Specialization in Electrical Engineering

Automatic Object Detection for Video-based Intention Recognition

Submitted by:
Dipendra Yadav
Matrikel-Nr.: 217205512

Submitted on:
31.01.2019

Supervisor:
Dr.-Ing. Kristina Yordanova

University of Rostock | Department of Computer Science and Electrical Engineering
Institute of Computer Science Mobile Multimedia Information Systems
Albert-Einstein-Str. 22 18059 Rostock | Tel: +49 381 498 7511

ABSTRACT

Video-based Intention Recognition is the process of being able to identify the intentions of the subjects involved in a video through observing their actions and effects of their actions on their environment. An action basically involves interaction of the subjects of the video with different objects in their environment to attain a particular goal. Hence, being able to automatically detect objects in their affinity is of paramount importance in recognizing their intentions.

The aim of this study was to explore different modern open source based object detection solutions that can be used to automatically detect different objects in a video, in this case to detect 9 classes of objects in a video showing the recipe for cooking scrambled eggs from **Quality of Life Grand Challenge Data Collection** of Carnegie Mellon University. Different object detection models were explored for the purpose and their advantages and disadvantages were discussed. TensorFlow Object Detection API, an open source framework for object detection was used to train and test **Faster RCNN** (Faster Region based Convolution Neural Network) and **SSD** (Single-Shot Multibox Detector) with Inception-v2 model. The models were implemented on **Google Colaboratory** which is a Jupyter notebook based research tool that provides free GPU for training and testing of a model.

The performance of the models in terms of mAP (mean Accuracy Precision), Total loss and Learning Rate were discussed during their training and their performance on untrained dataset were also calculated in terms of Confusion Matrix, accuracy, precision, Recall and F1-score.

Keywords: Object Detection, Intention Recognition, Machine Learning, TensorFlow Object Detection API, Faster RCNN model, SSD model, Google Colaboratory.

TABLE OF CONTENTS

FOREWORD

I would like to extend my gratefulness to the following list of people and groups for their help in different ways for the implementation of this work for Specialization in Electrical Engineering:

Dr. -Ing. Kristina Yordanova (Supervisor)

Dr. Yordanova acted as the supervisor for this work. I am grateful for her guidance and support throughout the work in the form of face-to-face meetings, technical advice and review of the implementation of the work and the report. I would also like to thank her for especially helping me in determining the scope of the work and providing necessary suggestions with the required technical modifications.

MSc. Martin Adam

I would also like to thank Mr. Adam for providing me the initial guidance to explore different methods for the implementation of this work and helping me in getting started with it by providing relevant literature references.

CMU Grand Challenge Dataset Collection

Frames from the videos of CMU Grand Challenge Dataset Collection were used to train and evaluate the implemented models in this work. I would like to thank the maintainers of this dataset for making it open source and freely available for the use.

EdjeElectronics [53]

The great tutorial by Evan was used as a guide for fine-tuning the pre-trained models for the detection of objects in CMU Grand Challenge Dataset Collection. I am really grateful for its existence.

I would like to dedicate this work to my family and friends.

Rostock, 31.01.2019

Dipendra Yadav

ABBREVIATIONS

API	Application Program Interface
DNN	Deep Neural Network
CMU	Carnegie Mellon University
FRCNN	Faster Region-Based Convolution Neural Network
SSD	Single Shot Multibox Detector
IoU	Intersection over Union
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
SVM	Support Vector Machine
VOC	Visual Object Class
mAP	mean Average Precision
R-CNN	Convolution Neural Network based on Region proposal
SPP	Spatial Pyramid Pooling
RPN	Region Proposal Network
SVD	Singular Value Decomposition
VOC	Visual Object Classes Challenge
YOLO	You only Look Once
RoI	Region of Interest
GPU	Graphics Processing Units

LIST OF FIGURES AND TABLES

1 INTRODUCTION

Object detection is a field in computer vision which deals with identifying instances of objects in an image or a video from the appropriate class of objects in that particular image or video. Object detection has evolved to become an important and central research area in the computer vision [1], which has been tested in various domains like driverless vehicles, robotics, video surveillance and pedestrian detection [2], [3].

The open source libraries like TensorFlow Object Detection API and DNN library from OpenCV provides with convenient open source frameworks. The pre-trained models like, Tensorflow model zoo for object detections can provide very high accuracy in recognizing different classes of objects. The video footage used in this study is from the CMU Grand Challenge Data Collection shot by head mounted high spatial resolution (800 x 600 / 1024 x 768) camera at low temporal resolution (30 Hertz). [35] As the view was from a high angle, the objects in the image were often occluded by other objects which made it necessary to train the models with specific dataset. TensorFlow Object Detection API was selected for this specific work due to its high performance in object detection and also for its easy-to-approach structure.

In this study, nine classes of object were chosen for testing the detections by TensorFlow Object Detection API. The model used for this work was Faster RCNN model and SSD model with inception v2 architecture. The pre-trained models were trained, tested and evaluated by the images extracted from the videos in eggs recipes of CMU Grand Challenge Data Collection. Following tasks were accomplished in this particular study:

1. A brief literature research conducted on state of the art methods for automatic object detection.
2. Implementation of Object Detection using FRCNN model with inception architecture.
3. Implementation of Object Detection using SSD model with inception architecture.
4. The variation of their mAP, Total loss and Learning rate during training of the models were discussed.
5. Confusion matrix by class intersection over union (IoU) alongside their accuracy, precision, recall and F1-score was calculated for both the models.

2 LITERATURE REVIEW

The main aim of object detection in computer vision is to identify all instances of objects from an established class, such as faces, peoples or cars in an image or video. Generally, there are not many instances of the objects available in the image but the main problem is with the sheer large number of possible locations and scales at which they are present and this is what is driving the engineers and scientists to come up with new and efficient methods to tackle this issue. [33]

Generally, all detections are addressed with some sort of pose data. The information can be as elementary as a position of the object, a position and range, or the location of the object defined in a bounding box. In some other cases, the pose information is more elaborated and consists of the specifications of a linear or non-linear transformation, i.e. a face detector system may also find the locations of eyes, nose and mouth, along with the bounding box of the face. A collection of training cases are used to build the design of an Object detection system. The amount of needed training examples depends on the various facet of class variability, i.e. in established situations with rigid objects even an illustration is enough, but most of the times multiple datasets are required to comprehend particular aspects of class inconsistency. [33]

Prior to the development of deep learning technology, the approach to object detection was to implement mathematical models on the basis of knowledge acquired from observations. But the evolution of deep learning technology has replaced the conventional approach of object identification and object detection. [34] The AlexNet was the first rewarding deep learning based technology for image recognition which also won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, a worldwide competition on computer vision. [4] And thereon the deep learning based object detection technology excelled in the succeeding LSVRCs. The substantial development in object detection based on deep learning was achieved in 2013 when the LSVRC combined object detection in their competition. The biggest advantage of the deep neural network is that it doesn't require any special premeditated features and it has a very robust feature depiction ability [5] which facilitates the use as a regression or classifier device. The most important aspect of traditional techniques is that mostly it uses mathematical models to detect an object. The below mentioned traditional object detection techniques can be classified into two groups. The techniques like the Hough transform, frame-difference, background subtraction and optical flow uses the method of feature accompanied with mathematical model which uses some known information of the present data to form a mathematical model and by solving the model in object detection scenes it provides the result. The latter two methods, i.e. the sliding window model technique and the deformable part model technique, considers the manner of region selection with feature extraction and classification. This method can be used for machine learning applications as it integrates the hand-engineered characteristics with the classifier to achieve object detection.[34] The most commonly used classical technique for object detection are described briefly as follows:

2.1 Traditional methods of object detection

2.1.1 Hough Transform technique

The Hough transform [6] technique basically converts the present image based space into parameter based space by creating curves in parameter space for every pixel in image based space. Then the parameters of curve in the space of Image is the coordinates of the point of intersection of the most curves in the space of Parameters which is found by voting. The Hough transform is good for the objects whose contour can be asserted easily by the analytical function, i.e. straight line, roundness, etc. The generalized object transform[14], [15] is more advanced than the normal Hough transform as it combines the graphic edge details with the edge point details and this leads to better speed and accuracy. Hence, it can detect object's category alongside object's shape [16]-[17] [18].

2.1.2 Frame-difference technique

The main idea of frame-difference technique, is that the motion object area and the difference image can be obtained by subtracting the two adjoining image frames which can be later denoised by binaryzation processing and morphological filtering. Most commonly used methods are two-frame-difference, three-frame-difference and four-frame-difference techniques [7].

2.1.3 Background subtraction technique

The background subtraction technique is quite identical to the frame-difference method and the only contrast is that the background subtraction technique requires to characterize a background frame and also amend it appropriately. The background subtraction technique works in three phases, i.e. first background modelling then object detection and then finally background updating. [8]

2.1.4 Optical flow technique

The optical flow technique is devised by Horn and Schunck which basically presumes that the change in gray is hardly relevant to the object motion and lays out the motion of image pixel by forming an optical flow equation[9][10].

2.1.5 Sliding window model technique

The sliding window technique uses a window of rigid size and then slides it on the target image by using some scheme while deriving some characteristics from the window which is then later classified by some classifier. The main advantage is that the model can select SHIFT, gradient histogram and color histogram whereas the classifier can select the Adaboost and SVM classifier [11].

2.1.6 Deformable part model technique

The deformable part model technique is one of the most successful traditional technique for object detection as it has won the Visual Object Class (VOC) detection championship in the year 2007, 2008 and 2009 and has been in use extensively. It basically consists of two models, i.e. main model and sub-model. The main model detects the global features and the sub-model detects the local features. The object is divided into various parts by sub-model and then it extracts the features from all the parts. The extraction of feature and classification in this method is also done by using the sliding window model [34].

2.2 Deep Learning based techniques of Object Detection

The deep learning based method uses the approach of region selection + feature extraction + classification. Some approaches can be used to select the region, the convolution neural network can be used to extract feature and some special neural network or SVM can be used for the classification.[34] Earlier techniques of deep learning for object detection were Deep Neural Networks [23] and Overfeat [24] which began the era of object detection with deep learning. Deep neural networks (DNN) uses basically two subnetworks i.e., classification subnetwork that is used for recognition and the regression subnetwork that is used for locating. When the softmax layer in the stern is supplanted with a regression layer then the DNN can function as a regression subnetwork and if combined with classification subnetwork then it can detect the object. Figure 1 shows the DNN regression networks.

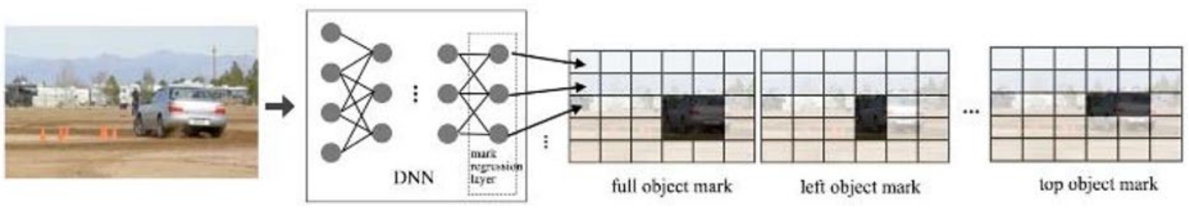


Figure 1. DNN regression networks. [23]

DNN can detect the object by masking the object region with binarizated grid sequence. Two adjoining objects can be differentiated simultaneously by the five ground truth marks which are used to make regression. Contrasting regions of target are covered by the general target mark, left target mark, right target mark and bottom target mark. But the efficiency of DNN is not so adequate and it was able to achieve just about 30% mean Average Precision (mAP) in the dataset of VOC2007. [34] The Overfeat technology was propounded by the team under LeCun which uses an advanced deep convolution model Alexnet. It can classify the objects by facilitating the slide window and offset on the images of different scales and it uses a regression network to pinpoint the objects. [24] The working of Overfeat for object detection is shown in the Fig. 2.

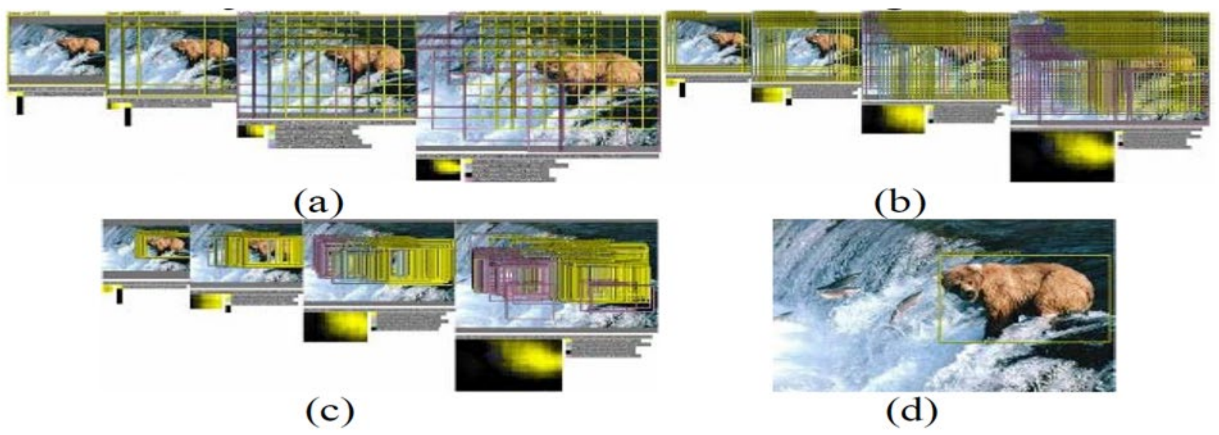


Figure 2. The working of Overfeat for object detection: (a) Multiscale; (b) Recognition; (c) Regression; (d) Merge. [24]

Fig. 2(a) shows the multiscale recognition process. We can see that in the first two images only bear is recognized but in the latter two images both fish and bear has been recognized simultaneously. Fig. 2(b) shows the identification process by utilizing the slide window and offset operation which increases the precision of recognition; Fig. 2(c) is regarding regression process which improves the precision of location Fig. 2(d) is the outcome of detection succeeding scale integration. The main drawback of this method is that it requires huge computation power for offset operation and slide window operation shown in Fig. 2(b) and Fig. 2(c). The mAP of the Overfeat for the test dataset of ILSVRC13 was just 19.4%.

The main problems with initial object detection techniques based on deep learning is that it utilizes sliding window method embraced by Overfeat to acquire the probable objects and this method would lead to the issue of data explosion due to its imperceptive and extensive method. As the preliminary model design was having problems with accuracy, the models subsequently tried to address this issue by either upgrading the existing techniques or propounding new ideas. Driven by the above disadvantages, new techniques and models for object detection based on deep learning evolved.

In the last few years, there has been a boom in deep learning technology for object detection and with that a lot of models are put forth for deep detection. This report focuses on the seven present-day established deep learning based object detection models which can be approximately put in two categories: (a) Models based on region proposal and (b) Models based on regression.

2.2.1 Models Based on Region Proposal

The main idea behind the object detection based on region proposal method is first to find the candidates for the region and then additionally to build the deep neural networks.

2.2.1.1 R-CNN

The notion of region proposal was intimated by Girshick in 2014 with the convolution neural network based on region proposal, i.e. R-CNN. The main idea of R-CNN is the region segmentation technique for selective search which is used to decide the region proposals in the image. Once the feasible object candidates are found, they are fed to the CNN to get the feature vectors which are then classified by SVM classifier to get the final region proposal. The output is detected by the object bounding boxes produced by the non-maximal suppression (NMS). [26] The complete procedure is shown in Fig. 3.The mAP result of R-CNN for object detection has significantly raised compared to previous methods. For VOC2007 test dataset mAP was 58.5%. [34] However, it has unsatisfactory real time capability as each region proposed has to go through CNN one after another and this is one of its main drawbacks.

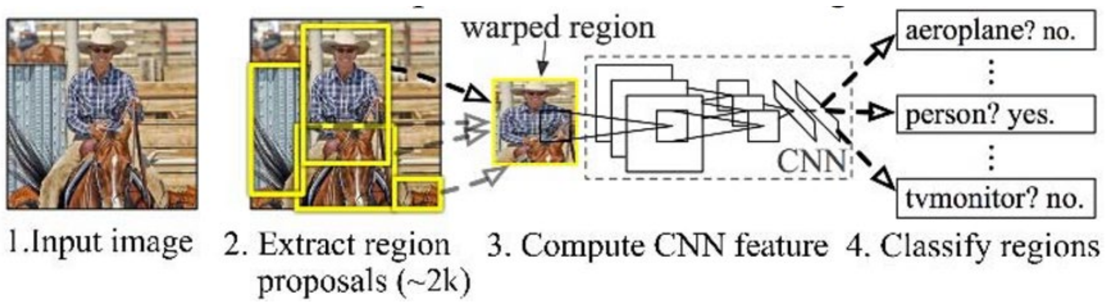


Figure 3. The complete process of R-CNN object detection [26]

2.2.1.2 SPP-Net

SPP-Net was proposed in 2014 by MSRA He which is a deep neural network which has an additional spatial pyramid pooling layer. The main advantage of this method is that it eliminates the need of crop/wrap operation on the input image which previously needed to customize the CNN to a fixed dimension. [27] Hence, it removes the possibility of inadequacy and object deformation of the image as shown in Fig. 4.

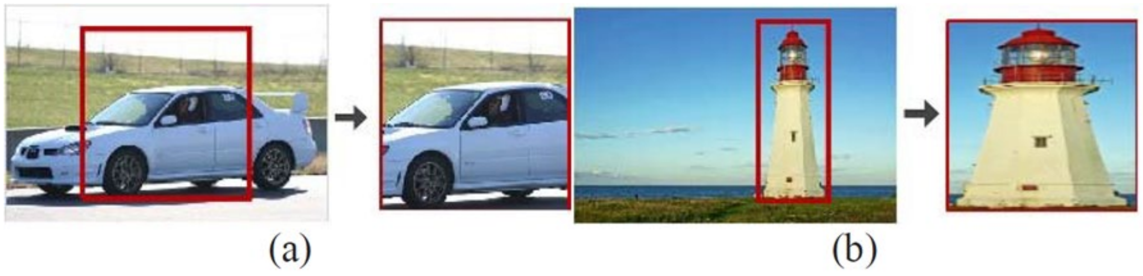


Figure 4. Crop and warp operation: (a) Crop operation; (b) Warp operation. [27]

The pros of this method is that it fixes the problem of image incompleteness and object deformation but the cons are that it utilizes the same technique for image processing as the former method and hence has unsatisfactory real-time capability.

2.2.1.3 Fast R-CNN

Fast R-CNN rectifies the issue of repetitive computation problem of the 2000 proposed regions going through the CNN in turn. Fast-CNN basically evolved from R-CNN and was proposed by Girshick and plays the similar role as that of Spatial pyramid pooling in SPP-Net. It outlines the outcome of selective search algorithm which is the proposed region of the image and feeds it to the feature layer of CNN. Then it conducts the ROI pooling on the mapped proposed region. It improves by extracting the fixed size feature vectors which is needed for indispensable connection with full connection. [28] The overall working of Fast R-CNN is presented in Figure. 5.

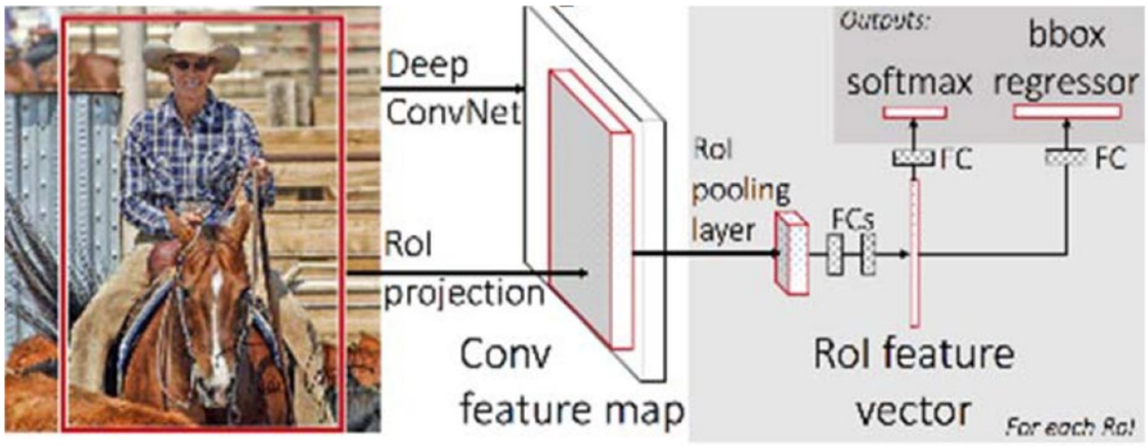


Figure 5. The working of fast R-CNN [28]

There is significant reduction in calculation due to implementation of mapping region proposal of image to feature layer and there is substantial deduction in the number of parameters by embracing curtailed SVD. This in turn facilitates in reduction in the network calculation as to a weight matrix which was reciprocal to a fully connected layer is replaced by two short fully connected layers. Furthermore , Fast R-CNN is 8.8 times swifter than R-CNN and 2.58 times swifter than SPP-net in training phase, 146 times swifter than R-CNN when with uncurtailed SVD, 213 times swifter than R-CNN when with curtailed SVD and 7 times swifter than SPP-net when with uncurtailed SVD and 10 times swifter than SPP-net when with curtailed SVD. [34]

2.2.1.4 Faster R-CNN

Faster R-CNN rectifies the problem of large computation and insufficient real-time in R-CNN and Fast R-CNN due to selective search method. It was put forth by Ren, He, Girshick, et al. which utilizes region proposal network (RPN) and end to end framework to solve the above mentioned problems and also to ease the training of the models. RPN succeeds the task of selective search in gathering region proposals by breaking down the feature layer to n×n regions and gathers feature regions of different scales and aspect ratio. The above mentioned technique is called anchor mechanism and these are utilized to bring forth object proposals which are then forwarded for object recognition and location to the endmost classification and regression networks. [29] The working of this method is shown in figure 6.

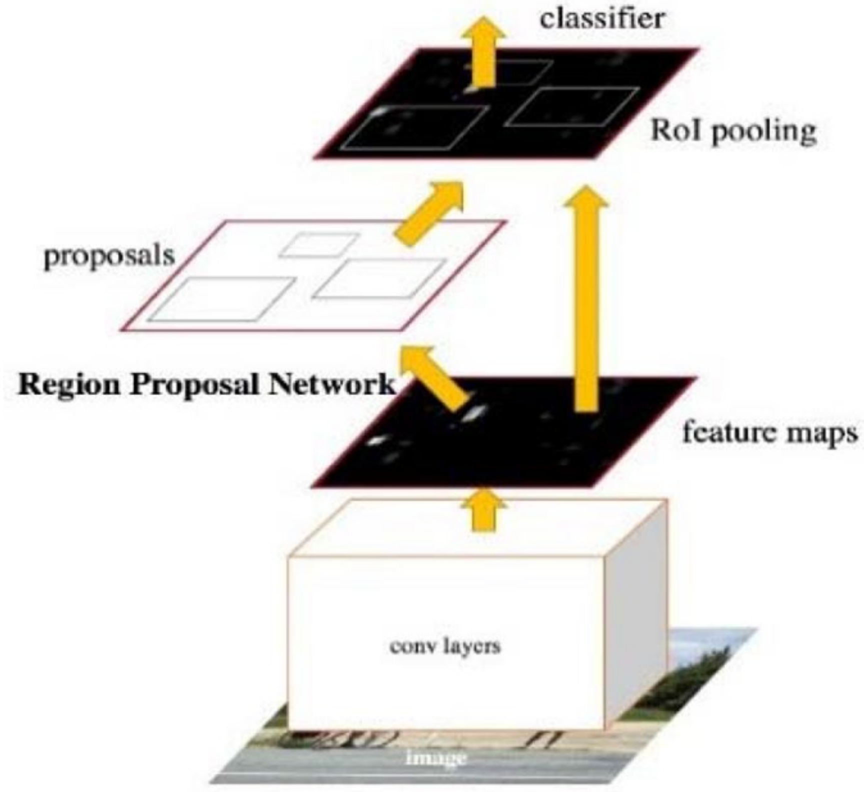


Figure 6. The working of faster R-CNN [29]

The number of region proposals are reduced to 300 from 2000 by using the RPN which remarkably improves the computation of the entire neural network. The mAP has improved by 2% to 3% compares to Fast R-CNN in the datasets of VOC2007 and VOC2012. [34]

2.2.1.5 R-FCN

R-FCN is a full convolution neural network introduced by Dai which solves the problem associated with Region of Interest. The region of interest is generated by RPN. It can document the response of all the objects present in different locations by using positive-sensitive score maps. It generally decides the feature vector by polling according the Region of Interests and embracing softmax classification to arrange the feature vectors in classes to detect the object. R-FCN is 2.5 times swifter than Faster R-CNN. Figure 7 depicts the working principle of R-FCN. [30]

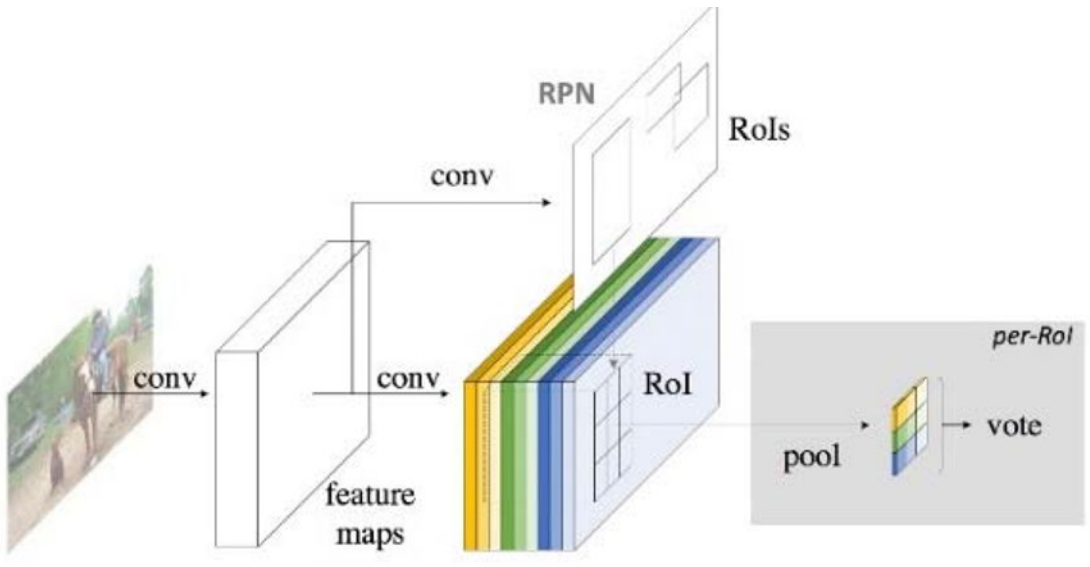


Figure 7. The working method of R-FCN [30]

2.2.2 Models Based on Regression

The current object detection techniques uses region proposals to get sufficient results but it performs poorly in context of real-time applications. The following two methods are based on regression and had achieved significant improvement with regard to real-time applications.

2.2.2.1 YOLO

You Only Look Once (YOLO) [31] is a convolution neural network based technique which can achieve end to end training and as it does not use any RoI module, it no longer needs to come up with proposals of object regions. It was proposed by Redmon, Divvala, Girshick, et al., for object detection. YOLO consists of a CNN in the front to extract feature and for the purpose of classification and regression, it consists of two full connected layers in the backend in grid regions. It groups the input image into 7∗7 grids and for each grid it generates two bounding boxes. The bounding boxes gives information about coordinate and confidence about object in form of four dimensional vector. All grids produces 30-dimensional vector which includes the four dimensional vector coordinate alongside with 20 class probabilities. And then finally YOLO refines the low confidence object proposals by the help of threshold and erases the repetitive object proposals to finally detect object. Summary of this whole process is shown in figure 8. [31]

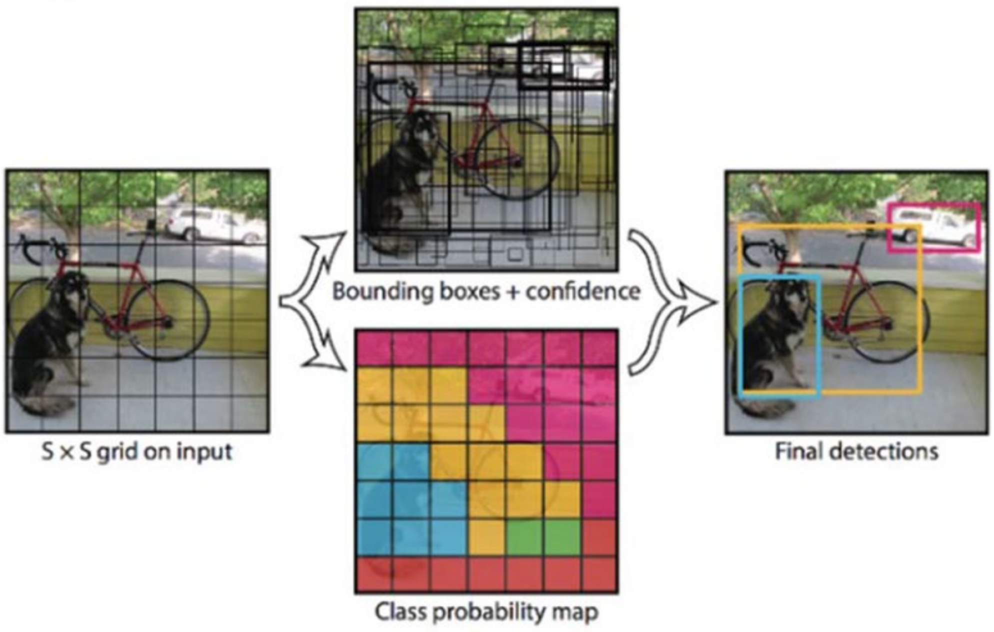


Figure 8. The operation principle diagram of YOLO [31]

YOLO speed of detection is 45 fps and the later version of it can reach 155fps to achieve the real-time detection but it doesn't perform as well as other state-of-the-art models due to the removal of region proposal. [34]

2.2.2.2 SSD

Single shot multi-box detector [SSD] was proposed by Liu Wei which basically combines the regression idea of YOLO and anchor mechanism of Faster R-CNN. The implementation of idea of regression reduces the computational complexity and makes sure that it is suitable for real-time applications and it can provide great detection accuracy by implementing the scales of various aspect and features ratios inherited with the anchor mechanism from Faster R-CNN. It has upper hand over YOLO in general feature extraction. It implements the method of multi-scale feature extraction which makes the detection of different-scale objects easier. [32] The overall working principle of SSD is shown in figure 9.

The SSD is the first object detection technique to achieve the detection speed of 59 fps and the accuracy of 74.3% [32] However, it needs to improve its detection capability in regard to small objects as it does not perform so well in this regard.

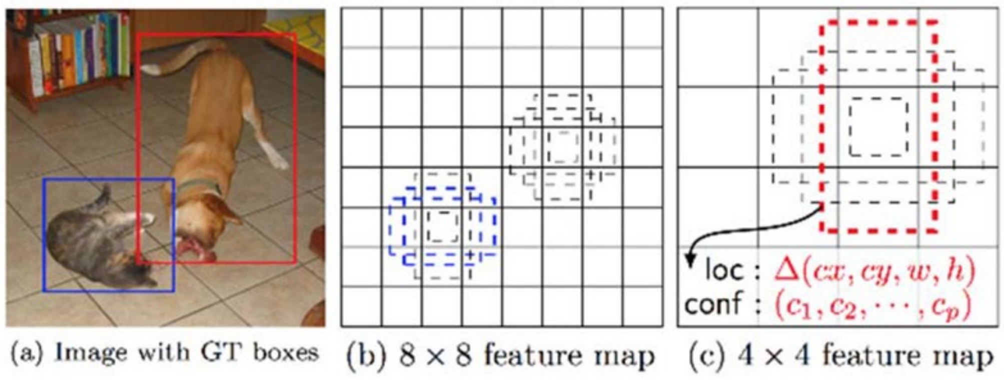


Figure 9. The working principle of SSD [32]

3 IMPLEMENTATION OF OBJECT DETECTION

The project was implemented using Faster RCNN model and SSD model in python 3 on Google colaboratory with Tensorflow being used to train the deep network. The specification of the virtual machine which Google colaboratory used to train, test and evaluate the model is:

1. GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM. [37]
2. CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e. (1 core, 2 threads). [37]
3. RAM: ~12.6 GB Available. [37]
4. Disk: ~33 GB Available. [37]

3.1 Tensorflow

Tensorflow is an open-source software mathematical library developed by Google Brain for high performance numerical computations. It uses C++ for the accomplishment of the applications and Python to setup applications with the framework which provides convenient Application Programming Interface (API). It provides high degree of flexibility with the implementation of computation irrespective of the platforms like Central processing units, Graphics processing units or Tensor Processing Units. It is accessible from most of the operating systems like Windows, macOS, Linux and also on mobile computing platforms like iOS and Android [36].

3.2 Google Colaboratory

Google colaboratory is a Jupyter notebook based research tool that can be used for machine learning related education and research. It can be accessed by major internet browsers like Chrome and Firefox. It supports both Python 2.7 and Python 3.6. A virtual machine is dedicated to a particular google account for code execution which has a maximum lifetime as enforced by the system and which is recycled when machine stays idle for some time. Colaboratory is basically for collective use and hence it may interrupt long running background computations on GPU. But this can be avoided by using a local runtime which supports continuous or long-running computations [37].

3.3 Data and preprocessing

For the purpose of this study, the publicly available CMU Grand Challenge Dataset were used. It consists of 18 subjects cooking five different recipes (brownies, pizza, sandwich, salad and scrambled eggs) which are recorded by three different cameras with different spatial and temporal resolution [35]. All the data images for training, testing and evaluation were extracted from the videos of subjects cooking egg recipes shot by the head mounted high spatial resolution (800 x 600 / 1024 x 768) camera at low temporal resolution (30 Hertz).

QUALITY OF LIFE TECHNOLOGY CENTER
Grand Challenge Data Collection

Figure 10. Dataset [35]

VLC Media Player was used to extract per frame raw images at 10 frames per second from the videos exhibiting recipe of eggs. As the objects in all the videos of this particular recipe class were almost similar and exhibited minor differences, hence, frames from three random videos were used for training and testing the model. However, frames used for evaluation and generation of confusion matrix were chosen from remaining videos. The final data consisted of 400 training data images, 100 test data images and 100 evaluation data images from three separate videos (total 600 video frames). For the purpose of better comparison of performances, same set of training, test and evaluation video frames were used for both FRCNN model and SSD model. As the subjects in the videos wearing the head mounted camera were moving while cooking, each training images, testing images and evaluation images had varying number of tagged objects. The total number of manual labelling performed for objects of each class in training, test and evaluation dataset is shown in Table 1.

CLASS	EGG	FORK	BOWL	P_S[1]	S_S[2]	O_B[3]	PAN	STOVE	PLATE
TRAINING	56	139	161	3	31	93	195	102	92
TEST	7	47	74	32	45	38	5	17	23
EVALUATION	201	36	59	28	45	13	44	48	10
TOTAL	**264**	**222**	**294**	**63**	**121**	**144**	**244**	**167**	**125**

Table 1. Total number of manually labelled classes; 1) Pepper_Shaker, 2) Salt_Shaker, 3) Oil_Bottle.

All the test, training and evaluation database was created by manually labelling the objects in the frames (images) using LabelImg [38] (Figure 11). LabelImg provides a ready-made program [39] that stores the labels as xml-files in PASCAL VOC format which was later used to create TFRecords (Tensorflow Record Format). The resizing of 800x608 images were not needed as the size of images were good enough for the model to train efficiently.

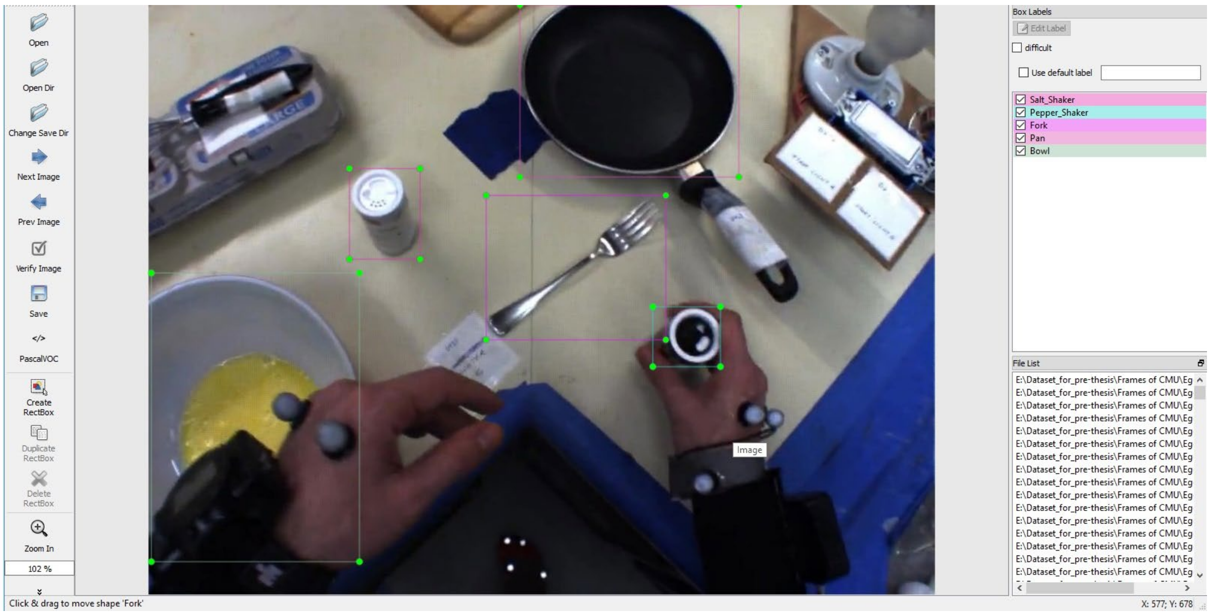


Figure 11. Labelling an image using LabelImg.

3.4 Object detection using TensorFlow Object Detection API

TensorFlow Object Detection API is “an open source framework built on top of TensorFlow” that aims to make it easy to “construct, train and deploy object detection models” [40]. It assists the users by maintaining multiple pre-trained object detection models on its GitHub page with guidance and examples on how to use and fine-tune the models for different object detection tasks [41]. The selection of the model is generally a tradeoff between accuracy and detection speed, higher the detection lower the accuracy.

For this particular study, SSD model with inception (ssd_inception_v2_coco) and Faster RCNN model with inception (faster_rcnn_inception_v2_coco) was chosen. Both the models had been trained on Microsoft COCO Dataset. The dataset contains 2.5 million labeled instances in 328,000 images, encompassing total of 91 objects types such as “Spoon”, “Bottle” or “person” [42]. The mean Average Precision (mAP) and speed of ssd_inception_v2_coco-model is reported to be of 24 and 42 respectively whereas that of faster_rcnn_inception_v2_coco-model is reported to be of 28 and 58 respectively on COCO dataset [43].

The general step by step procedure for implementation of both the models for this work is shown below in figure 12 [44] and the actual implementation on Jupyter notebook of Google colaboratory is attached to the appendix of this report.

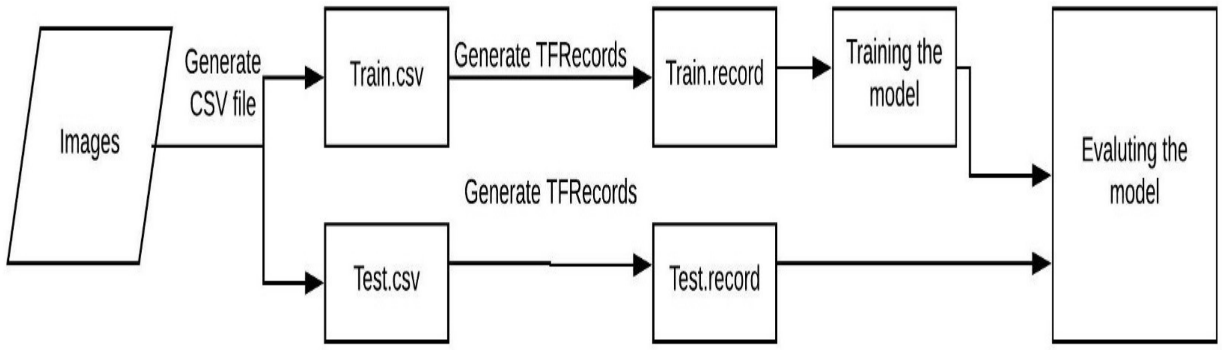


Figure 12. General flow diagram of the implementation of object detection [44].

3.4.1 Generation of CSV files

The main aim of this step is to use the information stored in .xml file about the co-ordinates of the objects in labelled images from the previous step and generate train.csv, test.csv and evaluate.csv files. CSV stands for “Comma-separated values” [45] and is basically a simple file format used to store basic information of the objects in the class and their location in the labelled images. The script used for csv file generation in this work can be found in the appendix.

3.4.2 Generation of TFRecords

TFRecords file is Tensorflow’s basic binary storage format which stores all the information about the images in binary file format as a sequence of binary strings. It helps to improve the performance of the import pipeline and hence affects the training time of the model. The main reason behind using binary data is that it consumes less space on the drive, needs less time to work with and can read and written efficiently from the storage systems. Furthermore for larger datasets, it makes easier to load the specific required data at a particular time and hence whole file system is not needed to be stored fully in memory [46]. The script used for TFRecords file generation can be found in the appendix.

3.4.3 Training the Models

Next step is to create a label map for the total class of objects which the model is to be trained to detect. For this implementation, model had to be trained on detecting total of 9 classes of objects. According to the selection of the model to be used for training, respective config file has to be downloaded from the TensorFlow model zoo hence, 'ssd_inception_v2_coco.config' and 'faster_rcnn_inception_v2_coco.config' files were downloaded for SSD and Faster RCNN models respectively. Following changes were made in both the config files:

a) 'num_classes' were set to nine.

b) 'fine_tune_checkpoint' were directed to the models.

c) In the 'train_input_reader' function, 'input_path' was directed to 'train.record' file.

d) In the 'eval_input_reader' function, 'input_path' was directed to 'test.record' file.

e) 'label_map_path' for both the above mentioned functions were directed to the 'labelmap'.

The label map and config files used in this work can be found at the appendix of this report. Once, the above steps are implemented, models can be started to train. The script used to train these models can be found at the appendix of this report.

3.4.4 Evaluation of the model

The performance of the trained models were evaluated on both trained and untrained datasets and most relevant metrics that were used to assess the models were precision, recall, F1-score and mAP.

Precision metrics can be used to describe the significance of the detection results and is calculated by following equation (Eq.1) where TP and FP signifies True Positive and False Positive respectively.

$$precision = \frac{TP}{TP+FP} \quad (1)$$

Recall metrics can be used to evaluate the percentage of relevant objects that were detected by the model and is given by the formula in following equation (Eq.2) where FN signifies False Negative.

$$recall = \frac{TP}{TP+FN} \qquad (2)$$

While dealing with precision and recall, there is a general tradeoff involved i.e. when the precision of a model is very high, recall tends to get significantly lower and vice versa. In other words, a very sensitive model would be able to detect a large number of objects in a frame but it would at the same time generate higher number of unwanted false positives. Similarly with the models which has higher threshold for detection will generate less number of false positives but on the other hand it will also not be able to detect higher percentage of objects. It is always a challenge to find the right balance between them and which mostly depends on the application. **F1-score** combines both the evaluation metrics as a single metric and is given by the following equation (Eq. 3).

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \qquad (3)$$

The TensorFlow Object Detection API uses “PASCAL VOC 2007 metrics” [43] where if Intersection over Union (IoU) (Eq. 4) of a prediction is higher than 50% only then the prediction is considered TP. If there are more than one bounding boxes predicting the object then one of them is considered as TP and rest are considered as FP whereas if an object is not predicted with any bounding box then it is considered as FN. [47]

$$IoU = \frac{area(true\ bounding\ box \cap predicted\ bounding\ box)}{area(true\ bounding\ box \cup predicted\ bounding\ box)} > 0.5 \quad (4)$$

Mean Average Precision (mAP) is the mostly preferred evaluation metrics for the performance evaluation of a model in object detection. It provides a concise value by extracting information from “precision-recall” curve. The curve is generated by arranging all predicted bounding boxes by their confidence rating and then computing precision and recall values for each predictions. Precision is given as the proportion of True Positives exceeding the given rank and recall is given as the proportion of True Positives exceeding the given rank among all user tagged objects. Average

Precision (AP) is found out from the curve created by these precision-recall pairs as “area under the precision-recall curve” [48].

Learning rate basically is a hyper-parameter that controls the degree to which weights of the network are adjusted in contrast with the loss gradient. The relationship can be shown by the formula in equation (5). If the learning rate is low, the rate of movement of the curve in the downward slope will also be low which is good as we don’t miss any local minima but it will need a long a long time to converge [49].

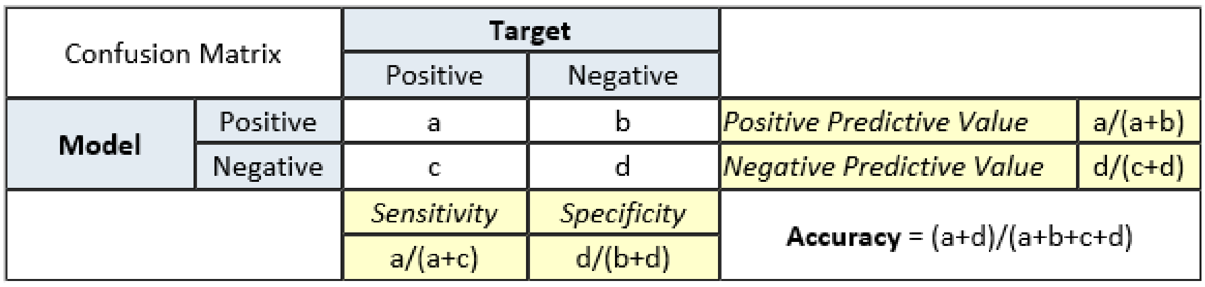
new_weight = existing_weight — learning_rate * gradient (5)

Total loss of a model during training is the addition of Localization and Classification losses where the classification losses are evaluated only for the detections with the same class label as the ground truth [50].

Accuracy of a model is the fractions of right predictions done by the mode with respect to the total predictions which is given by the following equation where TP is True positives, TN is True Negatives, FP is False Positives and FN is False Negatives (6) [51].

Accuracy = (TP + TN)/ (TP + TN + FP + FN) (6)

Confusion Matrix basically exhibits the total number of correct and incorrect predictions produced by the model in contrast to the expected outcomes in the dataset. The matrix is of *N x N* dimension where N represents the number of classes where the evaluation is done on basis of the data present in the matrix [52]. An extra last row and last column can also be added to matrix with more than two classes that will represent the class nothing which was used to represent when an object of particular class was not detected but the object was present and when the object was detected but the object was not part of the ground-truth.



Confusion Matrix		**Target**			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Table 2. A 2x2 confusion matrix [52]

4 RESULTS

The models were evaluated by using the script provided by TensorFlow Object Detection API and mAP curve was generated alongside with their Learning rate and Total loss. The script can be found in the appendix of this report. The following curves were recorded on the Tensorboard.

4.1 Faster RCNN

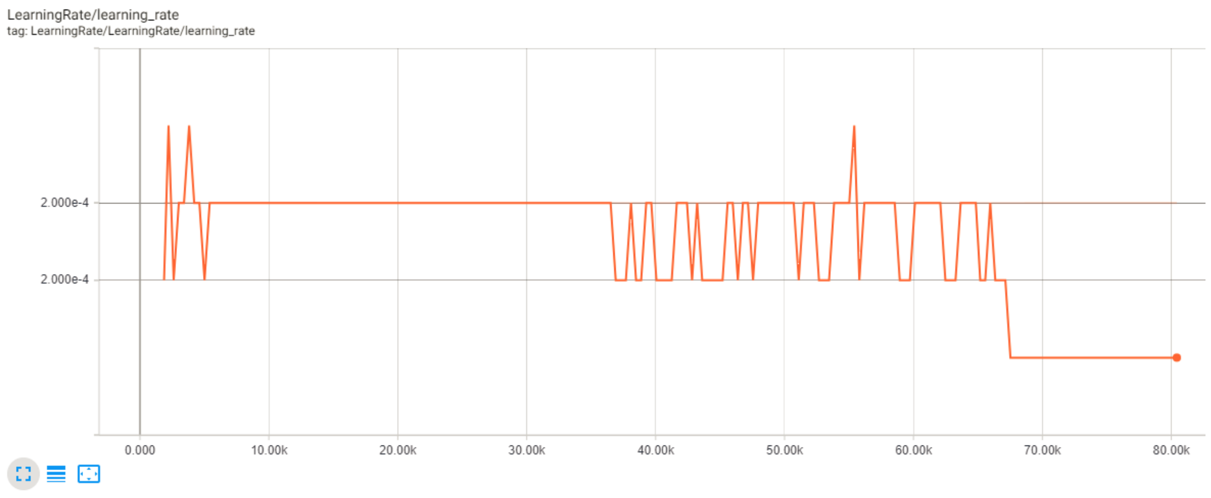


Figure 13 Learning rate of Faster RCNN model with respect to training steps.

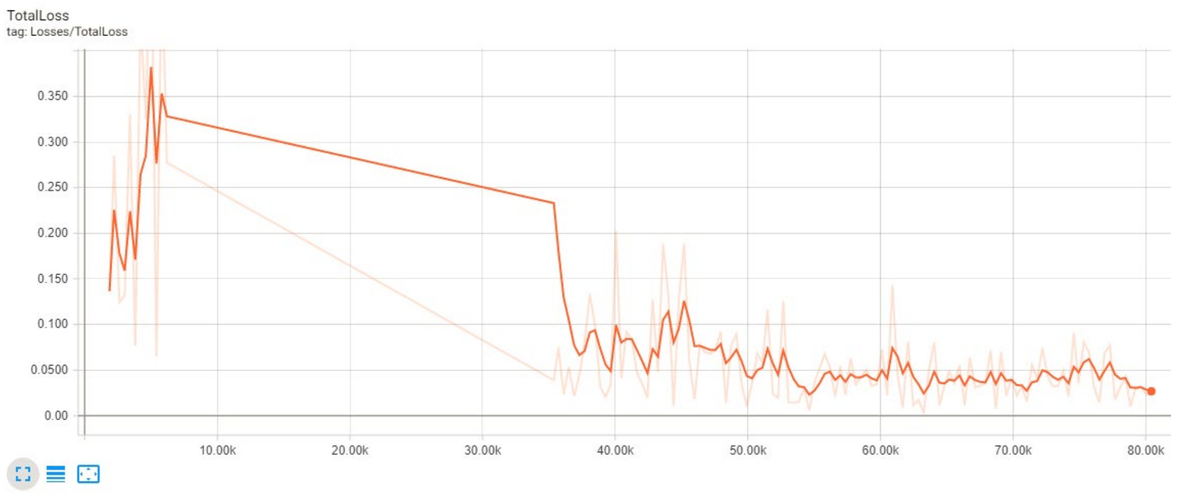


Figure 14. Total loss of Faster RCNN model with respect to training steps

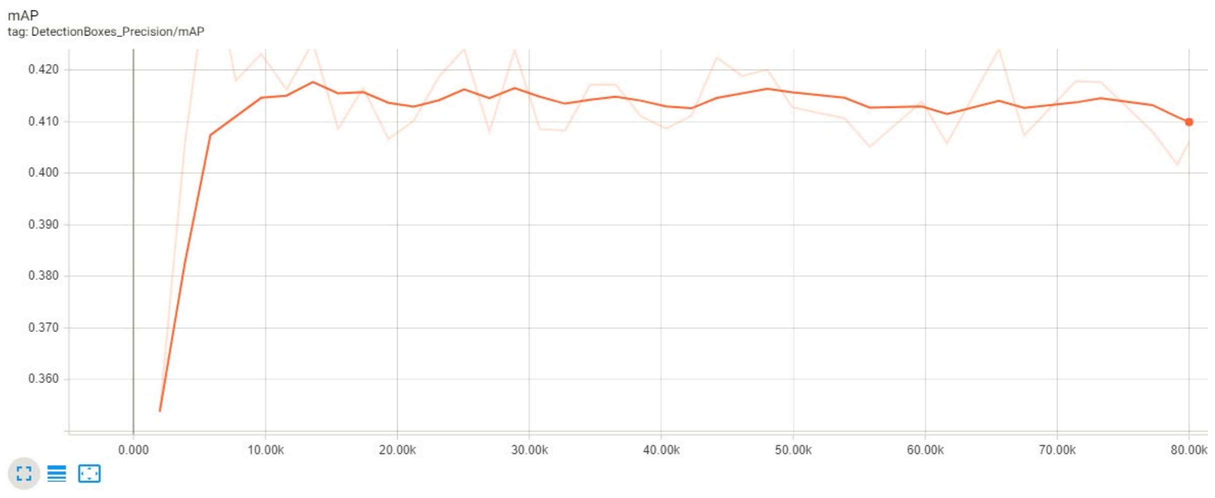


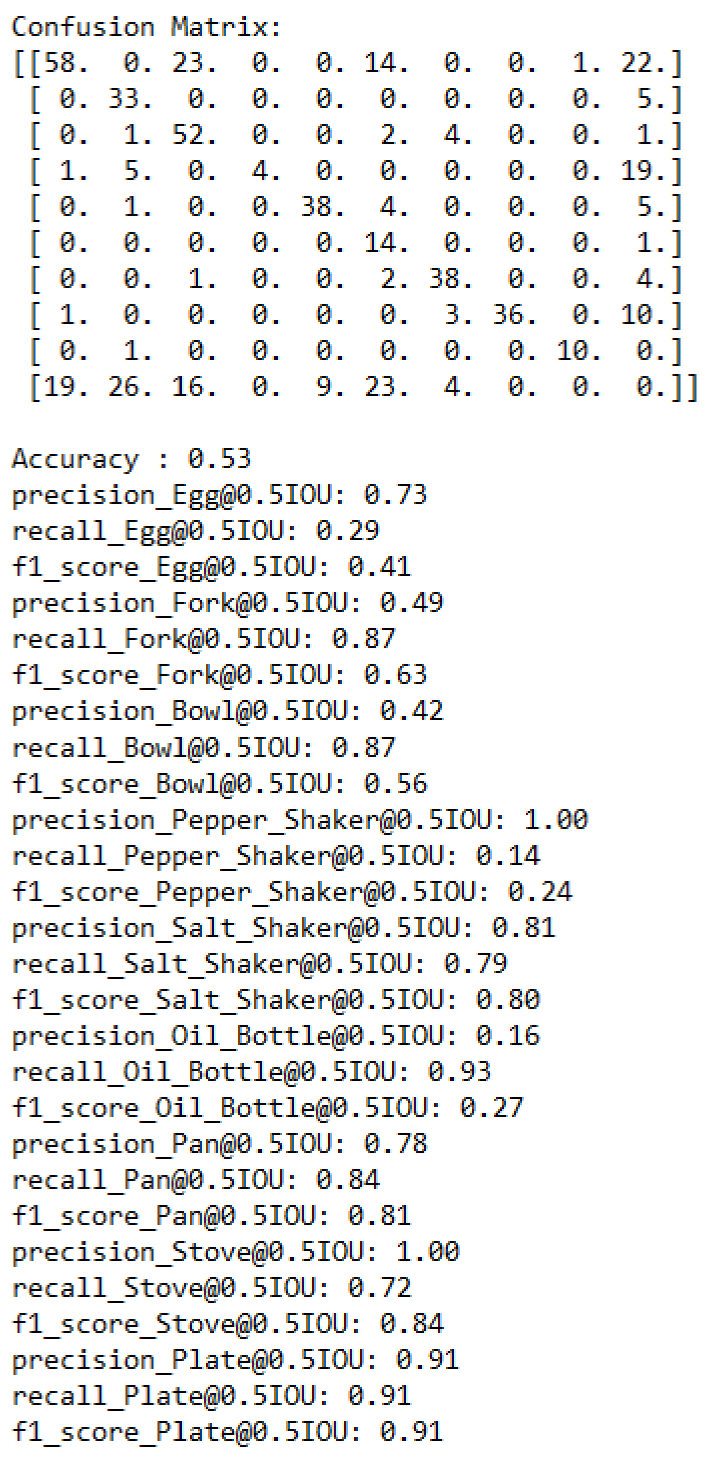
Figure 15. mAP of the Faster RCNN model with respect to training steps.

Figure 13 shows progress of the learning rate of Faster RCNN model during training which seems to be triangular in nature. Initially as the training dataset was unfamiliar for the model, the learning rate remained high but after around 65,000 steps the accuracy seems to drop stayed stable thereafter which signifies that the model has trained well enough on the dataset.

Figure 14 shows the variations total loss of Faster RCNN model during training which seems to be very high in the beginning of the training signifying that as the dataset were again unfamiliar and hence the model had very high loss during training. Later on as the model continued with the training the losses seemed to have dropped drastically and somewhat stayed stable at around 0.02.

Figure 15 shows the mAP of the changes in the model's mean average accuracy during training. At the beginning the model's mAP seems to be around 0.355 which is quite low. Later on at around 12000 steps the accuracy seems to have improved to 0.42 and stayed stable thereafter with minor variations.

The confusion matrix of the model on untrained dataset of 100 images at 0.5 IoU was also calculated alongside with accuracy, precision, recall and F1-score for each class of object which is given below. The script used for this purpose is attached in the appendix of this report.



```
Confusion Matrix:
[[58.  0. 23.  0.  0. 14.  0.  0.  1. 22.]
 [ 0. 33.  0.  0.  0.  0.  0.  0.  0.  5.]
 [ 0.  1. 52.  0.  0.  2.  4.  0.  0.  1.]
 [ 1.  5.  0.  4.  0.  0.  0.  0.  0. 19.]
 [ 0.  1.  0.  0. 38.  4.  0.  0.  0.  5.]
 [ 0.  0.  0.  0.  0. 14.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.  2. 38.  0.  0.  4.]
 [ 1.  0.  0.  0.  0.  0.  3. 36.  0. 10.]
 [ 0.  1.  0.  0.  0.  0.  0.  0. 10.  0.]
 [19. 26. 16.  0.  9. 23.  4.  0.  0.  0.]]

Accuracy : 0.53
precision_Egg@0.5IOU: 0.73
recall_Egg@0.5IOU: 0.29
f1_score_Egg@0.5IOU: 0.41
precision_Fork@0.5IOU: 0.49
recall_Fork@0.5IOU: 0.87
f1_score_Fork@0.5IOU: 0.63
precision_Bowl@0.5IOU: 0.42
recall_Bowl@0.5IOU: 0.87
f1_score_Bowl@0.5IOU: 0.56
precision_Pepper_Shaker@0.5IOU: 1.00
recall_Pepper_Shaker@0.5IOU: 0.14
f1_score_Pepper_Shaker@0.5IOU: 0.24
precision_Salt_Shaker@0.5IOU: 0.81
recall_Salt_Shaker@0.5IOU: 0.79
f1_score_Salt_Shaker@0.5IOU: 0.80
precision_Oil_Bottle@0.5IOU: 0.16
recall_Oil_Bottle@0.5IOU: 0.93
f1_score_Oil_Bottle@0.5IOU: 0.27
precision_Pan@0.5IOU: 0.78
recall_Pan@0.5IOU: 0.84
f1_score_Pan@0.5IOU: 0.81
precision_Stove@0.5IOU: 1.00
recall_Stove@0.5IOU: 0.72
f1_score_Stove@0.5IOU: 0.84
precision_Plate@0.5IOU: 0.91
recall_Plate@0.5IOU: 0.91
f1_score_Plate@0.5IOU: 0.91
```

Figure 16. Output on untrained dataset for fine-tuned Faster RCNN model

4.2 SSD

The time required for per step training for SSD was considerably higher than Faster RCNN and the total time any model can use Google Colab’s free GPU is limited, this the reason why the model could be trained only for approximately 16000 steps.

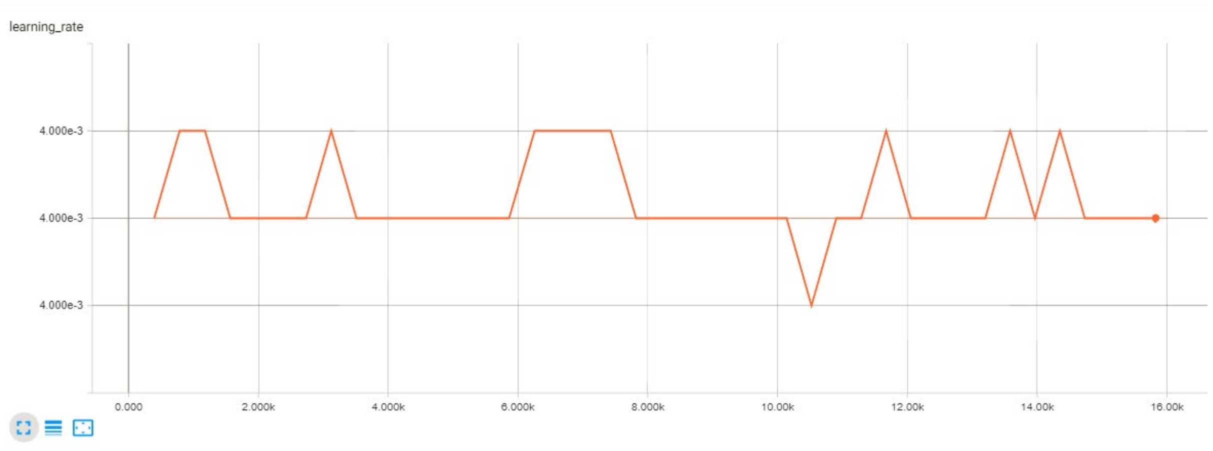


Figure 17. Learning rate of the SSD model with respect to training steps.

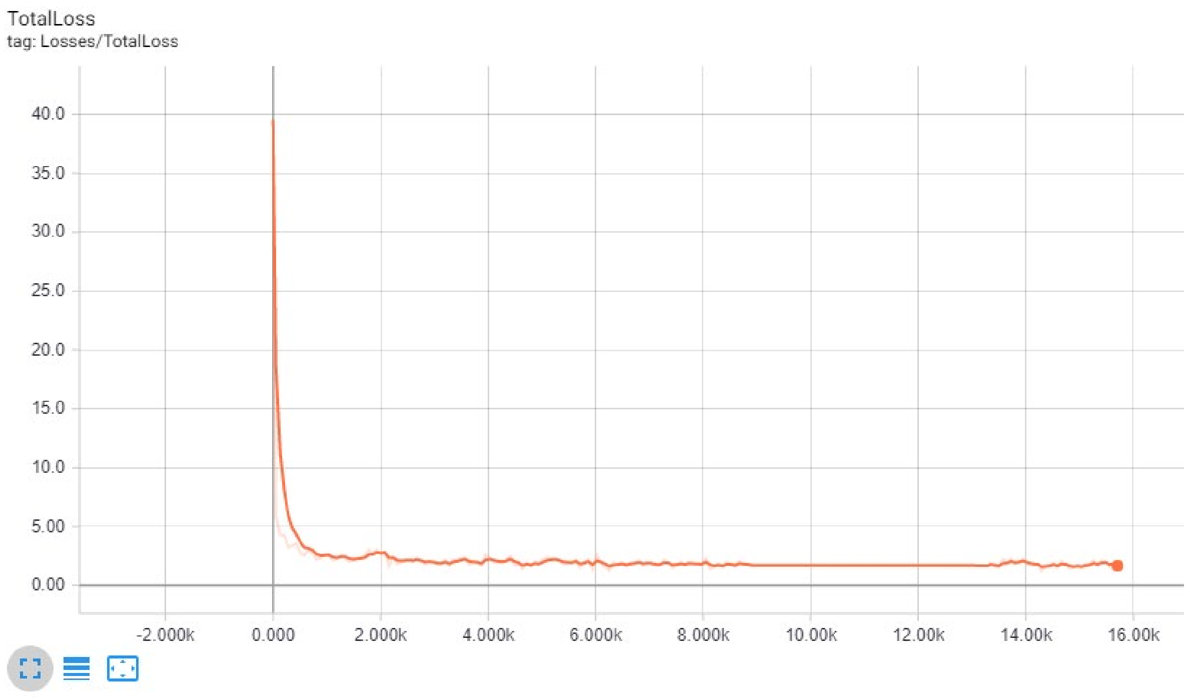


Figure 18. Total loss of the SSD model with respect to training steps.

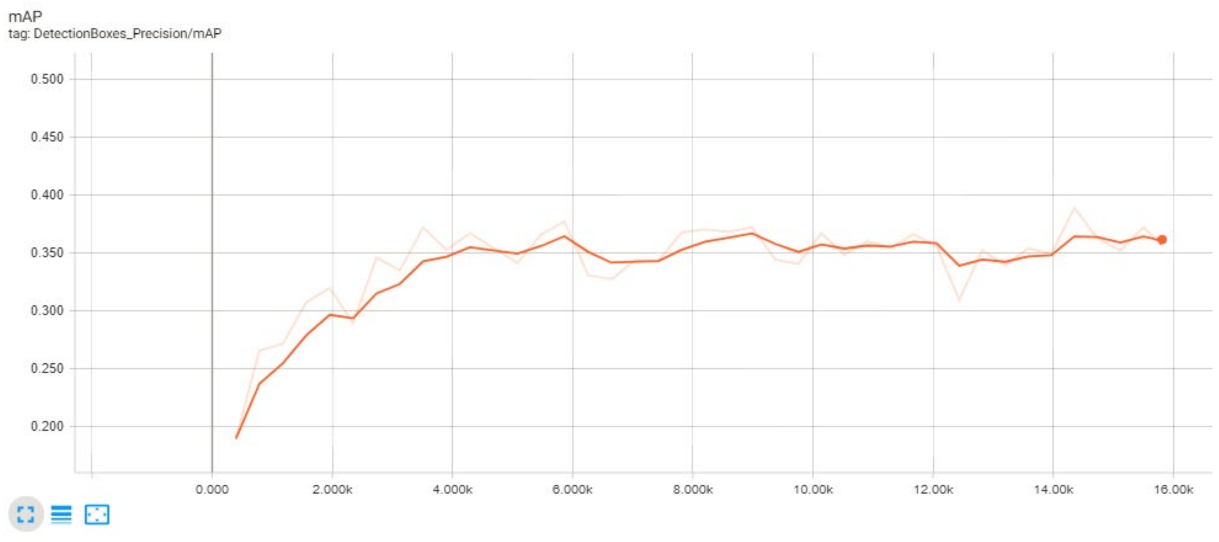


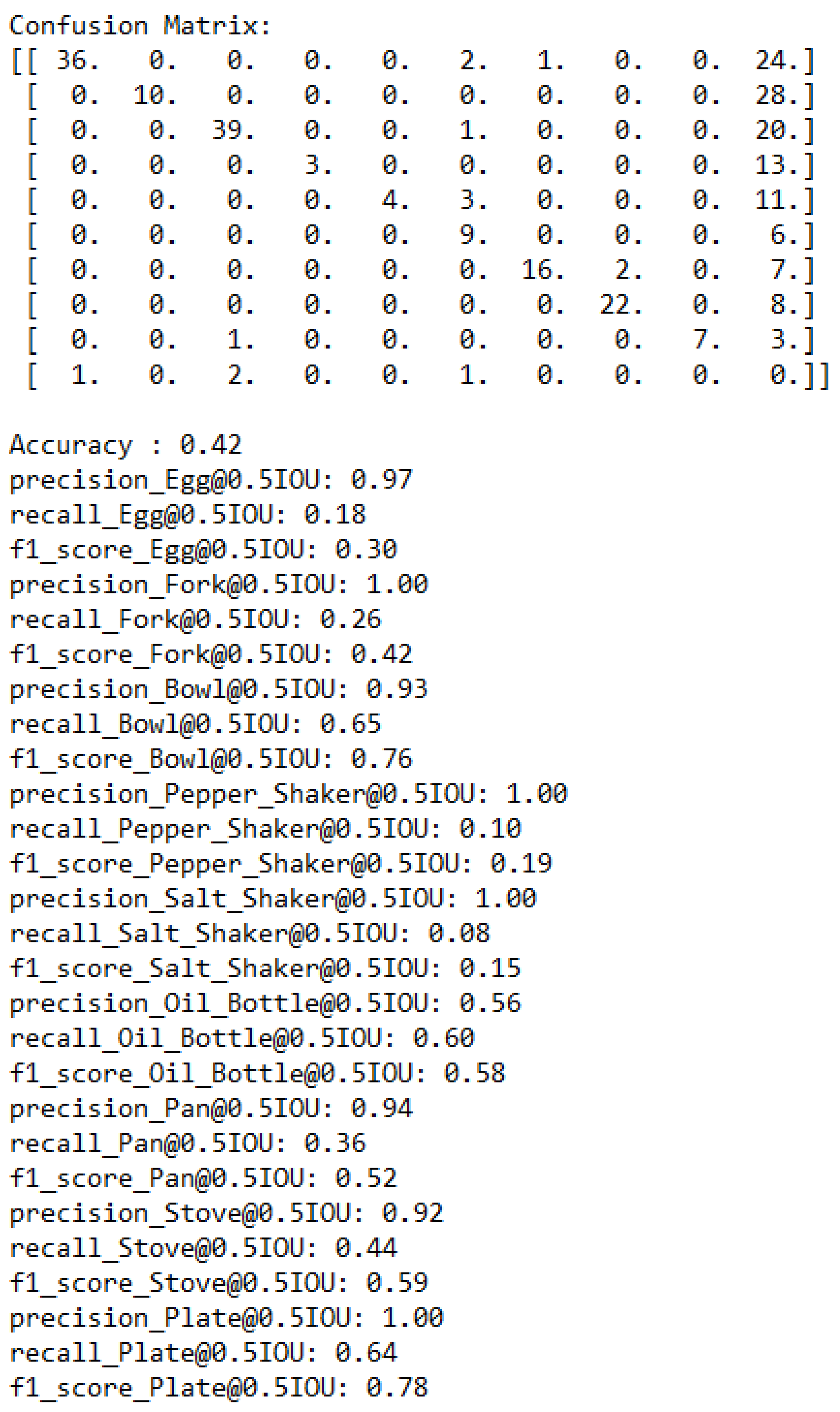
Figure 19. mAP of the SSD model with respect to training steps.

Figure 16 shows the learning rate of the SSD model during training which seems to be triangular in nature where the learning rates are being restarted every few iterations.

Figure 17 shows the total loss of the model during training which seems to have reduced exponentially over the training steps. Loss seems to have stabilized at around 1000 steps and stayed stable there on.

Figure 18 shows the mean average precision of the model with respect to the training steps and it seems to have increased significantly up to 4000 steps and then stayed stable thereon at around a rough value of 0.352.

The confusion matrix of the model on untrained dataset of 100 images at 0.5 IoU was also calculated alongside with accuracy, precision, recall and F1-score for each class of object which is shown below.



```
Confusion Matrix:
[[ 36.   0.   0.   0.   0.   2.   1.   0.   0.  24.]
 [  0.  10.   0.   0.   0.   0.   0.   0.   0.  28.]
 [  0.   0.  39.   0.   0.   1.   0.   0.   0.  20.]
 [  0.   0.   0.   3.   0.   0.   0.   0.   0.  13.]
 [  0.   0.   0.   0.   4.   3.   0.   0.   0.  11.]
 [  0.   0.   0.   0.   0.   9.   0.   0.   0.   6.]
 [  0.   0.   0.   0.   0.   0.  16.   2.   0.   7.]
 [  0.   0.   0.   0.   0.   0.   0.  22.   0.   8.]
 [  0.   0.   1.   0.   0.   0.   0.   0.   7.   3.]
 [  1.   0.   2.   0.   0.   1.   0.   0.   0.   0.]]

Accuracy : 0.42
precision_Egg@0.5IOU: 0.97
recall_Egg@0.5IOU: 0.18
f1_score_Egg@0.5IOU: 0.30
precision_Fork@0.5IOU: 1.00
recall_Fork@0.5IOU: 0.26
f1_score_Fork@0.5IOU: 0.42
precision_Bowl@0.5IOU: 0.93
recall_Bowl@0.5IOU: 0.65
f1_score_Bowl@0.5IOU: 0.76
precision_Pepper_Shaker@0.5IOU: 1.00
recall_Pepper_Shaker@0.5IOU: 0.10
f1_score_Pepper_Shaker@0.5IOU: 0.19
precision_Salt_Shaker@0.5IOU: 1.00
recall_Salt_Shaker@0.5IOU: 0.08
f1_score_Salt_Shaker@0.5IOU: 0.15
precision_Oil_Bottle@0.5IOU: 0.56
recall_Oil_Bottle@0.5IOU: 0.60
f1_score_Oil_Bottle@0.5IOU: 0.58
precision_Pan@0.5IOU: 0.94
recall_Pan@0.5IOU: 0.36
f1_score_Pan@0.5IOU: 0.52
precision_Stove@0.5IOU: 0.92
recall_Stove@0.5IOU: 0.44
f1_score_Stove@0.5IOU: 0.59
precision_Plate@0.5IOU: 1.00
recall_Plate@0.5IOU: 0.64
f1_score_Plate@0.5IOU: 0.78
```

Figure 20. Output on untrained dataset for fine-tuned SSD model

5 CONCLUSION

Automatic object detection is immensely important aspect in the process of video-based intention recognition as there is a need to detect objects in the video to be able to recognize the intentions of subjects in the video. Many different methods which can be used for automatic object detection were explored in this work with their pros and cons. The pre-trained SSD inception v2 model and Faster RCNN inception v2 model from TensorFlow Object Detection API were selected to implement this work. The models were fine-tuned on the video frames from the CMU Grand Challenge Data Collection which improved the models mean Average Precision considerably. With the availability of pre-trained models in TensorFlow alongside better library management and computational graph visualization using Tensorboard it has been really favorable to work with the models. And with free GPU provided by Google Colaboratory, it has become relatively labor-saving to train and test different object detection models on various kind of datasets.

However, the increase in the mAP of the models leads to the reduction in the speed of execution of each steps which is not beneficial for real-time intention recognition. Tensorflow has its drawbacks like no support for windows and lower computational speed which again can be a troublesome for video based real-time intention recognition. Google colaboratory uses google drive as its sole source and target of storage which always shows storage full if training continued for long time which was one of the problem faced during training SSD model and it would be required to buy more storage if the models has to work on longer on bigger datasets.

Hence, it can be concluded that for the purpose of Object Detection for video-based intention recognition both the models have performed sufficiently well to be considered for the purpose. Tensorflow with its well-built libraries and support for Object Detection API alongside with free GPU from Google colaboratory, it can be considered the right choice for the implementation of object detection for video-based intention recognition.

REFERENCES

[1] D. Erhan, C. Szegedy, A. Toshev et al., "Scalable object detection using deep neural networks", 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2155-2162, 2014.

[2] Borji, M.M. Cheng, H. Jiang et al., "Salient object detection: A benchmark", IEEE Transactions on Image Processing, vol. 24, pp. 5706-5722, Dec 2015.

[3] Y. Tian, P. Luo, X. Wang et al., "Deep learning strong parts for pedestrian detection", 2015 IEEE International Conference on Computer Vision, pp. 1904-1912, 2015.

[4] P. Ahmadvand, R. Ebrahimpour, P. Ahmadvand, "How popular CNNs perform in real applications of face recognition", 2016 24th Telecommunications Forum (TELFOR), pp. 1-4, 2016.

[5] W. Ouyang, X. Wang, X. Zeng et al., "Deepid-net: Deformable deep convolutional neural networks for object detection", 2015 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2403-2412, 2015.

[6] P.M. Merlin, D.J. Farber, "A parallel mechanism for detecting curves in pictures", IEEE Transactions on Computers, vol. C-24, pp. 96-98, Jan 1975.

[7] N. Singla, "Motion detection based on frame difference method", International Journal of Information & Computation Technology, vol. 4, no. 15, pp. 1559-1565, 2014.

[8] D.S. Lee, "Effective Gaussian mixture learning for video background subtraction", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pp. 827-832, May 2005.

[9] B.K.P. Horn, B.G. Schunck, "Determining optical flow", Artificial intelligence, vol. 17, pp. 185-203, 1981.

[10] J.L. Barron, D.J. Fleet, S.S. Beauchemin et al., "Performance of optical flow techniques", 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 236-242, 1992.

[11] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. I-511-I-518, 2001.

[12] P.F. Felzenszwalb, R.B. Girshick, D. McAllester et al., "Object detection with discriminatively trained part-based models", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, pp. 1627-1645, 2010.

[13] P. Felzenszwalb, D. McAllester, D. Ramanan, "A discriminatively trained multiscale deformable part model", 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8, 2008.

[14] M. Ulrich, C. Steger, A. Baumgartne, "Real-time object recognition using a modified generalized Hough transform", Pattern Recognition, vol. 36, pp. 2557-2570, Nov. 2003.

[15] J. Xu, X. Sun, D. Zhang et al., "Automatic detection of inshore ships in high-resolution remote sensing images using robust invariant generalized Hough transform", IEEE Geoscience and Remote Sensing Letters, vol. 11, pp. 2070-2074, Dec. 2014.

[16] B. Leibe, A. Leonardis, B. Schiele, "Robust object detection with interleaved categorization and segmentation", International Journal of Computer Vision, vol. 77, pp. 259-289, 2008.

[17] S. Maji, J. Malik, "Object detection using a max-margin hough transform", 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1038-1045, 2009.

[18] A. Lehmann, B. Leibe, L. Van Gool, "Fast prism: Branch and bound hough transform for object class detection", International Journal of Computer Vision, vol. 94, pp. 175-197, Feb. 2011.

[19] K. Dai, G. Li, D. Tu et al., "Prospects and current studies on background subtraction techniques for moving objects detection from surveillance video", Journal of Image and Graphics, vol. 11, pp. 919-927, July 2006.

[20] Q. Ji, S. Yu, "Object detection algorithm based on surendra background subtraction and four-frame difference", Computer Applications and Software, vol. 31, pp. 242-244, Dec. 2014.

[21] C. Stauffer, W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 246-252, 1999.

[22] M. Heikkila, M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects", IEEE Transactions on Pattern Aanalysis and Machine Intelligence, vol. 28, pp. 657-662, April 2006.

[23] C. Szegedy, A. Toshev, D. Erhan, "Deep neural networks for object detection", Advances in Neural Information Processing Systems, pp. 2553-2561, 2013.

[24] P. Sermanet, D. Eigen, X. Zhang et al., "Overfeat: Integrated recognition localization and detection using convolutional networks", ICLR, 2014.

[25] R. Girshick, J. Donahue, T. Darrell et al., "Rich feature hierarchies for accurate object detection and semantic segmentation", 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580-587, 2014.

[26] J.R.R. Uijlings, K.E.A. Van De Sande, T. Gevers et al., "Selective search for object recognition", International Journal of Computer Vision, vol. 104, pp. 154-171, Feb. 2013.

[27] K. He, X. Zhang, S. Ren et al., "Spatial pyramid pooling in deep convolutional networks for visual recognition", European Conference on Computer Vision, pp. 346-361, 2014.

[28] R. Girshick, "Fast r-cnn", 2015 IEEE International Conference on Computer Vision, pp. 1440-1448, 2015.

[29] S. Ren, K. He, R. Girshick et al., "Faster r-cnn: Towards real-time object detection with region proposal networks", Advances in Neural Information Processing Systems, pp. 91-99, 2015.

[30] Y. Li, K. He, J. Sun, "R-FCN: Object detection via region-based fully convolutional networks", Advances in Neural Information Processing Systems, pp. 379-387, 2016.

[31] J. Redmon, S. Divvala, R. Girshick et al., "You only look once: Unified real-time object detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, 2016.

[32] W. Liu, D. Anguelov, D. Erhan et al., "SSD: Single shot multibox detector", European Conference on Computer Vision, pp. 21-37, 2016.

[33] A. Yali, F. Pedro et al., “Object Detection”, University of Chicago.

[34] C. Tang, Y. Feng, X. Yang, C. Zheng and Y. Zhou, "The Object Detection Based on Deep Learning," 2017 4th International Conference on Information Science and Control Engineering (ICISCE), Changsha, 2017, pp. 723-728.

[35] Kitchen.cs.cmu.edu. (2019). Quality of Life Grand Challenge | Kitchen Capture. [online] Available at: http://kitchen.cs.cmu.edu/main.php [Accessed 27 Jan. 2019].

[36] TensorFlow. (2019). TensorFlow. [online] Available at: https://www.tensorflow.org/ [Accessed 27 Jan. 2019].

[37] Colab.research.google.com. (2019). Google Colaboratory. [online] Available at: https://colab.research.google.com/notebooks/welcome.ipynb [Accessed 27 Jan. 2019].

[38] GitHub. (2019). tzutalin/labelImg. [online] Available at: https://github.com/tzutalin/labelImg [Accessed 27 Jan. 2019].

[39] GitHub. (2019). tensorflow/models. [online] Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/dataset_tools/create_pascal_tf_record.py [Accessed 27 Jan. 2019].

[40] GitHub. (2019). tensorflow/models. [online] Available at: https://github.com/tensorflow/models/tree/master/research/object_detection [Accessed 29 Jan. 2019].

[41] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,Song Y & Guadarrama S (2016) Speed/accuracy trade-offs for modern convolutional object detectors. arXiv preprint arXiv:1611.10012 .

[42] Lin T, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P & ZitnickCL (2014) Microsoft coco: Common objects in context. European conference on computer vision. , Springer: 740-755.

[43] GitHub. (2019). tensorflow/models. [online] Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md [Accessed 29 Jan. 2019].

[44] Becoming Human: Artificial Intelligence Magazine. (2019). TensorFlow Object Detection API tutorial — Training and Evaluating Custom Object Detector. [online] Available at: https://becominghuman.ai/tensorflow-object-detection-api-tutorial-training-and-evaluating-custom-object-detector-ed2594afcf73?fbclid=IwAR0fOnt_LCsz4XMKv2lA7WNyZGZeyGwtGH1s5Ap0eciolsvZ6mQA3QC7qjk [Accessed 29 Jan. 2019].

[45] Help, S., Help, F. and Hope, C. (2019). How to create a CSV file. [online] Computerhope.com. Available at: https://www.computerhope.com/issues/ch001356.htm [Accessed 29 Jan. 2019].

[46] Medium. (2019). Tensorflow Records? What they are and how to use them. [online] Available at: https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564 [Accessed 29 Jan. 2019].

[47] Everingham M, Van Gool L, Williams CK, Winn J & Zisserman A (2010) The pascal visual object classes (voc) challenge. International journal of computer vision 88(2): 303-338.

[48] McCann, S. (2019). Average precision. [online] Sanchom.wordpress.com. Available at: https://sanchom.wordpress.com/tag/average-precision/ [Accessed 30 Jan. 2019].

[49] Towards Data Science. (2019). Understanding Learning Rates and How It Improves Performance in Deep Learning. [online] Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 [Accessed 30 Jan. 2019].

[50] Becoming Human: Artificial Intelligence Magazine. (2019). TensorFlow Object Detection API: basics of detection (1/2). [online] Available at: https://becominghuman.ai/tensorflow-object-detection-api-basics-of-detection-7b134d689c75 [Accessed 30 Jan. 2019].

[51] Google Developers. (2019). Classification: Accuracy | Machine Learning Crash Course | Google Developers. [online] Available at: https://developers.google.com/machine-learning/crash-course/classification/accuracy [Accessed 30 Jan. 2019].

[52] Saedsayad.com. (2019). Model Evaluation. [online] Available at: https://www.saedsayad.com/model_evaluation_c.htm [Accessed 30 Jan. 2019].

[53] GitHub. (2019). EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10. [online] Available at: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10 [Accessed 30 Feb. 2019].

APPENDICES