Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications

Conference Paper · July 2020

DOI: 10.1109/ICESC48915.2020.9155832

5 authors, including:

Mohana Mohana
R. V. College of Engineering
51 PUBLICATIONS **852** CITATIONS
SEE PROFILE

Ayush Jain
Rashtreeya Vidyalaya College of Engineering
4 PUBLICATIONS **142** CITATIONS
SEE PROFILE

Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC 2020)
IEEE Xplore Part Number: CFP20V66-ART; ISBN: 978-1-7281-4108-4

Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications

Harsh Jain, Aditya Vikram, Mohana, Ankit Kashyap, Ayush Jain
Telecommunication Engineering, RV College of Engineering®
Bengaluru, Karnataka, India.

***Abstract*- Security is always a main concern in every domain, due to a rise in crime rate in a crowded event or suspicious lonely areas. Abnormal detection and monitoring have major applications of computer vision to tackle various problems. Due to growing demand in the protection of safety, security and personal properties, needs and deployment of video surveillance systems can recognize and interpret the scene and anomaly events play a vital role in intelligence monitoring. This paper implements automatic gun (or) weapon detection using a convolution neural network (CNN) based SSD and Faster RCNN algorithms. Proposed implementation uses two types of datasets. One dataset, which had pre-labelled images and the other one is a set of images, which were labelled manually. Results are tabulated, both algorithms achieve good accuracy, but their application in real situations can be based on the trade-off between speed and accuracy.**

Keywords— Computer vision, weapon detection, Faster RCNN, SSD, CCTV, Artificial Intelligence (AI).

I. INTRODUCTION

Weapon or Anamoly detection is the identification of irregular, unexpected, unpredictable, unusual events or items, which is not considered as a normally occurring event or a regular item in a pattern or items present in a dataset and thus different from existing patterns. An anomaly is a pattern that occurs differently from a set of standard patterns. Therefore, anomalies depend on the phenomenon of interest [3] [4]. Object detection uses feature extraction and learning algorithms or models to recognize instances of various category of objects [6]. Proposed implementation focuses on accurate gun detection and classification. Also concerned with accuracy, since a false alarm could result in adverse responses [11] [12]. Choosing the right approach required to make a proper trade-off between accuracy and speed. Figure 1 shows the methodology of weapons detection using deep learning. Frames are extracted from the input video. Frame differencing algorithm is applied and bounding box created before the detection of object [7] [8] [14].

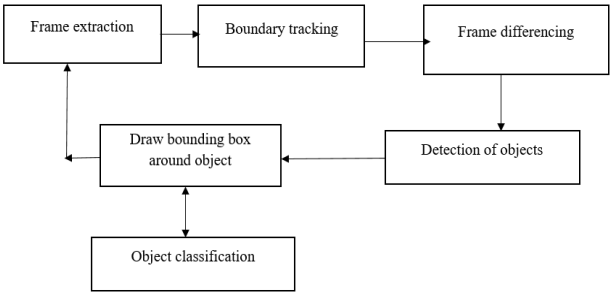

Fig.1.Methodology

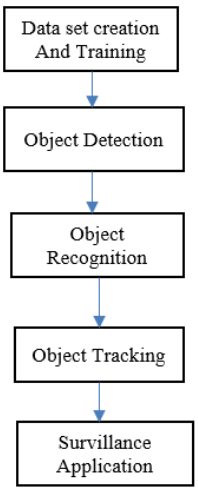

Fig.2. Detection and Tracking

The flow of object detection and tracking as shown in figure 2. Dataset is created, trained and fed to object detection algorithm. Based on application suitable detection algorithm (SSD or fast RCNN) chosen for gun detection. The approach addresses a problem of detection using various machine-learning models like Region Convolutional Neural Network (RCNN), Single Shot Detection (SSD) [2][9][15].

II. IMPLEMENTATION

A. Resources or components used for implementation

- *OpenCV 3.4*- Open source computer vision library version 3.4.
- *Python 3.5*- High level programming language used for various image-processing applications.
- *COCO Dataset*- Dataset consisting of common objects with respective labels.
- *Anaconda and Tensorfflow 1.1*
- *NVIDIA GeForce 820M GPU*-GeForce is a brand of graphics processing units designed by Nvidia.

B. Dataset Specifications

Case – I: Video specifications

- System Configuration- Intel i5 7th Generation (4 Cores)
- Clock Speed- 2.5 GHz
- GPU- NVIDIA GeForce 820M
- Input Frames per Second- 29.97 fps
- Output Frames per Second- 0.20 fps
- Video Format- .mov
- Video Size- 4.14 MB
- COCO and self-created image dataset

- Number of classes trained- 5

Case – II: Image specifications

- System Configuration- Intel i5 7th Generation (4 Cores)
- Clock Speed- 2.5 GHz
- GPU- NVIDIA GeForce 820M
- Input Image Size- 200-300 KB
- Training Time- ~0.6 seconds(SSD)
- ~1.7 seconds(RCNN)
- Image Format - .JPG
- COCO and self-created image dataset
- Number of classes trained for- 5

C. Assumptions and Constraints made for implementation

- The gun is in line of sight of camera and fully/partially exposed to the camera.
- There is enough background light to detect the ammunition.
- GPU with high-end computation power were used to remove lag in the ammunition detection.
- This is not a completely automated system. Every gun detection warning will be verified by a person in charge.

D. FASTER R-CNN

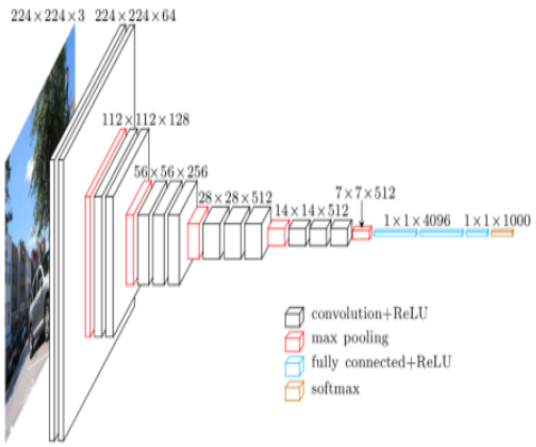

Fig 3. Layers in CNN Architecture [5]

Layers of CNN and faster RCNN architecture depicted in figure 3 and 4 respectively. It has two networks RPN to generate region proposals and network for object detection. To generate region proposals it uses selective search approach. Anchors or region boxes are ranked by RPN network.

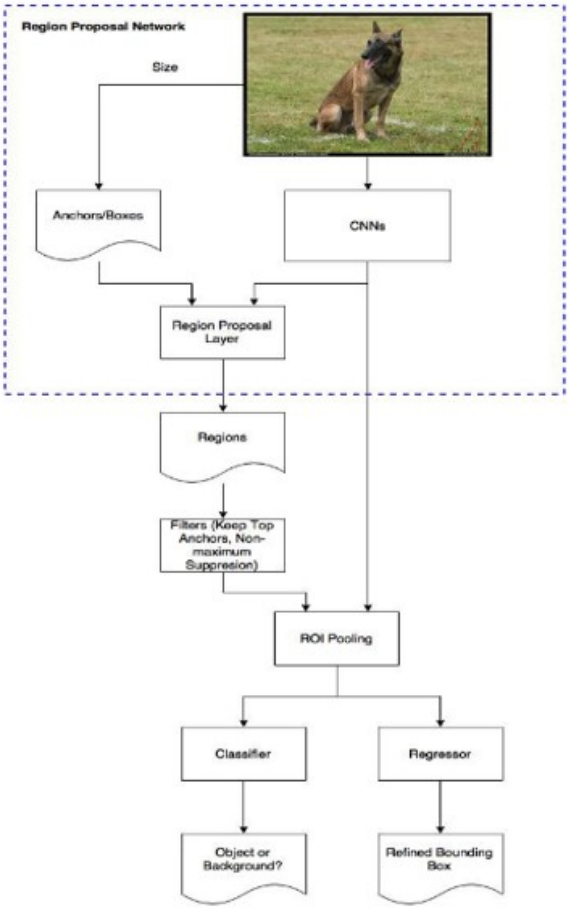

Fig 4. Faster R-CNN [5]

Dataset Creation and Training

Images are downloaded in bulk using Fatkun Batch Image Downloader (chrome extension) which can download multiple Google Images at once. Then the downloaded images are labelled. 80% of total images used for training and 20% images for testing. The created ammunition dataset was then trained using Single Shot Detector (SSD) model and made 2669 iterations/steps on the model to ensure that the loss is less than 0.05 in order to increase the accuracy and precision. Figure 5 shows folder with test and train images. Figure 6 shows image with labels.

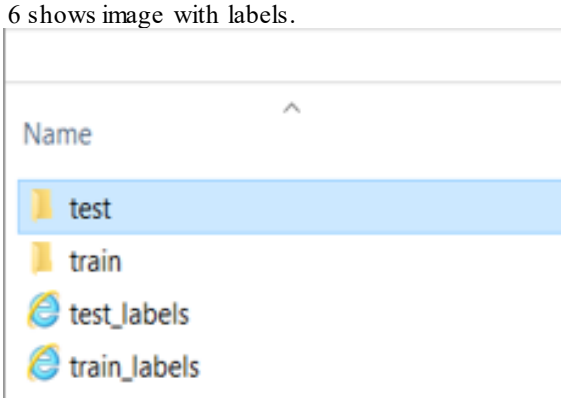

Fig.5. Folder with test and train images

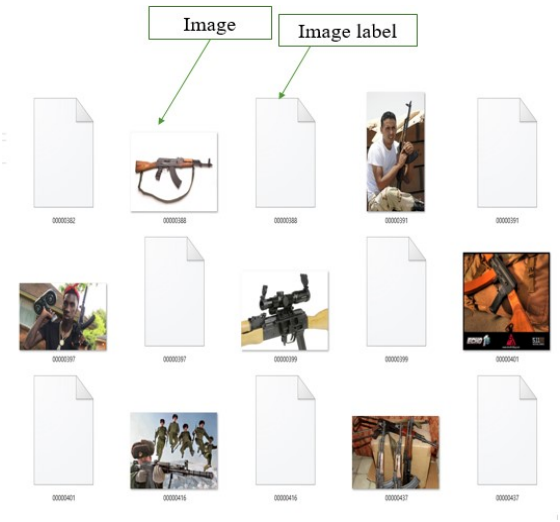

Fig.6. Image along with its label

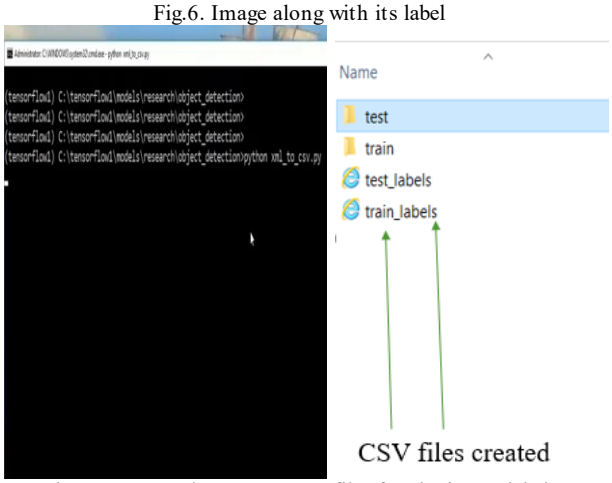

Fig.7. Command to create CSV files for the image labels

XML data is converted into CSV file by executing this command in Anaconda Prompt: python xml_to_csv.py as shown in figure 7.

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	00000022.	600	450	ak47	142	197	567	300
3	00000028.	600	439	ak47	251	11	388	392
4	00000030.	600	900	ak47	90	221	467	374
5	00000034.	500	389	ak47	56	42	444	322
6	00000038.	600	450	ak47	19	9	597	402
7	00000039.	600	600	ak47	160	240	380	382
8	00000039.	600	600	ak47	245	288	400	434
9	00000039.	600	600	ak47	6	160	367	381
10	00000052.	600	438	ak47	325	11	388	101
11	00000052.	600	438	ak47	383	1	435	191
12	00000055.	482	200	ak47	263	147	318	180
13	00000079.	480	480	ak47	2	332	480	436
14	00000079.	480	480	ak47	1	198	478	310
15	00000098.	240	240	ak47	5	94	235	147
16	00000099.	600	427	ak47	259	73	417	206
17	00000112.	600	800	ak47	175	258	494	503
18	00000112.	600	800	ak47	1	200	293	323
19	00000112.	600	800	ak47	379	293	545	587
20	00000121.	600	376	ak47	1	46	599	259
21	00000122.	300	257	ak47	119	54	200	103
22	00000127.	380	570	ak47	195	218	372	570
23	00000130.	480	480	ak47	21	246	176	295
24	00000130.	480	480	ak47	11	12	194	70
25	00000130.	480	480	ak47	13	87	158	165
26	00000144.	600	450	ak47	21	19	597	359
27	00000147.	360	170	ak47	8	59	344	163
28	00000151.	600	337	ak47	1	43	305	301
29	00000163.	600	963	ak47	238	423	419	869
30	00000169.	480	480	ak47	4	132	478	330

test_labels

Fig.8. CSV file of testing dataset

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	00000001.	600	346	ak47	48	25	562	297
3	00000003.	600	396	ak47	42	111	573	308
4	00000011.	320	212	ak47	22	67	306	148
5	00000014.	600	450	ak47	2	144	599	331
6	00000018.	600	400	ak47	170	137	490	397
7	00000020.	600	450	ak47	13	107	582	368
8	00000031.	221	160	ak47	16	7	212	153
9	00000040.	600	225	ak47	29	32	580	199
10	00000041.	600	600	ak47	5	228	597	375
11	00000051.	600	178	ak47	9	6	595	174
12	00000059.	599	448	ak47	89	31	404	186
13	00000059.	599	448	ak47	444	124	569	239
14	00000062.	600	337	ak47	4	115	553	226
15	00000066.	200	200	ak47	7	74	194	127
16	00000071.	600	428	ak47	153	104	550	387
17	00000072.	600	718	ak47	87	32	543	686
18	00000075.	600	216	ak47	11	22	587	195
19	00000076.	320	213	ak47	6	57	315	150
20	00000078.	600	450	ak47	20	33	554	415
21	00000083.	600	376	ak47	72	140	562	267
22	00000084.	600	800	ak47	244	126	348	778
23	00000084.	600	800	ak47	244	126	348	778
24	00000086.	600	300	ak47	13	83	584	255
25	00000088.	600	337	ak47	224	3	413	336
26	00000088.	600	337	ak47	224	3	413	336
27	00000092.	600	234	ak47	34	42	586	218
28	00000092.	600	234	ak47	34	42	586	218
29	00000096.	600	337	ak47	179	138	560	320
30	00000096.	600	337	ak47	179	138	560	320

train_labels

Fig.9. CSV file of Training dataset

Figure 8 and 9 shows generated CSV file of test and training dataset. Figure10 shows the training of Faster R-CNN algorithm with training loss less than 0.15.Figure 11 describes pseudo code of faster RCNN.

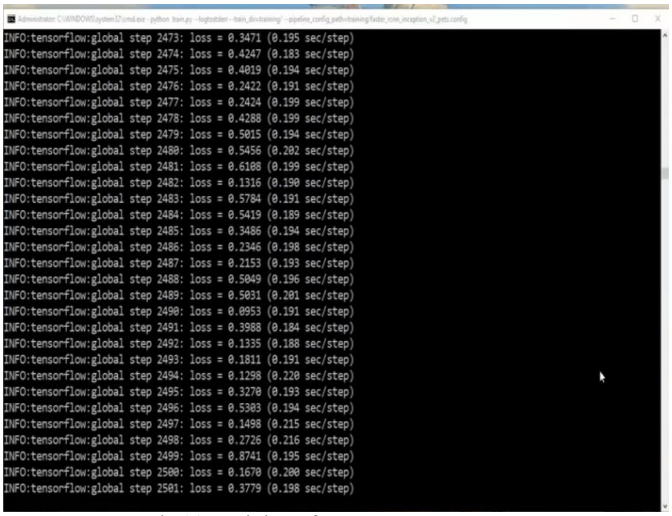

Fig.10.Training of Faster R-CNN

Pseudo code of faster RCNN

Initialize the parameters
confThreshold =>0.5 #Confidence threshold for bounding box predictions
maskThreshold -> 0.3 # Mask threshold for binary masks
Load the models
weightsPath -> .../frozen_inference_graph. pb #pre-trained weights
configPath ->.../rcnn_inception_v2_coco_pets. pbtxt #text graph file to load model onto OpenCV

Initialize the video stream
vs -> cv2.VideoCapture(.../ weapon_video.mov) #loading the video

Process each frame
grabbed, frame -> vs. read () #reading each frame and returning the coordinates of the frames
blob -> cv2.dnn.blobFromImage(frame) #creation of 4D blob from a frame
net. setInput (blob) #passing the blob as an input to the ConvNets

Extract the bounding box and drawing the box for each detected object
for i in range(numDetections):
box -> boxes [0, 0, i]
mask -> masks[i]
left -> int (frameW * box [3]) #Acquiring bounding boxes
top ->int (frameH * box [4])
right -> int (frameW * box [5])
bottom -> int (frameH * box [6])
cv2.rectangle(frame, (startX, startY), (endX, endY), colour, 2) #drawing bounding boxes

Fig.11. Pseudocode of faster RCNN

E. SSD (Single Shot Detector)

SSD algorithm reached new milestones in terms of precision and performance detection. SSD speeds up the process by eliminating the need of region proposal network. To overcome the drop in accuracy SSD brings a few technology including default boxes and multi-scale features. These improvements allow SSD to match the Faster R-CNN's accuracy using lower resolution images, which further pushes speed higher. Average scoring is around 74% MAP and 59 fps on COCO dataset. Figure 12 shows SSD VGG-16 Architecture [1] [5] [10].

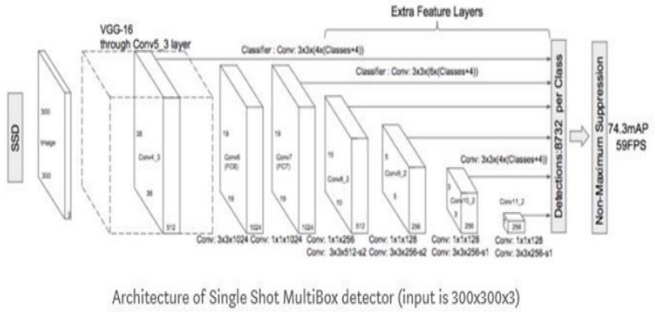

Fig.12. SSD VGG-16 Architecture [10]

Training of SSD

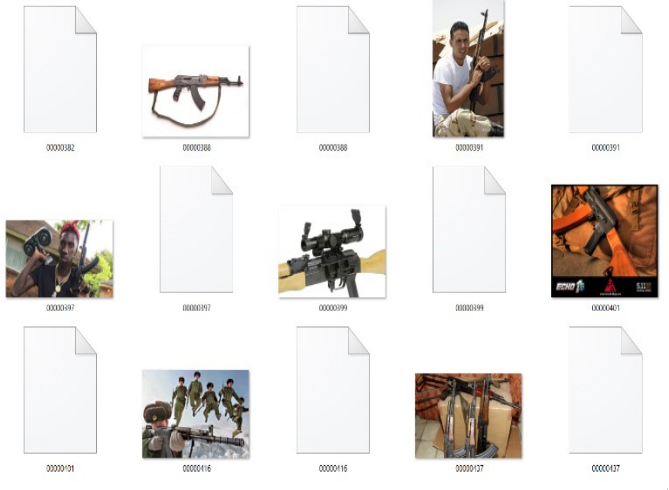

Fig.13. Manually labelled images

Figure 13 shows manually labelled images in xml format. Total 838 considered for training and 241 images for testing (22% testing and 78% training). XML data is converted into CSV file by executing this command in Anaconda Prompt:- python xml_to_csv.py. Ground truth box and scale is represented by

$$predicted_{center} * variance0 = \frac{real_{center} - prior_{center}}{prior_{hw}}$$

$$\exp(predicted_{hw} * variance1) = \frac{real\_hw}{prior_{hw}}$$

"variance0" and "variance1" are 0.1 and 0.2 in general.

Object detection and recognition

- To make sure object is detected, changes are made in the label map and tf_record file.
- Label map is the file which stores the total number of types of objects that will be detected. AK47 is added in the label map.
- The images are converted into tf_record format in Tensorflow so that they can processed in batches. AK47 is added in the tf_record format.

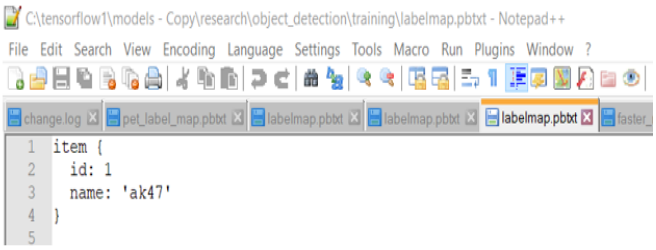

Fig.14.Label map file to record all the names of classes

Figure 14 shows label map to identify classes. Image dataset is created and converted into CSV format. It is now ready to train by SSD model by using command as shown in figure 15.

Fig.15. Command used to run SSD Model

SSD model trained until to obtain loss training 0.05 as shown in figure16. Corresponding pseudo code of SSD implementation as shown in figure 17.

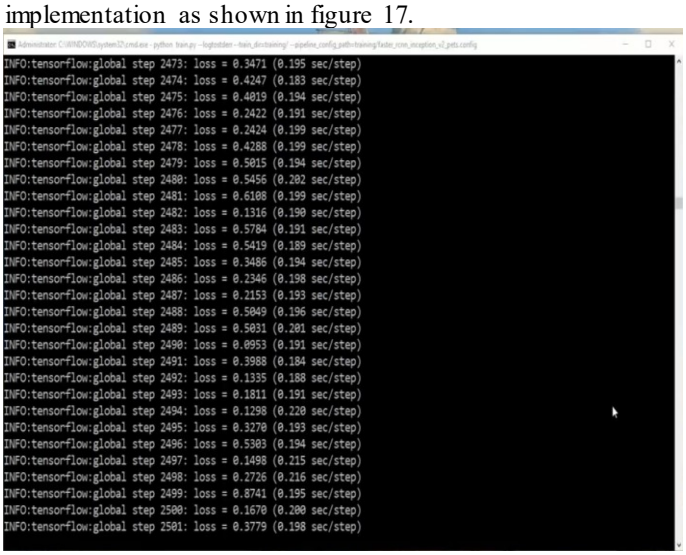

Fig.16. SSD model is trained until loss is reduced to 0.05

Pseudo code of SSD

Some of the drawbacks of faster RCNN compared to SSD model are training data is unwieldy and too long. SSD algorithm is a faster option, training occurs in more phases. Network is too slow at inference time and cannot provide accurate real time detection due to time spent on region proposals. SSD fills these shortcomings.

```
Initialize the parameters
confThreshold ->0.5
maskThreshold -> 0.3
Load the models
weightsPath -> …/frozen_inference_graph. pb
configPath ->…/ssd_inception_v2_coco_2018_01_28. pbtxt
Initialize the video stream
vs -> cv2.VideoCapture(…/ traffic_video_source.mp4)
Process each frame
grabbed, frame -> vs. read ()
blob -> cv2.dnn.blobFromImage(frame) #creation of 4D blob from a frame
net. setInput (blob)
Extract the bounding box and drawing the box for each detected object
for i in range(numDetections):
box -> boxes [0, 0, i]
mask -> masks[i]
left -> int (frameW * box [3])
top ->int (frameH * box [4])
right -> int (frameW * box [5])
bottom -> int (frameH * box [6])
cv2.rectangle(frame, (startX, startY), (endX, endY), colour, 2)
```

Fig.17. Pseudo code for SSD implementation

III. Results and Analysis

A. *Detection of weapons using SSD algorithm*

Case 1: Using pre-labelled dataset

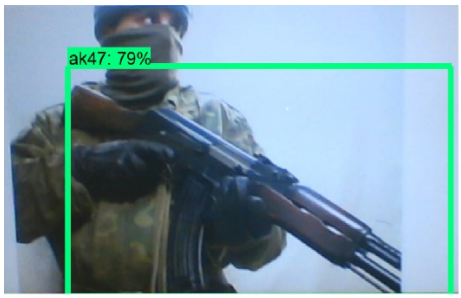

Fig.18. Detection of AK47 gun

Figure 18. Shows detection of a gun AK47 using SSD algorithm. Accuracy of detection is 79%. Further accuracy can be increased by increasing more number training samples.

Case 2: Using self-created dataset

Fig.19. COLT M1911 Detected with 72% accuracy

Fig.20. Smith & Wesson Model Detected with 67% accuracy

Figure 19 and 20 shows detection of COLT M1911 and Smith & Wesson Model gun with an accuracy of 72% and 67%.

B. *Detection of weapons using Faster R-CNN*

Case 1: Using pre-labelled image dataset

Fig.21. Detection of AK47 using Faster R-CNN

Figure 21 shows detection of AK47 gun using faster RCNN with an accuracy of 99% which shows Faster R-CNN provides the superior accuracy.

Fig.22. Detection of AK47 using Faster R-CNN

Figure 22 shows detection of AK47 in hands of army with accuracy of 99% and 81%.

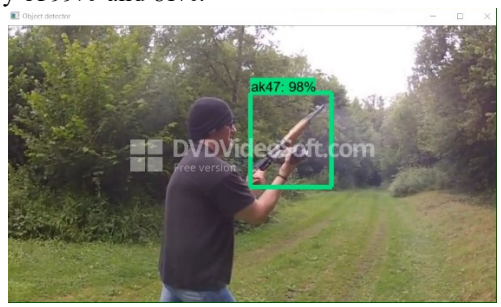

Fig.23. AK47 gun detection from video stream

Figure 23 shows detection of AK47 gun from video stream with an accuracy of 98%.

Case 2: Using self-created image dataset

Fig.24. Detection of Colt M1911 gun

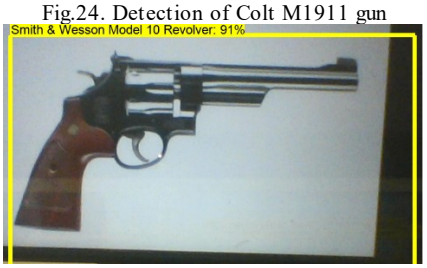

Fig.25. Detection of Smith & Wesson Model 10 gun

Figure 24 and 25 shows detection of Colt M1911 gun Smith & Wesson Model 10 gun using Faster R-CNN algorithm with the accuracy of 74% and 91%.

C. *Performance Analysis*

Faster R-CNN

TABLE I. PERFORMANCE ANALYSIS: FASTER R-CNN ALGORITHM

Gun Type	Average Accuracy	Speed (S)	Gun Detection	Correct Classification
AK-47	94%	1.28	Yes	Yes
Colt M1911	74%	1.63	Yes	Yes
S&W Model 10	91%	1.74	Yes	Yes
UZI Model	88%	1.49	Yes	No
Remington Model	76%	1.89	Yes	No

From Table 1, it shows that highest average accuracy is obtained for pre-labelled dataset (i.e. AK47) and the Colt M1911, Smith & Wesson Model 10, UZI Model, Remington Model obtained accuracy in the range of 76% to 91%. Faster R-CNN achieves an average Accuracy of 84.6% and average speed 1.606s/frame. This concludes that the pre labelled dataset provided better accuracy because it is trained for millions of images in comparison to the self-created dataset.

SSD

TABLE II. PERFORMANCE ANALYSIS: SSD ALGORITHM

Gun Type	Average Accuracy	Speed (S)	Gun Detection	Correct Classification
AK-47	80%	0.67	Yes	Yes
Colt M1911	70%	0.89	Yes	Yes
S&W Model 10	66%	0.78	Yes	Yes
UZI Model	81%	0.61	Yes	No
Remington Model	72%	0.73	Yes	No

Table 2 shows performance Analysis for the SSD algorithm. Obtained SSD Average Accuracy is 73.8% and average Speed 0.736 s/frame. Trained model for 5 classes of guns such as AK47, Smith and Wesson Model 10, Colt M1911, UZI Model and Remington model and obtained maximum confidence level for AK47 gun. Used SSD and RCNN Inception V2 models to train the guns. SSD took 12 more hours to train model in comparison with RCNN model but provided lower accuracy. Faster R-CNN achieved 10.8% more average accuracy than SSD algorithm. SSD provides faster speed than Faster R-CNN by 0.7 seconds. Pre-labeled dataset like AK47 gun provides higher accuracy in SSD and Faster R-CNN models, compared to self-created image dataset.

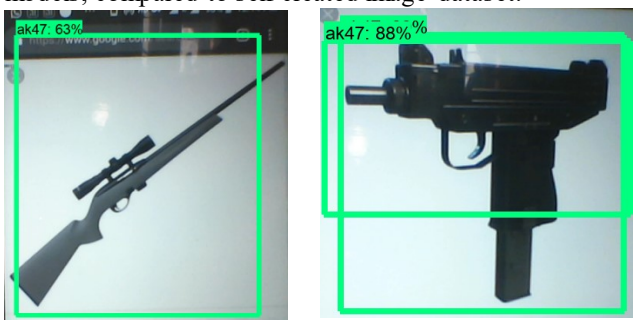

Fig.26 (A) UZI erroneously detected as AK 47 (b) Remington Model erroneously detected as AK47

Figure 26 shows UZI and Remington model gun detection the detection for UZI Model gun and Remington Model gun is successful but the classification is incorrect. These guns are detected as AK47. Further Edge analytics concepts may be applied [13].

IV. Conclusions

SSD and Faster RCNN algorithms are simulated for pre labeled and self-created image dataset for weapon (gun) detection. Both the algorithms are efficient and give good results but their application in real time is based on a tradeoff between speed and accuracy. In terms of speed, SSD algorithm gives better speed with 0.736 s/frame. Whereas Faster RCNN gives speed 1.606s/frame, which is poor compared to SSD. With respect to accuracy, Faster RCNN gives better accuracy of 84.6%. Whereas SSD gives an accuracy of 73.8%, which is poor compared to faster RCNN.SSD provided real time detection due to faster speed but Faster RCNN provided superior accuracy. Further, it can be implemented for larger datasets by training using GPUs and high-end DSP and FPGA kits [16] [17].

REFERENCES

[1] Wei Liu et al., "SSD: Single Shot MultiBox Detector", European Conference on Conputer Vision, Volume 169, pp 20-31 Sep. 2017.

[2] D. Erhan et al., "Scalable Object Detection Using Deep Neural Networks," *IEEE Conference on Computer Vision and Pattern Recognition*(CVPR),2014.

[3] Ruben J Franklin et.al., "Anomaly Detection in Videos for Video Surveillance Applications Using Neural Networks," *International Conference on Inventive Systems and Control,2020.*

[4] H R Rohit et.al., "A Review of Artificial Intelligence Methods for Data Science and Data Analytics: Applications and Research Challenges,"*2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud),* 2018.

[5] Abhiraj Biswas et. al., "Classification of Objects in Video Records using Neural Network Framework," *International conference on Smart Systems and Inventive Technology,2018.*

[6] Pallavi Raj et. al.,"Simulation and Performance Analysis of Feature Extraction and Matching Algorithms for Image Processing Applications" *IEEE International Conference on Intelligent Sustainable Systems,2019.*

[7] Mohana et.al., "Simulation of Object Detection Algorithms for Video Survillance Applications", *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud),2018.*

[8] Yojan Chitkara et. al.,"Background Modelling techniques for foreground detection and Tracking using Gaussian Mixture model" *International Conference on Computing Methodologies and Communication,2019.*

[9] Rubner et.al, "A metric for distributions with applications to image databases", International Conference on Computer Vision,2016.

[10] N. Jain et.al., "Performance Analysis of Object Detection and Tracking Algorithms for Traffic Surveillance Applications using Neural Networks," *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, 2019.

[11] A. Glowacz et.al., "Visual Detection of Knives in Security Applications using Active Appearance Model",Multimedia Tools Applications, 2015.

[12] S. Pankanti et.al.,"Robust abandoned object detection using region-level analysis,"*International Conference on Image Processing*,2011.

[13] Ayush Jain et.al.,"Survey on Edge Computing - Key Technology in Retail Industry" *International Conference on Intelligent Computing and Control Systems,2019.*

[14] Mohana et.al., Performance Evaluation of Background Modeling Methods for Object Detection and Tracking," *International Conference on Inventive Systems and Control,2020.*

[15] J. Wang et.al., "Detecting static objects in busy scenes", Technical Report TR99-1730, Department of Computer Science, Cornell University, 2014.

[16] V. P. Korakoppa et.al., "An area efficient FPGA implementation of moving object detection and face detection using adaptive threshold method," *International Conference on Recent Trends in Electronics, Information & Communication Technology,*2017.

[17] S. K. Mankani et.al., "Real-time implementation of object detection and tracking on DSP for video surveillance applications,"*International Conference on Recent Trends in Electronics, Information & Communication Technology,*2016.