

**REAL TIME WEAPON DETECTION USING
CONVOLUTIONAL NEURAL NETWORK**

BY

**A. N. M. JUBAER
ID: 162-15-7850**

**ABU SAYEM
ID: 162-15-7682**

**MD. ASHIKUR RAHMAN
ID: 162-15-7723**

This Report Presented in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Dr. Sheak Rashed Haider Noori
Associate professor & Associate Head
Department of CSE
Daffodil International University

Co-Supervised By

Ms. Nusrat Jahan
Senior Lecturer
Department of CSE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

MAY 2020

APPROVAL

This Project titled “Real Time Weapon Detection Using Convolutional Neural Network”, submitted by A. N. M. Jubaer, Abu Sayem and Md. Ashikur Rahman to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 8th July, 2020.

BOARD OF EXAMINERS



Dr. Syed Akhter Hossain

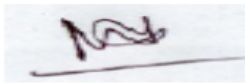
Chairman

Professor and Head

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University



Dr. Md. Ismail Jabiullah

Internal Examiner

Professor

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University



Nazmun Nessa Moon

Assistant Professor

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Internal Examiner



Dr. Mohammad Shorif Uddin

Professor

Department of Computer Science and Engineering

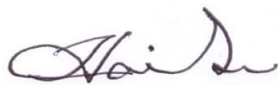
Jahangirnagar University

External Examiner

DECLARATION

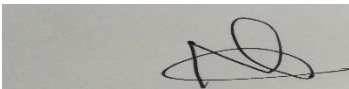
We hereby declare that; this project has been done by us under the supervision of **Dr. Sheak Rashed Haider Noori, Associate Professor and Associate Head, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Dr. Sheak Rashed Haider Noori
Associate Professor and Associate Head
Department of CSE
Daffodil International University

Co-Supervised by:



Ms. Nusrat Jahan
Sr. Lecturer
Department of CSE
Daffodil International University

Submitted by:



A. N. M. Jubaer
ID: 162-15-7850
Department of CSE
Daffodil International University

Abu Sayem

Abu Sayem

ID: 162-15-7682

Department of CSE

Daffodil International University



Md. Ashikur Rahman

ID: 162-15-7723

Department of CSE

Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Dr. Sheak Rashed Haider Noori, Associate Professor and Associate Head**, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of “*Deep Learning*” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to Prof. **Dr. Syed Akhter Hossain, Head**, Department of CSE, for his kind help to finish our project and also to other faculty members and the staffs of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

Security is a big issue these days. People are becoming violent and the tendency to be armed is being noticed. Carrying a pistol or knife is now very common. Security guards are trying their best, but we think technology can be used very effectively in this case, such as machine learning. We designed this model to detect real-time weapons. It takes video input from a camera, CCTV, or any other device and can detect if there are any weapons. Here we use Deep Learning techniques based on CNN (Convolutional Neural Network). Since so many algorithms exist, it was difficult for us to choose an algorithm that suited our work. Finally, we choose “YOLOv3”, an algorithm that uses convolutional neural networks for object detection. Then we faced the biggest hurdle, "DATA". There was not enough data on the internet to train our model. Then, we decide to produce the data ourselves. We used the VLC media player to collect images from videos and then used LabelImg, a python library to label those images. After meeting all the pre-requirements, we started writing our scripts and training the model. Eventually, we were able to build what we expected. We have built a model that can detect weapons in real-time and it is very much possible to send signals to specific destinations or play an alarm.

TABLE OF CONTENTS

CONTENTS	PAGE
Board of examiners	i
Declaration	iii
Acknowledgements	v
Abstract	vi
CHAPTER	
CHAPTER 1: Introduction	1-2
1.1 Introduction	1
1.2 Motivation	1
1.3 Rationale of the Study	1
1.4 Research Questions	2
1.5 Expected Output	2
Chapter 2: Background	3-7
2.1 Introduction	3
2.2 Related Works	3
2.3 Research Summary	6
2.4 Scope of the Problem	6
2.5 Challenges	7
Chapter 3: Research Methodology	8-22
3.1 Introduction	8

3.2 Research Subject and Instrumentation	8
3.3 Data Collection Procedure	9
3.4 Statistical Analysis	14
3.5 Implementation Requirements	21
Chapter 4: Experimental Results and Discussion	23-28
4.1 Introduction	23
4.2 Experimental Results	23
4.3 Descriptive Analysis	28
4.4 Summary	28
Chapter 5: Summary, Conclusion, Recommendation and Implication for Future Research	29-30
5.1 Summary of the Study	29
5.2 Conclusions	29
5.3 Recommendations	29
5.4 Implication for Further Study	30
APPENDIX	
REFERENCES	31-32

LIST OF FIGURES

FIGURES	PAGE NO
Figure 2.2.1.1: Basic block diagram of this IVS system	3
Figure 2.2.2.1: Demonstration of positive results in an environment with multiple objects.	4
Figure 2.2.2.2: Object detection result chart	5
Figure 2.2.3.1: Performance graph in terms of ROC and AUC C (a) SVM (b) KNN and (c) Ensemble Tree	6
Figure 3.3.1: Poor image quality	10
Figure 3.3.2: VLC Media Player logo	10
Figure 3.3.3: VLC Media Player recording images settings	11
Figure 3.3.4: Labelling with labellmg software	12
Figure 3.3.5: Image with clear view of weapon	13
Figure 3.3.6: Open Image Dataset explore option (weapon selected)	13
Figure 3.4.1: YOLOv3 bounding box	15
Figure 3.4.2: An image in RGB channels	16
Figure 3.4.3: Formulas for converting network output to bounding box prediction	17
Figure 3.4.4: Anchor concept	17
Figure 3.4.5: Upsample with stride of 2	19
Figure 3.4.6: Training log for batch number 3905	20
Figure 3.4.7: Training log for loss	21
Figure 4.2.1: Input for experiment 1	24
Figure 4.2.2: Result of experiment 1	24
Figure 4.2.3: Input of experiment 2	26
Figure 4.2.4: Result of experiment 2	27
Figure 4.2.5: Input of experiment 3	27
Figure 4.2.6: Result of experiment 3	27

CHAPTER 1

Introduction

1.1 Introduction

Although the concept of machine learning is quite old, its use has been increasing at a significant rate over the past decade. Its use has increased day by day in almost all sectors of technology. Its use in other sectors is also quite noticeable. From the present situation it can be inferred that machine learning is going to be the most effective technology in the future. Violence, on the other hand, is a primitive human nature, but over the past decade, people have become increasingly violent. The rate of carrying firearms has increased at a significant rate. The crime rate is rising, the administration is failing to provide security to the public due to lack of skilled manpower and technology. In this research-based project we have tried to show how it is possible to control the growing problem by utilizing machine learning.

1.2 Motivation

Our supervisors have repeatedly encouraged us to do things that can be applied in real life. We were excited to work with machine learning. We wanted to do something with machine learning that would help people in the context of Bangladesh.

The current security situation in Bangladesh and the lack of manpower in the administration have made us interested here.

1.3 Rationale of the Study

It is much easier and cheaper to build efficient and intelligent machines than to create skilled workers, at least in the long run. We think it makes more sense to use a machine in uncreative work since it can work seamlessly in the same accuracy for a long time. Thus it is quite a great choice to train a machine to do the boring but important in accuracy without delay works. And our research is an example of this.

1.4 Research Questions

How can we overcome the lack of skilled manpower by utilizing machine learning in the fight against terrorism?

1.5 Expected Output

A model that can detect weapons in real-time almost perfectly by using Convolutional Neural Network.

CHAPTER 2

Background

2.1 Introduction

Currently, there is a lot of research going on with machine learning. Research is also underway on machine learning in the security sector. Areas of research include cybersecurity, military robot, intelligent CCTV, etc. We are keen to use this technology to make CCTV intelligent.

2.2 Related Works

2.2.1 Automatic Weapons Detection [1]

Designing an integrated system that allows for fast and reliable processing of high-quality video data and in doing so detects and reacts to the presence of a firearm or other weaponry when used in a threatening or dangerous manner. The formulated block diagram of their IVS system is shown in the figure: 2.2.1.1 below.

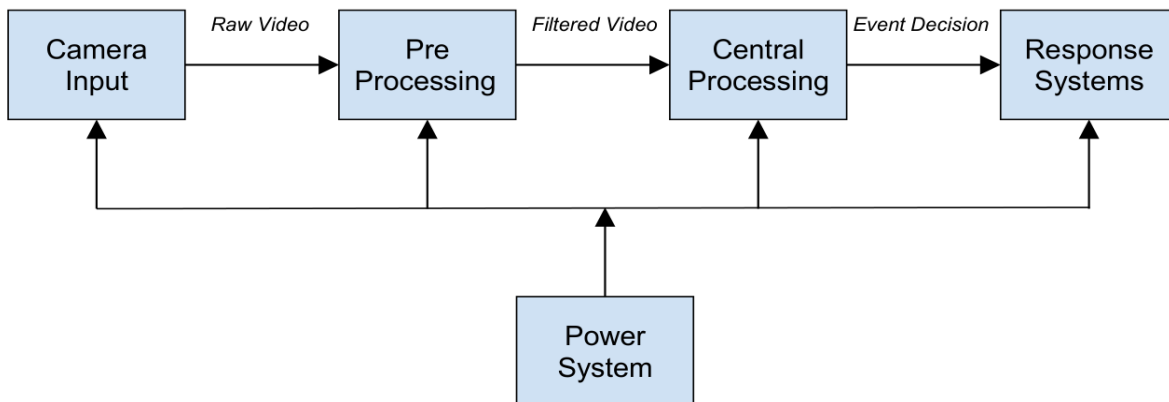


Figure 2.2.1.1: Basic block diagram of this IVS system

2.2.2 Firearm Detection using Convolutional Neural Networks [2]

Designing and Creating a firearm detection system, demonstrating its effectiveness in this task using the YOLO algorithm. They also constructed a dataset based on

the website Internet Movie Firearm Database (IMFDB) for this study. They show that a rapid response from law enforcement agents is the main factor in reducing the number of victims. The demonstration of positive results in an environment with multiple objects of their object is shown in figure 2.2.2.1 below. And the overall object detection result chart is shown in figure 2.2.2.2.



Figure 2.2.2.1: Demonstration of positive results in an environment with multiple objects.

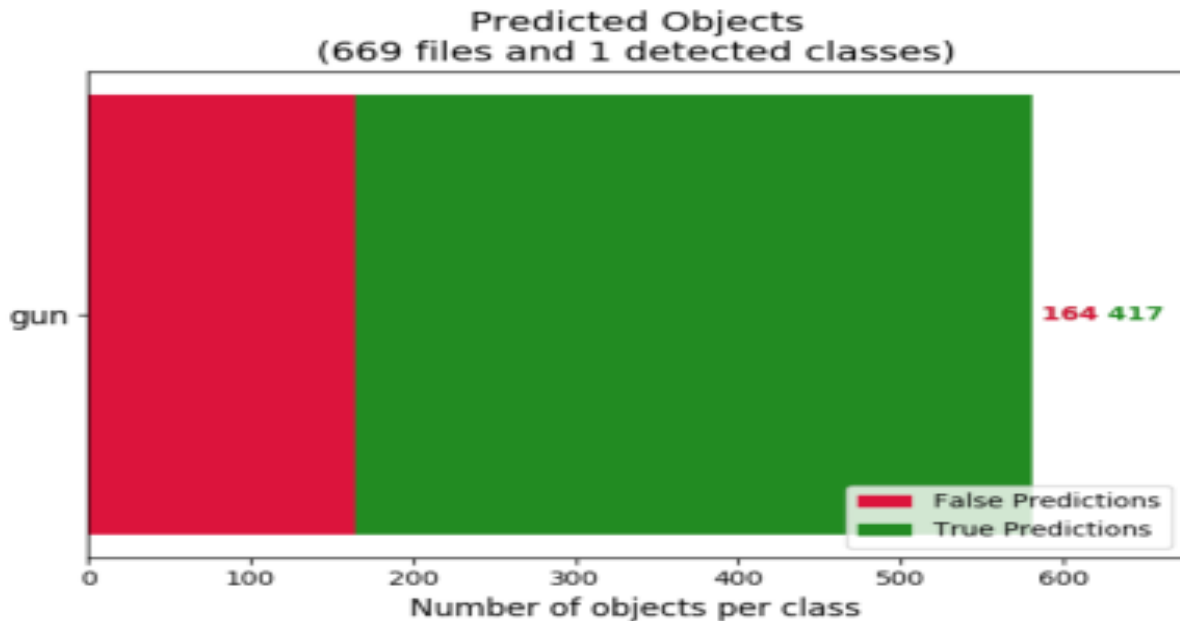


Figure 2.2.2.2: Object detection result chart

2.2.3 A Handheld Gun Detection using Faster R-CNN Deep Learning [3]

They have used Deep Convolutional Network (DCN), a state-of-the-art Faster Region-based CNN model, through transfer learning. They demonstrate that, against the number of several training images, the CNN model magnifies the classification accuracy, which is most advantageous in those practices where generous liberal is often not available. They tested the performance of their system is against various conditions such as varied backgrounds with guns, occlusion, etc. They have used three different classifiers namely Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Ensemble tree for classification and the results are shown in the figure 2.2.3.1.

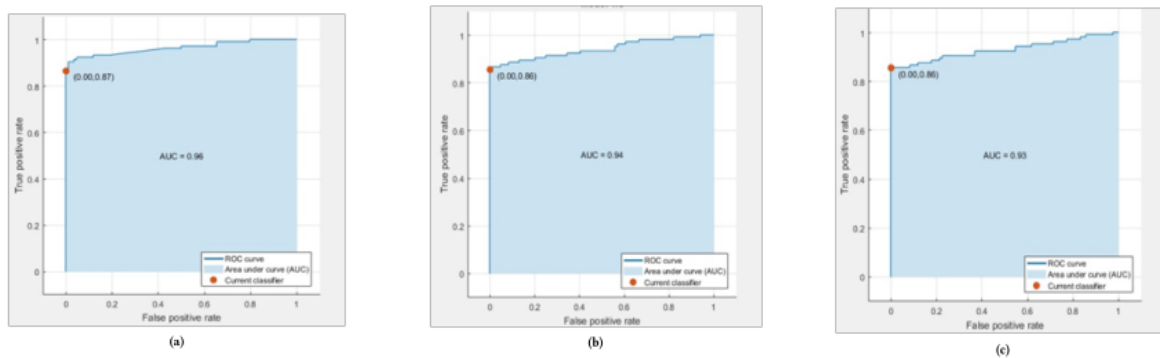


Figure 2.2.3.1: Performance graph in terms of ROC and AUC C

(a) SVM (b) KNN and (c) Ensemble Tree

2.3 Research Summary

Our goal is to effectively address the shortage of skilled manpower in the security sector using machine learning. We want to design such a model that can detect weapons quickly and almost accurately on the fly. We have decided to solve this problem by using any existing algorithm effectively. To train our model we have decided to collect some data from online but we will produce most of the data ourselves. If we succeed our model will be able to detect any object based on train data, but we are more focused on weapon detection.

2.4 Scope of the Problem

- Existing CCTVs are stupid boxes. They can do nothing but capture videos. If they become disabled or unable to capture video they even cannot say so. But using some small machine learning algorithms it is possible to turn it into a smart device. Then they can understand what they capturing. And if they can detect something dangerous, they can send it to a specific destination in a predetermined.
- Drones are a promising technology of the present. But in most cases, it has to be managed by a human. It is possible to automate it using our model. Although we are working on weapon detection, our model can detect any object based on train data.

- Another upcoming technology is the 'military robot' or 'security robot'. Our model can also be applied here.

2.5 Challenges

- The first challenge is to find an algorithm that is compatible with our project. Since there are so many algorithms, we have so many options, we have to work hard to find the right algorithm. We need a fast algorithm with acceptable accuracy that provides output in real-time.
- Data is the most important element of our project. Since the machine learns based on the data, the more the data, the more efficient the machine will be, and the more accurately the machine will be able to detect a object. We need a lot of data for a truly functional model and the kind of data we need is not available on the Internet. Thus, data production is a big challenge.

CHAPTER 3

Research Methodology

3.1 Introduction

Our project, weapon detection is a problem where we tried to identify weapons in picture frames as well as in videos. We do it by using convolutional neural network framework on YOLOv3. YOLOv3 is a convolutional neural network where it uses only one neural network to do the whole job. We used YOLOv3 to train our own custom weapon detector on our own dataset. It does quite a good job. We tried many things before YOLOv3. One of them is TensorFlow. We used faster_rcnn_inception_v2_coco to detect weapons. We chose it because it has some decent speed with decent accuracy. But in the end, as we are focusing on building a real time object detector that can perform detecting weapons in real time mainly from a cc camera, it failed to meet our goals in terms of speed. So, we switched to YOLOv3 to implement our work as it is the state of the art nowadays. It is super-fast with decent accuracy. We used the darknet framework to implement our work. We built our own dataset and labeled them. We also merge some data from OID (Open Image Dataset) to enlarge our dataset. After that, using darknet we trained our own model and got yolo weights. Using that weight, we then ran detection on images or on videos. We used OpenCV to detect weapons using that weight. The train was done in google COLAB as it is high end GPU enabled. But the detection with OpenCV was done in a local computer which has moderate CPU and GPU.

3.2 Research Subject and Instrumentation

We tried to find a way to detect weapons from real time CC footage. In this system, no one needs to be seated in front of the display to detect weapons. The machine will do it as an assistance to the human. Also, humans make more mistakes than computers. So, a good machine which understands the problem very well and can perform detection can actually save some times, some life's as well. We tried many things but YOLOv3 worked well so far with our problem set.

We collected data which is pictures of guns or have guns in them. We collected them from YouTube CC camera footage and converted them to pictures frame by frame. We did that using VLC media player [4]. Then we label the images with LabelImg [5] into xml files in PASCAL VOC format. We did that to implement TensorFlow fashion. But to train in YOLOv3 fashion we need yolo format data labelling. We then convert them to yolo format using python script [6]. We installed darknet [7] in Google COLAB [8]. Set .names, .data file as to our need. We also changed the original .cfg file according to our need. We set the train.txt file, which basically holds the relative address of the train images. After setting everything up, we started our training. After training got our .weights file. Then we developed a python script [9] file to detect weapons using that .weights file. The script used OpenCV [10], NumPy and argparse for command line options.

For training purpose, we also used CUDA-version: 10000 (10010), cuDNN: 7.6.5, CUDNN_HALF for GPU accelerated training process in COLAB [8].

3.3 Data Collection Procedure

For any research on AI field data plays a vital role. And in Neural Networking, it has even more significance. More data means great training and lastly great accuracy. So, we focused on creating a decent sized dataset on weapon detection. We first gathered some videos of cc camera footage from YouTube related to weapons. They were as poor in quality as we can see from the figure 3.3.1 below.



Figure 3.3.1: Poor image quality

But to start our work, we started with them. Then we downloaded them from YouTube using online website [11]. They were in .mp4 format but to train our weapon detector we need images in .jpg format. Basically, needed to break down the videos into images. One way to do it is using VLC media player [4].



Figure 3.3.2: VLC Media Player logo

VLC media player serves well as image extractor from videos. VLC has an option named scene video filter. Using that the output name of image files, format and location can be set. Also, one can easily change the recording ratio. It means the frames per second of the recording can be manipulated according to our need. Our videos were basically 20-25 frame per seconds. We wanted to have 3-4 frames per second from that 20-25 frames. So, we set the ratio to 6-7. We preferred the .jpg format and set a location for image recording. All the setting is showed in figure 3.3.3 below.

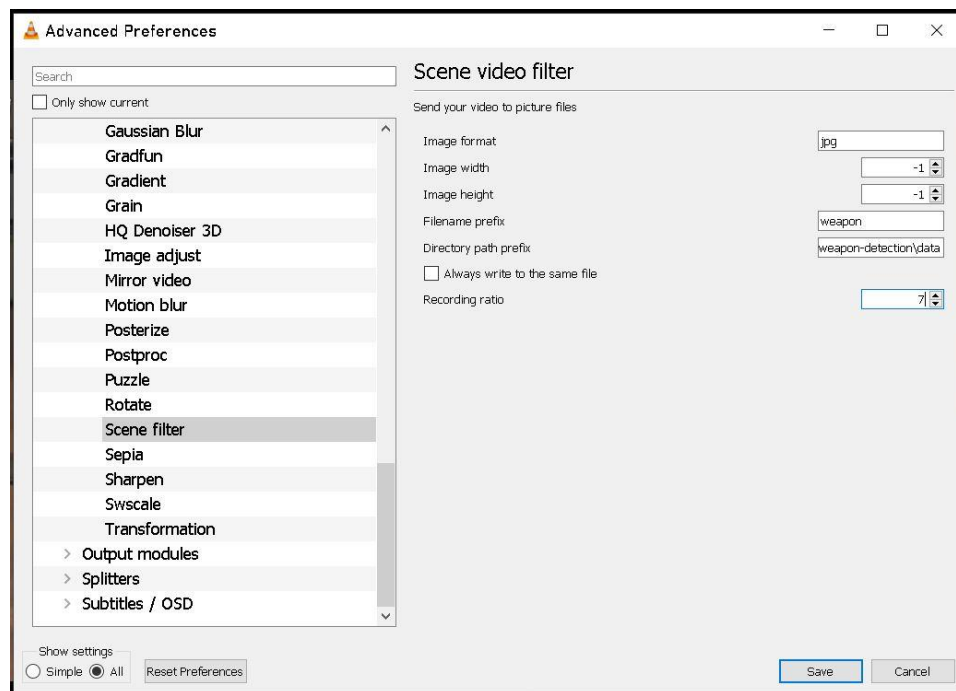


Figure 3.3.3: VLC Media Player recording images settings

After getting all the images it is time for the next work, labelling. To perform Convolutional Neural Networking, we need ground truth box. The labelling is considered as that ground truth box. Labelling includes the box drawing and the labelling the box with class name as well. We used the labelImg [5] opensource project to do this. It is free and simple. It gives two option of format to label images. PASCAL VOC format and YOLO format. As mentioned earlier, we tried to implement the TensorFlow version of object detection at first so we labelled our images in PASCAL VOC format in .xml files as showed in figure 3.3.4.

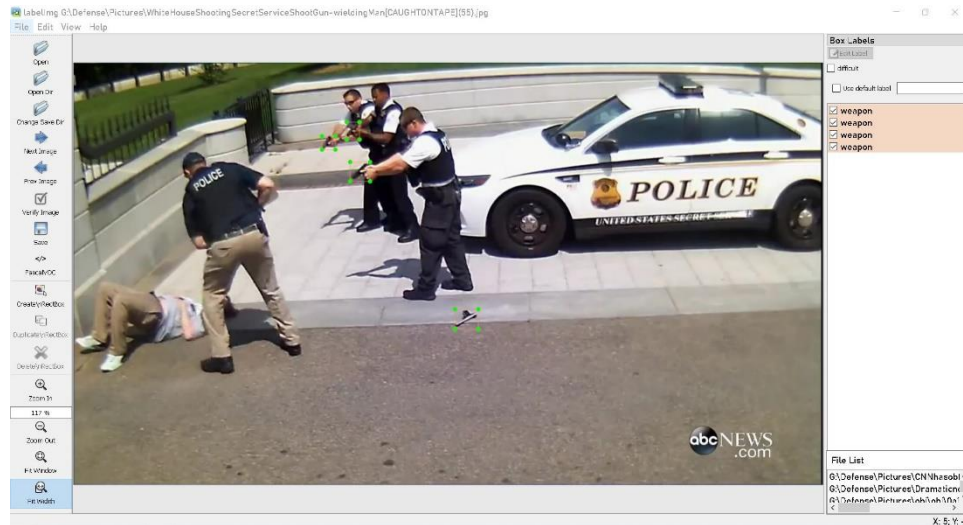


Figure 3.3.4: Labelling with labelImg software

These files hold the information about the boxes, and labels. There are four objects in figure [3.3.4]. So, the .xml file of this image containing information about all four weapons. But later we switched to YOLOv3 then we needed the YOLO format of the PASCAL VOC formal which is in .txt format. We did that with the help of a GitHub repo with changes according our needs [6]. The .txt files contains the object number as well as object location information in the image each on new line. The format is,

<object number> <x> <y> <width> <height>

Object number is type of object in integer format starts at 0. x, y are the center of the object and width and height are the float relative value.

After that we trained our detector for a test run and get not a bad result. One problem was that, the train images we used all are very low-quality small weapon images. As a result, our detector was failing to detect large and interestingly more clear weapons. To resolve this issue, it needed more data especially more clearly and bigger images of weapon for training as shown in figure 3.3.5.



Figure 3.3.5: Image with clear view of weapon

So, we used the Open Images Dataset (OID) V6 [12] to get more images related to weapons. The helping thing here is it is already labelled with slightly different annotations and about ready to feed into our desired model. The OID page is shown below figure 3.3.6.

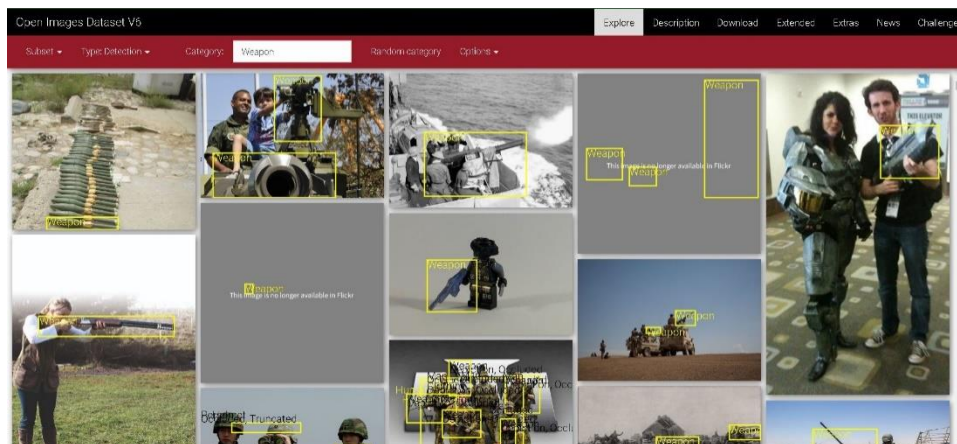


Figure 3.3.6: Open Image Dataset explore option (weapon selected)

One thing left to do with the downloaded dataset is converting the .txt file in YOLO format. It was done with the help of a GitHub repo [13].

Even after that to enlarge our dataset as more data is good, we took images from different movie clips which contains weapons show and videos with clear images of gun. Again, those were annotated with labelImag [5]. After all these steps, our dataset stands at a decent state with decent size with correctly annotation. It has about 4.5k images. It is not a very big dataset. But for our purpose, it serves well.

3.4 Statistical Analysis

We used Convolutional Neural Network to get our job done. The YOLOv3 is a Convolutional Neural Network that serves our purpose well. Our preferred model has 106 layers. 75 of them are convolutional layer and 31 of them are other layers. The model is FCN or Fully Convolutional Network. Means that there is no dense layer like typical CNN or Convolutional Neural Network. The other 31 layers are nor dense layer. They are shortcut, route, upsampling and yolo layers.

The final output of the network is a feature map. A 1x1 convolution layer is used to determine the final output feature map. As it is 1x1 convolution layer, the size of the feature map is exactly as the feature map of previous layer. The result is then interpreted as considering each element in the feature map as a cell in image at last layer. Each cell produces a fixed number of bounding boxes. Each bounding box has certain number of attributes. They are the center coordinates, the dimensions, the objectness score and class confidences. So, each bounding box has 5 (center coordinates, dimensions and objectness score) + C (class confidences) attributes.

Let the size of the input image is 416 x 416 and the stride of the network is 32. Stride of the network act as a factor by which the out image is smaller than input image. So, the output image in this case will be 13 x 13. We can see the bounding boxes in figure 3.4.1.

Image Grid. The Red Grid is responsible for detecting the dog

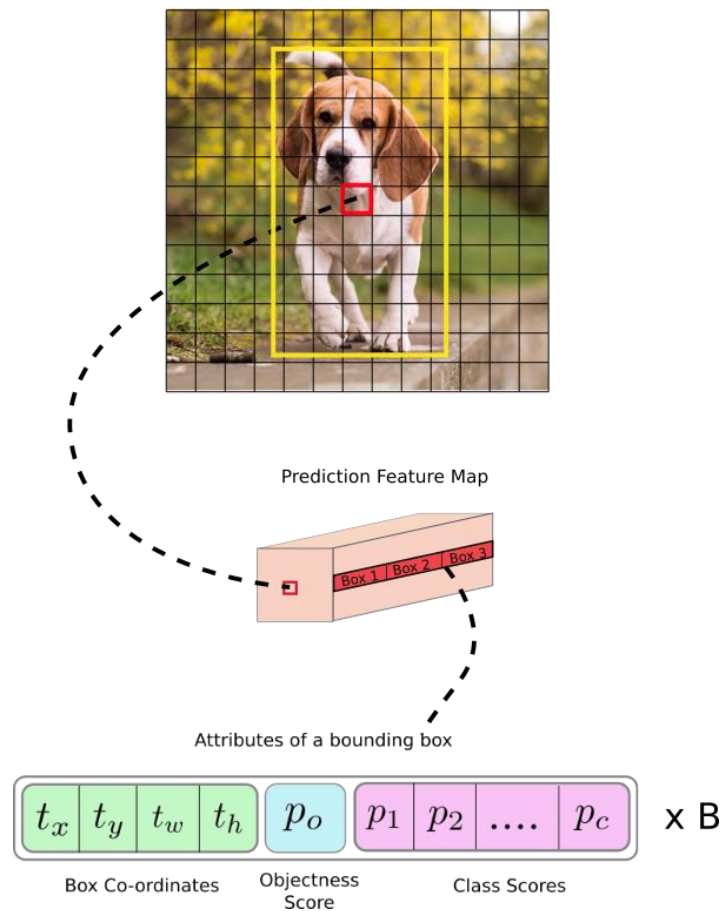


Figure 3.4.1: YOLOv3 bounding box [14]

Here, the t_x , t_y , t_w , t_h are the center and height, width, P_0 is the objectness score and P_1 , P_2 to P_c are the class score of the bounding box. B is the number of the bounding boxes in each cell. Our model has 3 bounding boxes for each cell. Which cell contains the center of the object is responsible for drawing the bounding box for that object. Our Images were all RGB image. We worked with all three channels like in figure 3.4.2.

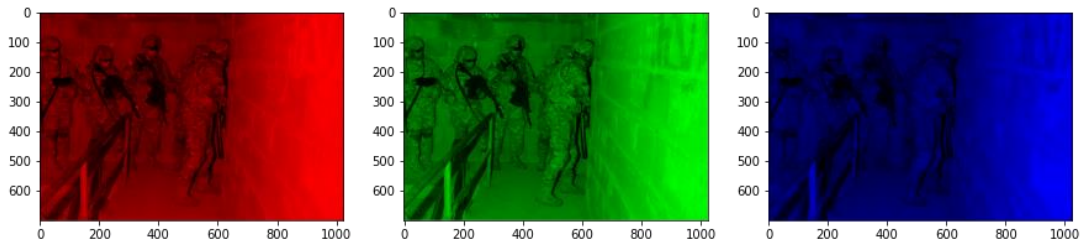


Figure 3.4.2: An image in RGB channels

The network doesn't predict the absolute value for coordinates of bounding box, rather it gives offset value of coordinates. They are relative to the upper left corner of the cell and normalized to be in range 0 to 1. The objectness score is the value between 0 and 1 and means the probability of the object inside that bounding box. Class confidences means the probability of object being a class type. In our model, we have only one class, weapon. So, the model only gives prediction for object being a weapon. It is also normalized between 0 and 1.

The network has another extraordinary thing, the anchors. Anchors are the pre-defined bounding boxes of different sizes. We can determine the bounding box from scratch by convolution but this takes much more resource and time. But using anchors makes it simple. We just need to modify the anchors according to our needs. They are the sizes of the objects (width, height) in images that are converted to the network size (width, height in cfg). Formula showing in figure 3.4.3 is the rules for making bounding box predictions from network output.

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

Figure 3.4.3: Formulas for converting network output to bounding box prediction [15]

Here, b_x , b_y , b_w , b_h are the co-ordinates of center, height and width prediction. t_x , t_y , t_w , t_h is the network outputs. c_x and c_y are the top-left co-ordinates of the cell. p_w and p_h are anchors value of the bounding boxes.

The use of anchors is shown in figure 3.4.4 below,

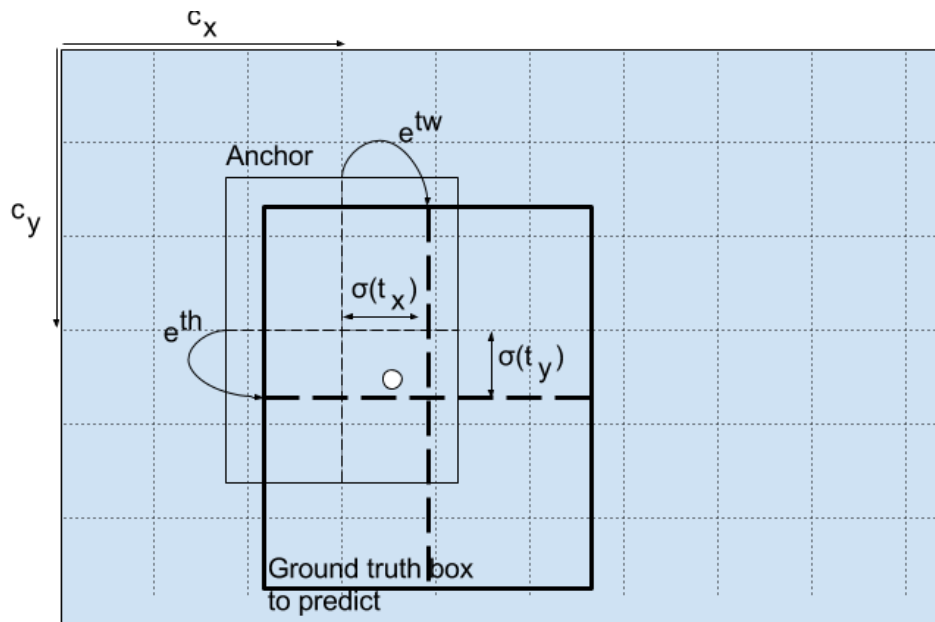


Figure 3.4.4: Anchor concept [15]

There is different level of prediction. It is because to make prediction better in different levels. Initially the network size is 416 x 416. The detection layer use three different sized feature map. They have stride of 32, 16, 8. They produce feature map of 13 x 13, 26 x 26 and 52 x 52. They can detect object even they are small.

Total number of bounding boxes will be $(13 \times 13 + 26 \times 26 + 52 \times 52) \times 3 = 10647$. But We need only those bounding boxes that holds our weapons. Here we used the objectness score. We put a threshold of 0.35 to choose only those boxes have probability greater than the threshold. We faced another problem of getting multiple bounding boxes for same object. We used Non-maximum Suppression to overcome this problem.

The structure of the network is written in a file, .cfg format. It holds all the information and layers details of the network. There are 5 different kinds of layers in our network as we implement using YOLOv3.

Convolutional: There are 75 of them in the network. They are the basic layer of the network. They are different in size, padding, stride and activations.

Shortcut: They are skip connections.

Upsample: Upsample the feature map of previous layer by a fixed stride. We can see that in 3.4.5.

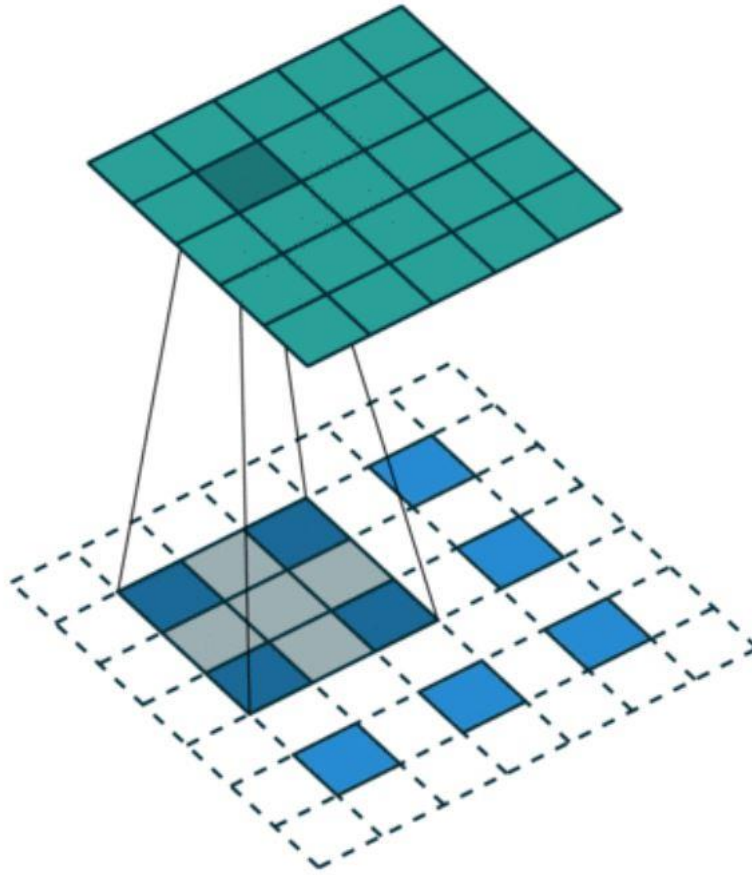


Figure 3.4.5: Upsample with stride of 2 [16]

Route: Route layers basically return the feature maps. They can also concatenate the feature maps of different layers and return one feature map.

YOLO: YOLO layer is considered as the prediction layer of the network. It has mask values, anchors, number of classes etc. Mask values defines the index of anchors to be used.

The other info about the network is written in a block called net. It holds the information about batch size, subdivisions, network size and training parameters like learning rate etc. To start training, we needed data file. We named it obj.data. It contains the info about class number, train-validation.txt, names file and location where the weights will be saved. train-validation.txt is text file containing the location of train and validation images. name file

was named `obj.name` that holds the name of the classes. In our case, we have only one class, weapon. We used a pretrained weight file to initiate our training process. That weight file contains only weights for convolutional layers not for fully connected layers. This weight file is not completed. They just hold weight of prior layers. After setting all the ingredients, we started training. We divided our whole training in 4000 batches. Here, in figure 3.4.6, we can see the few information about training for batch 3905.

```
3905: 0.507463, 0.552476 avg loss, 0.000010 rate, 2.170689 seconds, 249920 images
Loaded: 0.000054 seconds
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.769230, GIOU: 0.757829), Class: 0.996593, Obj: 0.164265, No Obj: 0.001172, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.373812, iou_loss = 0.112150, total_loss = 0.485763)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.712696, GIOU: 0.703550), Class: 0.998675, Obj: 0.853458, No Obj: 0.000222, .SR: 1.000000, .75R: 0.800000, count: 2, class_loss = 0.467586, iou_loss = 0.181485, total_loss = 0.649071)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.692598, GIOU: 0.688470), Class: 0.999336, Obj: 0.390738, No Obj: 0.000144, .SR: 1.000000, .75R: 0.250000, count: 4, class_loss = 0.424990, iou_loss = 0.333407, total_loss = 0.758397)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.765861, GIOU: 0.759234), Class: 0.999719, Obj: 0.809753, No Obj: 0.004048, .SR: 1.000000, .75R: 0.500000, count: 4, class_loss = 0.383907, iou_loss = 0.157885, total_loss = 0.459713)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.765959, GIOU: 0.761538), Class: 0.998925, Obj: 0.841014, No Obj: 0.001536, .SR: 1.000000, .75R: 0.600000, count: 5, class_loss = 0.157876, iou_loss = 0.12451, total_loss = 0.378329)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.554643, GIOU: 0.447500), Class: 0.991832, Obj: 0.130172, No Obj: 0.000062, .SR: 0.500000, .75R: 0.000000, count: 2, class_loss = 0.477374, iou_loss = 0.587675, total_loss = 1.065049)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.754991, GIOU: 0.745789), Class: 0.999080, Obj: 0.257500, No Obj: 0.001939, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.379343, iou_loss = 0.852462, total_loss = 0.431805)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.870986, GIOU: 0.868050), Class: 0.997216, Obj: 0.32134, No Obj: 0.000516, .SR: 1.000000, .75R: 1.000000, count: 2, class_loss = 0.292760, iou_loss = 0.196968, total_loss = 0.312458)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.642716, GIOU: 0.624470), Class: 0.999323, Obj: 0.239357, No Obj: 0.000044, .SR: 1.000000, .75R: 0.000000, count: 2, class_loss = 0.266351, iou_loss = 0.562394, total_loss = 0.562394)
Can't open label file. (This can be normal only if you use MSCOCO): data/obj/PUBG_GunsInRealLife18711.txt
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.813918, GIOU: 0.811449), Class: 0.999043, Obj: 0.661642, No Obj: 0.005418, .SR: 1.000000, .75R: 0.600000, count: 5, class_loss = 0.348674, iou_loss = 0.218289, total_loss = 0.566963)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.815363, GIOU: 0.815363), Class: 0.99481, Obj: 0.117181, No Obj: 0.000394, .SR: 1.000000, .75R: 1.000000, count: 1, class_loss = 0.329448, iou_loss = 0.839206, total_loss = 0.369054)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: -nan, GIOU: -nan), Class: -nan, Obj: -nan, No Obj: 0.000018, .SR: -nan, .75R: -nan, count: 0, class_loss = 0.000380, iou_loss = 0.000000, total_loss = 0.000380)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.482997, GIOU: 0.482997), Class: 0.998543, Obj: 0.820533, No Obj: 0.000056, .SR: 0.000000, .75R: 0.000000, count: 1, class_loss = 0.309051, iou_loss = 0.140400, total_loss = 0.457532)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.722045, GIOU: 0.715813), Class: 0.999755, Obj: 0.409442, No Obj: 0.000658, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.377803, iou_loss = 0.897706, total_loss = 0.475509)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.579160, GIOU: 0.536447), Class: 0.993512, Obj: 0.046353, No Obj: 0.000087, .SR: 0.666667, .75R: 0.000000, count: 3, class_loss = 0.707276, iou_loss = 0.477192, total_loss = 1.184467)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.796134, GIOU: 0.780576), Class: 0.999078, Obj: 0.727329, No Obj: 0.002026, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.075235, iou_loss = 0.855317, total_loss = 0.130552)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.909023, GIOU: 0.906962), Class: 0.999098, Obj: 0.377197, No Obj: 0.000437, .SR: 1.000000, .75R: 1.000000, count: 2, class_loss = 0.220509, iou_loss = 0.008932, total_loss = 0.229440)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: -nan, GIOU: -nan), Class: -nan, Obj: -nan, No Obj: 0.000026, .SR: -nan, .75R: -nan, count: 0, class_loss = 0.001662, iou_loss = 0.000000, total_loss = 0.001662)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.847364, GIOU: 0.841865), Class: 0.999990, Obj: 0.992658, No Obj: 0.001345, .SR: 1.000000, .75R: 1.000000, count: 1, class_loss = 0.021814, iou_loss = 0.824445, total_loss = 0.845459)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.831564, GIOU: 0.825213), Class: 0.999957, Obj: 0.273416, No Obj: 0.000391, .SR: 1.000000, .75R: 1.000000, count: 1, class_loss = 0.209441, iou_loss = 0.841968, total_loss = 0.251418)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.525561, GIOU: 0.466677), Class: 0.998049, Obj: 0.142221, No Obj: 0.000071, .SR: 0.500000, .75R: 0.000000, count: 2, class_loss = 0.428903, iou_loss = 0.284303, total_loss = 0.713206)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.781775, GIOU: 0.769377), Class: 0.998071, Obj: 0.956315, No Obj: 0.004283, .SR: 1.000000, .75R: 0.500000, count: 4, class_loss = 0.069932, iou_loss = 0.282747, total_loss = 0.289679)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.775834, GIOU: 0.764141), Class: 0.999951, Obj: 0.760259, No Obj: 0.000341, .SR: 1.000000, .75R: 1.000000, count: 1, class_loss = 0.029964, iou_loss = 0.826082, total_loss = 0.855965)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.660520, GIOU: 0.627668), Class: 0.996636, Obj: 0.121883, No Obj: 0.000160, .SR: 1.000000, .75R: 0.200000, count: 5, class_loss = 1.075822, iou_loss = 0.461028, total_loss = 1.537643)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.827337, GIOU: 0.822879), Class: 0.990450, Obj: 0.590333, No Obj: 0.003207, .SR: 1.000000, .75R: 1.000000, count: 3, class_loss = 0.240807, iou_loss = 0.130780, total_loss = 0.378867)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.815901, GIOU: 0.817905), Class: 0.999273, Obj: 0.686948, No Obj: 0.000961, .SR: 1.000000, .75R: 0.666667, count: 3, class_loss = 0.079673, iou_loss = 0.152154, total_loss = 0.231837)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.715883, GIOU: 0.708242), Class: 0.999205, Obj: 0.708317, No Obj: 0.000156, .SR: 1.000000, .75R: 0.000000, count: 2, class_loss = 0.201933, iou_loss = 0.158933, total_loss = 0.352065)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.741884, GIOU: 0.727845), Class: 0.998067, Obj: 0.315249, No Obj: 0.001379, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.366556, iou_loss = 0.864783, total_loss = 0.461439)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.799243, GIOU: 0.787341), Class: 0.990411, Obj: 0.281856, No Obj: 0.000729, .SR: 1.000000, .75R: 0.600000, count: 5, class_loss = 0.908900, iou_loss = 0.481074, total_loss = 1.389974)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: -nan, GIOU: -nan), Class: -nan, Obj: -nan, No Obj: 0.000039, .SR: -nan, .75R: -nan, count: 0, class_loss = 0.012881, iou_loss = 0.000000, total_loss = 0.012881)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.805119, GIOU: 0.801794), Class: 0.999549, Obj: 0.657794, No Obj: 0.002174, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.805822, iou_loss = 0.893710, total_loss = 0.124732)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.727900, GIOU: 0.710294), Class: 0.999599, Obj: 0.324792, No Obj: 0.000707, .SR: 1.000000, .75R: 0.250000, count: 4, class_loss = 0.759562, iou_loss = 0.235827, total_loss = 0.985589)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.593088, GIOU: 0.552793), Class: 0.996107, Obj: 0.167746, No Obj: 0.000333, .SR: 0.800000, .75R: 0.000000, count: 10, class_loss = 2.070378, iou_loss = 1.486687, total_loss = 3.557065)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.577641, GIOU: 0.577642), Class: 0.996412, Obj: 0.002121, No Obj: 0.000459, .SR: 1.000000, .75R: 0.000000, count: 1, class_loss = 0.262934, iou_loss = 0.122127, total_loss = 0.385861)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.708336, GIOU: 0.683364), Class: 0.994067, Obj: 0.306355, No Obj: 0.000434, .SR: 1.000000, .75R: 0.333333, count: 3, class_loss = 0.454591, iou_loss = 0.275599, total_loss = 0.760181)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: -nan, GIOU: -nan), Class: -nan, Obj: -nan, No Obj: 0.000008, .SR: -nan, .75R: -nan, count: 0, class_loss = 0.000327, iou_loss = 0.000000, total_loss = 0.000327)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.864086, GIOU: 0.860194), Class: 0.999616, Obj: 0.693627, No Obj: 0.002663, .SR: 1.000000, .75R: 1.000000, count: 3, class_loss = 0.121614, iou_loss = 0.861402, total_loss = 0.183016)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: -nan, GIOU: -nan), Class: -nan, Obj: -nan, No Obj: 0.000035, .SR: -nan, .75R: -nan, count: 0, class_loss = 0.001800, iou_loss = 0.000000, total_loss = 0.001800)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.636771, GIOU: 0.594444), Class: 0.999408, Obj: 0.222456, No Obj: 0.000048, .SR: 1.000000, .75R: 0.000000, count: 1, class_loss = 0.174931, iou_loss = 0.213309, total_loss = 0.388241)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.859549, GIOU: 0.850950), Class: 0.998025, Obj: 0.677048, No Obj: 0.002017, .SR: 1.000000, .75R: 1.000000, count: 2, class_loss = 0.863718, iou_loss = 0.821518, total_loss = 0.885236)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.721128, GIOU: 0.695976), Class: 0.987913, Obj: 0.451775, No Obj: 0.000564, .SR: 1.000000, .75R: 0.750000, count: 4, class_loss = 0.710495, iou_loss = 0.297204, total_loss = 1.007699)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.383388, GIOU: 0.071379), Class: 0.999793, Obj: 0.083495, No Obj: 0.000043, .SR: 0.000000, .75R: 0.000000, count: 1, class_loss = 0.281877, iou_loss = 0.758447, total_loss = 1.039524)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.728273, GIOU: 0.728273), Class: 0.999395, Obj: 0.671669, No Obj: 0.000900, .SR: 1.000000, .75R: 0.000000, count: 1, class_loss = 0.845651, iou_loss = 0.832895, total_loss = 0.878547)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.623708, GIOU: 0.623750), Class: 0.998996, Obj: 0.322679, No Obj: 0.000587, .SR: 1.000000, .75R: 0.000000, count: 3, class_loss = 0.427064, iou_loss = 0.372126, total_loss = 0.799189)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.467188, GIOU: 0.434485), Class: 0.993559, Obj: 0.238598, No Obj: 0.000093, .SR: 0.500000, .75R: 0.000000, count: 2, class_loss = 0.358799, iou_loss = 0.667055, total_loss = 1.025054)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.763731, GIOU: 0.746247), Class: 0.999617, Obj: 0.516525, No Obj: 0.002077, .SR: 1.000000, .75R: 0.500000, count: 2, class_loss = 0.216754, iou_loss = 0.892405, total_loss = 0.309159)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.778105, GIOU: 0.774776), Class: 0.997195, Obj: 0.226513, No Obj: 0.000417, .SR: 1.000000, .75R: 0.333333, count: 3, class_loss = 0.570406, iou_loss = 0.150340, total_loss = 0.720745)
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.489489, GIOU: 0.464577), Class: 0.995912, Obj: 0.000078, No Obj: 0.000014, .SR: 0.000000, .75R: 0.000000, count: 1, class_loss = 0.259332, iou_loss = 0.175080, total_loss = 0.425940)
```

Figure 3.4.6: Training log for batch number 3905

We can see the output of IOU, GIOU, class probability, recall at 50%, 75%, class loss, IOU loss and total loss in each step. Total loss has an overall tend to decrease as the model is training. We expect to have equal or loss less than 0.05 as we can see in figure 3.4.7.

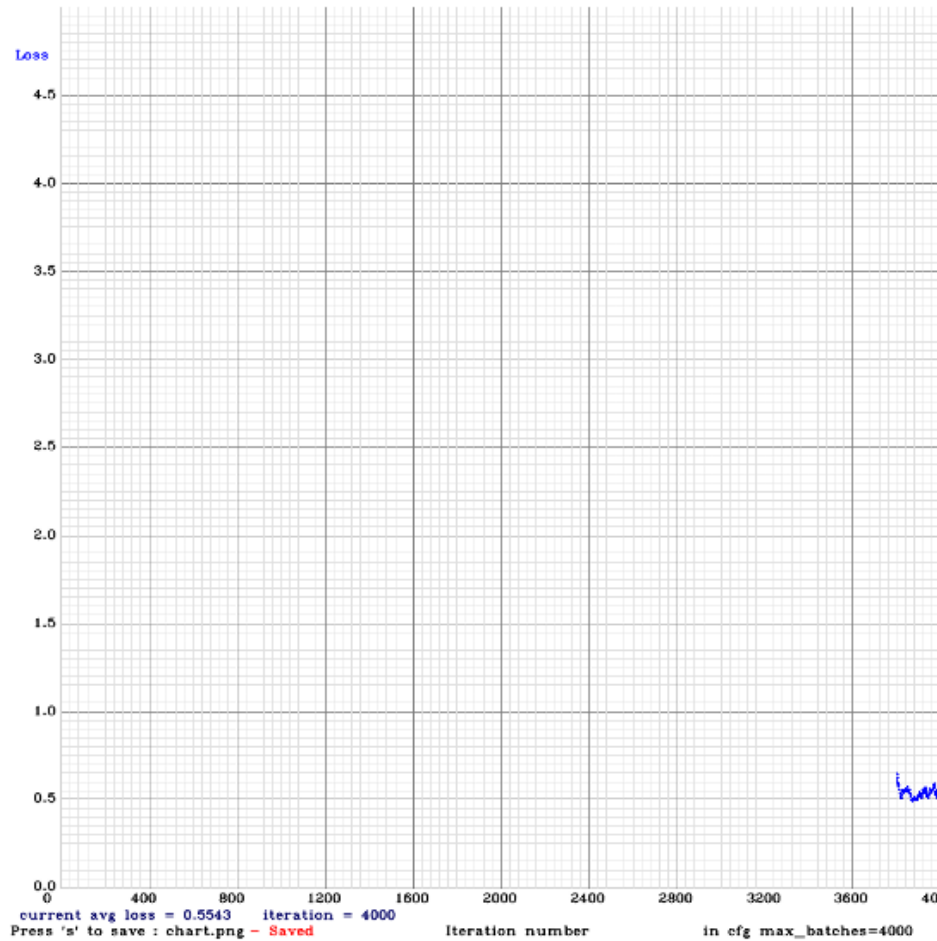


Figure 3.4.7: Training log for loss

It shows loss after 4000 iteration. In early iteration it was extremely high. But with each iteration, it was decreased. And at our last iteration number 4000 it decreased to our expected value. After iteration 4000, average loss was 0.5543 and it is quite good. As we have only one class, we needed less iterations.

3.5 Implementation Requirements

By following our project website [17] one can implement our process. In the website all the necessary files have been uploaded.

One need to go to GitHub repository [18] for this. This repository contains all the files and links to implement this project.

There is a link of config file. It is already modified for single class weapon detection. It can be used to re-produce the weight file. Or, one can use the fully trained weight file on weapon detection and detection script to detect weapons.

To do this, one need to download the python script file for detection. Get the dataset, the name and data file from the link. To implement, one also need the train.txt file. After gathering all this files, one needs to install the dependencies of the python script. After that provide all the file path to appropriate location in script. One need a sample image or video to detect in image or video. Our model also supports the command line options.

CHAPTER 4

Experimental Results and Discussion

4.1 Introduction

Model is working or not can be told by doing experiments on that. Basically, by the results of the experiments. We tried TensorFlow model to get our job done, but it didn't give us good result. It was quite slow in speed because of pipeline it uses to detect weapons. Average FPS we got using TensorFlow faster R-CNN is about 4-7 FPS. But using our latest model, we got about 14-20 FPS on a moderate GPU enabled computer. For detecting weapons in real time, this speed is good.

4.2 Experimental Results

We trained our weapon detection model based on YOLOv3. After collecting data and labelling it we fed it into our network and ultimately got final weight files. These weight files are used later to detect weapons in frames of videos in real time. Videos are nothing but frames or images on sequence. We can detect on images and if we do it iteratively then it is similar to video. If we feed a video stream to our model, then the video stream is broken into frames and perform detection on each individual frame. We are here showing 3 experimental results of our work.

First experimental result is shown below,

The input of first experimental is shown in figure 4.2.1.



Figure 4.2.1: Input for experiment 1

Output for experiment 1 is shown in figure 4.2.2.



Figure 4.2.2: Result of experiment 1

We see that in input, there are actually 4 weapons. But our model could detect 3 of them. It is because our model needs more data to detect all kind of objects. But for initial purpose, it has done a good job.

We can see the log file for experiment no. 1 below. Here is log data of first 20 layers.

CUDA-version: 10010 (10010), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1 OpenCV version: 3.2.0 compute_capability = 610, cudnn_half = 0
 net.optimized_memory = 0 mini_batch = 1, batch = 16, time_steps = 1, train = 0

layer	filters	size/strd(dil)	input	output
0 conv	32	3 x 3/ 1	416 x 416 x 3 ->	416 x 416 x 32 0.299 BF
1 conv	64	3 x 3/ 2	416 x 416 x 32 ->	208 x 208 x 64 1.595 BF
2 conv	32	1 x 1/ 1	208 x 208 x 64 ->	208 x 208 x 32 0.177 BF
3 conv	64	3 x 3/ 1	208 x 208 x 32 ->	208 x 208 x 64 1.595 BF
4 Shortcut Layer: 1, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.003 BF				
5 conv	128	3 x 3/ 2	208 x 208 x 64 ->	104 x 104 x 128 1.595 BF
6 conv	64	1 x 1/ 1	104 x 104 x 128 ->	104 x 104 x 64 0.177 BF
7 conv	128	3 x 3/ 1	104 x 104 x 64 ->	104 x 104 x 128 1.595 BF
8 Shortcut Layer: 5, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF				
9 conv	64	1 x 1/ 1	104 x 104 x 128 ->	104 x 104 x 64 0.177 BF
10 conv	128	3 x 3/ 1	104 x 104 x 64 ->	104 x 104 x 128 1.595 BF
11 Shortcut Layer: 8, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF				
12 conv	256	3 x 3/ 2	104 x 104 x 128 ->	52 x 52 x 256 1.595 BF
13 conv	128	1 x 1/ 1	52 x 52 x 256 ->	52 x 52 x 128 0.177 BF
14 conv	256	3 x 3/ 1	52 x 52 x 128 ->	52 x 52 x 256 1.595 BF
15 Shortcut Layer: 12, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF				
16 conv	128	1 x 1/ 1	52 x 52 x 256 ->	52 x 52 x 128 0.177 BF
17 conv	256	3 x 3/ 1	52 x 52 x 128 ->	52 x 52 x 256 1.595 BF
18 Shortcut Layer: 15, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF				
19 conv	128	1 x 1/ 1	52 x 52 x 256 ->	52 x 52 x 128 0.177 BF

[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00

Total BFLOPS 65.304

Allocate additional workspace_size = 52.43 MB

Loading weights from yolov3_obj_final.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done!

Loaded 107 layers from weights-file

©Daffodil International University

Enter Image Path: /content/mydrive/yolov3/weapon8.jpg

/content/mydrive/yolov3/weapon8.jpg: Predicted in 32.492000 milli-seconds.

Weapon: 49%

Weapon: 47%

Weapon: 74% BF

Here, we can see that the BFLOPS is 65.304, means our model calculate floating operation at that speed. It is the measurement of Billions Floating point Operation Per Second. Our model detected weapon in about 33 milli-seconds and all the detected weapons accuracy is also shown. Let's do another experiment, experiment no. 2.

Input image for experiment 2 is in figure 4.2.3.

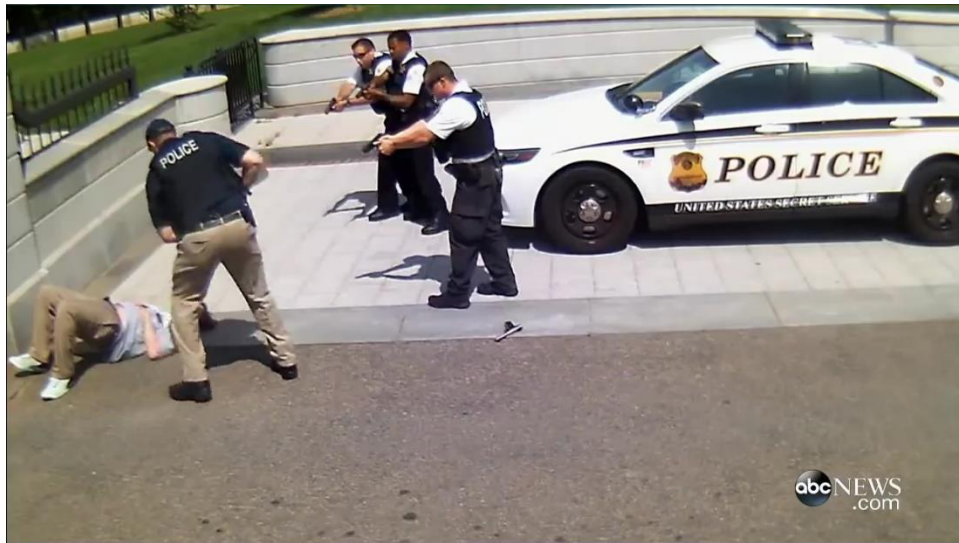


Figure 4.2.3: Input of experiment 2

We feed the image in figure 4.2.3 to our model and get figure 4.2.4 as result after detection.



Figure [4.2.4]: Result of experiment 2

For third experiment, figure 4.2.5 and figure 4.2.6 showing input and result respectively,



Figure 4.2.5: Input of experiment 3



Figure 4.2.6: Result of experiment 3

In experiment 3, we feed some normal weapons photo and get some good predictions. In this 3-experiment which input weapon image our model missed are for lack of proper data relating to that type weapon.

4.3. Descriptive Analysis

From the figures in section 4.2, we can see that the model is doing a decent job. We had mAP (mean Average Precision) of about 85%. This number works well for our purpose, at least for initial state. mAP can be increased by putting more data, specially variety in data. Means, there need more data from different shapes of weapon from many angles. It will be helpful if part of a weapon image is also part of data, to detect weapon with more abstract data. We can see that our model can handle most of the weapons in images we provided for experimenting. And the accuracies of detected weapons are pretty good. It is doing well in both 50% and 75% recall.

4.4 Summary

After doing bunch of experiments, it is clear that our trained model can handle its job. It can detect weapons from frames of images. It is also fast with decent accuracy and precision. The main purpose of this model is to detect the weapons in live video stream. That is done by script that convert the video to frames. The trained weight holds the value for different layers of model in floats. By reading these floats, the model predicts the weapon location and draws the bounding box. And by doing these intermediate works, it gets its job done.

CHAPTER 5

Summary, Conclusion, Recommendation and Implication for Future Research

5.1 Summary of the Study

Our goal was to implement a method to detect weapons from CC camera footage in real-time. We chose YOLOv3 as our main framework because it is fast and efficient, state of the art. We collected data (images with weapons) from various videos and OID (Open Image Dataset). We labeled them according to their weapons size. Then we put all the data to pipeline for training. We trained our model for enough time to reach down total loss under 0.5, more or less. After training we got our weight file that contains values for parameter to detect weapons in images. We developed a python script [18] that allows one to detect weapons in frames. It is also command line option enabled.

5.2 Conclusions

We selected this work from a realization. A realization that concerns security of our public places. We have most of the important places under CC camera coverage. We tried to do a favor for this sector. We developed a system that can detect weapons from videos in real time. We were able to detect weapons in a frame in about 35 milli-seconds. On an average we got 15-20 fps. This level of speed of detection is good for detecting weapons in real time.

We believe use of this model can save lot of human time from cumbersome and boring work of monitoring CC camera footage for detecting weapons in footage.

5.3 Recommendations

We recommend to use a good setup which contain good amount of RAM and GPU to implement our work. The more the CPU and RAM, the better means the faster the detection. To detect in real time, it requires a heavy amount of resources. Unless it is not possible to get expected result.

If one wants to train furthermore with more images, we recommend to use versatile weapons data. The benefit of this is the good detection. It is good to detect wrong object as weapons rather than not detecting real weapons.

Another important thing is labeling. It is highly recommended to label all the images correctly. The labeling precision plays a vital role here. The ground truth box should include all the part of weapons but not more than that. This would increase the rate of correct detections.

5.4 Implication for Further Study

We got that convolutional neural network is a good way to detect weapons in images. We implemented the YOLOv3 version for detecting weapons. There is an improve version of that YOLO, YOLOv4 out there. It has more precision. The mAP of YOLOv4 is higher than YOLOv3.

It is well proved that the accuracy depends on fine data. We had few kinds of weapons images. To get even better result, more clear data is required. More data is always good for convolutional neural network.

REFERENCES

- [1] Skyler Alsever, Georgios Karapanagos, Carlos Monterrosa Diaz, Glen Mould, Paul Raynes, “AWeD: Automatic Weapons Detection”, Worcester Polytechnic Institute, April 2017.
- [2] Rodrigo Fumihiro de Azevedo Kanehisa, Areolino de Almeida Neto, “Firearm Detection using Convolutional Neural Networks”, Proceedings of the 11th International Conference on Agents and Artificial Intelligence, Czech Republic, vol. 2, pp. 707–714, January 2019.
- [3] Gyanendra K Verma, Anamika Dhillon, “A Handheld Gun Detection using Faster R-CNN Deep Learning”, Proceedings of the 7th International Conference on Computer and Communication Technology, pp. 84–88, November 2017.
- [4] Videolan.org. 2020. Official Download Of VLC Media Player, The Best Open Source Player - Videolan. [online] Available at: <<https://www.videolan.org/vlc/index.html>> [Accessed 13 May 2020].
- [5] GitHub. 2020. Tzutalin/Labelimg. [online] Available at: <<https://github.com/tzutalin/labelImg>> [Accessed 26 April 2020].
- [6] Gist. 2020. Convert Pascal Voc Dataset To Yolo Format. [online] Available at: <<https://gist.github.com/M-Younus/ceaf66e11a9c0f555b66a75d5b557465>> [Accessed 05 May 2020].
- [7] GitHub. 2020. Alexeyab/Darknet. [online] Available at: <<https://github.com/AlexeyAB/darknet>> [Accessed 08 May 2020].
- [8] Colab.research.google.com. 2020. Google Colaboratory. [online] Available at: <<https://colab.research.google.com/>> [Accessed 13 May 2020].
- [9] GitHub. 2020. Jubaer-Ad/Weapon-Detection-Real-Time. [online] Available at: <<https://github.com/jubaer-ad/weapon-detection-real-time/blob/master/opencv-detection.py>> [Accessed 13 May 2020].
- [10] Partnership, H., 2020. Opencv. [online] Opencv.org. Available at: <<https://opencv.org/>> [Accessed 10 May 2020].

[11] Y2mate.com. 2020. Youtube Converter - Convert Youtube Videos To MP3, MP4 And More. [online] Available at: <<https://www.y2mate.com/en11/convert-youtube>> [Accessed 13 May 2020].

[12] storage.googleapis.com. Open Images Dataset. Available at: <<http://storage.googleapis.com/openimages/web>> [Accessed 26 April 2020]

[13] theAIGuysCode, “theAIGuysCode/OIDv4_ToolKit,” GitHub. [Online]. Available: https://github.com/theAIGuysCode/OIDv4_ToolKit/blob/master/convert_annotations.py. [Accessed: 14-May-2020].

[14] “How to Implement a YOLO (v3) Object Detector from Scratch in PyTorch: Part 1,” KDnuggets. [Online]. Available: <https://www.kdnuggets.com/2018/05/implement-yolo-v3-object-detector-pytorch-part-1.html>. [Accessed: 14-May-2020].

[15] Redmon, Joseph and Farhadi, Ali, “YOLOv3: An Incremental Improvement”, arXiv: 1804.02767, 2018

[16] M. T. M. Thoma, “What are deconvolutional layers?” Data Science Stack Exchange, 01-Feb-1965. [Online]. Available: <https://datascience.stackexchange.com/questions/6107/what-are-deconvolutional-layers>. [Accessed: 14-May-2020].

[17] A. N. M. Jubaer, A. Sayem, and Md. Ashikur RAHMAN, “mlproject,” mlproject. [Online]. Available: <https://sites.google.com/diu.edu.bd/believersmlproject>. [Accessed: 14-May-2020].

[18] Jubaer-Ad, “jubaer-ad/weapon-detection-real-time,” GitHub, 02-May-2020. [Online]. Available: <https://github.com/jubaer-ad/weapon-detection-real-time>. [Accessed: 14-May-2020].

Report-Final.docx

ORIGINALITY REPORT

23%	17%	6%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	8%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
3	github.com Internet Source	2%
4	Gyanendra K. Verma, Anamika Dhillon. "A Handheld Gun Detection using Faster R-CNN Deep Learning", Proceedings of the 7th International Conference on Computer and Communication Technology - ICCCT-2017, 2017 Publication	2%
5	blog.paperspace.com Internet Source	1%
6	www.scitepress.org Internet Source	1%
7	Submitted to University of Colombo Student Paper	1%

8	Submitted to Curtin University of Technology Student Paper	1%
9	Submitted to University of Bradford Student Paper	<1%
10	koreascience.or.kr Internet Source	<1%
11	www.ukessays.com Internet Source	<1%
12	Submitted to SAE Institute, London Student Paper	<1%
13	"Pattern Recognition and Computer Vision", Springer Science and Business Media LLC, 2019 Publication	<1%
14	Submitted to University of Surrey Student Paper	<1%
15	"Intelligent Traffic Management System", International Journal of Recent Technology and Engineering, 2019 Publication	<1%
16	Submitted to University of Queensland Student Paper	<1%
17	Submitted to Marymount College Student Paper	<1%

18	Submitted to De Montfort University Student Paper	<1%
19	"Contents", 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), 2019 Publication	<1%
20	M. Milagro Fernandez-Carrobles, Oscar Deniz, Fernando Maroto. "Chapter 38 Gun and Knife Detection Based on Faster R-CNN for Video Surveillance", Springer Science and Business Media LLC, 2019 Publication	<1%
21	Submitted to University College London Student Paper	<1%
22	Submitted to Kensington College of Business Student Paper	<1%
23	Submitted to Monash University Student Paper	<1%
24	hdl.handle.net Internet Source	<1%
25	Submitted to University of Southern California Student Paper	<1%
26	Submitted to University of Hertfordshire Student Paper	<1%

27	openjicareport.jica.go.jp Internet Source	<1%
28	resources.fiorano.com Internet Source	<1%
29	www.mdpi.com Internet Source	<1%
30	othes.univie.ac.at Internet Source	<1%
31	scholar.sun.ac.za Internet Source	<1%
32	Submitted to Babes-Bolyai University Student Paper	<1%

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words