Border Security System Using Real-Time Image Processing

BY

Abdul Halim
ID: 161-15-980

Mahady Hasan
ID: 161-15-954

AND

Sohel Rana
ID: 161-15-862

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

Supervised By

Md. Reduanul Haque
Senior Lecturer
Department of CSE
Daffodil International University

Co-Supervised By

Tajim Md. Niamat Ullah Akhund
Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

DECEMBER 2020

APPROVAL

This Project titled "**Border Security System Using Real-Time Image Processing**", submitted by **Abdul Halim**, **Mahady Hasan** and **Sohel Rana** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfilment 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 5/12/2020.

BOARD OF EXAMINERS

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Md. Reduanul Haque, Senior Lecturer, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

`

Md. Reduanul Haque
Senior Lecturer
Department of CSE
Daffodil International University

Co-Supervised by:

`

Tajim Md. Niamat Ullah Akhund
Lecturer
Department of CSE
Daffodil International University

Submitted by:

`

Abdul Halm
ID: -161-15-980
Department of CSE
Daffodil International University

`

Mahady Hasan
ID: -161-15-954
Department of CSE
Daffodil International University

Sohel Rana
ID: -161-15-862
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First, we express our deepest thanks and gratefulness to Almighty Allah for His divine blessing that makes us possible to complete the final year project successfully. We are grateful and wish our profound indebtedness to **Md. Reduanul Haque**, **Senior Lecturer**, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "*Object Detection*" 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 stages have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr Syed Akhter Hossain, Professor and Head, Department of CSE, and Dr S. M. Aminul Haque, Associate Professor and Associate Head Department of CSE** for their kind help to finish our project and also to other faculty members and the staff of CSE department of Daffodil International University.

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

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

ABSTRACT

In recent times, it has occurred to us that county borders are now more deadly with security flaws. There are millions of stuff being smuggled in and out, neighbour countries security forces having many ways to advance with gunning down people and entering the border area, seizing people and extortion. That's why it’s our effort to detect forces with a faster image processing method so the security personnel can use it to their benefit. Object detection in modern computer science has developed quite a lot in recent years. With the development of neural network algorithms, some notable of them are CNN, RCNN, and faster-RCNN algorithms. Our motive was to develop a real-time object detection system which can be used at the country border to help detect targets in order to increase security. That's why we used the YOLO (you only look once) algorithm to train and test out data. The YOLO algorithm takes a different approach in order to detect objects faster. In this research project, we trained raw data in YOLO and SSD (Single shot multibox detector) and compared their advantages and disadvantages for having a real-time level of detection and accuracy. This detection scheme can be applied in surveillance systems such as cameras, drones and video surveillance, which will require cloud and server-based processing in object detection Application Programming Interface. we’re hopeful that This research project could be one of the early steps to increase border area security.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES	**PAGE NO**
Figure 2.1: Convolutional layers	12
Figure 2.2: Regions with CNN Features	13
Figure 2.3:Convolutional layer to SVM	13
Figure 2.4: YOLO bounding boxes	14
Figure 2.5: SSD detection layers	15
Figure 3.1: Anchor boxes	18
Figure 3.2: SSD feature layers	19
Figure 3.3: Pyramid detection layers	20

CHAPTER 1

Introduction

Application of computer technology in the modern-day security system is being applied in a large number of fields. It is our effort to find out which way we have to apply in border security. The whole process has been done in the field of object detection applying machine learning processes and algorithms.

1.1 Motivation

There have been many object detection methods developed over the decade. So being properly aware of that we looked for a method which can produce real-time detection systems in the border area with a modest amount of frame rate and accuracy. The Motivation Behind this work is by observing the security issues in Border areas that need to be equipped with better models to work with. That's why it is our effort to test better models being one of the first steps to progress the surveillance systems in those areas

1.2 Objectives

The core objective of this project is to find out the ways to Implement popular object detection methods, evaluate them, compare them and find out their prediction accuracy. That includes manually crafted training datasets and training them in both You Only Look Once (YOLOv4) and Single Shot Multibox Detector (SSD) algorithm based models. Also, Monitoring Frame rates, detection numbers and Min-max confidence falls under objectives

1.3 Expected Outcome

The outcome of this whole project is to be a facilitating factor in any Country border Security especially in a conflicting area or simply in surveillance in general. we expect to have at least 30 frames reach to have real-time detection. We also take at least 70% of classification accuracy and 60% accuracy in confusion matrix calculation.

CHAPTER 2

Literature Review

This has been at least two decades where object detection algorithms are apparent in applications. Our purpose is to use neural network algorithms for object detection in a certain way which is faster and usable in real-time. The trick is to use neural networks in a forward propagation pass through neural networks to detect objects faster performing instant bounding box application. This whole process is done in a faster-RCNN based algorithm called YOLOv4 and modified TensorFlow API based SSD models.

2.1 Related Works

There are three main object detection methods that you can find when it comes to deep learning-based object detection, CNN’s are one of the first deep learning-based object detectors and are an example of a two-stage detector. The Standard R-CNN Method typically brings a slow result and becomes problematic when it comes to end-to-end object detection. That’s when it comes to YOLO and SSD to help increase speed which is renowned for detecting objects in one stage. In Bochkovsky et al, deliver a quicker, state-of-the-art detector (FPS) and more accurate (MS COCO AP50...95 and AP50) than all the alternative detectors available.

2.2 Object detection

Deep learning-based object detection has developed in recent years facing to solve the problems of all the stages of generic object detection which can be identified as Informative region selection, feature extraction and classification

2.2.1 Informative region selection

The problem with informative region selection is it applies an exhaustive amount of sliding windows because of a large number of candidate windows which results in too many redundant windows. but if only a few numbers of sling windows applied it can result in unnecessary regions

2.2.2 Feature Extraction

It is required to extract visual features from an image to provide a visual representation. SIFT, HOG, and Haar-like features can be taken as an example. These features can work with convolutional neural networks and help detect a different class of objects in diverse groups. but it is difficult to make a sturdy feature extractor which can perfectly describe all kinds of objects

2.2.3 Classification

A classifier is required to distinguish between a target object and other categories. It is also required to make the depiction of objects hierarchical, semantic, and more revealing. There are some popular choices of classifiers in object detections which are AdaBoost, Support Vector Machine (SVM), and Deformable part-based model (DPM). DPM offers a graphical model, good low-level designed features and for a variety of classes in objects a high precision part-based model.

2.3 Convolutional neural network

The convolutional neural network requires multiple layers to function firstly as an input layer and a minimum of one hidden layer and an output layer. the layers perform detections such as shapes, edges and colours. The hidden convoluted layers act as filters that transform the input in certain patterns which often are referred to as pooling layers that work in different layers to detect a certain object. The more the layer increases it lessens the pooling size leading towards the correct detection output.

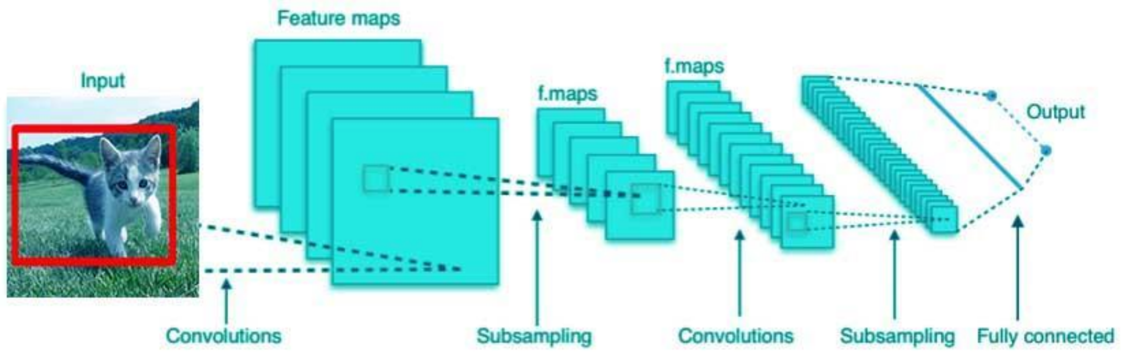


Figure 2.1: Convolutional layers

2.3.1 Regional Convolutional neural network

R-CNN models first select several proposed regions from an image (anchor boxes can be used for such selection procedure) and then label their categories and bounding boxes. Then, CNN is used to perform forward computation that extracts features from every proposed area. Then we use the features computed from each proposed region to predict their categories and bounding boxes.

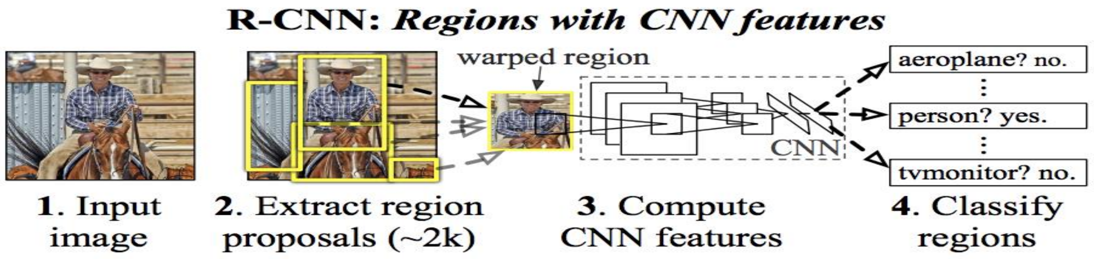


Figure 2.2: Regions with CNN Features

What Regional Neural Network does differently from traditional Neural Network is it creates a certain amount of Bounding boxes based on a limited number of proposed regions (about 2000) they become generated using selective search algorithms. Convolutional neural networks act as a feature extractor and generate convolution layers which are later fed into the Support Vector Machine (SVM) to classify the object to the target region proposal. Then It proceeds into creating bounding boxes and makes some necessary adjustments.

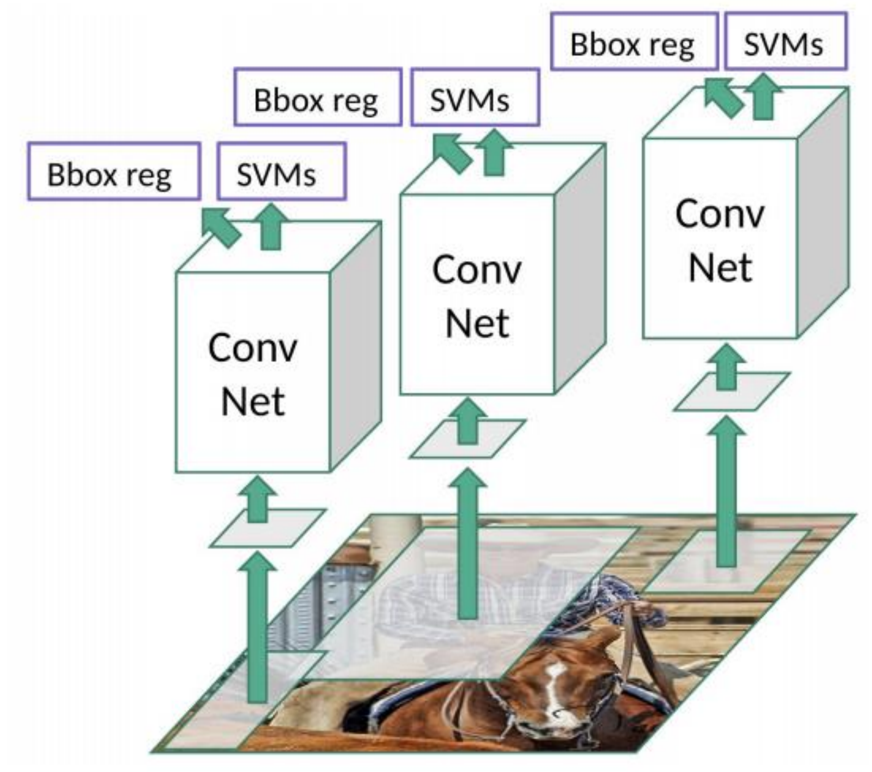


Figure 2.3:Convolutional layer to SVM

2.3.2 Fast Regional Convolutional Neural Network

One major bottleneck R-CNN exhibits that the model needs to independently extract features for each proposed area. If a high degree of overlap is present among the regions, which is most often the case, repetitive computations are performed in a large number in the process of independent feature extraction. Fast R-CNN improves the bottleneck, by only performing CNN forward computation on the image as a whole.

2.3.3 Faster Regional Convolutional Neural Network

To get accurate object detection results, Fast R-CNN generally requires that many proposed regions work in detective search. Faster R-CNN uses a region proposal network to replace the selective search procedure. This reduces the number of proposed region generation while ensuring precise object detection.

2.3.4 You Only Look Once (YOLO)

YOLO is an object detection algorithm where a single coevolutionary network predicts bounding boxes and class probabilities for these boxes. It's working in the following way. It takes an image and splits it into an SxS grid, inside each grid It takes m bounding boxes. For each bounding box, the network outputs the probability class and the bounding box offset values. After having the class likelihood above the threshold value, the selected bounding boxes are selected and used to locate the object within the image.

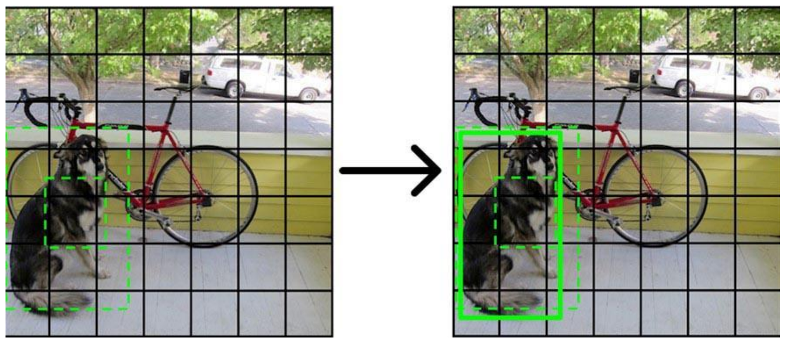


Figure 2.4: YOLO bounding boxes

2.3.5 Single Shot Multibox Detector

This is a system for police investigation and classifying objects in pictures or videos employing a single deep neural network. It's known as SSD as a result of its discreet output area of bounding boxes in an exceeding assortment of default boxes over completely different facet ratios and scales per operate map position. At the time of prediction, the network generates scores for the inclusion of every kind of object in each default box and creates changes to the box to best match the form of the thing. Additionally, the network incorporates projections from many feature maps with completely different resolutions to naturally manage artefacts of various sizes. SSD is just relative to strategies that need object proposals as a result of it removes proposal generation and future component or operate re-sampling stages and encapsulates all computation in an exceedingly single network. This makes SSD easy to train and easy to incorporate into systems that require the detection component. Experimental findings on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy in methods that use an additional object proposal stage and is much quicker by offering a single context for both training and inference. For 300 × 300 inputs, SSD achieved 74.3 per cent of mAP1 on the VOC2007 test at 59 FPS on the Nvidia Titan X and for 512 × 512 inputs, SSD achieved 76.9 per cent of mAP, outperforming a comparable state-of-the-art Faster R-CNN model. Compared to other single-stage approaches, SSD is much more accurate even with smaller input image sizes.

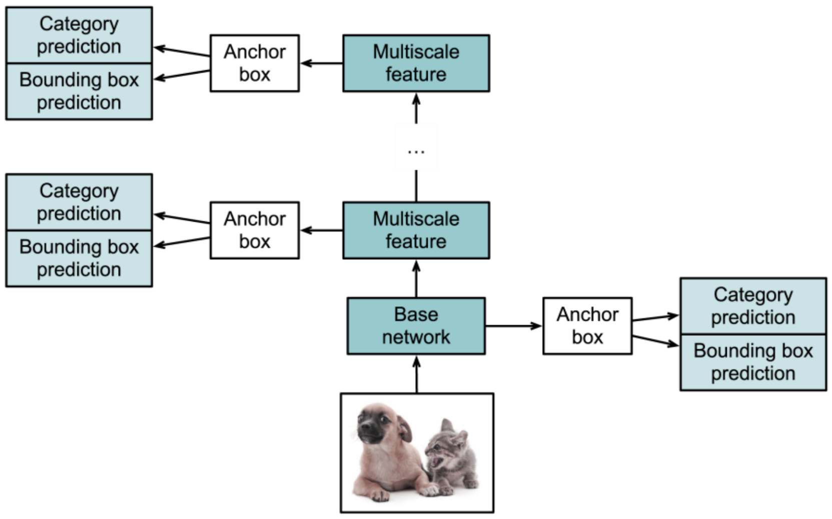


Figure 2.5: SSD detection layers

CHAPTER 3

Problems

Since we are venturing into the Non-Deterministic algorithm region, some key issues inherently follow any optimizer we build. Here we mention some of the key issues

3.1 Object classification and Localization

Its target is the first complication of object detection. Not only do we want to identify image objects, but we also want to decide the location of objects, commonly referred to as object localization tasks. To fix this problem, researchers most frequently use a multi-task loss feature to penalize both misclassification and localization errors. Regional-based CNNs are a common class of object detection frameworks. These approaches consist of creating regional proposals where objects are likely to be found, accompanied by CNN processing to identify and further refine object locations. Ross Girshick et al. developed Quick R-CNN to improve their initial findings with R-CNN. As its name suggests, Fast R-CNN offers dramatic speed-up, but accuracy also improves because the classification and position tasks are configured using a single unified multi-task loss feature. Each candidate region that may contain an object is compared to the actual objects of the image. Candidate regions would then incur penalties for both incorrect classifications and misalignment of bounding boxes. As a consequence, the loss function consists of two types of terms: loss function with one term for classification and one for a location where the classification term imposes log loss on the expected likelihood of the true object class u and the localization term is a smooth L1 loss for the four positional components that characterize the rectangle. Note that the localization penalty does not extend to the context class when there is no object present, u=0. Notice also that the parameter 5-007 can be changed to give priority to either classification or localisation.

$$\mathcal{L}(p,u,t^u,v) = \overbrace{\mathcal{L}_c(p,u)}^{classification} + \lambda \overbrace{[u \geq 1]\,\mathcal{L}_l(t^u,v)}^{localization},$$

3.2 Real-time detection

Object detection algorithms need not only to precisely identify and locate important objects, they also need to be extremely fast at the time of prediction to satisfy the real-time demands of video processing. Several key improvements over the years have improved the performance of these algorithms, increasing the test time from 0.02 frames per second (fps) of R-CNN to an impressive 155 fps of Fast YOLO. Fast R-CNN and Faster R-CNN seek to speed up the initial R-CNN approach. R-CNN uses selective search to produce 2,000 candidate regions of interest (ROIs) and transfers each individual RoI through a CNN base, which creates a huge bottleneck as the processing of CNN is very slow. Quick R-CNN instead sends the entire image via the CNN base just once and then matches the ROIs generated with a selective search on the CNN feature map, resulting in a 20-fold reduction in processing time. Although Quick R-CNN is much faster than R-CNN, there is still another speed barrier. It takes about 2.3 seconds for Quick R-CNN to perform object detection on a single image, and selective search accounts for a full 2 seconds of that time! Faster R-CNN replaces selective search with a separate sub-neural network to produce ROIs, generating a further 10x speed and thus testing at a rate of about 7–18 fps. Despite these remarkable changes, videos are usually shot at least 24 fps, which means that Faster R-CNN would probably not keep pace. Regional-based approaches consist of two different phases: proposing and processing areas. This division of tasks proves to be somewhat inefficient. Another big form of object detection framework relies instead on a single one-state approach. These so-called single-shot detectors completely locate and identify objects during a single pass over the image, which significantly reduces test time. One such single-shot detector, YOLO, starts by setting up a grid over the image and allows each grid cell to detect a fixed number of objects of varying sizes. The grid cell associated with the centre of the object is responsible for predicting this object for each true object present in the image. A dynamic, multi-term loss mechanism ensures that all locations and classifications take place within a single process. One variant of this system, Quick YOLO, even achieved a rate of 155 fps; however, classification and localization accuracy drops sharply at this elevated level. At the end of the day, object detection algorithms are trying to strike a balance between speed and accuracy. These findings are informed by a range of design choices outside the detection system. For example, YOLOv3 makes images of varying resolution: high-resolution images usually see better accuracy but slower processing times, and vice versa, low-resolution images. The option of CNN base also affects speed-accuracy tradeoffs. Very deep networks like the 164 layers used in Inception-ResNet-V2 offer amazing precision but pale in terms of speed compared to VGG-16 frames. Design choices for object detection must be made in context, depending on whether the focus is speed or accuracy.

3.3 Multiple spatial scales and aspect ratios

For several applications of object detection, objects of interest can appear in a broad variety of sizes and aspect ratios. Practitioners use a variety of methods to ensure that detecting algorithms can capture artefacts on different scales and views.

3.3.1 Anchor boxes

Instead of selective search, Faster R-modified CNN's area proposal network uses a narrow sliding window across the coevolutionary image map to produce candidate ROIs. Multiple ROIs can be predicted at each location and represented with the reference anchor boxes. The shapes and sizes of these anchor boxes are deliberately selected to cover a variety of different scales and aspect ratios. This allows different types of objects to be detected, with the hope that the bounding box coordinates do not need to be much changed during the localization task. Other frameworks, like single-shot detectors, also follow anchor boxes for the initialization of regions of interest.

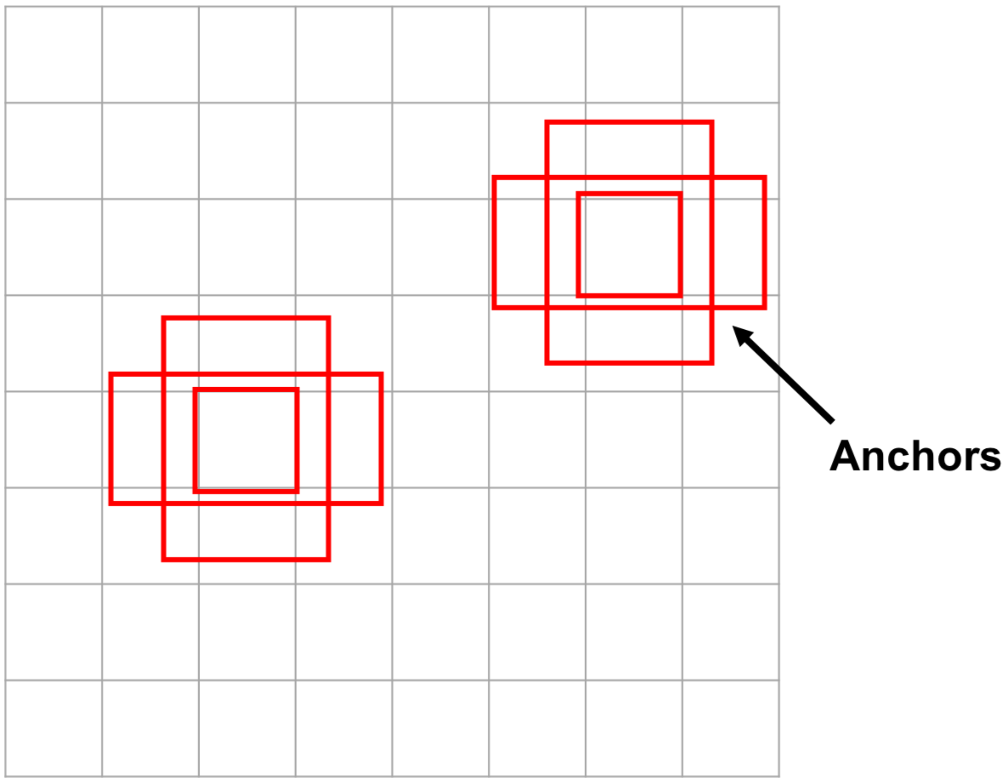


Figure 3.1: Anchor boxes

3.3.2 Multiple feature maps

Single-shot detectors must put specific emphasis on the issue of multiple scales because they detect objects with a single pass through the CNN system. If artefacts are detected from the final CNN layers on their own, only large items will be identified as smaller items can lose too much signal during downsampling in pooling layers. To address this problem, single-shot detectors usually search for artefacts within multiple CNN layers, including earlier layers where higher resolution remains. Despite the caution of using multiple feature maps, single-shot detectors are notoriously struggling to detect small objects, particularly those in tight clusters sort of like a flock of birds.

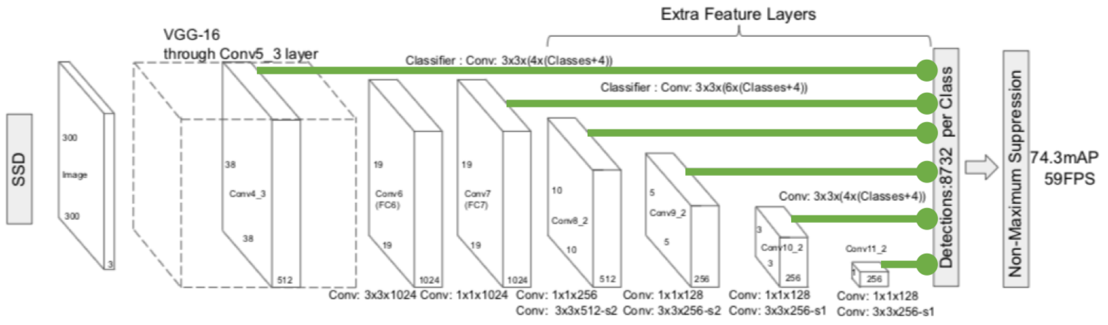


Figure 3.2: SSD feature layers

3.3.3 Feature pyramid network

The Function Pyramid Network (FPN) takes the idea of multiple feature maps one step further. Images first move through the standard CNN route, yielding semantically rich final layers. Then to restore better resolution, FPN produces a top-down direction by sampling this function diagram. Although the top-down pathway helps to detect artefacts of different sizes, spatial locations can be distorted. Lateral connections are inserted between the original feature maps and the corresponding restored layers to boost the position of the object. FPN currently offers one of the leading ways to detect objects on different scales, and YOLO has been expanded with this technique in its 3rd edition.

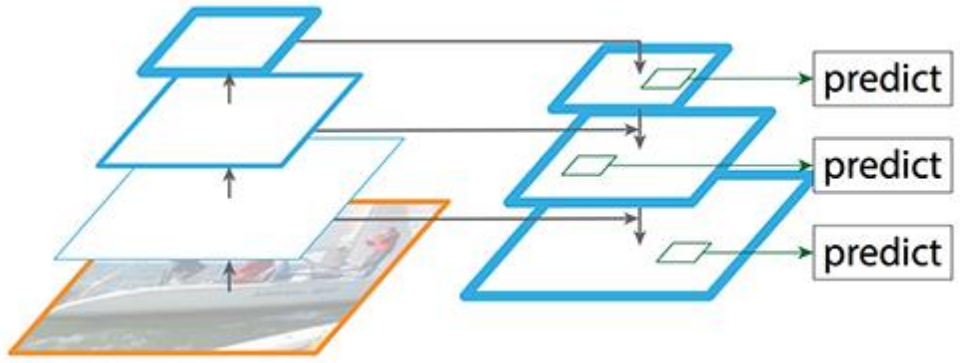


Figure 3.3: Pyramid detection layers

3.4 Data Limitations

Less amount of given data that is available for object detection shows to be another big problem. Object detection datasets normally hold empirical evidence examples for closely a dozen to a hundred groups of objects. while image classification datasets which include more than 100,000 classes. Besides, crowdsourcing also generates image classification equation tags for free (for example, by parsing the text of user-provided photo captions). Gathering ground truth labels along with correct bounding boxes for object detection, however, remains extremely time-consuming work. The COCO dataset is given by Microsoft currently leads to some of the best object detection data available. COCO comprises 300,000 segmented images of 80 different types of objects with very detailed location labels. Each picture contains around 7 objects on average, and the objects appear on a very large scale. As useful as this dataset is, object types outside of these 80 selected classes will not be recognized if they are trained solely on COCO. A very interesting approach to alleviating data scarcity has come from YOLO9000, the second edition of YOLO. YOLO9000 integrates several essential improvements into YOLO but also aims to narrow the dataset distance between object detection and image classification. YOLO9000 trains concurrently on both COCO and ImageNet, an image classification dataset of tens of thousands of object classes. COCO information helps to precisely locate objects, while ImageNet improves YOLO's "vocabulary" classification. A hierarchical WordTree allows YOLO9000 to first detect an object's concept (such as "animal/dog") and to then drill down into specifics (such as "Siberian husky"). This method tends to work well for concepts familiar to COCO as animals, but performs poorly on less prevalent concepts because RoI's recommendation is focused solely on COCO training.

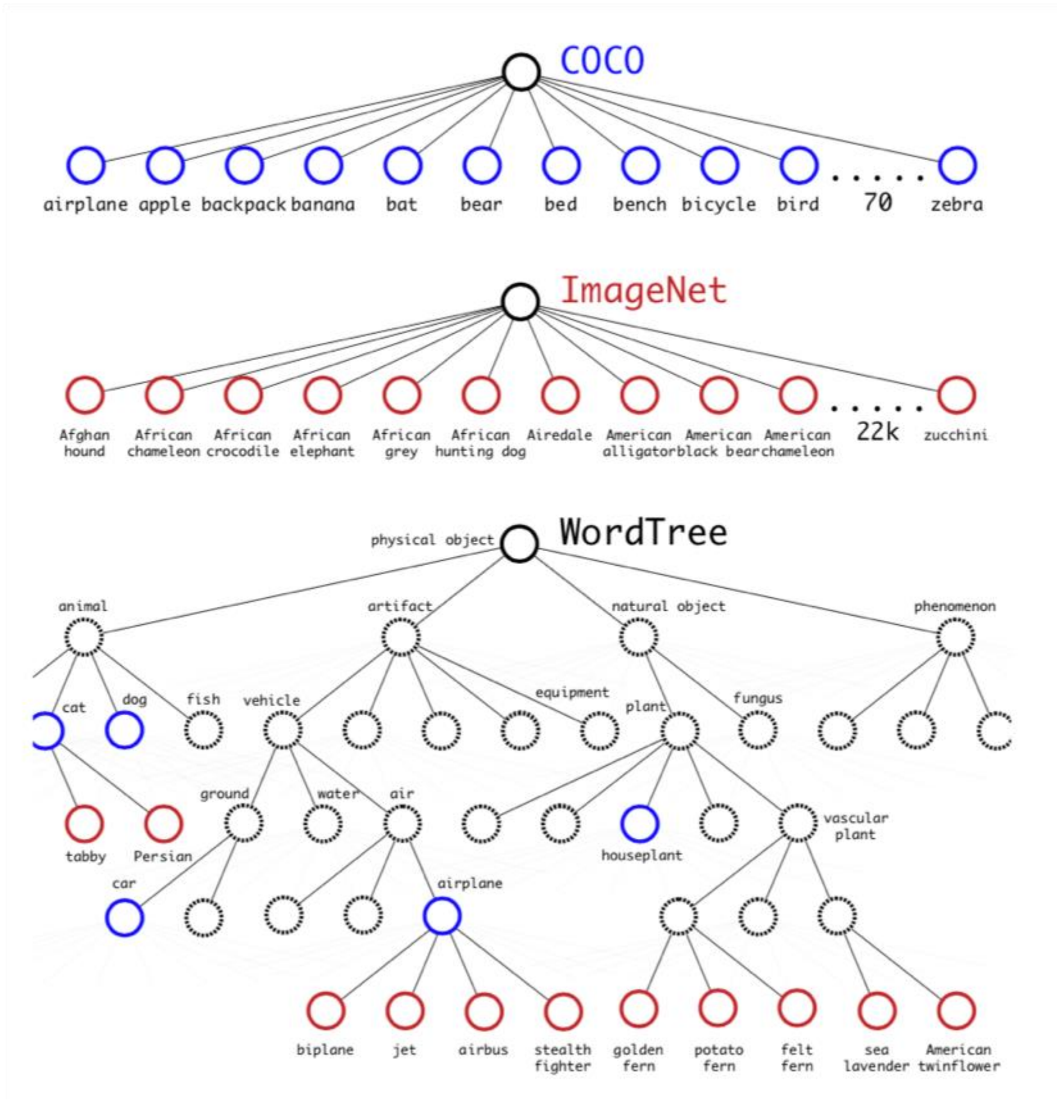


Figure 3.4: A variety of datasets

3.5 Class imbalance

Class imbalance proves to be a concern for most classification problems, and object detection is no exception. Consider a typical illustration. More probable than not, the photograph includes a few key objects and the majority of the picture is filled with background. Recall that a selective quest in R-CNN produces 2,000 candidate ROIs per image – just imagine how many of these regions do not contain objects and are considered negative! A recent method called focal loss is being applied in RetinaNet and helps to reduce the effect of class imbalances. In the optimization loss function, focal loss replaces the conventional log loss by penalizing misclassifications:

$$FL(p_u) = -\overbrace{(1 - p_u)^{\gamma}}^{*} \log(p_u)$$

The focal loss equation where p_u is the expected likelihood class for the true class and 0 for the true class. The additional factor (*) decreases the loss of well-classified examples with high probabilities, and the overall effect de-emphasizes classes with several examples that the model knows well, such as the context class. Objects of interest occupying minority groups are often more important and have increased accuracy.

Chapter 4

Implementation

As we worked with TensorFlow object detection API we used Python 3.7, pip, Pandas, NumPy, Cython, OpenCV, Google Colab. to train data and implement it on real videos and pictures. We also used Labellimg to level raw images to the format for YOLO and SSD training.

4.1 Setting Up Environment

Firstly we set up the TensorFlow API using NVIDIA CUDA library integrating with GPU for better performance.

4.2 Training model

The First step for training models is label data (images) for YOLO and SSD Coordination systems. For YOLO it has a distinctive system and for SSD it's called Pascal Voc coordinate system. it all can be done in "labellimg" software.

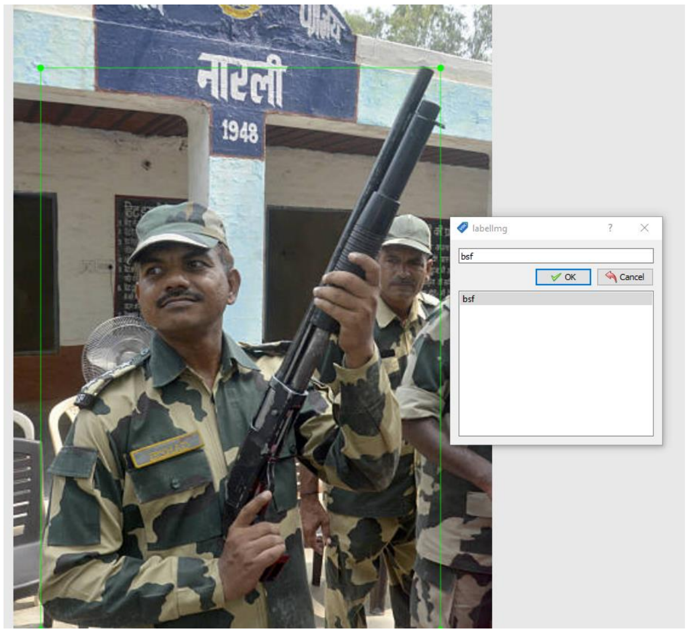
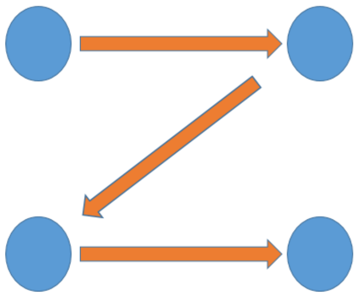


Figure 4.1: Image labelling

There are a few differences between the PASCAL VOC format and YOLO format. Pascal VOC format uses .xml files rather than .txt files, it refers to call class names directly in the file and uses a sequential coordination system. Which are upper left to the upper right and upper right to lower left and lower left to lower right. YOLO on the other hand uses .txt file to save Coordinates and refers to having a different file to store class names. YOLO also has a different approach for Coordination sequence. it's upper left to the upper right and upper right to lower right and lower right to lower left



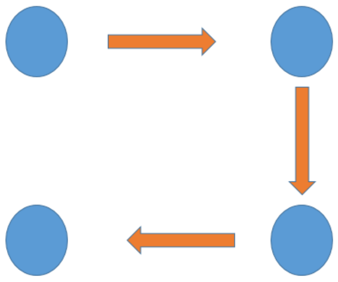


Figure 4.2: YOLO vs SSD Coordination system

4.3 Train YOLOv4

We manually label images first then train with YOLOv4 to generate weight data. First, we put the images into a repository and configure the YOLO for training. Then we use pre-trained YOLOv4 weights and train the new labelled images with them to get new weights. In the process, we have to check for overfitting and underfitting, and if such anomalies are detected, we train again until they are not present.

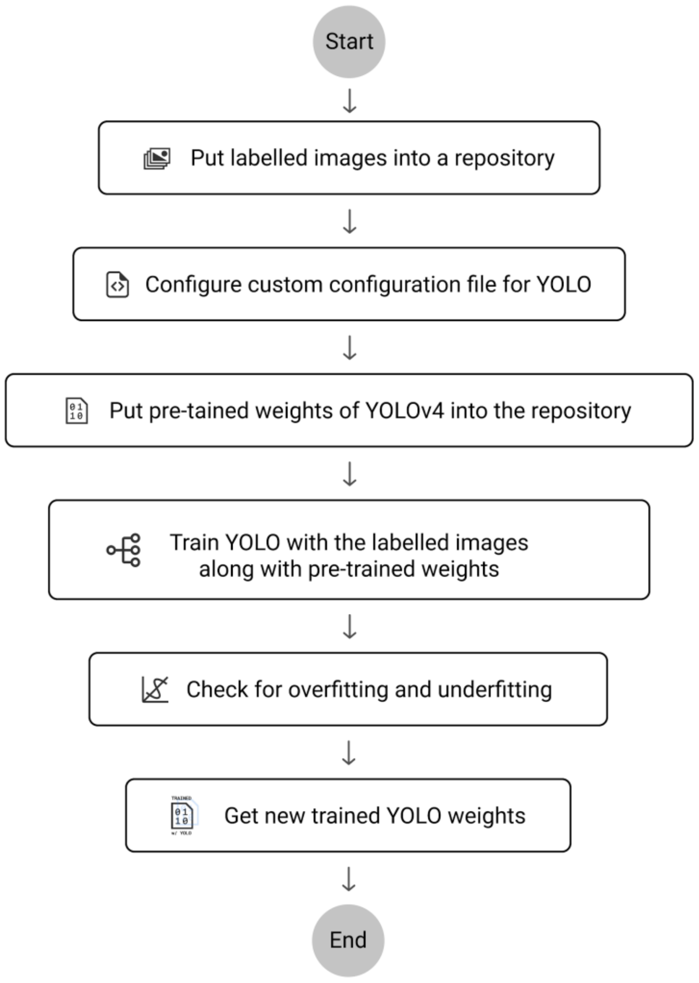
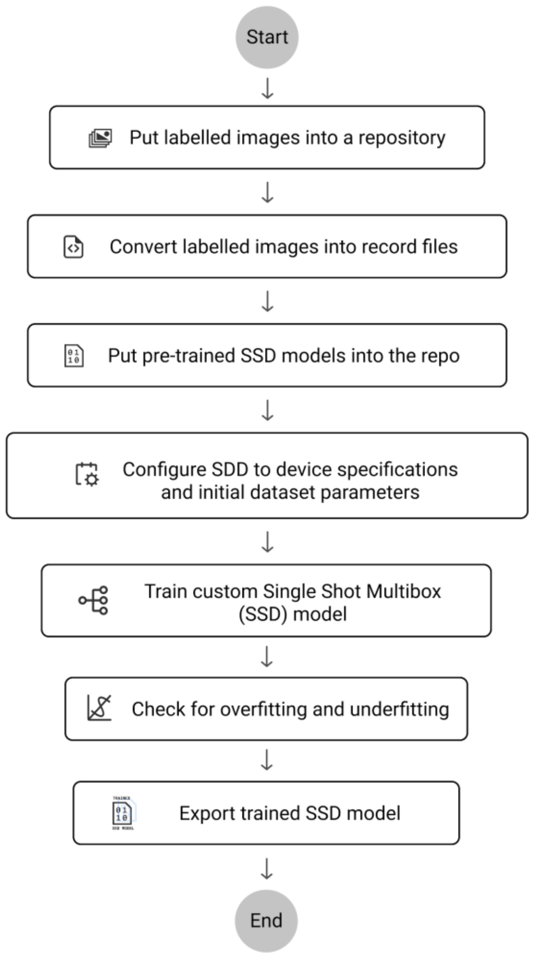

Start
Put labelled images into a repository
Configure custom configuration file for YOLO
Put pre-tained weights of YOLOv4 into the repository
Train YOLO with the labelled images along with pre-trained weights
Check for overfitting and underfitting
Get new trained YOLO weights
End

Figure 4.3: YOLO train Procedure

4.4 Train SSD in TensorFlow API

We put manually labelled images into the repository and convert them into .record file to cope with TensorFlow API system. then put pre-trained SSD models and configure it following the labelled data and classes. then train it by the script and wait for data to be trained but carefully noticing whether the trained model becomes overfitted or under-fitted. If they do so then we have to change some parameter and configuration and run the whole process again. Finally, we have the trained model ready to be exported.



Start
Put labelled images into a repository
Convert labelled images into record files
Put pre-trained SSD models into the repo
Configure SDD to device specifications and initial dataset parameters
Train custom Single Shot Multibox (SSD) model
Check for overfitting and underfitting
Export trained SSD model
End

Figure 4.4: SSD training Procedure

4.5 Output

We tested out custom trained models on a single image and compared the output and compared which one works on what conditions and detection number.

4.5.1 YOLOv4

On yolov4 it successfully detected all the desired objects with the confidence of 100%, 100%, and 98%, iou_norm: 0.07, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05 nms_kind: greedynm, beta = 0.600000 Total BFLOPS 59.563 avg_outputs = 489778, 162 layers from weights-file Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28, Predicted in 33.189000 milli-seconds

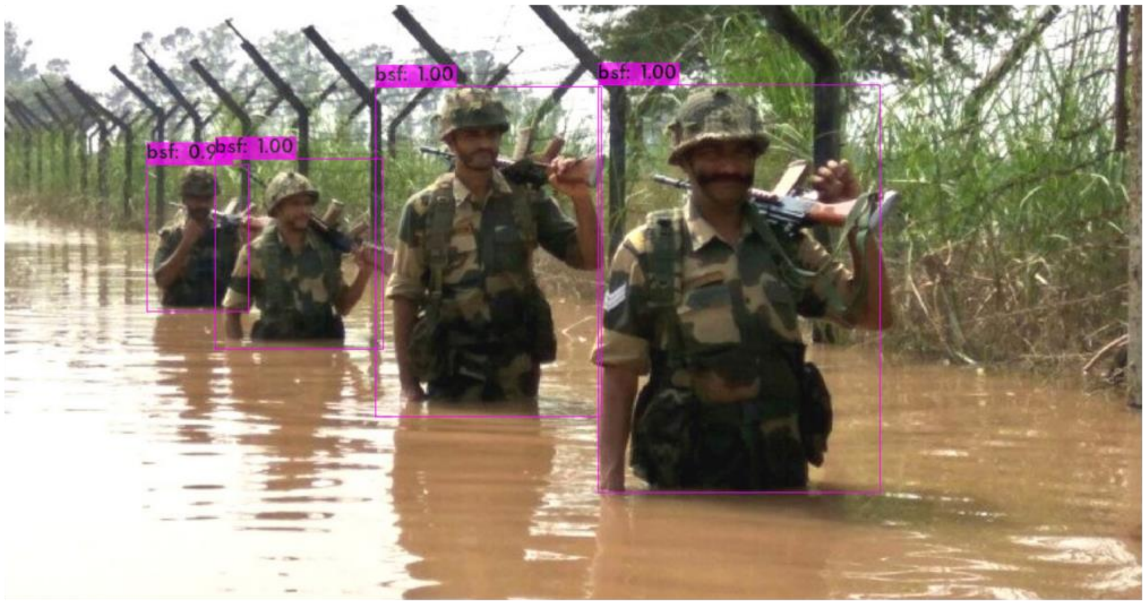


Figure 4.5: YOLOv4 detection in a low-resolution image

This is a Yolov4 model in lower resolution image and we also ran it by a higher resolution and pixel rate image and here is the outcome.

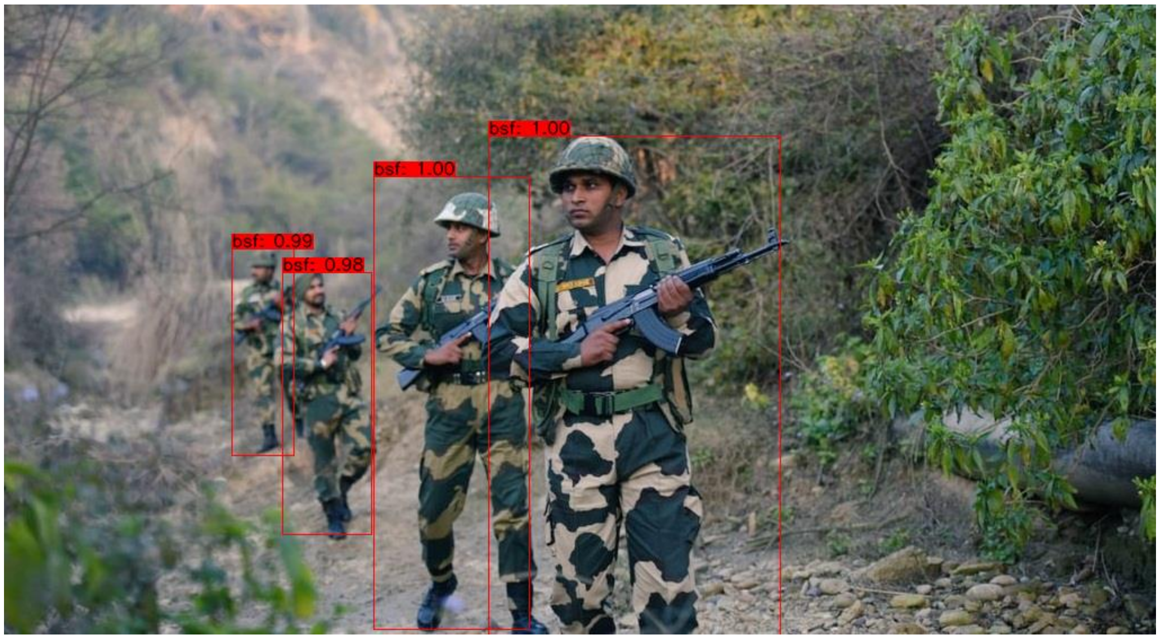


Figure 4.6: YOLOv4 on High-resolution image

4.5.2 SSD (TensorFlow API)

Training dataset in TensorFlow API was a bit problematic with having a risk of loss function reaching an overfitting level and when it can do the testing with different images it showed resistance on low-quality images like above which is done in YOLOv4. So it came out with zero detection. on the other hand, detecting higher quality images it had one detection and three missed detection making it inefficient compared to custom trained YOLOv4

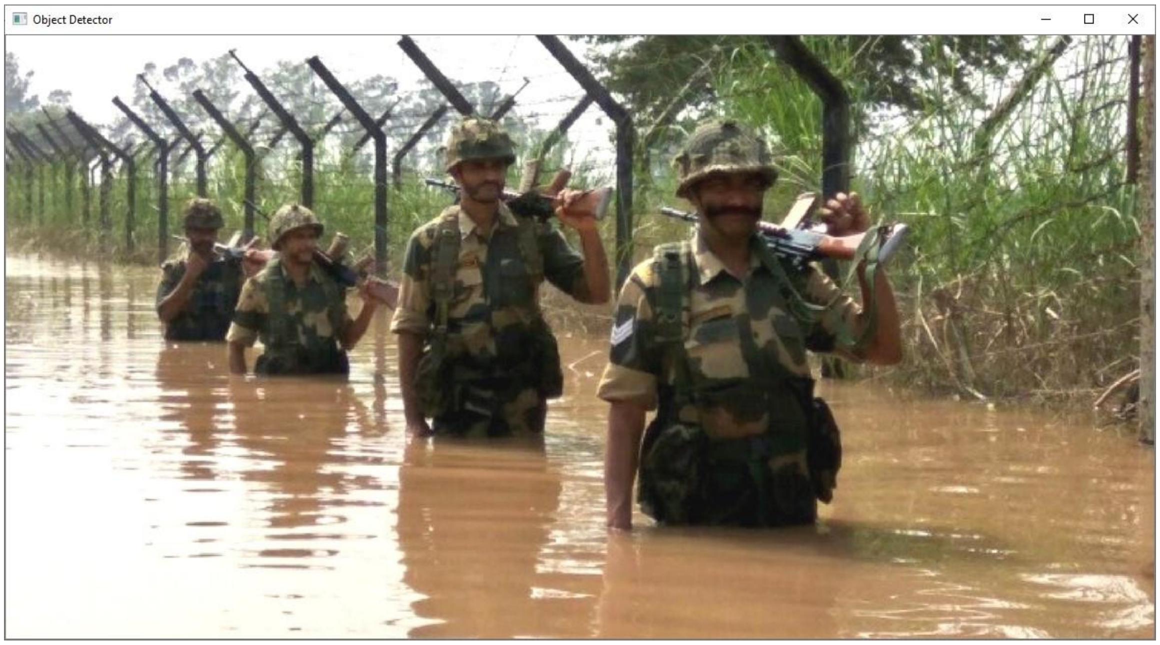


Figure 4.7: SSD on lower Quality image

Here is a low-quality image with zero detection. and Now we look into higher quality images which have one detection with 97% confidence rate

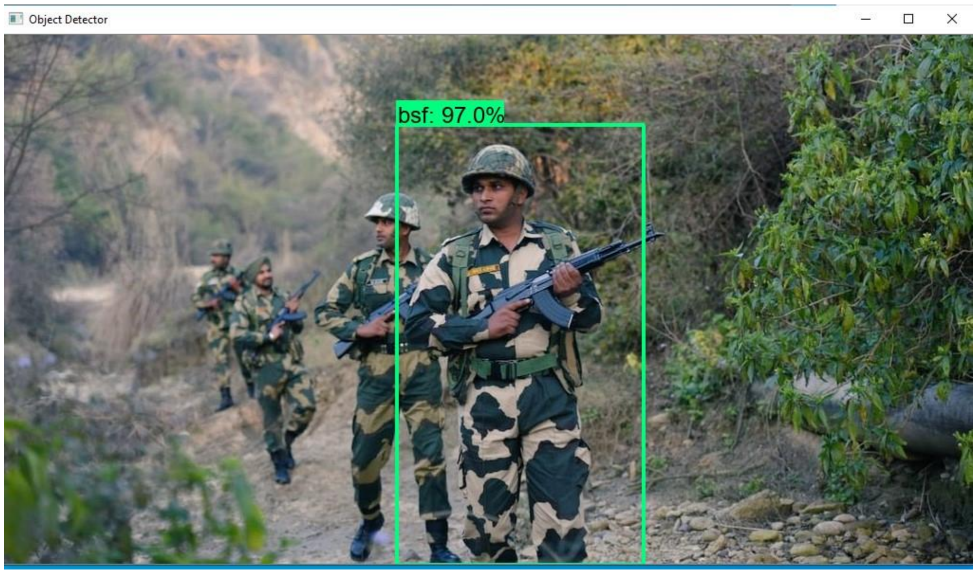


Figure 4.8: SSD on Higher Quality Image

Chapter 5

Evaluation and Result

In this section, we showed the comparison between YOLOv4 and the SSD Model. In Common Objects in Context (COCO) based Dataset mode, we noticed a big difference running and saving the detection events in a video which showed YOLOv4 had an average of 30 frames per second and SSD detection had an average of 20 frames per second. We also evaluated the total detection accuracy, how many detections they had and how much error, the confidence they produced. we also Evaluated matrices to find the final result of our process

5.1 Data Collection

We ran both YOLOv4 and SSD algorithms on a particular video (https://youtu.be/rMDiFFZbRcU) where we observed the detections by each scenario and by which we counted everything manually. The observation was based on how many events there were, our desired objects were on there and how many the algorithms detected

YOLOv4 Detection Data:

Events	TotalDetectionObjects	ActualObjects	correct	Wrong	NoDetection	maxConfidence	minConfidence
1	4	6	4	0	2	99	32
2	1	3	1	0	2	29	29
3	1	2	1	0	1	91	24
4	5	5	5	0	0	100	26
5	8	8	8	0	0	100	87
6	0	1	0	0	1	0	0
7	1	1	1	0	0	43	28
8	2	5	2	0	3	96	28
9	7	6	4	3	2	100	25
10	8	8	8	0	0	100	71
11	1	1	1	0	0	83	79

12	3	4	3	0	0	100	64
13	1	2	1	0	1	43	43
14	2	2	2	0	0	100	99
15	5	6	5	0	1	100	97
16	1	3	0	0	2	100	100
17	1	1	1	0	0	100	55
18	4	4	4	0	0	100	25
19	0	1	0	0	1	0	0
20	2	2	2	0	0	100	33
21	1	1	1	0	0	100	100
22	2	2	2	0	0	100	97
23	4	4	4	0	0	100	27
24	1	1	0	1	0	32	32
25	1	2	1	0	1	37	37
26	2	2	2	0	0	99	28
27	4	2	2	2	0	95	38
28	1	1	1	0	0	91	97
29	1	1	1	0	0	98	100
30	1	1	1	0	0	98	71
31	5	6	5	0	1	99	48
32	2	3	2	0	1	97	38
33	2	2	2	0	0	48	23
34	1	2	1	0	1	94	64
35	1	1	1	0	0	80	70
36	1	1	1	0	0	80	70

37	1	1	1	0	0	99	94
38	1	1	1	0	0	99	97
39	1	1	1	0	0	99	96
40	3	3	3	0	0	99	42
41	2	2	1	0	1	91	35
42	2	2	0	1	0	90	35
43	3	4	3	0	1	99	28
44	1	0	0	1	0	81	80
45	2	2	2	0	0	99	85
46	1	1	1	0	1	98	84
47	8	9	4	0	5	100	28
48	19	14	14	5	0	100	43
49	1	0	0	1	0	85	85
50	1	1	1	1	0	99	98
51	3	2	2	1	2	96	32
52	1	1	1	0	0	100	75
53	1	1	1	1	0	85	44
54	2	0	0	2	0	91	69
55	0	3	0	0	3	0	0
56	4	3	3	1	0	92	81
57	1	1	1	0	0	100	89
Total	145		120	20	33		

SSD Detection Data:

Events	TotalDetectionObjects	ActualObjects	correct	Wrong	NoDetection	maxConfidence	minConfidence
1	1	6	1	0	5	99	99
2	0	3	0	0	3	0	0
3	0	2	0	0	2	0	0
4	0	5	0	0	5	0	0
5	1	8	1	0	6	80	80
6	0	1	0	0	1	0	0
7	0	1	0	0	1	0	0
8	0	5	0	0	5	0	0
9	2	6	2	0	4	89	75
10	2	8	2	0	5	99	74
11	1	1	1	0	0	83	79
12	4	4	4	0	0	98	75
13	0	2	0	0	2	0	0
14	2	2	2	0	0	100	75
15	1	6	1	0	5	95	75
16	0	3	0	0	3	0	0
17	1	1	1	0	0	99	88
18	1	4	1	0	3	95	91
19	0	1	0	0	1	0	0
20	1	2	1	0	1	95	95
21	0	1	0	0	1	0	0
22	0	2	0	0	2	0	0
23	3	4	3	0	1	99	76
24	1	1	0	1	0	80	80

25	0	2	0	0	2	0	0
26	0	2	0	0	2	0	0
27	0	2	0	0	2	0	0
28	0	2	0	0	2	0	0
29	1	1	1	0	0	94	76
30	1	1	1	0	0	81	76
31	0	6	0	0	6	0	0
32	0	3	0	0	3	0	0
33	1	2	0	1	2	98	98
34	0	2	0	0	2	0	0
35	0	1	0	0	1	0	0
36	1	1	1	0	0	80	80
37	1	1	1	0	0	79	76
38	1	1	1	0	0	97	80
39	0	1	0	0	1	0	0
40	1	3	1	0	2	98	85
41	0	2	0	0	2	0	0
42	1	2	0	1	0	98	73
43	0	4	0	0	2	0	0
44	1	0	0	1	0	81	80
45	2	2	0	0	0	99	85
46	0	1	0	0	1	0	0
47	4	9	4	0	5	100	77
48	4	14	4	0	10	77	94
49	1	0	0	1	0	85	85

50	2	1	1	1	0	92	76
51	0	2	0	0	2	0	0
52	1	1	1	0	0	100	83
53	0	0	0	0	0	0	0
54	1	0	0	1	0	77	77
55	1	3	1	0	2	80	80
56	3	3	3	0	0	98	77
57	1	1	1	0	0	97	81
Toal	50		41	7	105		

Now that we have the raw data based on the observation. It's time to draw a visualization graph to have a visual comparison. So we used Matplotlib to draw the graph based on their detection status.

5.2 Total Detection

We took 57 events of Object Detection Scenario and ran it by SSD and YOLOv4. After that, we compared with another to find out which model is more frequent at catching trained objects. Here the data Shows that YOLOv4 for is more frequent at detecting objects.

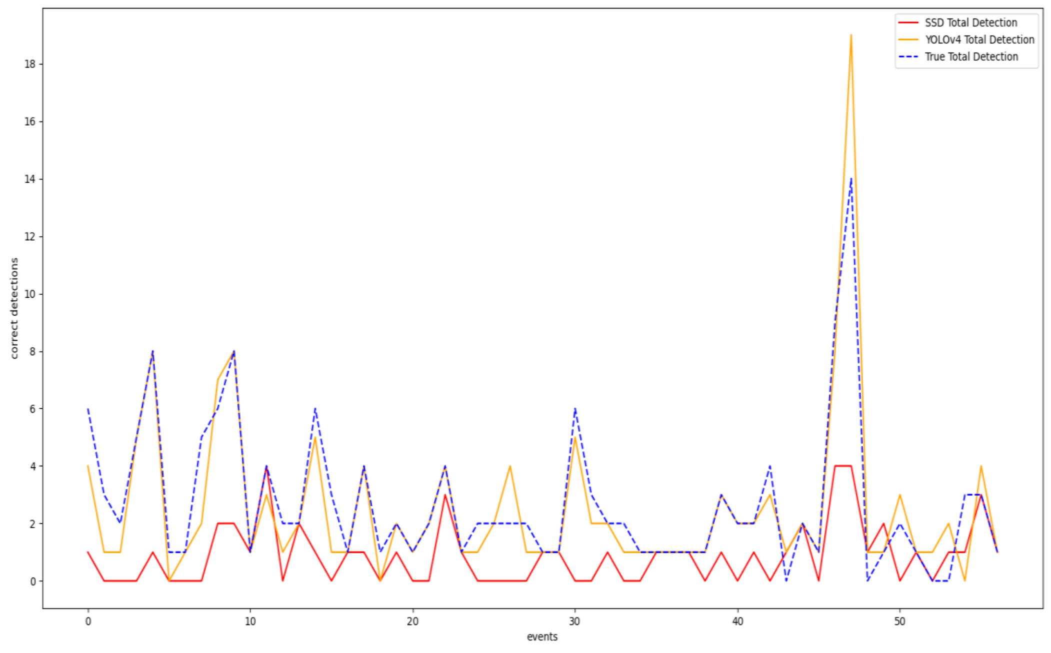


Figure 5.1: Total Detection YOLOv4 vs SSD

5.3 Correct Detection

Correct detection Differentiates between the total number of detection and no detections. here, SSD has more ‘no detections’ so YOLOv4 peaks higher

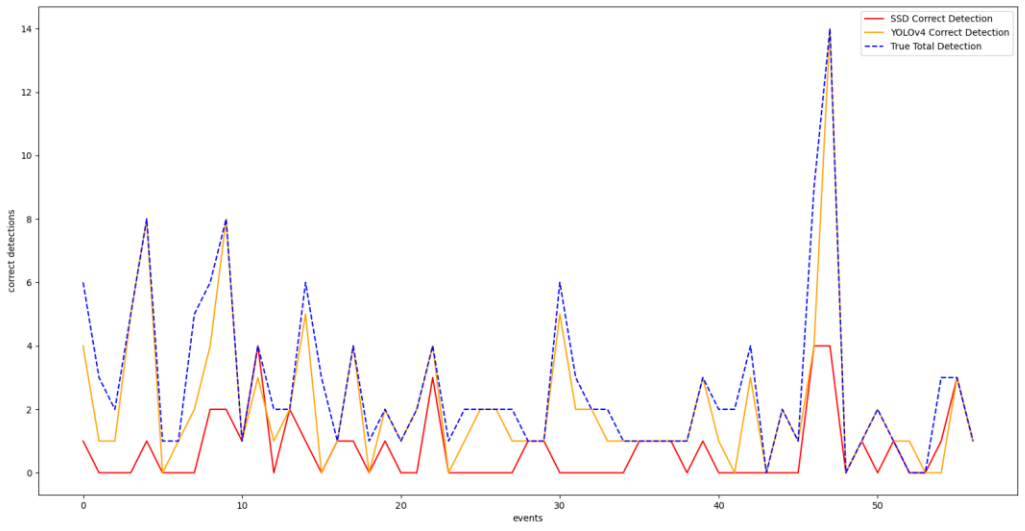


Figure 5.2 Correct Detection YOLOv4 vs SSD

5.4 Wrong and no Detection

YOLOv4 has more wrong detection than SSD because it has more spontaneous detection,

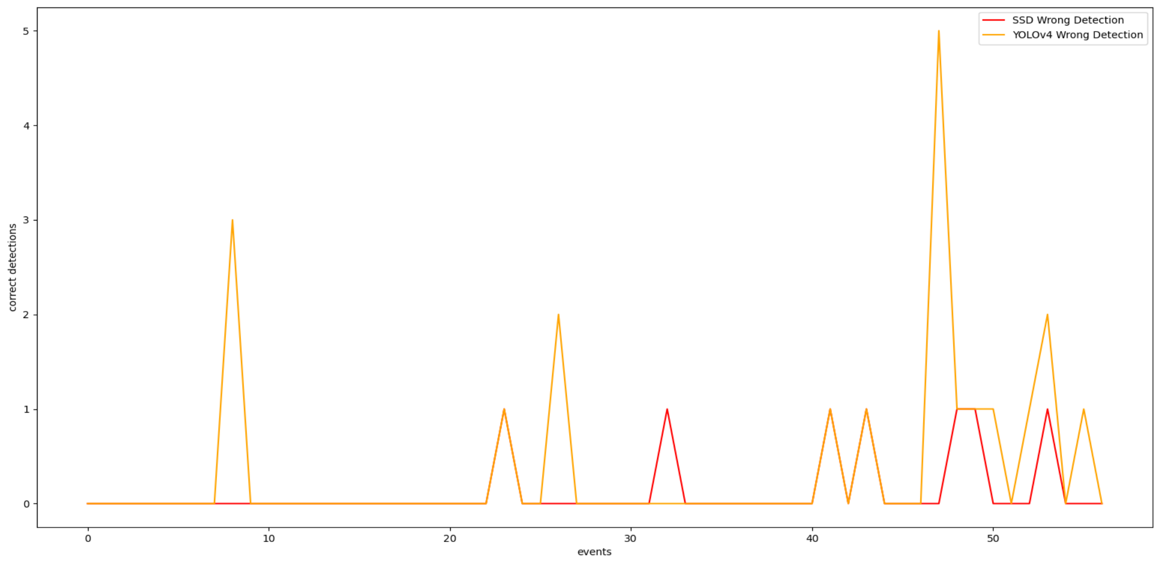


Figure 5.3: Wrong Detection YOLOv4

on the other hand, SSD has more no detection which makes us think about the pros and cons before thinking about choosing one model over the other

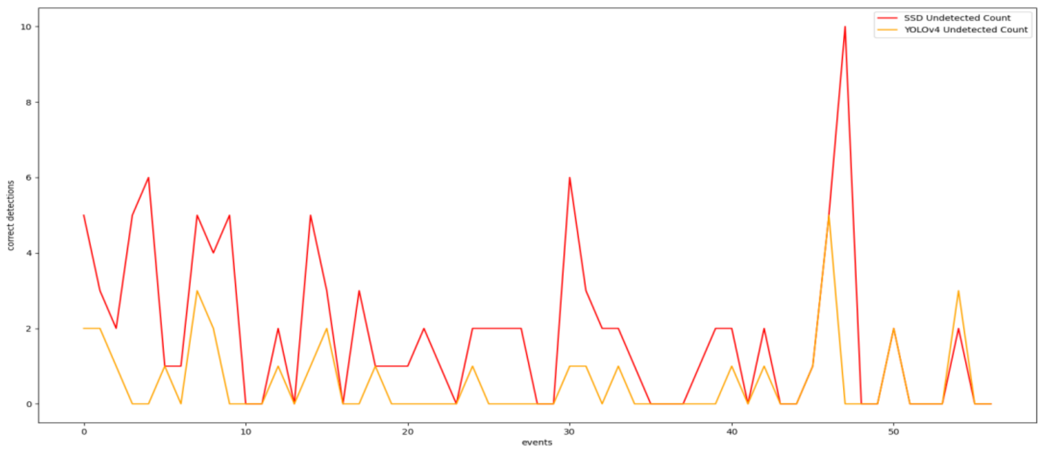


Figure 5.4: No Detection YOLOv4 vs SSD

5.5 Minimum and Maximum Confidence

Min-Max confidence and Average shows a variety of distances each model can go before

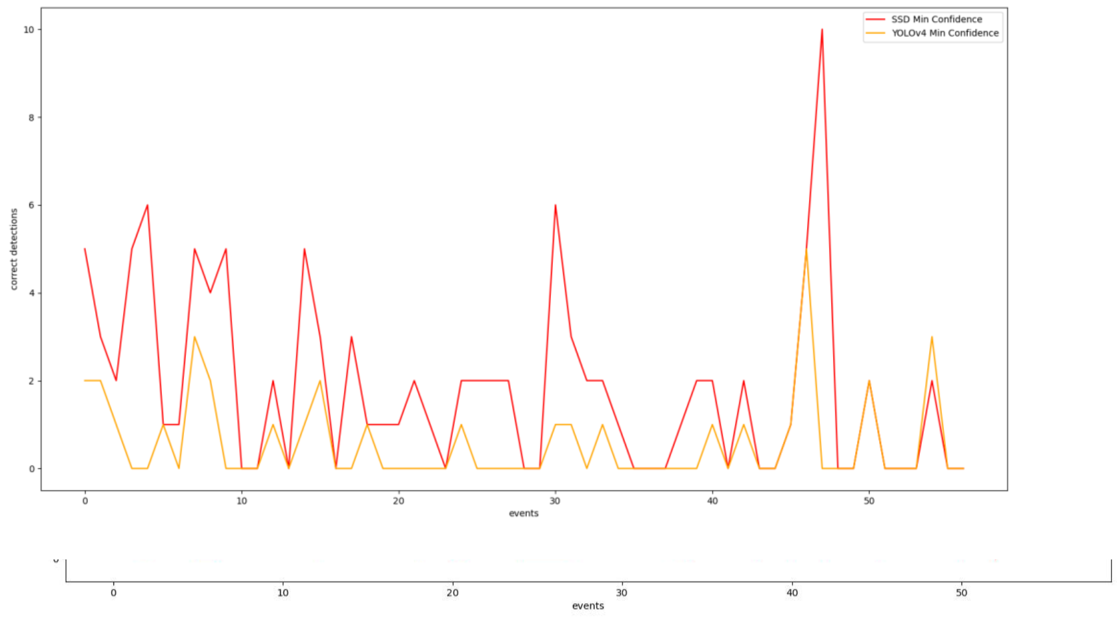


Figure 5.5: Min Confidence YOLOv4 vs SSD

detecting data. It is the percentage of a surety that the algorithm can give based on the training. These graphs show the limit of algorithms' correction detection flexibility and reach and ability to analyze the scenario of detection and their reach.

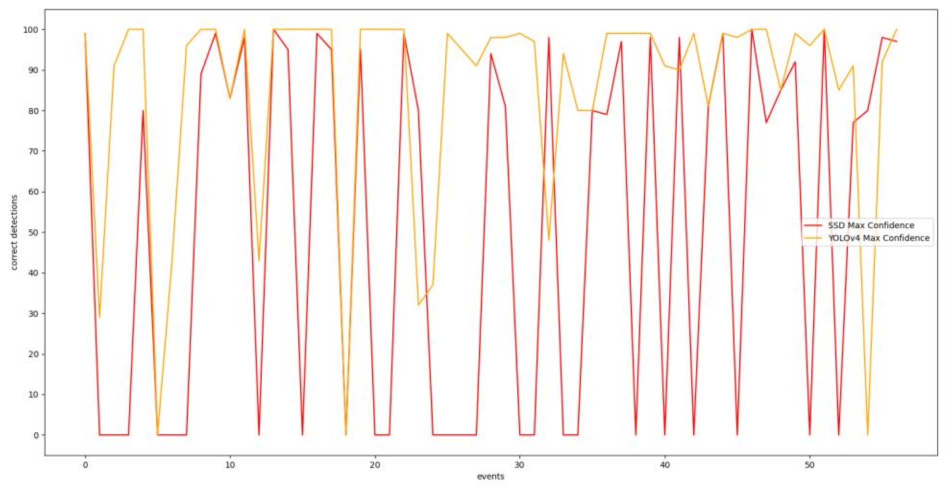


Figure 5.6: Max Confidence YOLO4 vs SSD

5.6 Classification Accuracy

Classification accuracy is the ratio of the correct number of predictions and Total number of predictions it shows how trained models are doing in the context of accuracy

Classification Accuracy = Number of Correct Predictions / Total Number of Predictions Made

Our 57 events show the classification accuracy of SSD

Classification Accuracy = 41/50 = 0.82 or

82 percent accurate predictions

and accuracy of YOLOv4

Classification Accuracy = 120/145 = 0.8275 or

82.75 percent accurate predictions

5.7 Confusion Matrix

A confusion matrix is a very good way to determine accuracy and precision. there is four terms should be considered before jumping into the calculations

True Positives: The cases in which we predicted **Correct** and the actual output was also **Correct**.

True Negatives: The cases in which we predicted **Wrong** and the actual output was **Wrong**.

False Positives: The cases in which we predicted **Correct** and the actual output was **Wrong**.

False Negatives: The cases in which we predicted **Wrong** and the actual output was **Correct**.

Now we calculate the SSD model for the accuracy and Precision

	Predicted Wrong	Predicted Correct
Actual Wrong	13	7
Actual Correct	105	41

Accuracy = 41 + 13 / (105 + 7 + 41 + 13) = 0.3253 or 32.53% accurate

precision = 41 / 41 + 13 = 0.7592 or 75.92%

For YOLOv4 we calculate the result

	Predicted Wrong	Predicted Correct
Actual Wrong	12	20
Actual Correct	33	120

Accuracy = 12 + 120 / = 0.7135 or 71.35% accurate

precision = 120 / 120 + 12 = 0.9090 or 90.9%

5.8 Result

So this analysis helps us to reach conclusions and allows us to tell the result forYOLOv4 is 71.35% accurate and SSD 32.53% accurate in detection. YOLOv4 90.9% precise and SSD 75.92% precise.

Chapter 6

Limitations and Future Work

From the beginning, we want to build a model with less error and more accuracy but having inaccessibility of more dataset makes it error-prone. There were very significant hardware limitations in case of building these state of the art level models using only 8 Gigabyte of RAM and NVIDIA Geforce 1050Ti which has 4 Gigabyte of Virtual RAM. whether it was recommended to have at least 64 Gigabytes of RAM and NVIDIA Tesla V100 which is enriched with tensor core GPU. Having These Hardware limitations slowed down our training process and caused various kinds of error and malfunction. We also had problems acquiring good quality datasets which had unavailability of getting access. Which had us become more invested in crafting datasets manually and configuring it to match up to our hardware. In the future, we determine to have more accurate results with having the right datasets and along with it to have more hardware build. We want to use it with IoT devices like drones and portable cameras but more importantly, we want to use it on actual border sites of the country.

Chapter 7

Conclusion

YOLOv4 has more accuracy and precision than SSD on our custom build model. we had evaluated and it made a satisfactory amount of 71.35% accuracy which is yet to be perfected with having more training data with all the ideal features. It makes adaptive change with the development of CNN models with constant updates of features, speed and accuracy. The project needs to be polished with the effort to have more accurate outputs and also need to discover the ways of implementations in real life. If the new Faster-RCNN based algorithms upgrade, it also should upgrade to maintain real-time detection and accuracy.

Bibliography

[1] Bochkovskiy, A., Wang, C., & Liao, H. M. (n.d.). YOLOv4: Optimal Speed and Accuracy of Object Detection. Retrieved April 23, 2020, from https://arxiv.org/abs/2004.10934

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, ScottReed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot Multi-Box Detector.arXiv e-prints, page arXiv:1512.02325, December 2015

[3] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. arXiv e-prints, page arXiv:1807.05511, July 2018

[4] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6, DOI: 10.1109/ICEngTechnol.2017.8308186.

[5] R. I. H. Abushahma, M. A. M. Ali, O. I. Al-Sanjary and N. M. Tahir, "Region-based Convolutional Neural Network as Object Detection in Images," 2019 IEEE 7th Conference on Systems, Process and Control (ICSPC), Melaka, Malaysia, 2019, pp. 264-268, DOI: 10.1109/ICSPC47137.2019.9068011

[6] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448, DOI: 10.1109/ICCV.2015.169.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXive-prints, page arXiv:1506.01497, June 2015.

[8] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, DOI: 10.1109/CVPR.2016.91

[9] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection.InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[10] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger.arXiv e-prints, page arXiv:1612.08242, December 2016

[11] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 936-944, DOI: 10.1109/CVPR.2017.106.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Gir-shick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Doll´ar. Microsoft COCO: Common Objects in Context. arXive-prints, page arXiv:1405.0312, May 2014.

[13]Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, MichaelIsard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete War-den, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In12th USENIX Symposium on operating systems Design and Implementation (OSDI 16), pages 265–283, 2016

[14] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks.arXiv e-prints, page arXiv:1710.05381, October 2017.

[15] Kotsiantis, Sotiris & Zaharakis, I. & Pintelas, P.. (2006). Machine learning: A review of classification and combining techniques. Artificial Intelligence Review. 26. 159-190. 10.1007/s10462-007-9052-3.

[16] Visa, Sofia & Ramsay, Brian & Ralescu, Anca & Knaap, Esther. (2011). Confusion Matrix-based Feature Selection. CEUR Workshop Proceedings. 710. 120-127.