW Document]. Medium. URL <https://towardsdatascience.com/understanding-confusion-matrix- a9ad42dcfd62>
- [34] Rezatofighi, H. et al. (2019) 'Generalized intersection over union: A metric and a loss for bounding box regression', 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) [Preprint]. doi:10.1109/cvpr.2019.00075.