applied sciences

MDPI

Article

Development and Optimization of Deep Learning Models for Weapon Detection in Surveillance Videos

Soban Ahmed [1], Muhammad Tahir Bhatti [1], Muhammad Gufran Khan [1,*], Benny Lövström [2] and Muhammad Shahid [1]

1 Department of Electrical Engineering, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan; f190851@nu.edu.pk (S.A.); f179113@nu.edu.pk (M.T.B.); muhammad.shahid@ieee.org (M.S.)
2 Department of Mathematics and Natural Sciences, Blekinge Institute of Technology, 371 41 Karlskrona, Sweden; benny.lovstrom@bth.se
* Correspondence: m.gufran@nu.edu.pk

Featured Application: This work has applied computer vision and deep learning technology to develop a real-time weapon detector system and tested it on different computing devices for large-scale deployment.

Abstract: Weapon detection in CCTV camera surveillance videos is a challenging task and its importance is increasing because of the availability and easy access of weapons in the market. This becomes a big problem when weapons go into the wrong hands and are often misused. Advances in computer vision and object detection are enabling us to detect weapons in live videos without human intervention and, in turn, intelligent decisions can be made to protect people from dangerous situations. In this article, we have developed and presented an improved real-time weapon detection system that shows a higher mean average precision (mAP) score and better inference time performance compared to the previously proposed approaches in the literature. Using a custom weapons dataset, we implemented a state-of-the-art Scaled-YOLOv4 model that resulted in a 92.1 mAP score and frames per second (FPS) of 85.7 on a high-performance GPU (RTX 2080TI). Furthermore, to achieve the benefits of lower latency, higher throughput, and improved privacy, we optimized our model for implementation on a popular edge-computing device (Jetson Nano GPU) with the TensorRT network optimizer. We have also performed a comparative analysis of the previous weapon detector with our presented model using different CPU and GPU machines that fulfill the purpose of this work, making the selection of model and computing device easier for the users for deployment in a real-time scenario. The analysis shows that our presented models result in improved mAP scores on high-performance GPUs (such as RTX 2080TI), as well as on low-cost edge computing GPUs (such as Jetson Nano) for weapon detection in live CCTV camera surveillance videos.

Keywords: weapon detection; object detection; deep learning; optimization; computer vision

check for updates

Citation: Ahmed, S.; Bhatti, M.T.; Khan, M.G.; Lövström, B.; Shahid, M. Development and Optimization of Deep Learning Models for Weapon Detection in Surveillance Videos. *Appl. Sci.* **2022**, *12*, 5772. https://doi.org/10.3390/app12125772

Academic Editor: Arcangelo Castiglione

Received: 3 May 2022
Accepted: 3 June 2022
Published: 7 June 2022

1. Introduction

Crime is a deed that is based on an offensive act, but to overcome such offensive acts it has always been necessary to utilize different means to minimize them in short time. Some of these crimes result in danger to both the environment and human life. Every country in the world seeks peace because it enables societies to flourish, and economies to grow and achieve new heights of success over time. Contrary to this, an unpeaceful environment full of illegal activities brings the downfall of societies, communities, and countries. For example, wars have the biggest impact on a society in which the whole country suffers from huge pain and loss. Peaceful people are forced to fight, which ends with the loss of life, property, and identity. Some people are forced to relocate and travel to other countries as refugees. Therefore, countries with good defense systems can protect their sovereignty and help others to live peaceful lives. Similarly, on a lower scale, people who are not educated or are mistreated by society try to take the law into their own hands and commit crimes such as robberies. In robberies, robbers use weapons to terrorize other humans to meet their end needs, which normally ends with a loss of life or money. In most of these events, handheld weapons like pistols are the commonly used firearms that are easy to carry and get away with. Likewise, weapons are used by terrorists against common citizens, which results in a massacre.

In the past few years, the incidents involving firearms have increased in public areas. A couple of years back were the attacks on mosques in New Zealand on 15 March 2019; at 1:40 pm, the attacker attacked the Christchurch AL-Noor Mosque during a Friday prayer, killing almost 44 innocent and unarmed worshippers, and with a break of just 15 min at 1:55 pm, another attack happened that killed seven more civilians [1]. In the USA and then in Europe, incidents of active shooters have occurred, such as in Columbine High School (USA, 37 victims), Andreas Broeivik's assault on Uotya Island (Norway, 179 victims), or the killing of 23 people in the Charlie Hebdo newspaper attack. UNODC statistics says that among 0.1 million people in a country, the crimes involving guns are very high, i.e., 1.6 in Belgium, 4.7 in the USA, and 21.5 in Mexico [2].

The reason and motivation behind this work is to detect various weapons in real time, thus reducing the aforementioned incidents. These incidents can be controlled using an early alarm system by alerting the operators and concerned authorities so that action can be taken immediately.

For decades, law enforcement agencies have installed CCTV cameras for surveillance that have helped them in noticing illegal activities in streets, airports, etc., and utilize the video as a piece of evidence in a court of law. People install CCTV cameras in their shops or markets for protection from thieves and robbers [3,4]. One common issue that arises with CCTV cameras is that they are not intelligent, which means a person should be available round the clock for monitoring live videos to take timely action against illegal activity. This turns out to be a tiresome job when a person must monitor multiple video screens for a longer duration of time [5].

Artificial intelligence (AI) and computer vision have enabled us to utilize video feeds in a way that we can detect and classify the objects of our interest in it. Therefore, it has been widely adopted and used in many applications such as autonomous vehicles, security feeds, etc. Many algorithms and architectural works have been done for the aforementioned tasks. In 2020, Murthy, Chinthakindi Balaram et al. [6] provided a detailed and comprehensive discussion and analysis of state-of-the-art techniques and algorithms used in the field of computer vision using deep learning technology, especially for the GPU-based embedded system. They covered many state-of-the-art algorithms that were trained and tested on COCO, PASCAL VOC datasets. The algorithms included RCNN, SPPNet, FasterRCNN, MaskRCNN, FPN, YOLO, SSD, RetinaNet, Squeeze Det, and CornerNet; these algorithms were compared and analyzed based on accuracy, speed, and performance for important applications including pedestrian detection, crowd detection, medical imaging, and face detection. Moreover, we also previously implemented a real-time weapon detector model based on YOLOv4 trained with a custom dataset. The model has a decent accuracy but lacks performance [7]. Therefore, we continued our research to further improve the model in terms of mAP and performance for both cloud and edge computing devices. In addition, the existing solutions are not highly accurate, lack real-time performance, and often utilize cloud computing in which resources are abundant. Cloud implementations have issues like performance bottlenecks caused by network bandwidth, latency and privacy breaches, etc. Hence, there is a need to improve the existing solutions by utilizing the state-of-the-art object detection algorithm deployed on edge computing devices to rectify the mentioned shortcomings of cloud computing. The notable limitations of the previous works are the usage of an older deep learning model that has lesser accuracy compared to the state-of-the-art models, very few and outdated preprocessing steps for dataset preparation, and no performance analysis on any embedded device. Furthermore, to the best of our knowledge, there is no previous work done to deploy and compare the performance of the weapon detection model on edge computing devices in real time.

The main contributions of this research work are as follows:

1. Improved the accuracy and performance of the existing weapon detection model by utilizing state-of-the-art algorithm and preprocessing techniques;
2. Improved number of frames per second FPS for real-time deployment;
3. Compared and analyzed the performance of the different deep learning models on different computing devices;
4. Utilized the TensorRT for network optimization that resulted in improved latency, throughput, power efficiency, and lower memory consumption.

The rest of the paper is organized as follows. Section 2 contains the related work and the literature review of different approaches that were being used to achieve the desired outcomes. Section 3 elaborates on the methodology of the research problem and what steps should be taken to successfully carry out this research. Section 4 contains the results of the research and a discussion on it that provides the reader with deep insight into the research findings. Finally, Section 5 concludes the paper and provides possible future directions in this area.

2. Related Work

The solution to overcome the problem of weapons in public places is to develop an automatic weapon detection system that works in real time with high accuracy and performance to quickly generate an alarm. Such detectors have many applications in security for the safety of human life that will enable the authorities to quickly act before a major incident can happen. Therefore, the weapon detector applications can be a valuable addition to society for developing a city to become much safer. The idea for real-time weapon detection using computer vision first came out in 2007 when scientists proposed a real-time firearm detector by utilizing a CCTV camera feed [8]. A year later, an idea was implemented in 2008 in which the scientists developed a pistol detector for RGB images, but the detector failed to detect multiple pistols in a single image [9]. Initially, researchers used machine learning-based algorithms, but with the development of deep learning algorithms, researchers shifted toward it.

Histogram of oriented gradients (HOG) and speeded-up robust features (SURF) as a feature extractor were utilized. For detecting the firearms in an image, SIFT- and SVM-based algorithms were used, but these algorithms were not highly accurate and had poor speed for real-time scenarios, taking more than 14 s per image for detection. Moreover, these algorithms do not learn features of their own; they use the handcrafted rules for feature extraction. The authors also pointed out that automatic feature representation provides better results than the manual approach [10,11]. Another publication, using HOG and neural networks, utilizes the possibility to only look for weapons in the vicinity of humans to save processing time [12].

With the advent of deep learning-based algorithms, researchers shifted toward it and utilized the algorithms that learn features automatically from an image, improving the performance and reducing a lot of manual work to save time. Researchers used CNN-based techniques to learn the features automatically and a ReLu-based activation function to introduce the nonlinearity that resulted in higher accuracy than machine learning-based algorithms but, still, performance in real time remained an issue because these algorithms are based on a sliding window approach that results in slow speed [13]. With the Faster RCNN algorithm and IMFDB database, researchers developed a weapon detection model that has 93.1% accuracy, but this database is not good for real-time cases and they did not consider the precision and recall in their design; therefore, it was prone to higher false alarms [14]. Region proposal methods in the algorithms helped in reducing the inference speed in Faster RCNN, which achieved 140 ms with 7 FPS on a powerful GPU [15]. Another researcher developed a weapon detection system by training the Faster RCNN using a feature pyramid network with Resnet50, but the performance of this was slow [16]. The researchers of [17] trained a weapon detection model on YOLOv3 and compared its performance with the YOLOv2 model. They used a customized dataset for their model but the dataset and model are not great because many state-of-the-art models are available that can perform much better in terms of mAP and real-time speed.

The authors of [18] utilized a combination YOLOv3 model and human pose to perform handgun detection. They highlighted that only detecting weapons is not enough because if a model has lower average precision (AP) it will be generating too many false alarms; therefore, to mitigate this problem they added human pose to improve the detection results, which showed them an improvement of 17.5% in AP. Likewise, in reference [19], the authors used the same YOLOv3 model to train a rifle detection classifier for their custom dataset of rifles that gave adequate AP; however, both systems are not good for real-time implementation and embedded devices as the base model have too many parameters that become overwhelming for a device with limited resources. The work presented in [20] utilized three different CNN-based models for weapon detection, namely Faster R-CNN, Retina Net, and YOLOv3; along with it they added pose estimation to further enhance the model. The authors compared these models in terms of different performance metrics such as precision, recall, and F1 score, in which YOLOv3 showed good results but the rate of false positives was high. This approach lacks real-time capabilities and has poor performance issues on embedded devices. Furthermore, a model that was trained on only 1220 images does not cover all the real-time scenarios, such as, for example, images in different backgrounds and orientations for the generalized solution.

The study of [21] shows the use of the transfer learning approach on AlexNet, VGG16, and VGG19 models to train gun and knife detection from images. The results show good accuracy, but accuracy is not a good metric to stand out the results in object detection models where mean average precision is preferred. Likewise, the model lacks the capabilities to be deployed in a real-time environment. In reference [22], the authors trained a model on a public dataset that used the open pose methodology for pose estimation that aids in weapon detection and improves the weapon detectors' detection capability. The authors of [23] identified that there is a lack of datasets in weapon detection problems; therefore, they proposed a way to generate synthetic datasets using a graphics engine. They have added an anomaly detector as well to ensure a reduction in the false-positive rate; however, the created dataset does not help much in the reduction in false positives but the anomaly detector helps in reducing it. In reference [24], the authors developed a weapon detector based on two models and compared them. They utilized a multi-contrast convolutional neural networks (MC-CNN) model and faster region-based convolutional neural networks (Faster R-CNN) model for concealed weapon detection. They have found that MC-CNN has a more complex architecture than the Faster R-CNN model; therefore, it performs much better than it. However, they developed a small, customized dataset for it that does not cover the real-time scenarios; moreover, the model is heavier for real-time scenarios and therefore not recommended for utilization in the real world. In work [25], the authors proposed a novel method for improving hand-held firearms detection. They have extracted the human hand and combined it with the human pose by using the open pose method to train the classifier that detects whether the person is holding a weapon or not. If both pose and hand suggest that the person is holding a weapon, the model classifies it as a weapon detected. Their work is only a recommendation that can assist in a better weapon detection model; therefore, it is not good for any real-time system and is prone to false positives. In reference [26], the authors utilized the YOLOv4 algorithm for weapon detection trained on the Google and Kaggle datasets. They used the internet of things (IoT) approach to collect video and fed it to their model. The method they have used is good but the dataset is not great because it does not cater to all real-time scenarios with different image qualities and backgrounds that affect the mAP of the model. The study of [27] shows the comparison of two different weapon detection models, YOLOv3 and YOLOv4, trained on images collected from Google. They have shown that YOLOv4 outperforms the YOLOv3 model in terms of processing time and sensitivity of detection because the YOLOv4 algorithm is a superior model to YOLOv3. The researcher of [28] used Alex-net in combination with some other techniques such as spatial pyramid pooling (spp) to train a weapon detection model. However, this approach is quite old and poor since many state-of-the-art algorithms outperform Alex-net now; therefore, it is not suitable for real-time systems. The work presented in [29] used YOLOv4 as the main weapon detection model on their custom dataset, providing them with adequate results; however, the dataset is not good for real-time scenarios. Moreover, they are getting 35 FPS on 2080 TI GPU, which is quite low compared to recent models such as in Scaled-YOLOv4.

In 2021, Jesus Salido et al. [30] proposed the detection of a handgun in deep learning surveillance images by training three convolutional neural network-based models (RetinaNet, FasterRCNN, and YOLOv3) and have done multiple experiments and claimed to have reduced the number of false positives, thus gaining best recall and average precision of 97.23 and 96.36%, respectively, using RetinaNet fine-tuned with unfrozen ResNet 50 as a backbone and, later adding the pose estimation training samples in the dataset, they have achieved 96.23% and 93.36% of precision and F1 score, respectively. The problem with this work was the use of high-definition images for training and also the 1220 data images might not cover the real-time case of having a lot of diverse incoming data [20]. Volkan Kaya et al. [30] proposed a weapon detection and classification technique by introducing a new model based on the VGGNet architecture trained on a dataset having seven different classes that include assault rifles, bazookas, grenades, hunting rifles, knives, pistols, and revolvers. They compared their model results to VGG-16, ResNet-101, and ResNet-50, examining the best classification results. Their proposed model achieved the highest accuracy of 98.4% among all, beating ResNet-50, VGG-16, and ResNet-101 with accuracies of 93.7, 89.75, and 83.33%, respectively. The problem with the work was the use of dataset images that are mostly animated or from a movie scene, hence making the model generalize less when implemented in real time [30]. In [31], the authors have proposed an object detection technique for abnormal situations such as guns and knives. They have introduced a new lightweight multiclass-subclass detection CNN (MSDCNN) model to extract and detect abnormal features in a real-time scenario. They have made a custom dataset and introduced a new evaluation method named detection time per interval (DTpI). Their proposed MSD-CNN has achieved the highest precision of 97.5% on imageNet and IMFDB datasets [31].

Recently, researchers utilized the YOLOv4 object detection algorithm in [7] to design a real-time weapon detection system that managed to achieve the 91.73% mAP and F1 score of 91%. They have trained many algorithms and provided a very good overview of different algorithms in terms of speed and accuracy on a powerful GPU, but they have not utilized the optimization techniques to improve performance for any embedded devices. The authors suggested in their future work that there is a lot of improvement required in terms of reducing the false positives and negatives [7].

Existing solutions can be improved by utilizing the latest model known as Scaled-YOLOv4, which has shown higher performance than its predecessors; better data augmentation techniques in the preprocessing step, such as mosaic augmentation, improve the accuracy of the model when the objects to detect are small. Likewise, adding TensorRT for network optimization for improved latency, throughput, power efficiency, and memory consumption can further enhance the model for an embedded device such as Jetson Nano. This approach will improve the accuracy of the existing solution by reducing the false positives and false negatives. It will enhance the performance of the existing solution for embedded devices by reducing the latency, increasing throughput, and privacy of an edge computing device. Edge computing will rectify the privacy and security issues as data will be stored locally and model deployment will be private. Moreover, minimal latency and power consumption on the edge computing device will further improve it. The bandwidth bottleneck issue will be resolved by deploying models on edge devices.

The Scaled-YOLOv4 algorithm with its higher accuracy enables the design of a highly accurate real-time weapon detector that will be deployable on an embedded device to achieve lower latency, higher throughput, better privacy, and security for humans. The comparison between the existing solution of YOLOv4 and Scaled-YOLOv4 in terms of accuracy and performance on an edge computing device will provide a thorough overview of different metrics along with a deeper insight into how current algorithms perform on an edge computing device.

3. Methodology

To mitigate the issues identified in our previous weapon detector [4], this work solves those problems by having an increased number of frames per second for real-time scenarios and a reduced number of false positives. The methodology provided herein making this work successful is divided into multiple steps listed as follows:

1. Dataset selection;
2. Preprocessing operations;
3. Model selection;
4. Model training and tuning;
5. Model optimization using TensorRT network.

3.1. Dataset Selection

In machine learning and computer vision applications, datasets play a significant role because without good datasets the algorithms will not be able to perform well because it is the only way a model learns to see the world with an eye. Therefore, the collection of a good quality dataset is an important task. The dataset used for the training and testing is the same as the previous weapon detector model consisting of a total of 8327 labeled images split into training and test datasets, as given in Table 1.

Table 1. Dataset Distribution.

Category	Data Distribution
Total Dataset	8327
Training Data	7328
Test Data	999
Split Size	12%

The dataset is collected by keeping in view the constraints of real-time detection scenarios where different kinds of colors, backgrounds, occlusion, flipped, and rotated objects might appear. This dataset is generalized because images captured have all kinds of background, resolutions, and occlusion that will help the model to generalize well and perform better in an unseen and unknown environment.

The dataset is categorized into two main categories; the first is the pistol and the second is the non-pistol category. The mAP of a model can be improved by reducing the number of false negatives and false positives. Therefore, along with the weapon class, we also included the other objects with which our model could confuse the pistol class. For the weapon category, the dataset includes revolvers, pistols, shotguns, and rifles, as shown in Figure 1 below.

Likewise, in the non-pistol class, we included items such as bicycles, cars, purses, selfie sticks, smartphones, etc. to help the model understand other similar things so that it can generalize well. Some samples of the dataset from non-pistol category are shown in Figure 2 below.

Figure 1. Data samples for pistol category.

Figure 2. Data samples for non-pistol category.

3.2. Preprocessing Operations

Raw datasets are not recommended to be directly used for training because either they contain less observable information or they contain unnecessary information that can bring a poor impact on our model. Instead of getting good results, we can be stuck with poor results, and therefore datasets should be preprocessed by using common preprocessing steps to enhance the quality of the raw dataset. For this purpose, our dataset is preprocessed to improve the quality of the dataset so the model can learn better details and generalize well for the unseen environment. We took different steps during preprocessing such as image resizing, mean normalization, bounding boxes allocation, labeling, image scaling, data augmentation and filtration such as Clahe, equalization, and RGB-to-grayscale conversion, as shown in Figures 3–5.

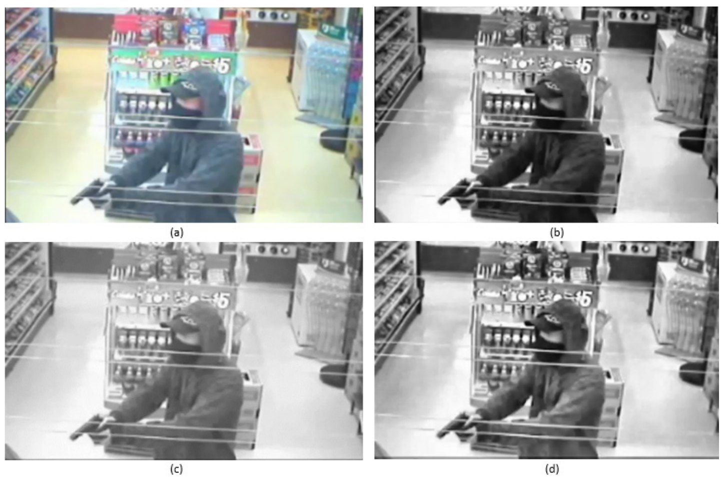


Figure 3. Image filtration using OpenCV filters; top left to bottom right (**a**–**d**): (**a**) actual image, (**b**) equalized filter result, (**c**) gray scale filter result, and (**d**) CLAHE filter result.

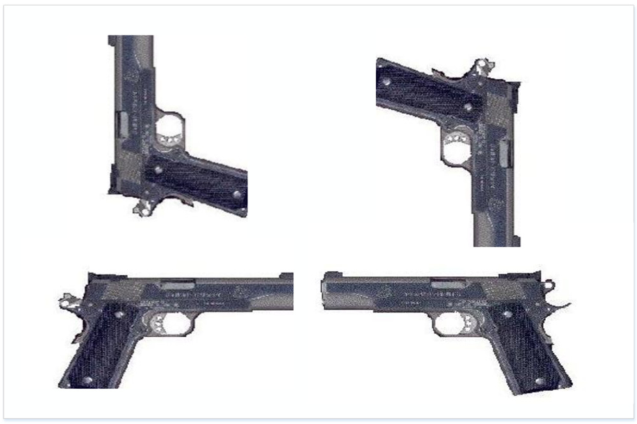


Figure 4. Image scaling and augmentation.

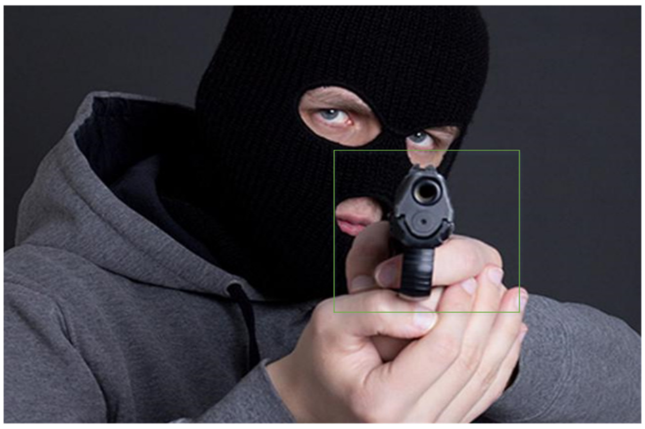


Figure 5. Image annotation and labelling.

3.3. Model Selection

After a thorough literature review, we chose state-of-the-art Scaled-YOLOv4 as an object detection model that has proven to be much better than YOLOv4 in terms of FPS and mAP. It should be noted that the model performance varies for different datasets; some perform well on small images while others perform well on large images. Scaled-YOLOv4 can be tuned to achieve the best result. The model used the cross stage partial (CSP) approach to improve the speed and accuracy. The network is scalable in terms of depth, width, resolution, and structure. The YOLOv4 was designed for object detection on general GPU, which is why it lacks speed and accuracy. The redesigning of YOLOv4 with the CSP approach enhances its speed and accuracy. The backbone of YOLOv4 is Darknet, which is replaced with CSPDarknet53 that has lower computations. The number of computations can be estimated by using the following equation:

$$whb^2\left(\frac{9}{4}+\frac{3}{4}+\frac{5k}{2}\right) \tag{1}$$

where w is the width, h is the height of the image, b is the base layer channels, and k is the number of layers. The neck of YOLOv4 consists of a path aggregation network (PAN) that is further CSP-ized to reduce the computation by 40%.

3.4. Model Training and Tuning

The models are normally trained multiple times with different parameter configurations to find the best suitable results. Similarly, we have trained our model with different parameters and further fine-tuned it to find the best mAP and FPS. We recognized the problem in the previous weapon detection model, utilized the right dataset, performed the data preprocessing, applied the state-of-the-art Scaled-YOLOv4 algorithm, and finally fine-tuned the algorithm parameters to get the best weapon detector model. The methodology adopted for training and optimizing the model is elaborated in Figure 6 below.

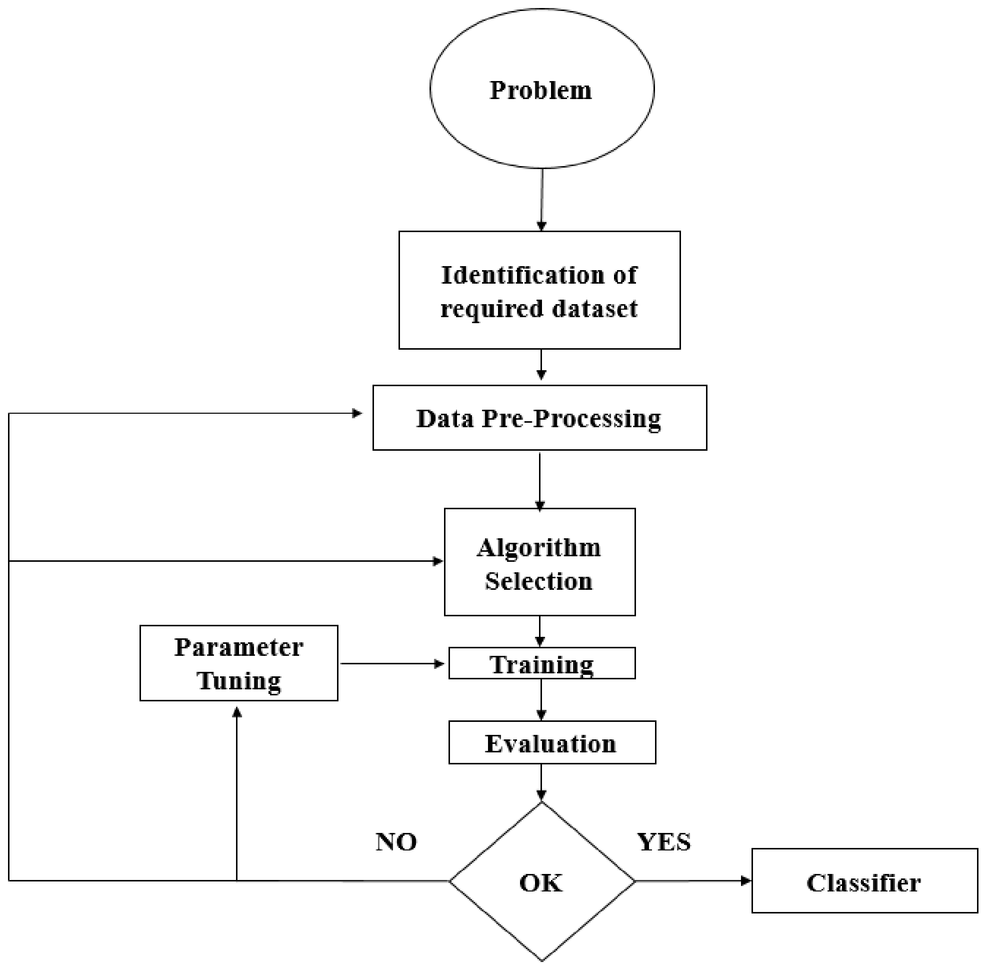


Figure 6. Training and optimization flowchart [7].

3.5. Model Optimization Using TensorRT Network

Nvidia developed the TensorRT, which is a machine learning/deep learning framework to run inference on Nvidia GPUs with the best optimization for the chosen hardware. It optimizes the inference and runtime that delivers low latency and higher throughput for deep learning models. TensorRT provides INT8 and FP16 optimizations by quantizing models while preserving accuracy for production deployment level deep learning model inference. It ensures the optimized use of GPU memory and bandwidth by layer and tensor fusion. It searches for the best data layers and algorithm for the selected GPU and tries to utilize less memory or efficiently reuse the same memory for tensors. The optimization workflow using TensorRT is depicted in Figure 7.

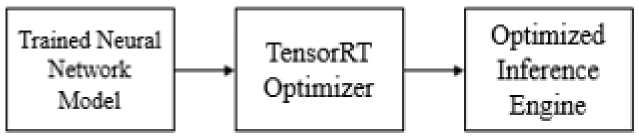


Figure 7. Model optimization workflow using TensorRT.

4. Results and Discussion

4.1. Experimentation

The objective was to train and fine-tune the model to improve the mAP and FPS of the weapon detector by following the training and optimization workflows as shown in Figures 6 and 7. We have trained several models to evaluate and find the best possible model for inference purposes. The training of the models was performed on different computing devices, first on high-performance machines like RTX 2060, RTX 2080TI, and Tesla T4, and then they were trained and tested on edge computing devices like Jetson Nano and Raspberry pi. The hyperparameters are set at the default setting for all the models to have an unbiased result and are listed in Table 2 below.

Table 2. Scaled-YOLOv4 Default Hyperparameters Settings for Models Training.

Sr. No.	Hyper Parameter	Value
1	Learning rate	0.001
2	Decay	0.0005
3	Momentum	0.949
4	Activation Function	Mish
5	Batch Size	64
6	Max Batches/Iterations	6000

Different parameters play important roles in Scaled-YOLOv4 such as image resolution, subdivision, intersection over union (IOU), intersection over union normalization, new coordinates, object smoothness, scaling x y, and loss functions. The first step of model training is to provide the dataset to the model and quickly finish the model training to observe and understand the trend, whether the model is improving the mAP and reducing the loss or not.

We followed the aforementioned process and first trained the model with different settings mentioned in the previous table for RTX 2060-based Scaled-YOLOv4 optimization process. Default hyperparameter settings along with subdivision and image resolution of 64 and 512 × 512, respectively, gave us 85.5% mAP and an average loss of 14.42.

After the first training and testing results, we conceived the idea to set the right number of iterations, which came out to be 4800 and 5400 according to our two classes. For the next training, we changed the subdivision to 32 and kept the resolution size the same as in the previous training along with the default hyperparameter setting and found that the mAP changed to 91.2 and the average loss of the model also dropped down to 16.2.

Since the last training session gave good mAP, we kept the subdivision to 32 and changed different parameter settings that could help in further improving the mAP and reducing the loss. Some of the parameters recommended by the authors are:

- Object smoothness = 0,
- Scale x y = 1.05,
- Remove IOU loss,
- Remove IOU normalization.

We observed that the mAP reduced to 90.4 while the average loss improved and dropped to near 4. There is a positive change, however, as mAP plays the main role and therefore we moved toward different parameter settings. Now we reset all the settings and only change the resolution of the input image 608 × 608 with subdivision 32 and retrain the model. We observed that changing only the resolution improved the mAP to 91.2; however, there is no effect on average loss, which is still high at around 14.

In the next step, we increased the number of iterations to 10,000 with the subdivision and resolution the same as the previous of 32 and 606 × 608 to check if increasing the iteration can help the model in learning new features; however, mAP reduced to 91.2 but loss slightly reduced so it did not show any major impact on the results.

To observe the effect of changing the resolution, we changed the input resolution to 640 × 640 with a subdivision of 64 and retrained the model; however, it negatively impacted the model, reducing mAP to 83.3. It helped us understand that image resolution can be increased to a certain limit; after that it can impact poorly on the model mAP and average loss. All the results described above and graphs for the best performing models on RTX 2060 machine are listed and shown in Table 3 and Figure 8.

Table 3. Model 1: Scaled-YOLOv4 Optimization Configuration, and Parameter Tuning using GPU RTX 2060.

Subdivision	Parameters Tuned	Resolution	mAP	Improvement
64	No	512 × 512	85.5	No
32	No	512 × 512	91.2	No
32	Object smoothness = 0 Scale x y = 1.05 Removed IOU loss and IOU normalization	512 × 512	90.4	No but average loss reduced
32	No	608 × 608	91.2	No
32	Max batches = 10,000	608 × 608	90.9	No
64	No	640 × 640	83.3	No

It can be observed from the above that the model performed best on RTX 2060 when having a subdivision of 32 and an image resolution of 608 × 608.

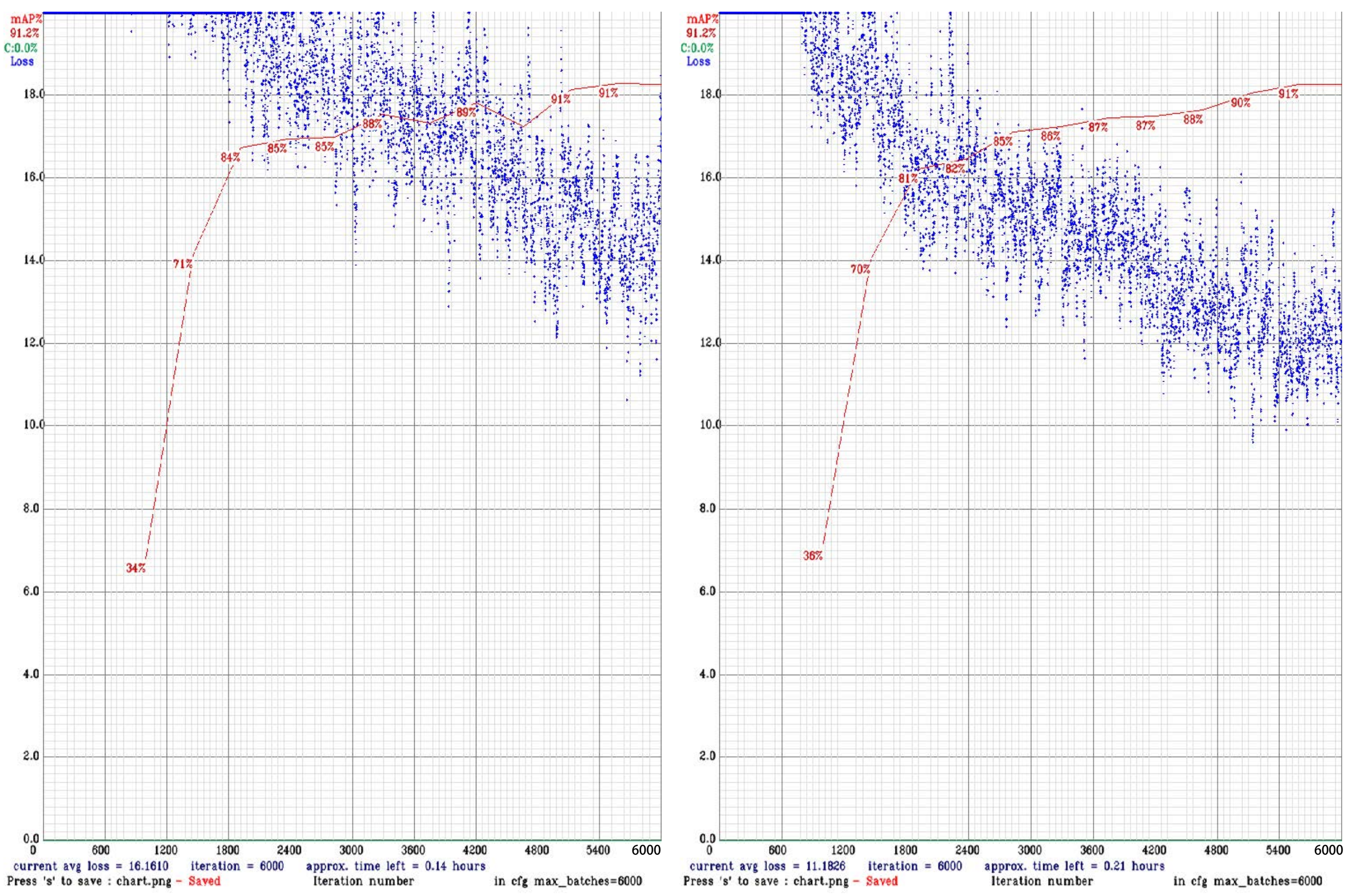
We further moved toward changing the image resolution, subdivision, and other parameters and found that RTX 2060 GPU was not able to load the model if we changed the network to do more complex work; therefore, we switched to RTX 2080TI GPU to further change the settings to get better results. We changed some hyperparameters as follows:

- IOU loss removed,
- IOU normalization 0.05 to 0.07.

In the first run, it gave us adequate results with mAP increasing to 90.4 and loss reducing to 4.57. We further tuned the parameters to get better results with mAP becoming 90.9 and loss hitting the lowest of 1.24 utilizing the following settings, i.e.,

- IOU loss removed,
- IOU normalize change from 0.05 to 0.07,
- Removed new coordinates,
- Changed logistic to linear loss function in YOLO layers.

All the results described above and graphs for the best performing models on RTX 2080TI machine are listed and shown in Table 4 and Figures 9 and 10.



(**a**) (**b**)

Figure 8. RTX 2060 trained the best performing models: mAP vs. avg. loss. (**a**) Model with subdivision of 32 and image resolution of 512 × 512 with default hyperparameters settings showing mAP of 91.2 and avg. loss of 16.2. (**b**) Model with subdivision of 32 and image resolution of 608 × 608 with default hyperparameters settings showing mAP of 91.2 with reduced avg. loss of 11.2.

Table 4. Model 2: Scaled-YOLOv4 Optimization Configuration, and Parameter Tuning using GPU RTX 2080TI.

Subdivision	**Parameters Tuned**	**Resolution**	**mAP**	**Improvement**
32	IOU loss removed IOU normalization changed from 0.05 to 0.07	512 × 512	90.4	No
32	IOU loss removed IOU normalization removed New coordinate removed Logistic to linear loss function change	512 × 512	90.9	Average loss reduced

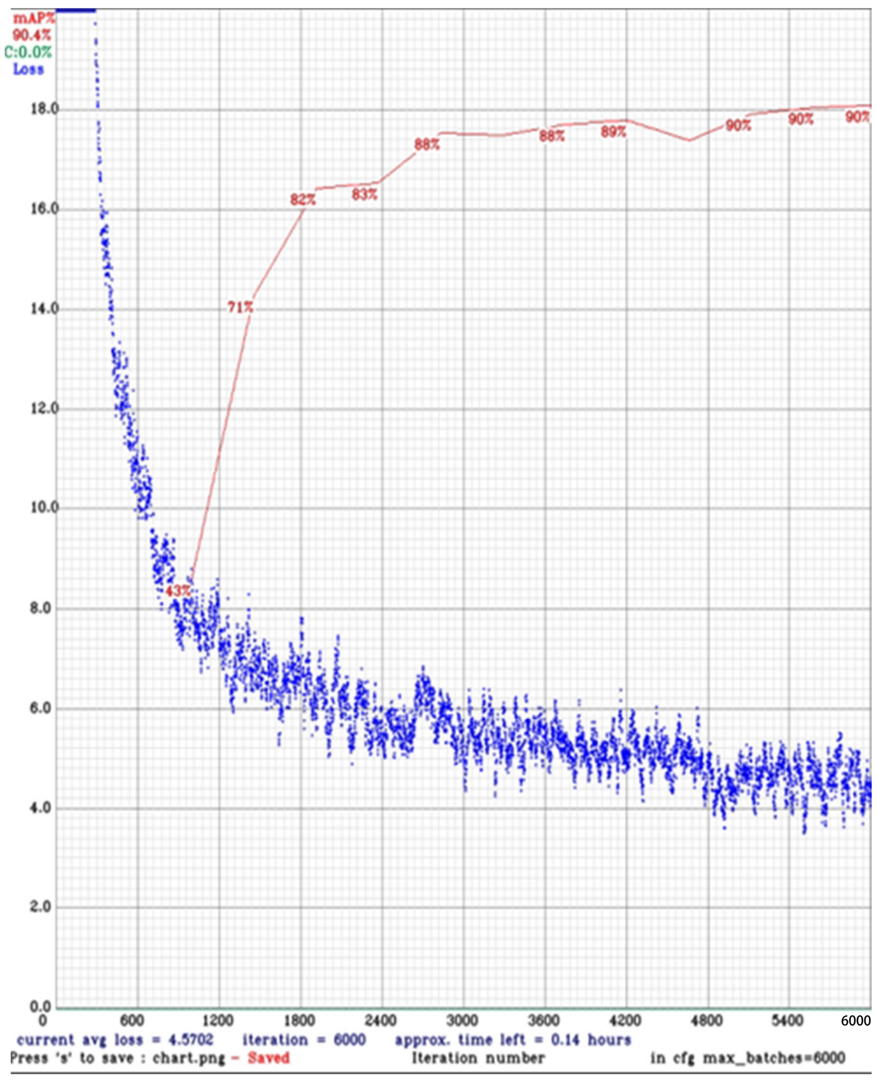


Figure 9. RTX 2080TI trained the best performing models: mAP vs. avg. loss: Model with IoU loss removed and IoU normalization change from 0.05 to 0.07 with default hyperparameters settings at an image resolution of 512 × 512 giving reduced mAP of 90.1 and better avg loss of 4.57.

Finally, after many different parameters tweaking, we changed the following hyper-parameters to get the highest mAP of 92.1, beating the previous model mAP. Some of the settings are as follow:

- Object smoothness = 0,
- Scale x y = 1.05,
- Subdivision = 8 with image resolution 512 × 512.

To go to an even lower subdivision made us move to a high-end GPU because RTX 2060 and RTX 2080TI GPUs were not capable of running the complex calculation. We ran the final training on Tesla T4, which gave us the expected results with parameter setting and model inference outcome as listed and shown in Table 5 and Figure 11.

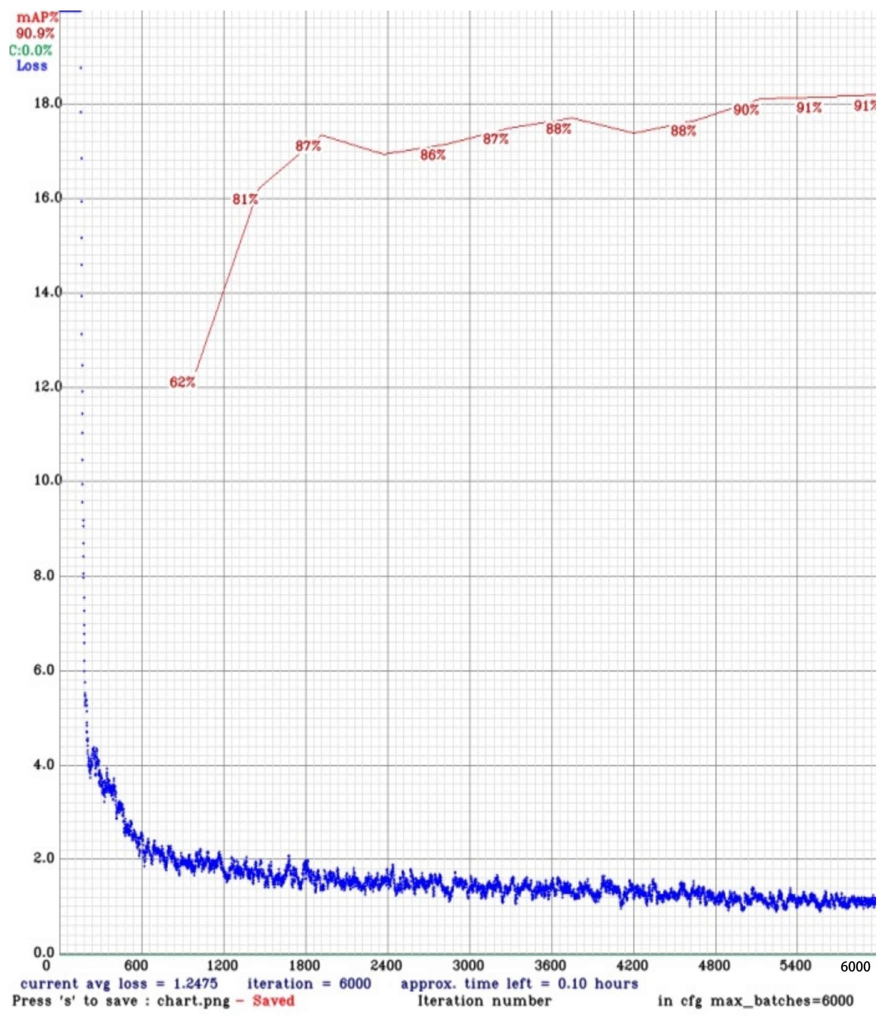


Figure 10. RTX 2080TI trained the best performing models: mAP vs. avg. loss: Model with IoU loss and new coordinates removed and IoU normalization change from 0.05 to 0.07 along with a change of loss function to linear from logistic with default hyperparameters settings at an image resolution of 512 × 512 giving mAP of 90.1 and lowest loss of 1.24.

Table 5. Model 3: Scaled-YOLOv4 Optimization Configuration, and Parameter Tuning using GPU Tesla T4.

Subdivision	Parameters Tuned	Resolution	mAP	Improvement
8	Object smoothness = 0 Scale x y = 1.05	512 × 512	92.1	Yes

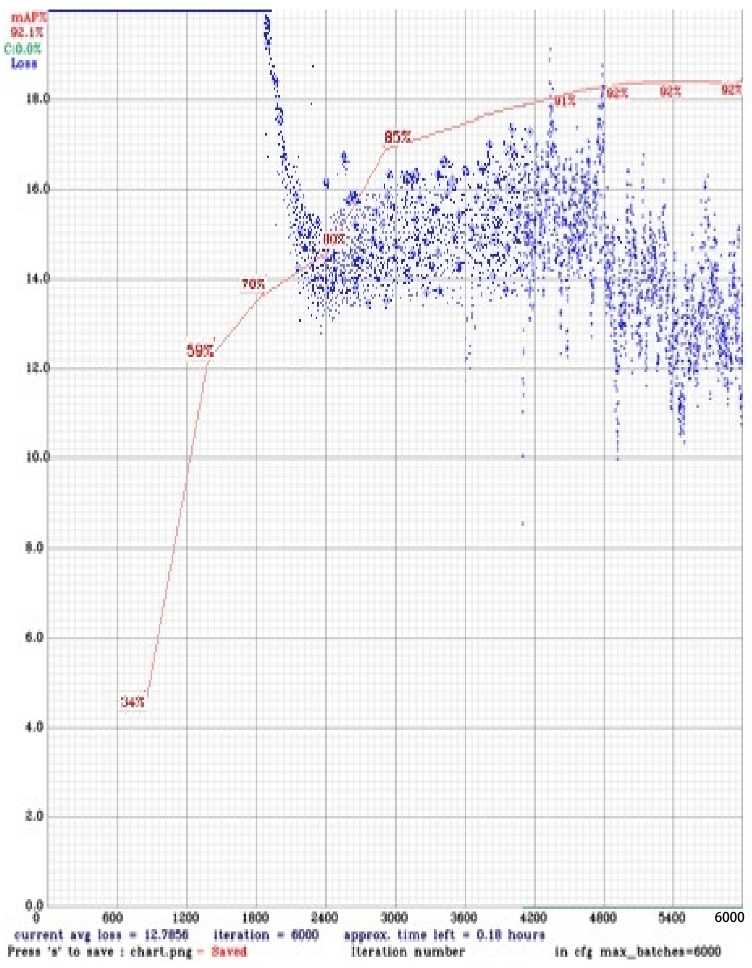


Figure 11. Tesla T4-trained best model graph of mAP vs. avg. loss: model trained with a subdivision of 8, image resolution of 512 × 512, number of iterations 6000, object smoothness as 0 and Scale x y as 1.05.

It has been noted from the results that subdivision plays an important role in improving the mAP of a model. However, if subdivision is a lower number, then more GPU memory will be used; therefore, a powerful GPU should be selected to support it.

4.2. Comparison of Mean Average Precision

The mAP of the weapon detector trained on Scaled-YOLOv4 is compared with its predecessor YOLOv4 as reported in Table 6.

Table 6. Comparison of mAP on best-performed Scaled-YOLOv4 vs. YOLOv4 [7].

Algorithm	mAP
Scaled-YOLOv4 (@0.5 IOU)	92.1
YOLOv4 (@0.5 IOU)	91.8

It can be noted from the results that the mAP is improved but it is closer to its predecessor [4]. In terms of precision, the previous model achieved 93% while ours reduced it to 90. However, in terms of recall, previous model achieved only 88% while our model gave 91%. Finally, the F1 score of our model is 91% while the previous model has 91. The result's comparison of the best performing Scaled-YOLOv4 vs. its predecessor YOLOv4 in terms is shown in Figure 12.

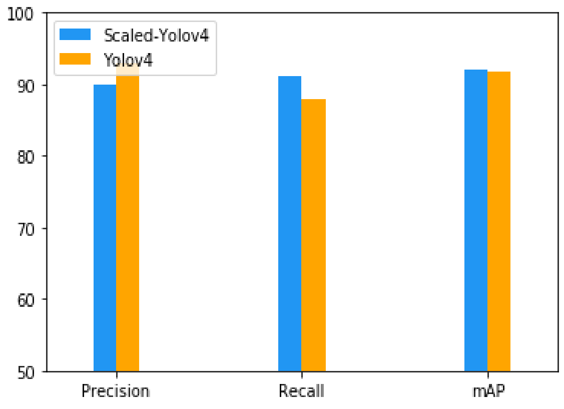


Figure 12. Comparison of Scaled-YOLOv4 vs. YOLOv4.

The confusion matrix of the weapon detection shows that the true positive rate is considerably higher than the false positives rates as shown in Figure 13.

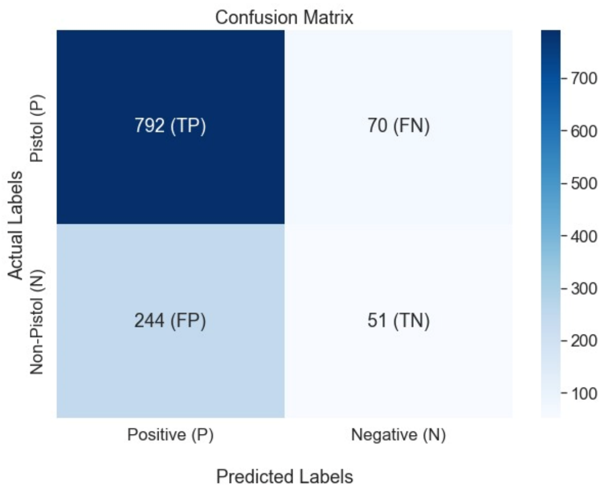


Figure 13. Confusion matrix of Scaled-YOLOv4 weapon detector.

4.3. TensorRT Network Optimization

To deploy the deep learning model on the edge computing device, especially on Jetson Nano, we utilized the TensorRT framework that enabled us to perform the network optimization. This optimization enables edge devices to perform high-speed inference by utilizing fewer resources efficiently. For this purpose, we applied FP16 optimization on our weapon detection model because Jetson Nano only supports FP16, and this was getting reasonable FPS on Jetson Nano.

4.4. FPS Evaluation on Different Machines

We evaluated the FPS of our weapon detection model on various machines. We got the highest FPS of 85.7 on RTX 2080TI, 59.3 FPS on RTX 2060, 0.69 FPS on Raspberry Pi 4, 4.26 FPS on Jetson Nano, and 22.5 FPS on Intel CoreI5 10th Generation. It can be noted that the powerful machines gave very high FPS while on resource-constrained edge devices FPS is quite low. However, if we compare Jetson Nano with Raspberry Pi 4 it can be noted that the Jetson Nano TensorRT network optimization keeps it ahead of the game. The graph showing FPS comparison of Scaled-YOLO v4 on different machines can be seen in Figure 14.

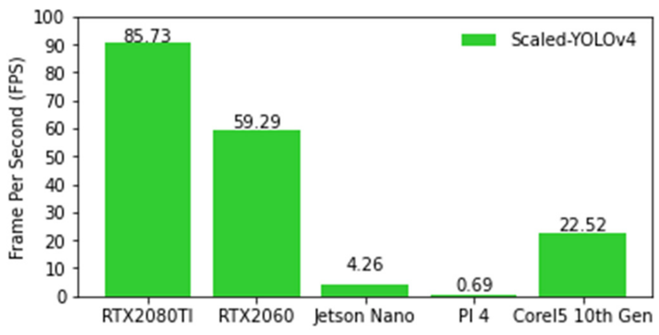


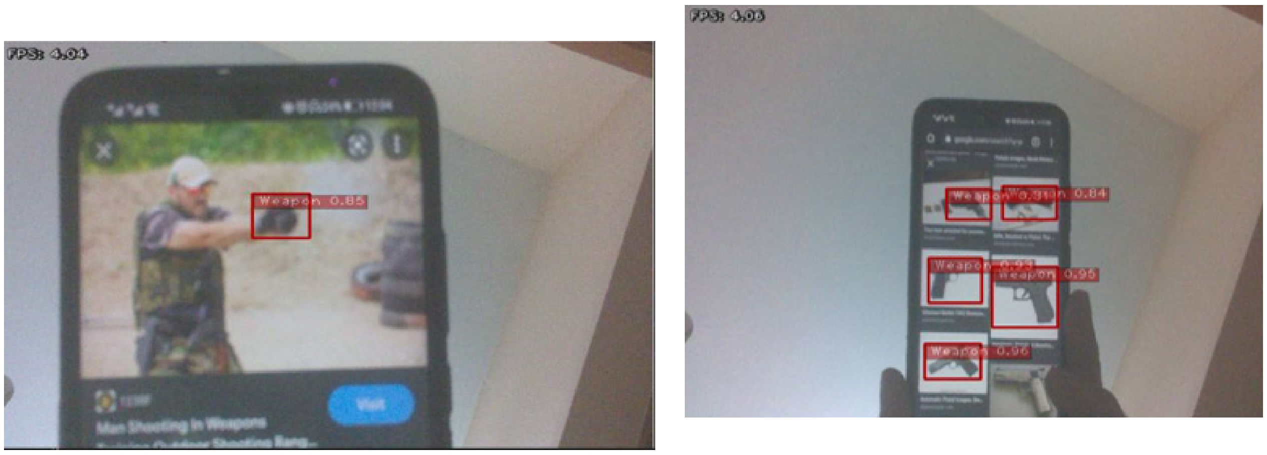
Figure 14. Comparison of FPS on different computing devices.

4.5. Comparison of Scaled-YOLOv4 and YOLOv4 FPS

The previous researchers were only focused on generating results on RTX 2080TI, so if we compare our results with theirs, it can be noted that the Scaled-YOLOv4-based model is ahead of it with 7 FPS. We reproduced the results and measured the FPS of the older YOLOv4 weapon detection model, which gave us 78.9 on an offline video.

4.6. Real-Time Weapon Detection and FPS Evaluation of Scaled-YOLOv4 on Jetson Nano

The weapon detection model's FPS is evaluated on Jetson Nano in real time by interfacing Raspberry Pi camera module version 2. Models are converted into ONNX format to use further on Jetson Nano. A different set of images was shown to the camera to detect the respected class of weapon and the frame per second, which can be seen in Figure 15.



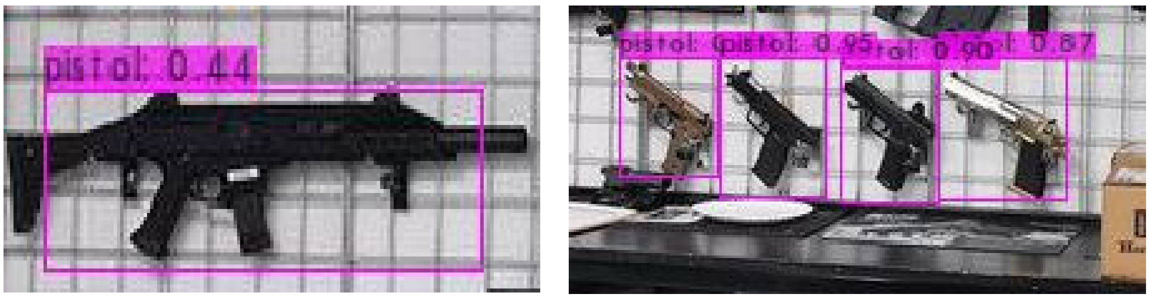
(**a**) (**b**)

Figure 15. Real-time results on Jetson Nano using PI Camera Scaled-YOLOv4. (**a**) Image with a person holding a weapon and pointing toward the aim with background data. (**b**) Image with a weapon on a surface with no background data at different orientations and angles.

We can see that the FPS is closer to the one measured in the videos and the model was able to detect many images of weapons in real time. It can be also observed from the real-time video that converting the model into the TensorRT format for weapon detection has reduced the mAP. Jetson Nano is observed to be confusing non-weaponry objects with the actual weapon class. After reproducing the work of previous researchers, we performed the TensorRT network optimization on the YOLOv4 model and deployed it on Jetson Nano. It can be noted the YOLOv4 gave 3.59 average FPS, which is lower than the Scaled-YOLOv4-based model.

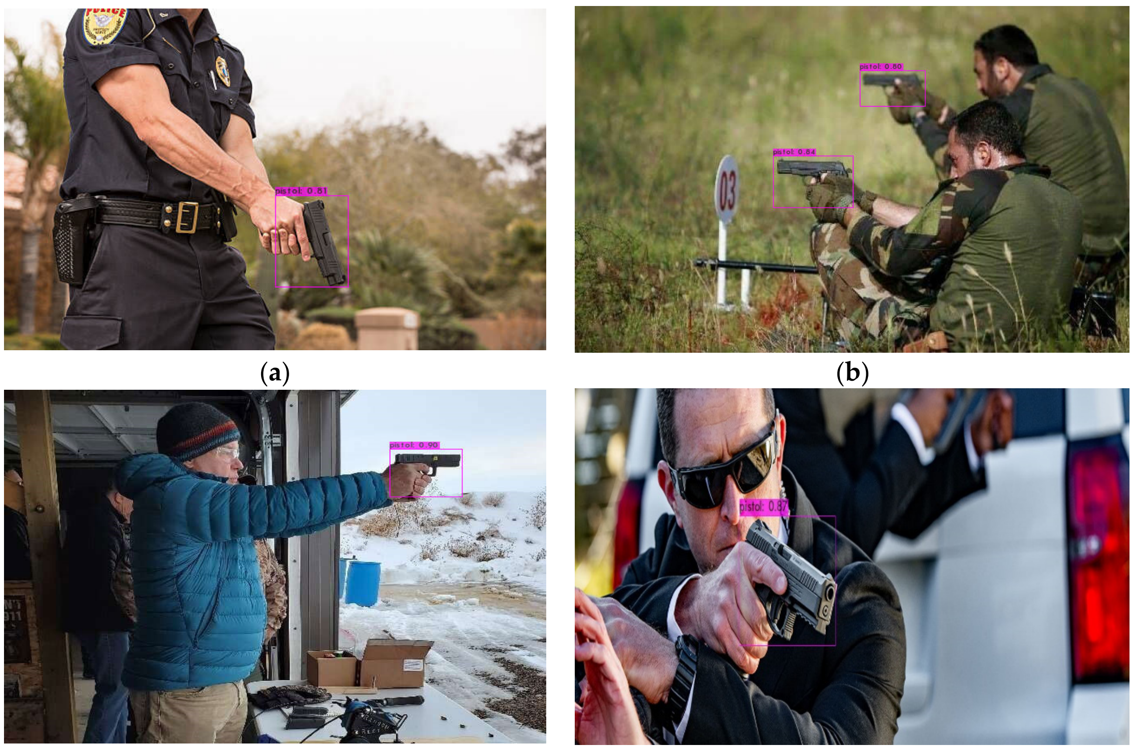
4.7. Weapon Detection of TeslaT4 Trained Scaled-YOLOv4 on RTX 2080TI on Images

Weapon images of different categories having different angles, resolutions, and backgrounds were tested. Some of the results with a weapon as pistol class label with no background and with the background are shown in Figures 16 and 17.



(**a**) (**b**)

Figure 16. Detection Results of Scaled-YOLOv4 with no background. (**a**) Image with a rifle as weapon hanging horizontally on wall. (**b**) Image with multiple pistols hanging diagonally on the wall.

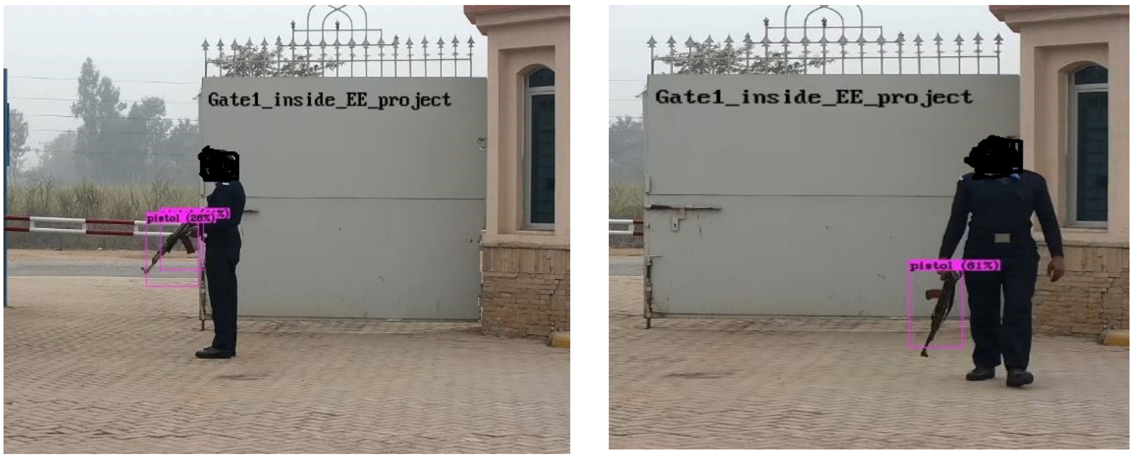


(**c**) (**d**)

Figure 17. Detection results of Scaled-YOLOv4 with background. (**a**) Image with pistol as a weapon held vertically down. (**b**) Image with multiple pistols as a weapon aimed horizontally showing left side angle. (**c**) Image with a pistol as a weapon aimed horizontally showing right side angle. (**d**) Image with a pistol as a weapon held diagonally showing front and side angles.

4.8. Weapon Detection of TeslaT4 Trained Scaled-YOLOv4 on RTX 2080TI in Real-Time CCTV

The ultimate goal of the work was to make it work successfully in real-time CCTV streams and we have successfully achieved that by testing it. The face of the security person holding the weapon is made blurred for privacy purposes. The results obtained after testing it in real time with pistol as label on the targeted object can be seen in Figure 18 below.



(**a**) (**b**)

Figure 18. Detection results of Scaled-YOLOv4 in real-time CCTV stream. (**a**) Image with a rifle as weapon held diagonally. (**b**) Image with a rifle as a weapon held upside down slightly tilted.

4.9. Misdetections

Even though it provides excellent detection results on the weapon in various categories including pistols, revolvers, and rifles, the system had very few false positives that can be improved in the future. An example of a false positive is depicted in Figure 19, which shows a smartphone detected as a pistol with a very low confidence of 25% that can be overcome by model improvement, as well as by setting the threshold.

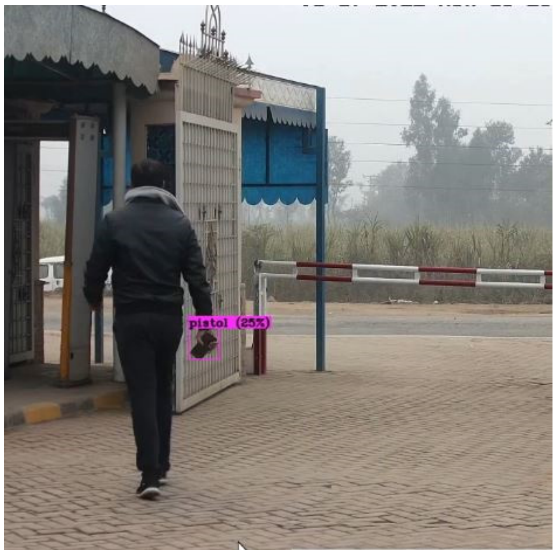


Figure 19. Misdetection results of Scaled-YOLOv4 in real-time CCTV stream.

5. Conclusions

Automatic weapon detection systems need to be installed in public places to prevent criminal activities before they occur. It is a challenging task to develop an accurate and efficient system with minimum false alarms. We have used a deep learning-based approach to CCTV videos to develop a highly accurate weapon detection model to achieve inference in real time. We implemented the Scaled-YOLOv4 algorithm to achieve high accuracy, as well as improved FPS compared to existing methods. Moreover, we investigated the model deployment on embedded edge computing devices to achieve lower latency, higher throughput, and improved privacy. The comparison is performed between our Scaled-YOLOv4-based weapon detector and the previously developed weapon detectors, in terms of accuracy and inference time on different machines. The results indicate that our weapon detection model is superior to the existing approaches in all aspects and generalizes well in detecting all weapon categories. We have also presented the results by changing different hyperparameters of the model and observed their effect on the overall performance of the model. In addition, we provided insights on TensorRT network optimization that has enabled deployment on the popular edge computing device, i.e., Jetson Nano. The network optimization resulted in only a slight reduction in the mAP score (causing the occurrence of higher false positives) and 4.26 FPS, which is a reasonable score for resource-constrained Jetson Nano device. Finally, it is concluded that the presented models give very good performance in terms of accuracy and inference time if implemented on a high-performance GPU (such as RTX 2080TI). On the other hand, acceptable performance can be achieved if the optimized model is implemented on the edge using low-cost Jetson Nano for weapon detection in live CCTV surveillance videos. Nevertheless, further improvement and optimization are required for better accuracy and higher FPS. In the future, we suggest that researchers should use pose estimation techniques in conjunction with object detection to further enhance the mAP score of the weapon detection surveillance system.

Author Contributions: Conceptualization, S.A. and M.G.K.; methodology, S.A. and M.T.B.; software, S.A.; validation, S.A., M.T.B. and M.G.K.; resources, M.G.K. and B.L.; data curation, M.T.B.; writing—original draft preparation, S.A. and M.T.B.; writing—review and editing, M.G.K., B.L. and M.S.; visualization, S.A.; supervision, M.G.K. and M.S.; project administration, M.G.K.; funding acquisition, M.G.K. and B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset for real-time detection was collected and constructed in different phases and data were collected from the internet, extracted from YouTube CCTV videos, through GitHub repositories, data by the University of Granada research group, and internet movie rearm database imfdb.org. This data will be available upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Christchurch Mosque Shootings. Available online: https://en.wikipedia.org/wiki/Christchurch_mosque_shootings (accessed on 10 July 2019).
2. Global Study on Homicide. Available online: https://www.unodc.org/unodc/en/data-and-analysis/global-study-on-homicide.html (accessed on 10 July 2019).
3. Deisman, W. A Report on Camera Surveillance in Canada: Part One. Surveillance Camera Awareness Network (SCAN). 2009. Available online: https://qspace.library.queensu.ca/handle/1974/1906 (accessed on 10 March 2022).
4. Ratcliffe, J. Video Surveillance of Public Places. US Department of Justice, Office of Community Oriented Policing Services: Washington, DC, USA, 2006. Available online: https://www.ojp.gov/ncjrs/virtual-library/abstracts/video-surveillance-public-places (accessed on 13 March 2022).
5. Cohen, N.; Gattuso, J.; MacLennan-Brown, K. CCTV Operational Requirements Manual 2009. Home Office Scientific Development Branch: St. Albans, UK, 2009. Available online: http://designforsecurity.org/downloads/CCTV_Requirements.pdf (accessed on 14 March 2022).
6. Murthy, C.B.; Hashmi, M.F.; Bokde, N.D.; Geem, Z.W. Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms—A Comprehensive Review. *Appl. Sci.* **2020**, *10*, 3280. [CrossRef]
7. Bhatti, M.T.; Khan, M.G.; Aslam, M.; Fiaz, M.J. Weapon Detection in Real-Time CCTV Videos Using Deep Learning. *IEEE Access* **2021**, *9*, 34366–34382. [CrossRef]
8. Darker, I.; Gale, A.; Ward, L.; Blechko, A. Can CCTV reliably detect gun crime? In Proceedings of the International Carnahan Conference on Security Technology, Ottawa, ON, Canada, 8–11 October 2007; pp. 264–271.
9. Darker, I.T.; Gale, A.G.; Blechko, A. CCTV as an automated sensor for firearms detection: Human-derived performance as a precursor to automatic recognition. *Unmanned/Unattended Sens. Sens. Netw. V* **2008**, *7112*, 71120V.
10. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, San Diego, CA, USA, 20–25 June 2005; Volume I, pp. 886–893.

11. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
12. Vajhala, R.; Maddineni, R.; Yeruva, P.R. Weapon Detection in Surveillance Camera Images. Dissertation, 2016. Available online: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-13565 (accessed on 17 March 2022).
13. Nakib, M.; Khan, R.T.; Hasan, M.S.; Uddin, J. Crime Scene Prediction by Detecting Threatening Objects Using Convolutional Neural Network. In Proceedings of the International Conference on Computer, Communication, Chemical, Material and Electronic Engineering, IC4ME2 2018, Rajshahi, Bangladesh, 8–9 February 2018.
14. Verma, G.K.; Dhillon, A. A Handheld Gun Detection using Faster R-CNN Deep Learning. In Proceedings of the 7th International Conference on Computer and Communication Technology, Allahabad, India, 24–26 November 2017; pp. 84–88.
15. Olmos, R.; Tabik, S.; Herrera, F. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing* **2017**, *275*, 66–72. [CrossRef]
16. González, J.L.S.; Zaccaro, C.; Álvarez-García, J.A.; Morillo, L.M.S.; Caparrini, F.S. Real-time gun detection in CCTV: An open problem. *Neural Netw. Off. J. Int. Neural Netw. Soc.* **2020**, *132*, 297–308.
17. Narejo, S.; Pandey, B.; Vargas, D.E.; Rodriguez, C.; Anjum, M.R. Weapon Detection Using YOLO V3 for Smart Surveillance System. *Math. Probl. Eng.* **2021**, *2021*, 9975700. [CrossRef]
18. Velasco-Mata, A.; Ruiz-Santaquiteria, J.; Vallez, N.; Deniz, O. Using human pose information for handgun detection. *Neural Comput. Appl.* **2021**, *33*, 17273–17286. [CrossRef]
19. Ma, Y.; Chen, H.; Huo, J. Assault Rifle Detection and Identification Based on Convolutional Neural Network YOLOv3. In Proceedings of the 2021 3rd World Symposium on Artificial Intelligence, WSAI 2021, Guangzhou, China, 18–20 June 2021; pp. 1–4.
20. Salido, J.; Lomas, V.; Ruiz-Santaquiteria, J.; Deniz, O. Automatic Handgun Detection with Deep Learning in Video Surveillance Images. *Appl. Sci.* **2021**, *11*, 6085. [CrossRef]
21. Ağdaş, M.T.; Türkoğlu, M.; Gülseçen, S. Deep Neural Networks Based on Transfer Learning Approaches to Classification of Gun and Knife Images. *Sak. Univ. J. Comput. Inf. Sci.* **2021**, *4*, 131–141. [CrossRef]
22. Narayanan, M.; Jaju, S.; Nair, A.; Mhatre, A.; Mahalingam, A.; Khade, A. Real-Time Video Surveillance System for Detecting Malicious Actions and Weapons in Public Spaces. *Lect. Notes Data Eng. Commun. Technol.* **2021**, *58*, 153–166.
23. Martín, C.; José, J. Weapon Detection with Deep Learning and Computer Graphics. 2021. Available online: http://hdl.handle.net/10578/27818 (accessed on 18 March 2022).
24. Reddy, R.; Vallabh, K.G.; Sharan, S. Multiclass weapon detection using multi contrast convolutional neural networks and faster region-based convolutional neural networks. In Proceedings of the 2021 2nd International Conference for Emerging Technology, INCET 2021, Belagavi, India, 21–23 May 2021.
25. Ruiz-Santaquiteria, J.; Velasco-Mata, A.; Vallez, N.; Bueno, G.; Alvarez-Garcia, J.A.; Deniz, O. Handgun Detection Using Combined Human Pose and Weapon Appearance. *IEEE Access* **2021**, *9*, 123815–123826. [CrossRef]
26. Singh, A.; Anand, T.; Sharma, S.; Singh, P. IoT Based Weapons Detection System for Surveillance and Security Using YOLOV4. In Proceedings of the 6th International Conference on Communication and Electronics Systems, ICCES 2021, Coimbatre, India, 8–10 July 2021; pp. 488–493.
27. Hashmi TS, S.; Haq, N.U.; Fraz, M.M.; Shahzad, M. Application of Deep Learning for Weapons Detection in Surveillance Videos. In Proceedings of the 2021 International Conference on Digital Futures and Transformative Technologies, ICoDT2 2021, Islamabad, Pakistan, 20–21 May 2021.
28. Madhushree, B.; Sowmya, K.N.; Chennamma, H.R. Automatic weapon detection in video using deep learning. In *Data Engineering and Intelligent Computing*; Springer: Singapore, 2021; pp. 503–510.
29. Dahlan, I.A.; Ariateja, D.; Arghanie, M.A.; Versantariqh, M.A.; David, M.; Fatmawati, U.D. Sistem Deteksi Senjata Otomatis Menggunakan Deep Learning Berbasis CCTV Cerdas. *J. Sist. Cerdas* **2021**, *4*, 126–141.
30. Kaya, V.; Tuncer, S.; Baran, A. Detection and Classification of Different Weapon Types Using Deep Learning. *Appl. Sci.* **2021**, *11*, 7535. [CrossRef]
31. Ingle, P.Y.; Kim, Y.-G. Real-Time Abnormal Object Detection for Video Surveillance in Smart Cities. *Sensors* **2022**, *22*, 3862. [CrossRef]