Smart Surveillance System for Identifying Bikers Without Helmets Using Deep Learning

by

MD. Iqbal Hossain
15301060
Raghib Barkat Muhib
15301058

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
August 2019

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

MD. Iqbal Hossain	Raghib Barkat Muhib
15301060	15301058

Approval

The thesis/project titled "Smart Surveillance System for Identifying Bikers Without Helmets Using Deep Learning" submitted by

1. MD. Iqbal Hossain (15301060)
2. Raghib Barkat Muhib (15301058)

Of Summer, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on August 28, 2019.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Jia Uddin
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Mahbubul Alam Majumdar
Professor and Chairman
Department of Computer Science and Engineering
Brac University

Abstract

Modern world is progressing quickly along with technology and one of the major sectors is transportation technology. Day by day the number of people are increasing and the number of vehicles are increasing too. As a result we have easy access to vehicles these days but at the same time it has increased the number of road accidents. Among the various types of road accidents, motorcycle accident is one of the main type which causes severe injuries and in some cases death. The only protection motorcyclists use is their helmet. Most countries has a law on wearing helmet otherwise it will be punished, but many people often break this law and as a result it increases the percentage of severe injury and death. In a populated country it's hard to keep track of the bikers who don't use helmets because of the huge number of bikes moving at a time. In this circumstances, we have developed a solution which can identify the bikers who don't use helmets. Our system uses image processing and deep Convolutional Neural Networks(CNNs) which is used to identify who breaks the law of wearing helmet, with helmet vs without helmet identification and finally motorcycle license plate recognition. We have used multiple models of object identification and evaluated in terms of speed vs accuracy. The results we have got indicates that due to the increase of law enforcement and awareness of the traffic police, the use of helmet has decreased a bit than before but there are still a lot of bikers who don't use helmets on regular basis. As we are going to detect people without helmet and then will keep the registration number from the license plate, we can easily identify the law breakers and the government can punish them accordingly. We have used the Tensorflow library for our system. The models we have used in our system are SSD Mobilenet v2 and Faster RCNN inception v2. For SSD Mobilenet V2 the accuracy for the helmet was 90 percent, human was 55 percent, bike was 80 percent and number plate was 95 percent. This model is quite light and we can use this trained model even in mobile devices. For Faster RCNN inception v2 it took more computations but the accuracy we got was slightly better as it is more heavyweight than SSD Mobilenet. The accuracy we got for this was 92 percent for the helmet, 58 percent for the human, 81 percent for bikes and 96 percent for number plates.

Keywords: Surveillance System; Deep learning; Convolutional Neural Networks(CNNs); Helmet; License Plate; Tensorflow; Tesseract; Object Detection

Acknowledgement

Firstly, all praise to the Great Almighty for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Dr. Amitabha Chakrabarty sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AI Artificial Intelligence

API Application Programming Interface

$CCTV$ Closed Circuit Television

CNN Convolutional Neural Network

GPU Graphics Processing Unit

$RCNN$ RegionConvolutional Neural Network

$ReLU$ Rectified Linear Unit

SSD Single Shot Multibox Detector

Chapter 1

Introduction

1.1 Problem Statement

Road safety has been one of the biggest issues throughout the World recently. Among the leading causes of unnatural deaths of present time, road accidents are leading the way. More precisely, bike accidents cause most of the fatal injuries and deaths throughout the World. According to Bangladesh Road Transport Authority (BRTA), the number of motorcycles saw an approximately threefold increase – from 7 lakhs to 22 lakhs – in two years (2016 to 2018). The number of deaths in road crashes was 4,144 in 2016, 5,645 in 2017, and 3,414 in 2018 (until September), according to the survey. Among the deaths in 2017, the survey said, 27 percent were motorcycle riders. Among the 15 deaths caused by road accidents in Dhaka city in the last two months, seven were motorcycle riders[1]. According to another statistics, among all the road accidents, bike accidents accounted for 34 percent in Western Pacific and Southeast Asian countries, 20 percent in the United States and 9 percent in Europe. Our proposed system aims at providing a solution to decrease the number of road accidents caused by riding motorcycles without taking proper safety measures like wearing helmet. Since the revolutionary ride sharing application Pathao was made, the usage of bike is increasing rapidly day by day. Also Bangladesh has some serious traffic jam problem. Bikes can play a vital role in transportation system of Bangladesh as it can take people to places in least possible time whatever the situation is. People are also comfortable riding a bike rather than using a public transport for which you have to fight for a spot or wait a long time before you get one. The only problem with motorbikes is that if you don't take proper safety measures like wearing a helmet, there's always a chance that you can face road accident anytime and can damage yourself severely and sometimes you get killed. Besides, a lot of Pathao riders don't provide extra helmet for the passenger. As a result, passengers are always at risk of getting injured. There are some people who don't use helmet at all even when they ride the bike alone. The government has made laws on wearing helmet. Wearing helmet is a must in our country Bangladesh or you get fined or punished for breaking the law. As our country has a huge population and the number of bikers at present is around 30 lakhs, it's easy to imagine that even if you try to enforce the law properly, you'll always miss a lot of law breakers if you try to stop them on spot.

1.2 Aim of study

Motorcycle accidents are the major part of our total road accidents right now in our country. The number of motorcycles is also increasing everyday as people use it earn money by sharing ride and some use it to save money and time. The only problem is the ride safety. If we can ensure that then we will be moving towards a better future with a safe environment. If we go through the newspapers we can see the news of bike accidents almost every day. Sometimes people get injured severely and sometimes they die. Our goal is to ensure that people are taking proper safety measures so that we don't have to lose another life because of road accidents. The law of wearing helmet has been made long time ago, but a lot of people disobey this law. Our main objective will be making those people vulnerable and accountable to that law. So that every time they go out with their bike they use a helmet must. We will also want the Pathao riders to give a mandatory helmet to the passengers they are sharing rides with. We are trying to implement our system in a way which can solve a major local issue. We can't stop people from taking their bike out recklessly but we can at least make sure that they will wear a helmet every time they go out.

1.3 Research Methodology

In this situation, our proposed system can play a vital role in identifying the bikers who break the law. Our system will keep surveillance by using the CCTV footage. It will analyze the video files taken by CCTV cameras. After some image processing methods it will identify the bikers with helmet and without helmet from the extracted frames from videos. If it detects any bikers without helmet then instantly it will detect the license plate of the motorcycle and will process further to note down the registration number from the plate. As we have a digital license plate system we can then easily get information of the biker from the registration number. This will ensure that everyone stays aware that they will be identified and later will be given punishment or fined. By identifying the person who broke the law the government can further decide whether to arrest the suspect's vehicle, impose helmet violation fines. System implementation will involve the most updated deep learning algorithms which require minimum computation cost. We will be keeping a database which will keep the records of the law offenders which includes images of motorcycles, license plate and the registration number. The system involves only a fixed view at the moment. But the CCTV in general rotates every second. In that case it can decrease the accuracy a bit. So if we can cameras from multiple angles then we can get maximum accuracy. We can identify the law breakers more accurately. Initially, our plan is to use the Tensorflow library for our object detection purpose and we will be using SSD Mobilenet V2 and Faster RCNN Inception V2 models. For number plate recognition we will be using Tessearact library.

1.4 Thesis Outline

We are going to use videos taken by us for our system initially. Overall we have divided the whole system in few parts which are data collection, annotation, verification of images, image processing, training the model so that it can identify the

object classes and number plate recognition and finally testing the system with test images.
literature review At first we have read a lot of papers with similar work. This involves image processing, number plate detection, helmet detection and comparison of algorithms.
Algorithms and models We will use the Tensorflow library for object detection and will use the models SSD Mobilenet V2 and Faster RCNN Inception v2. For number plate recognition we will use Tesseract.
System implementation After collecting data we will annotate the images. Then after proper pre processing we will train the data as long as we don't get enough accuracy.
Experiment After the training is done we will experiment our system with test images. This will detect helmet and if not found then it will go for number plate detection.

Chapter 2

Literature Review

This section includes the work already done on this system by various researchers using different methodologies and algorithms. Following is the brief description of some of them:
In paper [2] they present a system design using computer vision and deeply convolutionary neural networks (CNNs) to find bikers who violate the rules of the helmet. The mechanism includes identification of motorcycles, classification of helmets vs. no-helmet, and identification of bike registration plate. In terms of precision and speed, we assess the model. Several places in Bangkok and Phuket, Thailand have been using the program since 2016[2]. Early accounts show growing adherence with the legislation of bike helmets. Road deaths, mainly bike crashes, are one of the major triggers of abnormal deaths today. Bike crashes in Europe accounted for 9percent of all highway deaths, 20 percent in the United States, and 34percent in Western Pacific and Southeast Asian countries. Helmets are the primary motorcycle safety device, so where compliance is small, automatic handling has the ability to improve adherence significantly, thereby saving human life. Until lately, most of the techniques used to detect objects and classify objects used techniques such as haar, HOG, local binary models (LBP), scale invariant function conversion (SIFT), or speeded up robust features (SURF) to extract characteristics and then promote vector systems (SVM), random fields, or classifier AdaBoost. Silva et al. use techniques such as histograms of oriented gradient (HOG), LBP, and wavelet transform (WT) to extract features to classify helmet-free bikers. They use various core function mixtures such as HOG+LBP, HOG+WT, LBP+WT, and HOG+LBP+WT to get seven function sets. The seven characteristics are combined with six classifiers, namely SVM, RBNF, Multilayer Perceptron (MLP), Naive Bayes, Random Forest, and K-nearest Neighbors (KNN). The highest outcome was shown by the MLP classifier using HOG descriptors on a sample set with 255 pictures (151 with masks and 104 without gloves), with 91.3 percent precision. Since convolutionary neural networks (CNNs) have lately surpassed custom function-based techniques in many fields, there is proof that using CNNs may improve helmet or no-helmet classification precision[2]. All state-of - the-art techniques for classifying objects, detecting objects, classifying characters and segmenting objects are now focused on CNNs. The system was implemented at one Bangkok (Din Daeng) junction and five Phuket junctions. The Royal Thai Police has started awarding fares depending on the picture handling system performance. According to anecdotal accounts, compliance with the helmet is improving at these locations. From these studies on CNNs, we

can say that CNN depth is more important than width, and even without GPUs, the tiny designs can be efficient. While we get excellent precision for violation or non-violation ranking, mistakes are observed when a biker wears a hat, when the front rider wears a helmet and the second person is not. By enhancing the training data belonging to these courses, this problem can be fixed. That is to say, more information with caps and more information where the first cyclist wears a helmet and the second is not. The precision of latest CNN designs such as profound remaining networks or tightly linked convolution networks can be further improved. The precision can be enhanced for personality identification by raising the amount of characters from the courses with fewer instances in the current training data and using larger networks. By skipping the differentiation step, the system could be stronger as it introduces errors that the classification system cannot correct.
In the article [3] they suggest to use real-time surveillance videos to automatically detect motorcycle-riders without helmets. The suggested strategy first uses background subtraction and object differentiation to identify motorcycle drivers from monitoring video. Then it decides whether or not a motorcyclist uses a helmet and a binary classifier. They also introduce an infringement reporting consolidation strategy that allows to improve the accuracy of the suggested method.To assess our strategy, we presented a quality comparison for classification of three commonly used function depictions, namely focused histogram of oriented gradients(HOG), scale-invariant function transform (SIFT), and local binary patterns (LBP)[3]. The experimental findings indicate 93.80 percent precision of identification on information from real-world monitoring. It has also been demonstrated that the suggested strategy is less costly computationally and works with a processing moment of 11.58 m/s per image in real time.In almost every nation, two-wheeler is a very common form of travel. However, due to less security, there is a strong danger engaged. Using helmet is extremely suitable for bike riders to decrease the danger associated. Observing the effectiveness of the helmet, governments have rendered riding a bicycle without a helmet a punishable crime and have embraced manual approaches to capture the violators. However, the current techniques of video surveillance are passive and involve substantial human support. They suggest a structure to detect road law violators who drive a motorcycle without the use of a helmet in real time. The proposed structure will also help traffic officers identify such offenders in unusual environmental circumstances viz; warm sun, etc. Experimental findings indicate the precision for motorcyclist identification and offender identification of 98.88 percent and 93.80 percent respectively. The average processing period for a picture is 11.58 ms, which is appropriate to be used in real time. Also, if necessary, the suggested structure immediately adjusts with mild tuning to fresh situations. This context can be expanded to identify and record violator registration plates.
In [4] we find, one sort of intelligent transportation system (ITS) is a license plate recognition (LPR) system. It is a sort of technology where the software allows the computer system to automatically read from digital pictures the license number plate of the automobile. Automatically reading the number plate means converting the digital image pixel information into the number plate's ASCII text. This article uses mathematical morphological activities to discuss a technique for the car licence plate identification from the picture[4]. The primary goal is to use various morphological operations so that the vehicle's number plate can be correctly recognized.This is focused on multiple operations such as improving the picture, morphological con-

version, detecting the edge and removing the number plate from the picture of the vehicle. Using template matching, the letters current on the number plate are recognized after this segmentation is implemented. This algorithm can rapidly and precisely acknowledge the number plate from the picture of the vehicles.Over the previous few decades, automatic label identification scheme has been a feasible solution. The automatic number plate recognition (ANPR) is one form of intelligent transportation system (ITS) software that can differentiate each vehicle as different by acknowledging the number plate symbols.The fundamental components of an ANPR system are provided in this document including plate placement, personality differentiation and identification. The research's objective is to explore the option of creating an extensive system centered on license plate recognition for Indian car detection[4]. No extra hardware will be needed in that situation, such as antennas, installed on a vehicle, and responders. The scheme works well even on scratched, scaled plate pictures, etc., on different kinds of vehicle license plate pictures.
In [5] this work bargains with the issue from field of artificial Intelligence, Neural system Computer vision within the development of a programmed number plate detection system also known as Automatic Number Plate Recognition System (ANPR). This problem incorporates scientific standards and calculations, which guarantees different forms to carry out for the item. The securing of the picture takes put with any camera with capability of capturing picture with great quality. The accentuation of this paper is on the localization of number plate utilizing forms following procedure alongside edge location and honing of edge utilizing canny's edge detection algorithm. Additionally, the main focus on this paper is a new algorithm on artificial neural network (ANN) is utilized for acknowledgement of number plate characters[5]. This paper makes utilize of different calculation in each category from number plate location to actual detection of characters which improves the execution of the system up to the most extreme degree conceivable with less efforts and use of computational assets. The job being created is an image processing based job for the automatic recognition of the number plate from the picture of the object that enables the recognition of a distinctive number plate of the object for an effective traffic or vehicle management mechanism.The significant steps can be taken to achieve the suggested job as 1. Acquisition of images 2. Recognition of the number plate 3. Detection of the edge 4. Segmentation of characters 5. Character recognition and database matching. The research of the model proposed to date offers adequate information to determine which algorithms are best suited for execution for each ANPR model sub-part. For the first two implementation steps, the predefined functions available for pre-processing in MATLAB are to be used which reduces the computational cost to some extent which is the disadvantage of the above algorithms[5]. The Contour Tracing Algorithm which often detects the number plate from the picture gives output as number plate with different styles and sizes more efficiently and efficiently. The canny edge detection algorithm is used to detect and sharpen the corners.This algorithm alongside edge detection also enhances image quality by filtering past phase performance that is not achieved in any of the methodologies suggested. The characters are acknowledged separately and combined with the assistance of matchers and classifiers that are qualified and intelligent agents that have distinct practice sets that make them acknowledge individual personalities of distinct form and magnitude. The final stage will thus offer the digital representation of the amount of the number plate of the vehicle whose

photograph has been taken. The suggested algorithms overcome multiple drawbacks from prior articles researched to date by decreasing computational costs and offering greater precision.
There are more works which use more or less same algorithms wither for object detection or number plate detection. Another article suggested a strategy, where they first use adaptive background subtraction on video frames to get moving items. Again, for further acceptance of motorcyclists riding without a helmet, we check CNN on the upper quarter part.The efficiency of the suggested strategy is assessed on two datasets, with sparse traffic containing IIT H Helmet 1 and heavy traffic containing IIT H Helmet 2 respectively. Real video studies effectively identify 92.87 percent offenders with an average small false alarm rate of 0.5 percent and thus demonstrate the effectiveness of the strategy suggested[6]. Real video tests effectively identify some 92.87 percent violators on two real video information sets with a small false alarm rate of about 0.50 percent and thus show the effectiveness of the suggested method.Another research shows that they used the wavelet transform (WT) as the descriptive term for car ranking and the random forest as the classifier. The circular Hough transform (CHT) and the histogram of oriented gradients (HOG) descriptors have been used to obtain the picture characteristics for helmet identification, and the classification of the images is being used by the multi-layer perceptron (MLP).The studies for the classification of vehicles reached a precision level of 97.78 percent. The algorithm phase in the identification of the helmet obtained a precision level of 91.37 percent. The study results were acquired from the repository of the author[7].

Chapter 3

Algorithms and models

Our goal is to distinguish the bikers who are not wearing a helmet and when we succeed to identify one we also detect the number plate in the purpose of extracting the registration number from that. For this we used video files captured from the rear angle of moving motorcycles at different locations to begin our research. Initially, we obtained pictures from OpenCV video files. Then with LabelIMG software, which generates XML documents, we annotated the photos. Which we used for training and processed later. To detect whether the biker is wearing a helmet or not, we used the Tensorflow API, which is an open source API for object detection. We use the Tesseract API to recognize the license plate from where we can get the motorcycle's license number. Then in a database we store the registration number. Tensorflow has a lot of tools to choose from. There exists trade off of velocity vs. precision. To train our data, we will use SSD Mobilenet v2 and Faster RCNN Inception V2 models.

3.1 Tensorflow API

Regular neural networks are made up of one input, one concealed layer and one output layer, depending on shallow nets. Deep-learning networks are differentiated from those normal neural networks with more concealed layers, or so-called deeper layers. These kinds of networks are capable of finding hidden structures within unlabeled and unorganized information (i.e. pictures, audio, and writing), which comprise the majority of information. TensorFlow is among the finest deep learning libraries to execute. TensorFlow is a software library that uses information flow graphs to numerically compute mathematical expression. The graph nodes depict mathematical computations, whereas the corners portray the multilayered information arrays between them. It was developed for deep learning by Google and adapted to it. Indeed, Deep Learning is commonly used to create alternatives. TensorFlow is a free software library developed by only the Google Brain team for numerical composition and big-scale deep learning. TensorFlow combines several of the machine learning concepts and algorithms together with deep learning which is also known as neural networking and makes them helpful through a popular metaphor. It utilizes Python to provide enough framework with a useful front-end API to build apps while running high-performance C++ apps. TensorFlow can teach and operate deep neural networks for typewritten digital classification, image analysis, text embedding, recurring neural networks, machine translation sequence-to-sequence models,

natural language processing, and simulations based on PDE (partial differential equation). Greatest of everything, TensorFlow supports scale-based manufacturing forecast with the same systems used for professional development. TensorFlow enables designers to generate information flow graphs that portray how information moves through a chart or a sequence of processing nodes. Each node in the graph is a math operation, and each link or border between nodes is a multilayered information array or tensor. TensorFlow gives the software developer with all of this through the Python language. Python is easy to grasp and function with, and offers easy methods to convey how to combine high-level concepts. TensorFlow nodes and tensors are Python components, and TensorFlow apps are Python apps themselves. Nevertheless, Python does not execute the real math operations. TensorFlow's modification libraries are published as high-performance C++ binaries. Python simply guides congestion between the parts and offers the highest-level programming concepts to tie them together. TensorFlow apps can be operated on most of the appropriate targets: a local machine, a cluster in the cloud, android and apple systems, CPUs or GPUs. If you're using Google's own platform, you can operate TensorFlow for more enhancement on Google's custom TensorFlow processing unit (TPU). However, the corresponding models produced by TensorFlow can also be used to deliver impacts on most devices.

3.2 Convolutional Neural Network and Image Processing

It is essential to briefly introduce the foundation of Convolutionary Neural Networks before considering Faster R-CNN, since this model utilizes this neural network as a structure. In terms of image recognition and object tracking, Convolutional Neural Networks have been beneficial. Convolutionary Step, ReLU, Pooling, and. Fully Connected Layer. These are the four key parts of the neural network The Convolution Step takes features from the input picture and understands the picture characteristics using entry information matrices while enabling each feature's spatial connection within the picture to be cultivated-this will generate a feature map. Non-Linearity(ReLU) is then implemented in attempt to add non-linearity by exchanging all adverse pixel measurements in the function map with zero. As a consequence, the network will understand how to process entries in a non-linear manner comparable to how information is transmitted in the real universe. Pooling decreases the functionality map's dimensionality, but it holds each important data in addition to reducing over fitting by limiting the number of variables and calculations in the system. The final phase is the entry approaching the Fully Networked Layer, a multi-layer perception that uses a softmax activation feature in the output layer to guarantee that the outputs are up to 1. This is done by the softmax mechanism by taking a selectively valued scores vector and minimizing it to a value vector specifically between 0 and 1. This would generate a favorable conclusion for the identification class as one of the classes approaches a value of 1, leaving all other adverse values close to 0 to show a failure. The output reflects high-level characteristics of the input picture at this point of the process. The Fully Connected Layer's aim is to use the characteristics to classify the input picture into distinct classes depending on the training information.

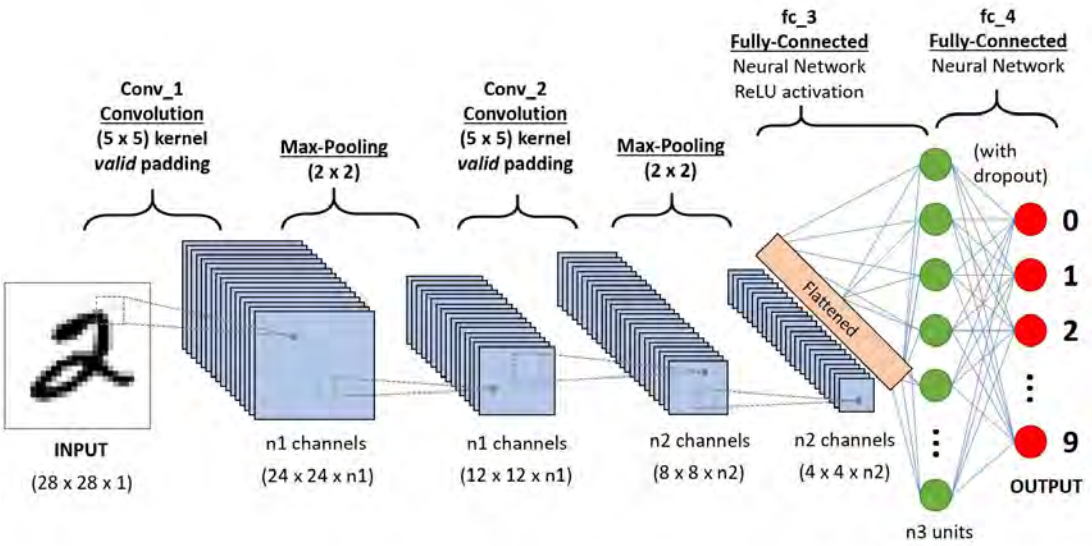


Figure 3.1: Typical feature extraction in a CNN passing through pooling layers, reaching the final neural network[8]

3.3 SSD Mobilenet V2

MobileNet V2 still utilizes profoundly separate and distinct convolutions, but now its primary construction structure looks like this: three convolutionary layers are in the block this moment.

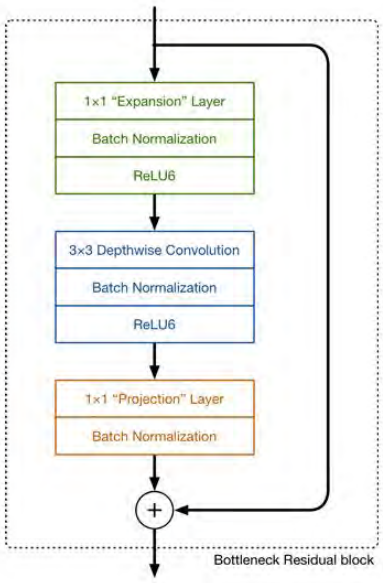


Figure 3.2: Main building block

The very last two are already known: a profound convolution that filters the inputs, followed by an 11 point-specific layer of convolution. This 11 layer has a distinct task now, though. The point specific convolution either maintained or doubled the amount of channels in V1. It does the reverse in V2: it reduces the amount of channels. This is the reason why this layer is now recognized as the reflection layer

it projects information with a large amount of dimensions (channels) into a tensor with a much smaller amount of dimensions. For instance, the deeper layer can operate on a tensor with 144 channels, which is then reduced to only 24 channels by the projection layer. This type of layer is also referred to as the bottleneck layer as it decreases the number of information flowing through the network. This was the very first layer of the fresh child on the block. (This is where the remaining unit of the bottleneck originally comes from: each block's yield is a bottleneck.) This is 11 convolution too. It aims to increase the amount of channels in the information before it gets deep-seated. Therefore this extension layer has almost always more performance than input channels the reverse of the projection layer is fairly much the case. The extension factor precisely indicates how much the information is extended. This was one of those hyper parameters for working with distinct architectures. The standard factor for expansion is 6. For instance, if a tensor runs into a block with 24 channels, the expansion layer first transforms it into a fresh tensor with 24 6= 144 channels. Next, the correct convolution of depth applies its filters to this tensor of 144 channels. And lastly, the layer of projection projects the 144 channels filtered back to a lower amount, again say 24[9].

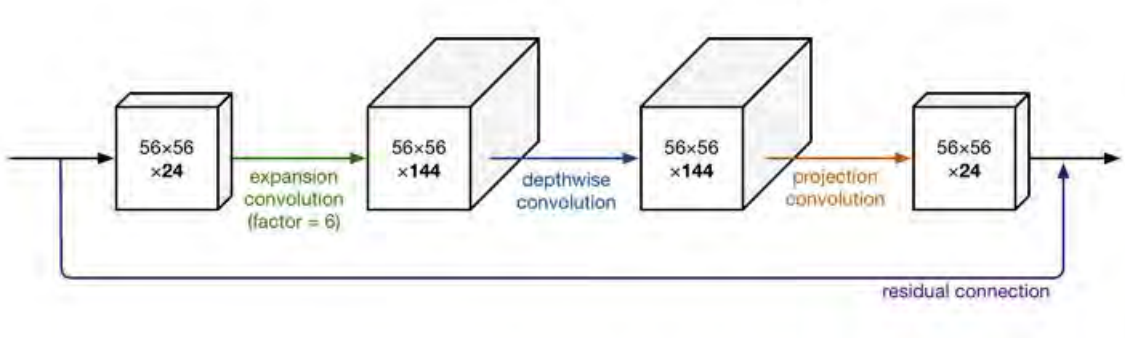


Figure 3.3: Convolution of channels

Thus the block's input and output are low-dimensional tensors, while a high-dimensional tensor is used to filter the move that occurs within the block. This operates as with ResNet and helps the network to fluctuate gradients. (The residual link only works if the number of channels entering the block is just like the number of platforms coming out from it, which may not be the case as all the blocks on the output channels are increased.) However, there is no activation function applied to the projection layer output. Because this layer generates small dimensional data, the writers discovered helpful information was effectively destroyed by a non-linearity following this layer. The whole architecture of MobileNet V2 therefore comprises of 17 blocks in a row. This comes with an eleven-fold regular, an average worldwide pooling layer and a classified layer. The use of small tensor is a key to reduce the amount of computations. The lower the tensor, the fewer multipliers the convolutional layers must. (Small detail: the initial block is slightly distinct, utilizes a periodic 33-channel convolution with 32 channels instead of a boost layer.) However, it doesn't operate quite well with only short-dimensional tensors. We prefer to use big tensors to filter the information. The block layout of MobileNet V2 provides the worldsfinest. Consider the low-dimensional information flowing between both the blocks as a compressed version of the actual information. We first must un-compress it in order to run filters across this information. This occurs within every block

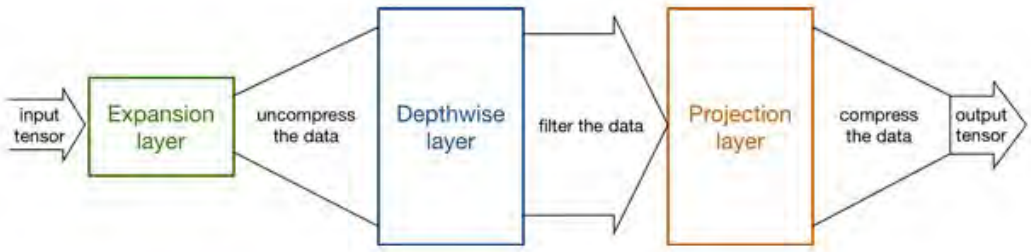


Figure 3.4: Uncompress of layers

The extension layer acts as an expression (like unzip), which first restores information to their complete form, then conducts any significant processing at this point in the system and lastly condenses the information to create it tiny again. The layer extension is like unzip. All this is done, of course, by means of convolutionary layers with learnable parameters, so that the model can learn how to best decompress the data in each phase of the network. SSD is intended to run on almost nothing, including MobileNet, independently of the base network. MobileNet+SSD utilizes even better the version called SSDLite that is used to use depth-specific layers rather than periodic convolutions in the object detection part of the network.

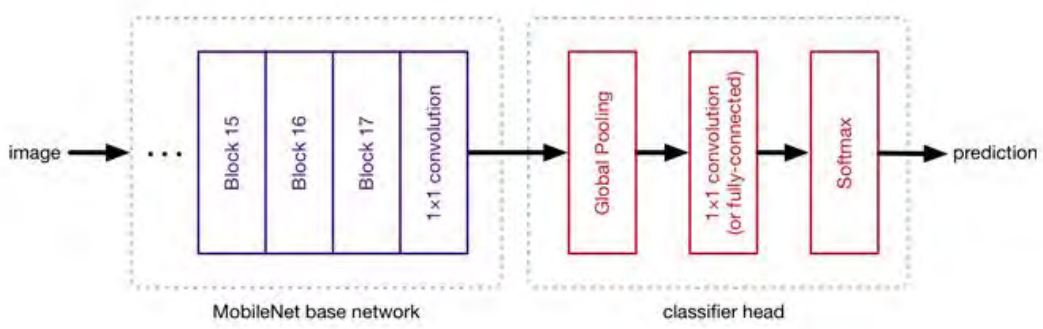


Figure 3.5: SSD classification phase

The base network output is usually an image of 77 pixels. First, the classifier utilizes a worldwide pooling layer to decrease the dimensions from 77 to 11 pixels basically a group of 49 predictors followed by a softmax and a classification layer. The last layers will look like that instead of using SSDLite with MobileNet.
The output from the last base network layer is not only received, but also the output from several prior layers and these inputs are fed into the SSD layers. The task of the layers in MobileNet is to transform pixels from the image input into functions that define the material of the photo and to transfer them towards the other layers. MobileNet is therefore have been using here as a 2nd neural network function extractor. In classification, we would like to learn about the characteristics which describe high-level ideas like "there is a face" and "there is fur," which can then be used by the classifier layer to draw a conclusion. This picture includes a cat. When we use SSD to detect objects, we want not only to understand those high-level but also low-level characteristics, which is why we have read them from earlier

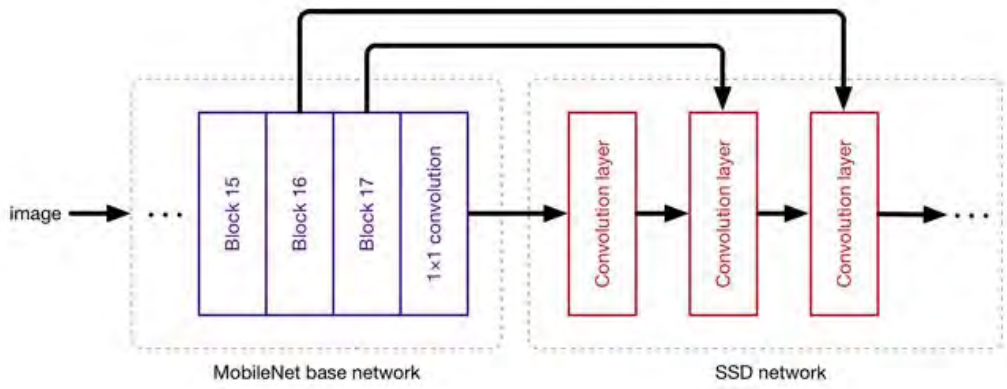


Figure 3.6: SSDlite classification phase

levels. SSD provides a lot of more convolutionary layers on top of the foundation network because object detecting is more complex than classification. Therefore, it is essential to have a quick function extractor, which is precisely what MobileNet V2 is.

3.4 Faster RCNN

The Faster R-CNN uses the CNN measured and the Region Proposition network to half use the template based characteristics gathered to identify the border rows that hold theobject(s) of concern by acquiring bounding boxes, marks allocated to rows and probabilities, (objectivity rating) to Builder of the Convolutional Neural networks The design is as follows: the Proposition Network for Region, Anchors, Training / Loss, RoI Pooling and the CNN-based Regions. Region Proposition Networks bring a picture as input and output suggestions for rectangular objects, supposed to be the areas with an objective value. It is achieved by glancing the convolutionary yield of the last convolutionary layer over a tiny network. The network then requires the entry convolutionary function map by a sliding space window. The floating window can then be converted to a smaller dimension with a ReLU. The freshly generated characteristic will then be inserted into fully linked layers of a box- and a box-classification layer. The network for object detection is a single uniform network. The anchors are corrected size reference boxes inside the window that can be placed in a uniform way in the original image, and the anchors are translational, so that the tasks and proposals of the anchor can be translated into various places within the image of an entity. Multi-scale anchors classifications and regress border boxes for various scales and look ratio anchor boxes for various image scales and dimensions. During the training course, an object class mark is assigned to each anchor in the Regional Proposal Network (RPN). Anchors with the greatest cross-sectional spread over a ground truth box or overlapping with a value above 0.7 with any Ground Truth Box have a favorable tag showing an item. If the IoU proportion is less than 0.3 for all ground-truth boxes, the adverse tag will be applied to indicate that no item has been identified. The RPN can be taught by propagation back and descent of stochastic gradients.Each mini-batch comes from the same picture, which includes many favorable and bad instance buttons to measure the loss function and

to achieve a percentage of almost 1:1. To teach the network. The Interest Pooling Region (RoI Pooling) utilizes convolutionary neural networks to identify several items in an picture, by maxing them on different-size outputs to achieve fixed-sized picture characteristics, by separating the regional proposition into identical parts, identifying the biggest value in each segment and copy the maximum value into the input buffer.(Deep-sense AI). The report also enters an Interest Region x 5 matrix representing a list of areas of interest where the first row is a picture index, and the four are the edges of the region co-ordinates. This is a streamlined graph of the Faster R-CNN model CNN-associated method:

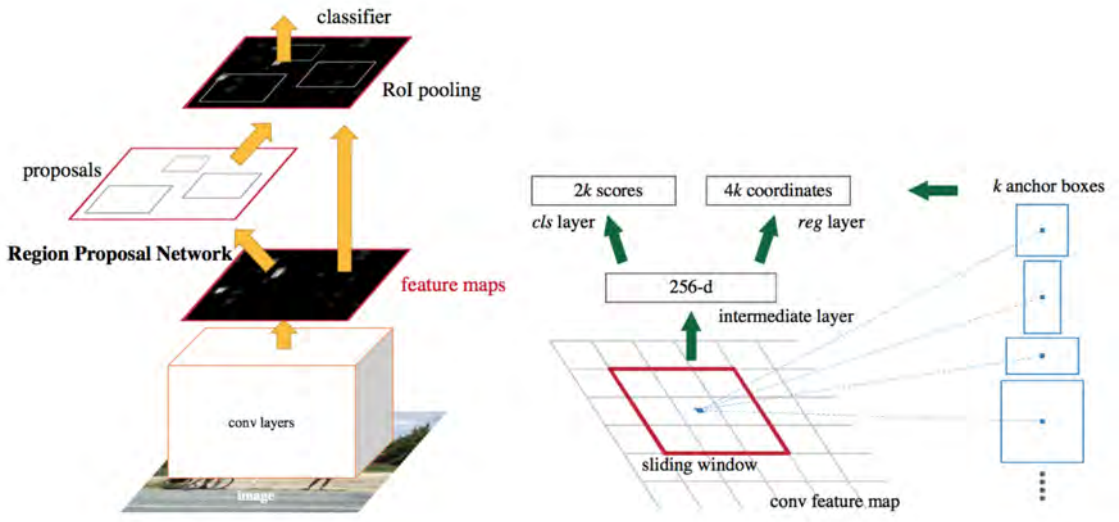


Figure 3.7: Faster RCNN model

In order to generate a checkpoint file and acceptable lost values, the advantages of introducing a Faster RCNN Tensorflow Model are that it gets half a time to get a training interval compared to the SSD MobileNetTensorflow model. It also detects greater amounts of items per picture and greater precision in contrast. However, because the model lacks the depth and point wise convolutionary layers, which also leads to the slower identification rate, it cannot be used for real-time image detection assessment (i.e. portable smartphones). This is a network which detects objects. It is quicker than its offspring RCNN and FastRCNN, as its name has stated. This network uses instances in vehicles, production, and safety as well as in Pinterest. This network has been used. The steps followed–1) Run the picture through CNN to get the feature map 2), run the activation map over a distinct network called the RPN, which produces exciting boxes / regions 3), use multiple completely linked layer to the output and bounding box coordination for the exciting boxes / regions from RPN. The measures followed are: So we should be evident in our knowledge of CNN to proceed further at Faster-RCNN. The first layer in which characteristics are extracted from a picture in Convolution. It maintains a connection between pixels through the use of tiny squares of input information to learn picture functions. It requires on two components like the picture matrix and a filter, or kernel, a computational operation[10].
From the picture above we notice that we have taken a 5x5x3 filter for our 32x32x3 entry and have it slid across the full picture and have taken the dot product between certain filter and the pieces of the picture. The resulting output is a 28x28x1 picture.

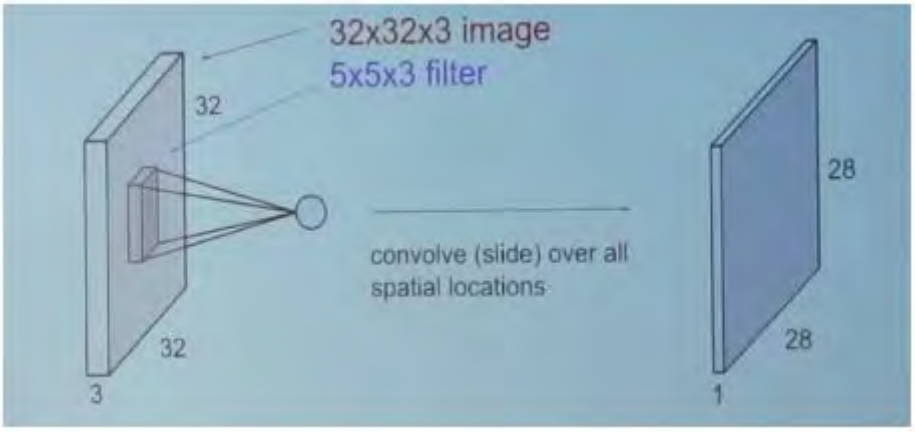


Figure 3.8: Initial Layer

Examples: Consider 5 x 5, the value of which is 0, 1 pixels and 3 x 3 filters, following sources.

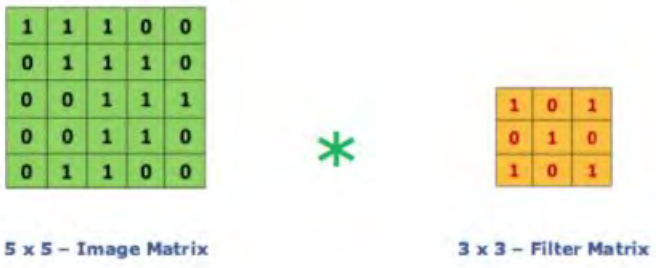


Figure 3.9: Image Matrix × Filter Matrix

The 5 x 5 matrix is then converted by 3 x 3 filters, known as "Feature Map", as a result below with a rhythm of 1.

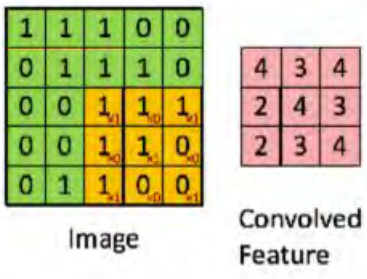


Figure 3.10: Feature Map

Stride: the amount of pixel changes across the entry matrix is Stride. If the run is 1 then filters are moved to 1 pixel at a moment. If the step is two, the filters will be moved to 2 pixels, etc. Padding: if we boost the step value the image size continues to decrease, this issue can be solved if zeroes are added. From the picture below we can see that the image size is maintained after step 1 padding ceros.

Figure 3.11: Size of images

ReLU Activation: ReLU is for the non-linear procedure corrected linear unit. This implies that we can simply propagate mistakes and have several levels of neurons triggered by the ReLU feature on our ConvNet. This implies we can trigger non-linearity. The real-world information would like to know that our ConvNet is a linear non-negative value. Pooling: Sometimes we have to decrease the amount of trainable parameters when the pictures are too big. Thereafter, pooling layers between layers of subsequent convolution are regularly introduced. Pooling is carried out solely to reduce the image's temporal magnitude. Pooling takes place on each aspect of depth separately and the image intensity is therefore unaltered. The most prevalent type of bundling is the maximum bundling.

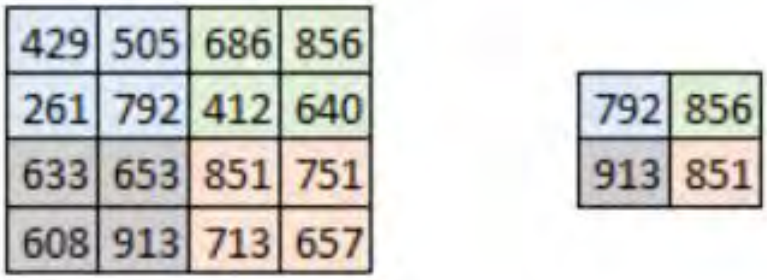


Figure 3.12: Pooling Layer

Here we took step 2, while we also took steps to pool size as 2. The maximum procedure is implemented to every depth of the converted product. So you see, after the max pooling procedure, the 4x4 converted score was 2x2. The picture below is a full CNN stream to process a picture and classify the items according to their merits.

Fully connected layer / output layer: vector (x1, x2, x3.) is flattered with the map matrix following a pooling layer. We have merged these characteristics to generate a template with the completely linked layers. Convolution layers produce 3D activity maps while only requiring production if a picture belongs to a certain category or not. The output layer is able to estimate the mistake in the forecast with a loss

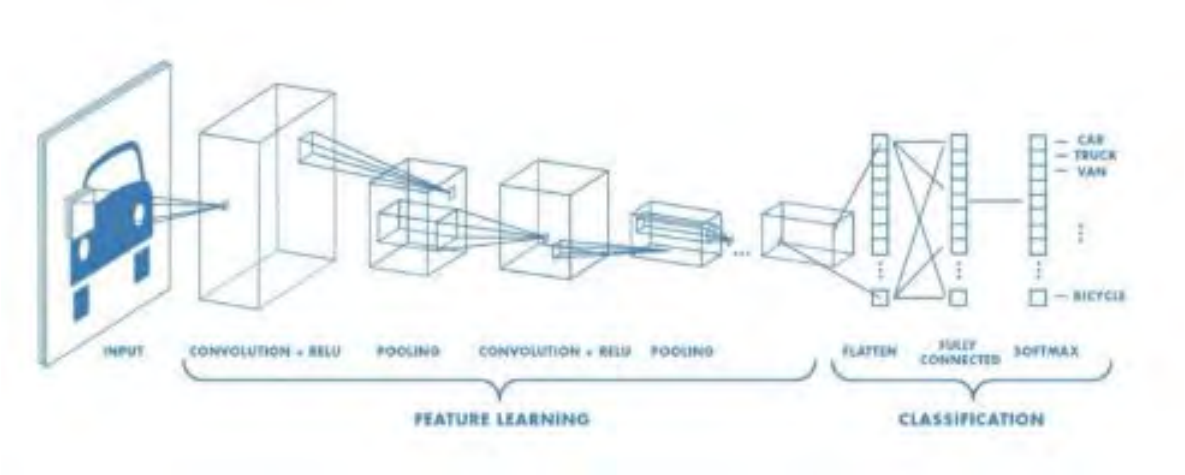


Figure 3.13: Flow of CNN

function such as category intersection entropy. The rear spread starts to adjust the weight and bias for error and loss decrease when the forward spread is full. Now, after receiving a CNN function chart, we must transfer it to the RPN, to discover exciting areas. It transfers areas of concern, loss of whether an item has been discovered and loss of the placement of an item. Then there is Faster-RCNN's top level design.

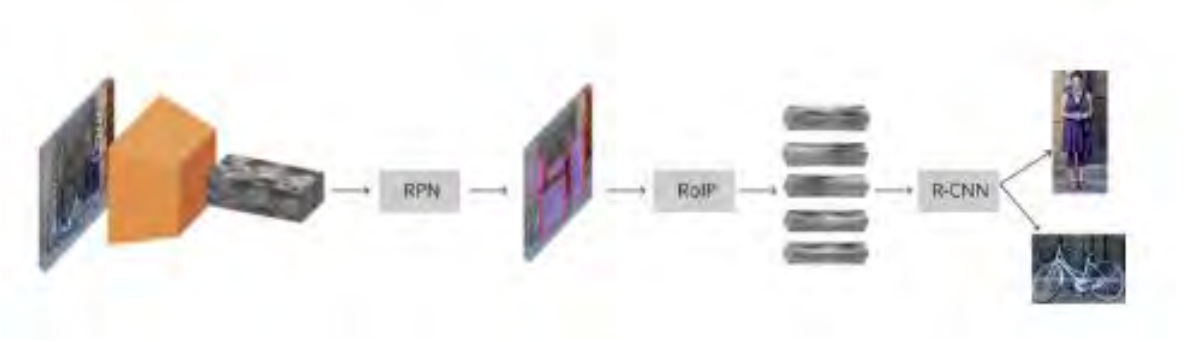


Figure 3.14: High level architecture for Faster-RCNN

When the CNN entry feature is available, the Region Proposal Network (RPN) layer can be used to generate regional ideas (Anchors / Bounding Box). The region suggestions for which the prediction is made are re-formed using an Interest Pooling Region (RoIP) layer to rank the picture in the suggested region and forecast the offset values for the boundary box with R-CNN. R-CNN: A technique used to remove just 2000 areas (area suggestions) from the picture by means of selective scanning. So you can only operate with 2000 areas now, rather than attempting to rank a large amount of areas. The 2000 applicant region suggestions are divided into a circle and fed into a 4096-dimensional output-producing CNN. The CNN functions as an extractor and the thick resulting layer contains characteristics obtained from the picture and the characteristics obtained are transmitted into an SVM to classify the object's existence within this proposed area. The algorithm also presides four values that are offset values to boost the accuracy of the galloping box, in relation to forecasting the presence of an Object within the region proposals. For instance, the algorithm would also have anticipated the existence of an individual in a regional

proposal, but that individual's face might have been cut by half within this region proposal. The offset measurements therefore assist to modify the bounding box of the proposed region. Fast R-CNN: We introduce a CNN input picture into the CNN to produce a convolutionary function map instead of loading the region's ideas. We define the proposal region and tweak it into squares on the convolutionary function chart and transform it into a set dimension via a RoIP layer, so that it can be feed into a fully linked layer. We have a softmax layer in the RoI feature vector to estimate the category of the region as well as the offset amounts for the bounding box. The reason that R-CNN' has to be fastened more quickly than R-CNN is because you don't need to fuel 2000 area ideas into the convertible Neural Network every moment you do it. Instead, a convolution procedure is only performed once per picture, and a function map is produced from this picture. Training is a pipeline for several phases. It includes three different designs for preparing and implementation. In room and time, training is costly. Deep CNN training is very slow on so many picture regional suggestions. Faster R-CNN architecture:

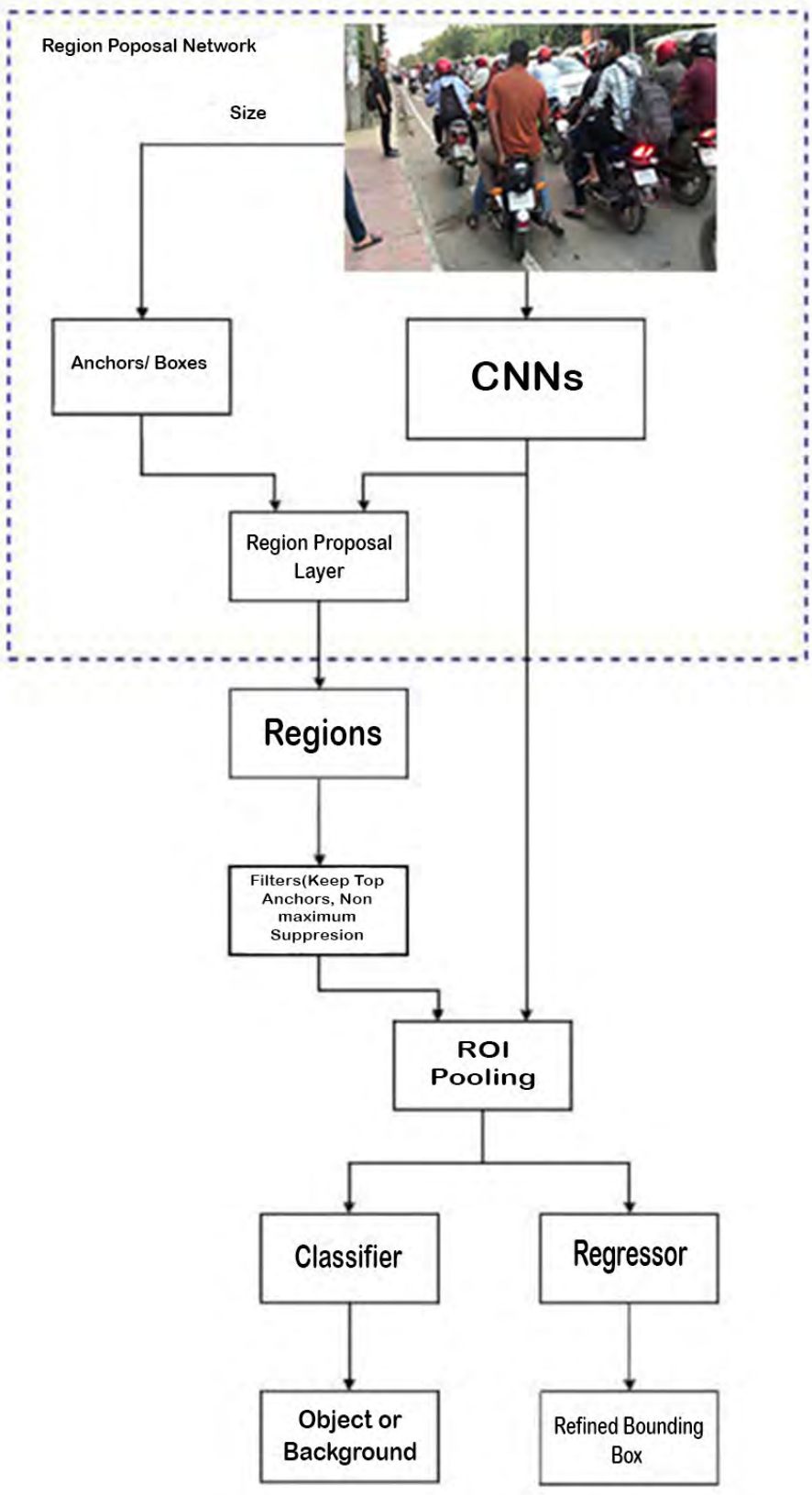


Figure 3.15: Architecture for Faster-RCNN

The previous architecture explains the operational sequence when the picture has

its CNN layer function matrix.

Working of RPN:

A floating window is running spatially on these characteristic maps when a range of convolutionary function maps (function matrix) is provided on the last convolutionary level. nn (here 33) is the sliding window's width. A series of nine bases with the same core (xa, ya) but 3 distinct dimension ratios and 3 distinct scales is produced for every floating window. As can be seen below :

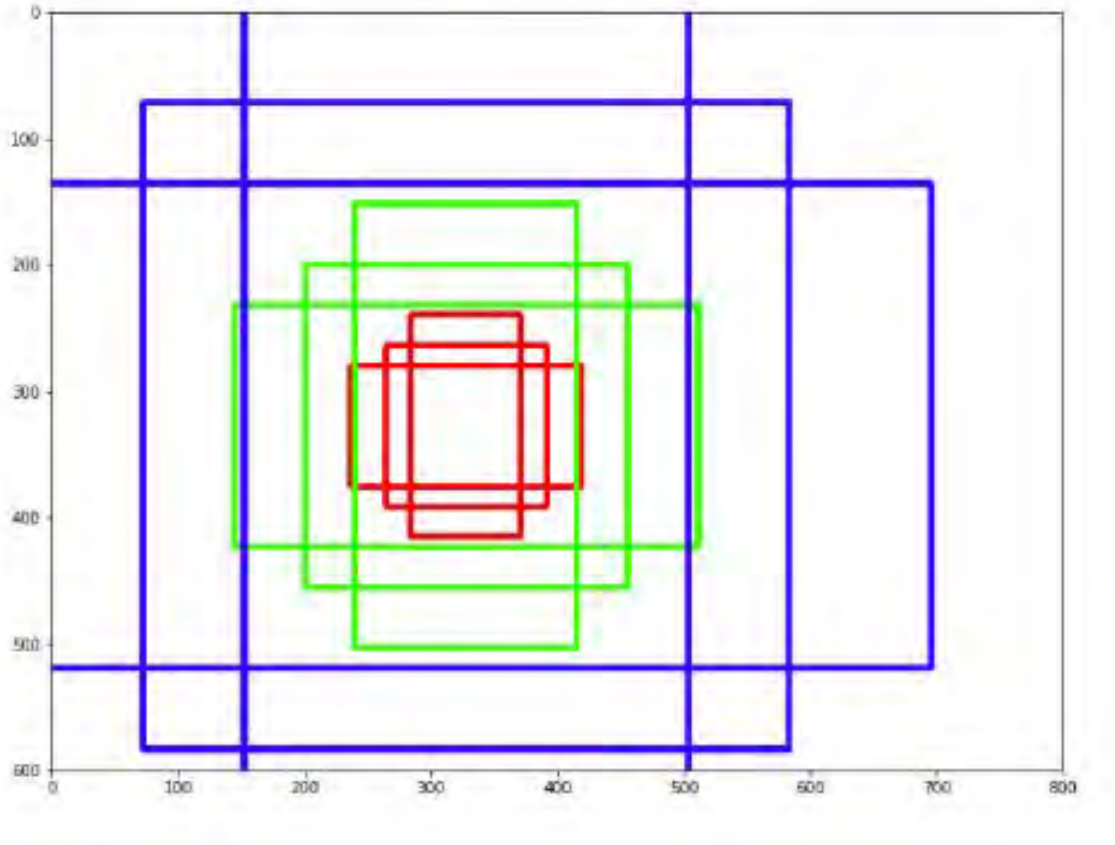


Figure 3.16: RPN

1. Three chromatic colors: 128x128 and 256x256 and 512x512. 2. The 3 boxes are proportions 1:1, 1:2 and 2:1 combination. The bases are well suited for Pascal VOC dataset and the COCO dataset and are also very effective at the height ratio of the set of data. However, you can layout various types of anchors / boxes. You may not be obliged to consider the very brief, large and rectangular boxes, for instance. You are developing a system to count passengers / pedestrians. The velocity and precision of the nice collection of anchors can be increased. Moreover, a value p is calculated for each of these columns which shows how much the bases overlap with the boxes of the ground wider truth. Finally, these features transmit the 33 spatial features to a smaller network, with two tasks: classification and regression. The outcomes of regression define a bounding box (x y, w, and h), a part-network signal shows a probability p that the anticipated box contains an object (1) or a background object (0 for no item). ROI pooling layer: We have separate regions after RPN.

Various areas imply distinct CNN function maps in size. A design that works on

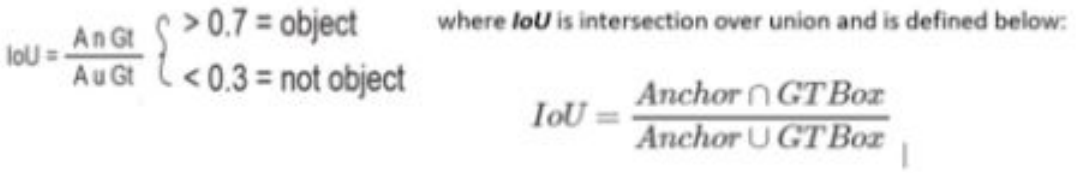


Figure 3.17: Equation

characteristics of distinct dimensions is not simple. By decreasing feature maps to the very same size, ROI pooling can modify the problem. The outcome is that we can rapidly get a list of function maps with a set volume from a list of rectangles with distinct dimensions. RoI pooling benefit? One is velocity computing. If the frame contains several objects (and there are usually many), we can still use the same map of input for them all. Since it is very costly to calculate the convolutions during early testing, we can take a great deal of moment with this strategy. Final layer R-CNN: The Faster R-CNN pipeline is the initial phase. Regional convolutive neural network (R-CNN). Used to obtain an image feature map and to receive item suggestions using the RPN, we lastly need to use these classification characteristics after receiving an involutive function map (by RoI Pooling). In the end of the classification of the CNNs, R-CNN is attempting to imitate a fully connected layer to provide a score for every possible object class.

3.5 ReLU

The corrected linear unit is the most frequently used profound learning feature. The feature returns 0 when it gets any adverse entry, but returns it for any beneficial value x. So it can be written as $f(x) = max(0, x)$.
Graphically it looks like this

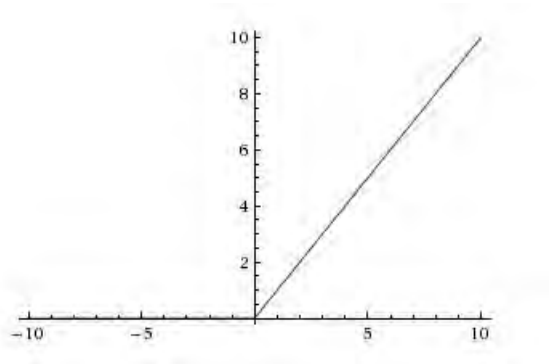


Figure 3.18: ReLU

Especially a basic function (and one consisting of two linear parts) can make the non-linearities and relationships so well known to your model. But in most apps, the ReLU feature operates fantastic and is therefore very commonly used.
Interactions and Non-linearities

Activation functions serve two primary purposes: 1) Help a model account for "interaction effects". What is an interactive effect? It is when one variable A affects a prediction differently depending on the value of B. For example, if my model wanted to know whether a certain body weight indicated an increased risk of diabetes, it would have to know an individual's height. Some body weights indicate elevated risks for short people, while indicating good health for tall people. So, the "effect of body weight on diabetes risk depends on height", and we would say that "weight and height have an interaction effect".
2) Help a model account for "non-linear effects." This just means that if I graph a variable on the horizontal axis, and my predictions on the vertical axis, it isn't a straight line. Or said another way, the effect of increasing the predictor by one is different at different values of that predictor.

Capturing Interactions and Non-Linearities

Interactions: Imagine one node in a template of a neural network. Suppose it has two outputs, A and B, in order to be simple. Our node is made up successively of 2 and 3 weights from A and B. The yield of the node is therefore $f(2A+3B)$. For our f, we will use ReLU. Thus, if $2A+3B$ are favorable, our node yield score is $2A+3B$ as well. The yield price of our node will be 0, if $2A+3B$ is incorrect. Take into account the case of A=1 and B=1 for concreteness. The yield is $2A+3B$ and the yield also rises if A rises. On the other hand, if the output of B=-100 is 0, and the output of A is moderately increased, then it is 0. So A could or could not boost our production. The importance of B only relies on it. This is a straightforward situation where the interaction is caught by the node. By adding further nodes and layers, there will only be increased complexity of relationships. Now, however, you need to see how the activation work helped to catch an event. Non-linearity: if the path is not continuous, a feature is non-linear. The ReLU feature is therefore non-linear at approximately 0, although the path is always either 0 or 1 (for adverse scores). This is an extremely restricted kind of non-linearity. However, two facts regarding deep learning designs enable us, as we merge the ReLU nodes, to generate many distinct kinds of nonlinearities. Firstly, for each node, most designs have a bias word. During model practice, the word is only a steady amount. Consider a node with an entry called A and a partiality for convenience. If the word is 7, the node yield is f (7+A). When the word is 7. If A is less than -7, the yield shall be 0 and the pitch shall be 0. If A exceeds-7 the yield of the node is 7 + A, and its pitch is 1. Thus, the preference word enables us to change the pitch. It seems that we can only have two distinct paths until now. Real designs have a lot of nodes, however. Each node (including in one layer) can have a distinct value to modify the path for the entry of each node. By adding the resulting features, we get a mixed feature that in many locations shifts pathways. These designs are flexible in creating nonlinear features and take good account of relationships (if this gives stronger forecasts). When additional nodes are added in each layer or more convolutions when a convolutionary system is used, the system becomes even more capable of representing such relationships and nonlinearities.

Facilitating Gradient Descent

This is a more technological part than the above. If it is hard to do so, note that apart from this technical context you can achieve a lot with deep learning. Deep learning designs have historically began in s-shaped bends (such as the tanh below) It seems that the tanh has a few benefits. While it's near to the plain, anywhere it is

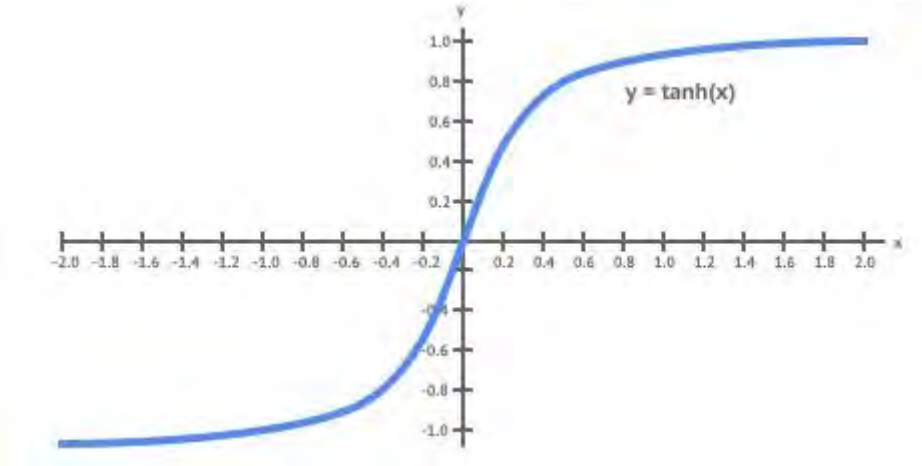


Figure 3.19: Tanh function

not entirely flat. So the production always represents modifications in the input that could be useful for us. It's non-linear (or everywhere bent) second. Non-linearity accounting is one of the primary reasons of the activation feature. We therefore predict a non-linear feature to function well. However, when the tanh feature was used, scientists had excellent trouble constructing designs with many layers. Except for a very small area (the range is about-2 to 2). It is comparatively flat. It is a very tiny derivative unless the input is within this limited scope and it is not possible to enhance weights via gradient descent through this plain derivative. This is exacerbated by the layer of the model. This has been called the "disappearing issue of gradient." The ReLU feature has a 0 over half-range derivative (adverse numbers). The derivative is 1. For beneficial outputs. Usually there is a certain amount of information points that give favorable scores to any node when studying on a fair batch. So the median derivative seldom reaches 0, which enables the progression of the gradient downward.

Alternatives

Many comparable options operate well, too. One of the most famous is the Leaky ReLU. For favorable figures that is the same as ReLU. But it has a steady curve (less than 1), rather than 0 for all adverse scores.

This pitch is a parameter that the template customer displays and is often referred to as α. The activation feature, for instance, is f(x) =max (0.3 x, x) if you set $\alpha = 0.3$. This has the theoretical benefit that the data stored inx can be used more completely by being affected byx at all levels.

There are other options but, usually speaking, researchers as well as scientists have discovered little to warrant anything other than RLU.

3.6 Tesseract

Tesseract was created between 1984 and 1994 at HP and is an outdoor OCR engine. Like supernova, the 1995 UNLV Annual Test of OCR Accuracy emerged from nowhere, gleaming brightly with its outcomes and then disappearing into the cloud of privacy in which it was created. Now the structure and algorithms information can be disclosed for the first moment. Tesseract has started as a PhD study venture at HP Labs, Bristol. As a software and hardware supplement for HP's flatbed

scanner line it has acquired momentum.Tesseract has a important advantage in the precision of business motors, but has not been converted to a brand, following a joint venture between HP Labs Bristol and the HP Scanner division. The next step was the study of OCR for compression in HP Laboratories Bristol. Work was focused more on enhancing refusal effectiveness than on precision at the basis level. Development completely stopped at the end of this initiative at the beginning of 1994. For the 1995 Annual Test of OCR Accuracy, the engine was sent off to UNLV to demonstrate its value against the then business motors. HP published Tesseract for open source at the end of 2005.

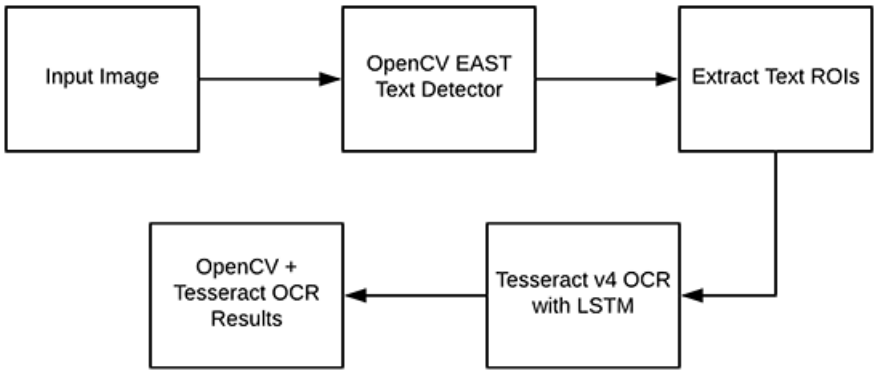


Figure 3.20: Tessearct workflow

Since HP had independently developed (and thus not available for open-source) screen design analyzing technology used in products, Tesseract never required an assessment of its own website. Thus, Tesseract expects its entry to be a binary picture with optionally specified polygonal text areas. A conventional step-by-step pipeline performs the processing but some phases were uncommon and may even stay so. The first stage is an assessment of the linked element which stores details of the parts.That's a very costly computer design choice at that moment, but it has had an important benefit: it's easy to find reverse text and acknowledge it as Black-on-white text by examining the borderlines and amount of children's and grandchildren's images. Probably the first OCR motor to manage writing so trivial in white-on-black was Tesseract. In this phase, contours are collected in blobs, solely through breeding. Blobs are structured into text rows, and for a set pitch or linear text, lines and areas are evaluated. Text lines are divided into phrases according to the type of distance between them.The text of the fixed pitch is cut by character cells right away. Proportional phrases with certain spaces and blurred spaces are split into phrases. Recognition continues then as a method of two passes. In the first step, every term is tried to acknowledge in turn. Each adequate term is carried on as training information to an adaptive classifier. The adaptive classifier then has an opportunity to identify text more correctly below the screen. As the adaptive classifier can have discovered something helpful too early to contribute to the top of the table, a second pass is taken over the website, which again recognizes phrases that are not sufficiently well known. A final stage resolves fluctuating rooms and examines alternative x-height hypotheses to find small caps[11].

3.7 OpenCV

OpenCV is a platform-based library that enables us to create machine views in real time. It relies primarily on image processing, video collection and evaluation, including the identification of face and objects. With the OpenCV library, pictures can be read, captured, saved or processed, pictures can be detected (filtered, transformed) and particular items such as faces, eyes or vehicles can be detected, pictures can be captured and saved, video can be analyzed, i.e. movement estimated, background subtracted, and items tracked. The first development of OpenCV was in C++. Python and Java have also been supplied with bindings. Operating systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD and many others are operated on[12].

Core Functionality

This module includes the fundamental information constructions used for OpenCV apps, such as Scalar, Point, Range, etc. In addition, the multidimensional Mat matrix is also included, which is used for saving the pictures. This module is also included with the title org.opencv.core in the Java library of OpenCV.

Image Processing

This module deals with different image processing activities such as filters, conversions of geometric images, conversions of colour space, histograms, etc. This module is included in OpenCV's Java library as a package called org.opencv.imgproc.

Video

This Module includes ideas for video analysis, like motion estimation, subtraction of the context and object monitoring. This module is included in the OpenCV Java library as a bundle called org.opencv.video.

Video I/O

The video capture and video codecs with OpenCV library is explained in this module. In the OpenCV Java library the module has the org.opencv.videoio title as a package.

Calib3d

This module contains algorithms concerning essential geometry algorithms for multiple views, calibration of the single and stereo cameras, evaluation of object poses, and interaction with 3D components. This configuration is included with the OpenCV Java library as a package called org.opencv.calib3d.

Features2d

The concepts for function recognition and description are included in this module. This module are included in the OpenCV Java library as a package called org.opencv.features2d.

Objdetect

This module detects predetermined class items and cases, like faces, eyes, cups, individuals, vehicles, etc. This module is included in the OpenCV Java library in the form of an org.opencv.objdetect title package.

Chapter 4

System Implementation

The whole process can be divided into 2 major parts -
1.Image processing and Object Detection
2.Number plate detection
We have collected our own dataset. At first we have taken multiple videos at different places in Dhaka. Then we have extracted images from those videos. Later we have selected around 5.5 thousands images for our system. Then we have annotated all the images for further processing and training. Then we have trained the images.
The working principle of our system -

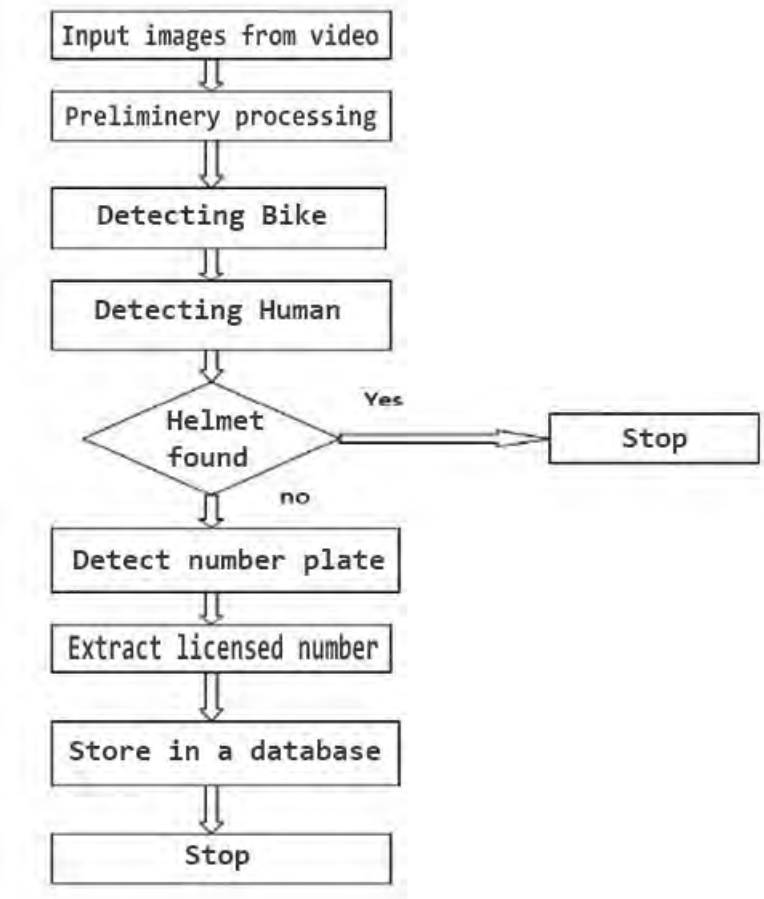


Figure 4.1: Work flow of our system

4.1 Data Collection

In the beginning we have collected as much videos as possible with our mobile. Then we have selected the best videos with best angles. Then we extracted images from our videos. Initially we got around 40000 images from the videos. Among which we have selected precisely 5626 images. These are the images which we have used for our overall system. We have used a python script with opencv for image extraction. The scrpit also uses Numpy library. The sample images initially looked like this -

Figure 4.2: Sample image initially

The GPU of our computers cannot take the load sometimes if it is given high resolution photos. So to optimize correctly, we have resized our images to 600×600. Initially the dimension was 960×540. Also we have scaled down the quality of the images by 2. As a result it takes much lesser space. This will also help the system to work fine even in a low quality environment where the images are not that good like the ones we have provided. After resize we get the images like this -

Figure 4.3: Sample image after resize

4.2 Verification of images

We have splitted our images into train images and test images where the ratio is 75:25. We have verified our images with histogram.
If we talk in image processing context, a histogram of the pixel intensity values can be referred as histogram of a particular image. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. There are 256 different possible intensities for an 8-bit grayscale image, and thus the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values. We can do istogram also on color images - we can do it by individual histograms of red, green and blue channels , or we can produce 3-D histogram, with the three axes representing the red, blue and green channels, and the pixel count is represented by brightness at each point. The exact output from the operation depends upon the implementation - it may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics.

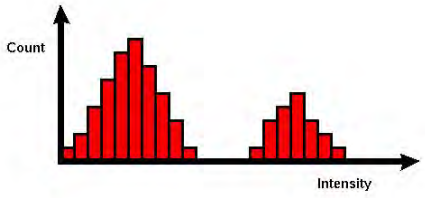


Figure 4.4: Histogram graph

It's a really straight forward operation. At first the image is scanned in a single pass and we keep a running count of the number of pixels found at each intensity value. After that we use this to construct a suitable histogram.
We have used Histogram so that we can make sure that the test images doesn't match with the train images and have enough difference.
Basically histogram compares images and gives a graph which shows that the images have some difference in them.
Here we can see that the process has generated 2 different graph for our 2 images.

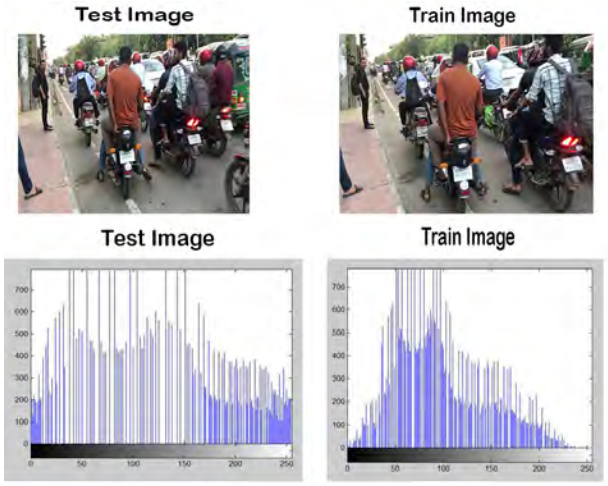


Figure 4.5: Histogram process

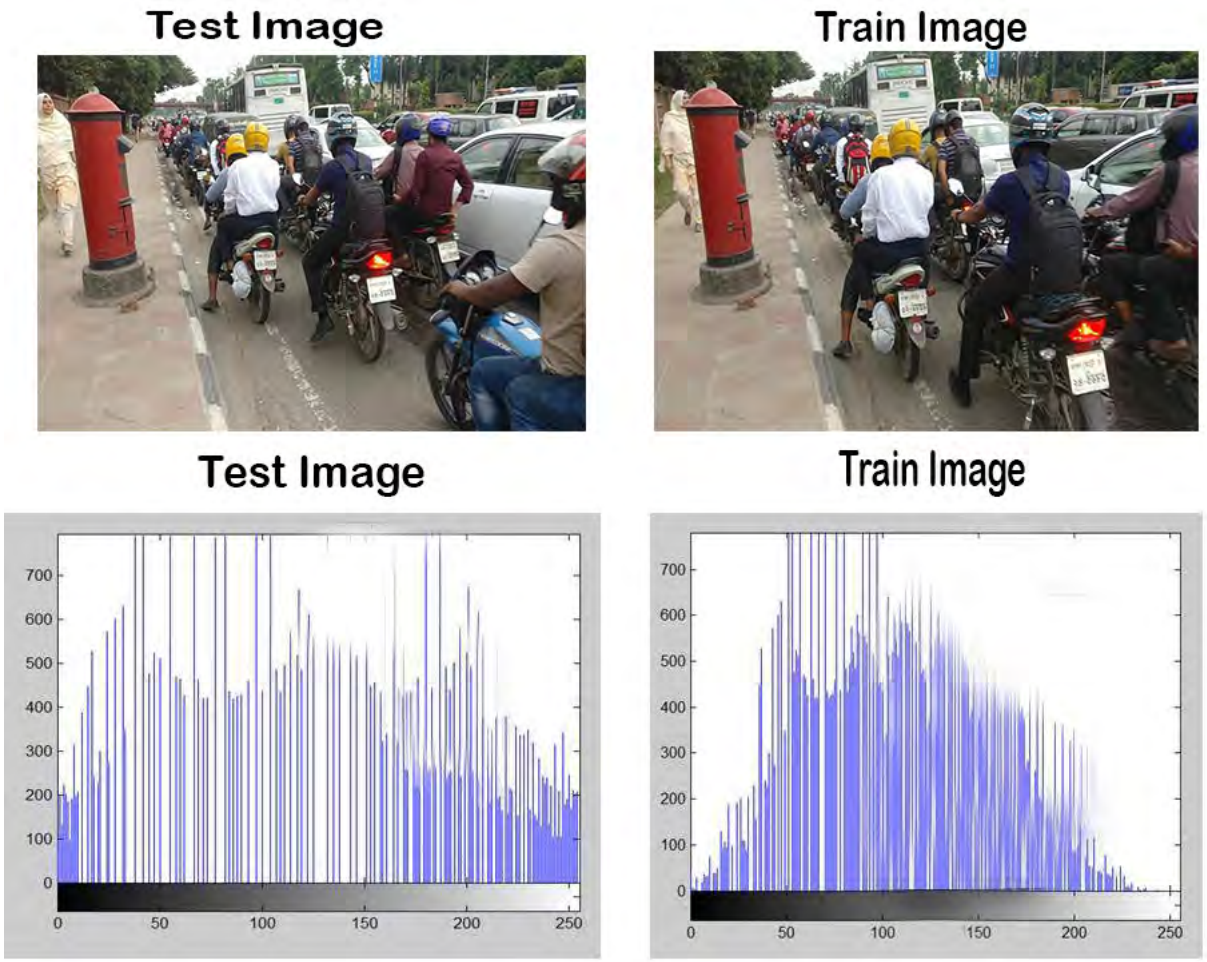


Figure 4.6: Histogram process with different images

If we see this another example then it makes it more clear.

4.3 Annotation

After extracting and resizing the images, we annotated the images. Annotation is done in a software named LabelIMG.
LabelImg is written in Python and an open source image labeling tool that has pre-built binaries for Windows and thus it's really easy to install. Though it has virtually no project management properties, it allows an easy way to import and visualize annotations and correct them if necessary. Specially, the simple offline interface makes the annotation process really fast, even though it does not support many hotkey shortcuts.
In this software we have to draw rectangle boxes around the classes we want to classify/detect. So in our case we have four classes in our images which are bike,number plate,human and helmet. At first we draw the rectangle box on the bike parts. Inside the bike part we draw the license plate part. We also draw a rectangle for human part. Inside that human part we draw another rectangle for helmet.As each of them are connected for our system, it is important to follow every single details and regioning of the rectangles. A sample annotated image looks like figure 4.7

Figure 4.7: Sample annotated image with multiple biker

Here we can see there are multiple bikes. Also there is a human who is not wearing any helmet. We have only annotated the images where all the parts are visible. Figure 4.8 is another sample image where there are two humans who are not wearing helmet.

Figure 4.8: Sample annotated image 3 bikers

It doesn't matter how many humans are there in the video. Even if there is only one human in the video with bike then we will annotate that one only and it's not a problem. Figure 4.9 is an image with only 1 human who is not wearing any helmet.Where the other parts are visible.

Figure 4.9: Sample annotated image with one biker

The final output of these annotated photos is a XML format(XML) which is all about x and y coordinates.
The labels look like this -
After labeling it generates XML files. XML files looks like this -
Here is another XML with multiple classes -

4.4 Pre-processing of data

After the annotations are done, we have generated a CSV file from the XML files. CSV files are files with tabulur format. These CSV file stores the values of our images in a tabular format. It stores the classes and their coordinates.
This CSV file shows that the XML file had only one item of each classes.
Now let's take a look at another CSV file with multiple items with same class and their ordering.
Then we create a TFRecords file from the CSV file. The API accepts inputs in the TFRecords file format. Finally we make a single CSV for all the XML files.
The TFRecord format is a simple format for storing a sequence of binary records. Protocol buffers are a cross-platform, cross-language library for efficient serialization of structured data. Protocol messages are defined by .proto files, these are often the easiest way to understand a message type. The machine can read the TFRecord file as it is written in only binary format using 0 and 1. It uses Protoc Buffer to

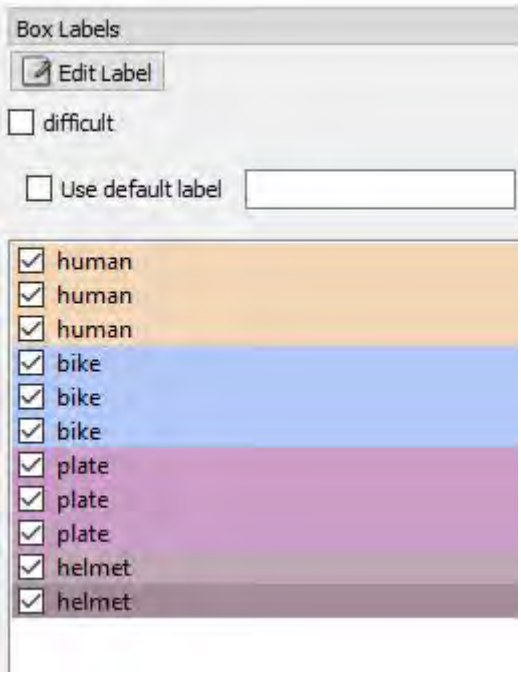


Figure 4.10: Labels in XML file

serialize or deserialize data. After we run the CSV to TFRecord script, we get 2 files - train.record and val.record.
We also create a Labelmap file which has all the classes we have annotated in the images. In our case it contains the bike,human,number plate and helmet classes.

4.5 Training the model

At first we chose the models we want to use for our system. As mentioned before, there is a trade off between speed and accuracy. Also, building and training an object detector from scratch would be extremely time consuming. So, the TensorFlow Object Detection API provides a bunch of pre-trained models, which you can fine tune to your use case. This process is known as Transfer Learning, and it speeds up training process by an enormous amount.[13]
Here is a chart on speed and computation of the varioud models of Tensorflow -
For our system we have chosen the SSD Mobilenet v2 and Faster R-CNN inception v2 models. Both the models are After this we use the model we chose for our project and keep all the files in order. In the model folder, there are the model checkpoints, a frozen inference graph, and a pipeline.config file. Then we defined the "training job" in pipeline config file which is used to map with train and test files. Fine tuning is major part for making a successful system.
Figure 4.17 shows us the tuning we have set up for our model of SSD Mobilenet v2. Fine tuning is a process to take a network model that has already been trained for a given task, and make it perform a second similar task.
Assuming the original task is similar to the new task, using a network that has already been designed trained allows us to take advantage of the feature extraction that happens in the front layers of the network without developing that feature extraction network from scratch. Fine tuning:

Figure 4.11: XML file of an image

Replaces the output layer, originally trained to recognize (in the case of imagenet models) 1,000 classes, with a layer that recognizes the number of classes you require. The new output layer that is attached to the model is then trained to take the lower level features from the front of the network and map them to the desired output classes, using SGD.
Once this has been done, other late layers in the model can be set as 'trainable=True' so that in further SGD epochs their weights can be fine-tuned for the new task too. Defining training job is also important because of the bass size and for the layers it will have. The better we define this job the better result we get.
In the pipeline config file we connect the train and test record files and the label map as shown in figure 4.18
After all these works we start the training job. We have trained our system for around 45000 steps for 6 hours. Here is how the training job looks -
After around 45000 steps when we the desired result, we stopped the training.

4.6 Inference Graph

Before using the model, we exported the trained checkpoint files to a frozen inference graph. We have trained our system for around 40000 steps for both the models. After we export the checkpoint to a frozen graph we get a new file named frozeninferencegraph.pb.
After doing all these we can put images we want to test in the test images folder for the model and run our test.
So the whole process of image processing and training pretty much seems like figure 4.10.

```
<annotation>
    <folder>Main</folder>
    <filename>img (3494).jpg</filename>
    <path>G:\New dataset\Main\img (3494).jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>600</width>
        <height>600</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>human</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>211</xmin>
            <ymin>1</ymin>
            <xmax>283</xmax>
            <ymax>187</ymax>
        </bndbox>
    </object>
    <object>
        <name>bike</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>225</xmin>
            <ymin>197</ymin>
            <xmax>304</xmax>
            <ymax>431</ymax>
        </bndbox>
    </object>
```

Figure 4.12: XML with more classes

A1 fx folder

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	folder	filename	path	source__d	size__wid	size__hei	size__dep	segmente	object__n	object__p	object__t	object__d	object__b	object__b	object__b	object__bndbox__ymax		
2	Main	img (1).jp	C:\Users\I	Unknown	600	600	3	0	bike	Unspecifi	0	0	156	343	267	547		
3									human	Unspecifi	0	0	160	135	274	342		
4									helmet	Unspecifi	0	0	205	143	245	203		
5									plate	Unspecifi	0	0	163	395	209	448		

Figure 4.13: CSV file content

4.7 Number Plate Recognition

In our trained model, when we test our test images everytime we get human without helmet we crop the number plate part from the images with the help of the coordinates we defined in the annotation. Then the cropped image part is processed for registration number extraction. Then we use the Tesseract library to extract the number from the number plate and store it. This is how we can keep records of the people who don't wear helmets for further punishment.

Tesseract is probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text.

At first we take the cropped part of a number plate like this -

Then we segment the number plate and get something like this -

This is how we can identify the registration number of a number plate.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
folder	filename	path	source__d	size__wid	size__hei	size__dep	segmente	object__n	object__p	object__tr	object__d	object__b	object__b	object__b	object__b	ndbox__ymax	
Main	img (2159	G:\New d	Unknown	600	600	3	0	human	Unspecifie	0	0	150	129	206	255		
								human	Unspecifie	0	0	241	108	315	274		
								human	Unspecifie	0	0	409	17	514	368		
								bike	Unspecifie	0	0	166	265	214	387		
								bike	Unspecifie	0	0	263	277	315	425		
								bike	Unspecifie	1	0	422	384	569	600		
								plate	Unspecifie	0	0	275	316	301	341		
								plate	Unspecifie	0	0	179	292	200	315		
								plate	Unspecifie	0	0	510	464	560	531		
								helmet	Unspecifie	0	0	257	109	293	153		

Figure 4.14: CSV with multiple class

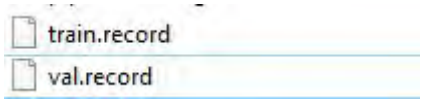
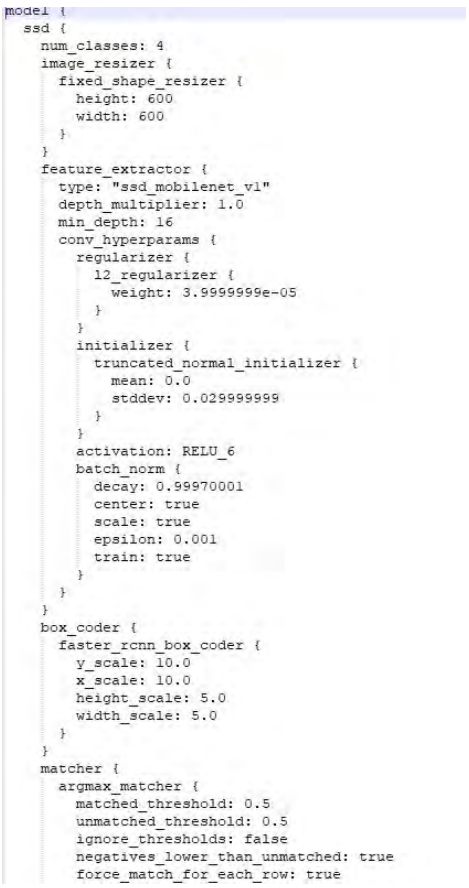


Figure 4.15: TFRecord files

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes

A bunch of models pre-trained on the MS COCO Dataset

Figure 4.16: Various models and their speed vs time



```
model {
  ssd {
    num_classes: 4
    image_resizer {
      fixed_shape_resizer {
        height: 600
        width: 600
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v1"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999999e-05
          }
        }
        initializer {
          truncated_normal_initializer {
            mean: 0.0
            stddev: 0.029999999
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.99970001
          center: true
          scale: true
          epsilon: 0.001
          train: true
        }
      }
    }
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
```

Figure 4.17: Fine Tuning inside Pipeline config file

```
  optimizer {
    rms_prop_optimizer {
      learning_rate {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0040000002
          decay_steps: 800720
          decay_factor: 0.94999999
        }
      }
      momentum_optimizer_value: 0.89999998
      decay: 0.89999998
      epsilon: 1.0
    }
  }
  fine_tune_checkpoint: "/home/headblcoks/Desktop/iqbal/models/research/object_detection/training/ssd_mobilenet_v1_coco_2018_01_28/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 100000
}
train_input_reader {
  label_map_path: "/home/headblcoks/Desktop/iqbal/models/research/object_detection/data/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "/home/headblcoks/Desktop/iqbal/models/research/object_detection/data/train.record"
  }
}
eval_config {
  num_examples: 8000
  max_evals: 10
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "/home/headblcoks/Desktop/iqbal/models/research/object_detection/data/label_map.pbtxt"
  shuffle: false
  num_readers: 1
  tf_record_input_reader {
    input_path: "/home/headblcoks/Desktop/iqbal/models/research/object_detection/data/test.record"
  }
}
```

Figure 4.18: Connecting train and test files inside pipeline file

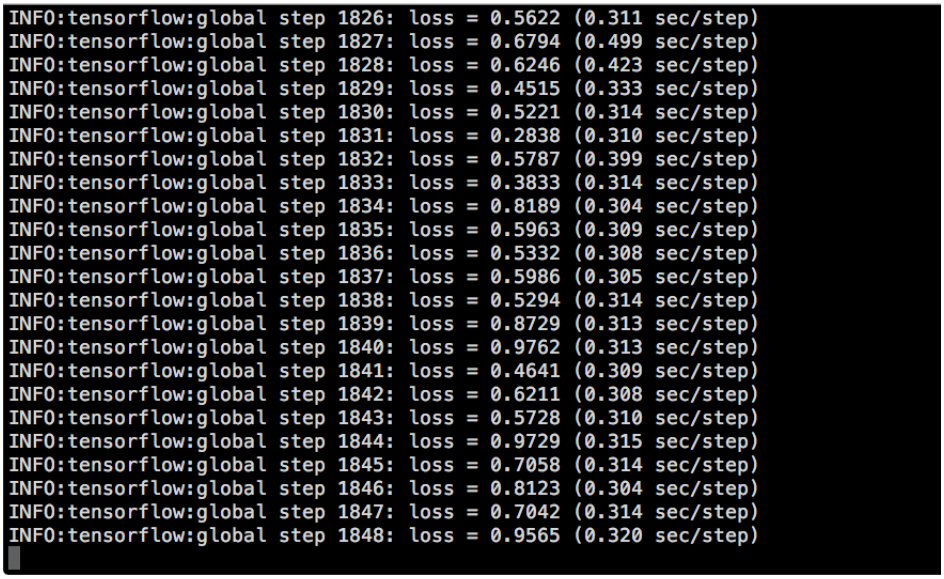


Figure 4.19: training of object detection

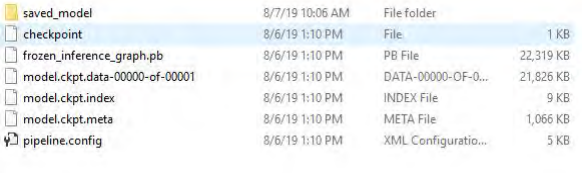


Figure 4.20: Generating frozen inference graph

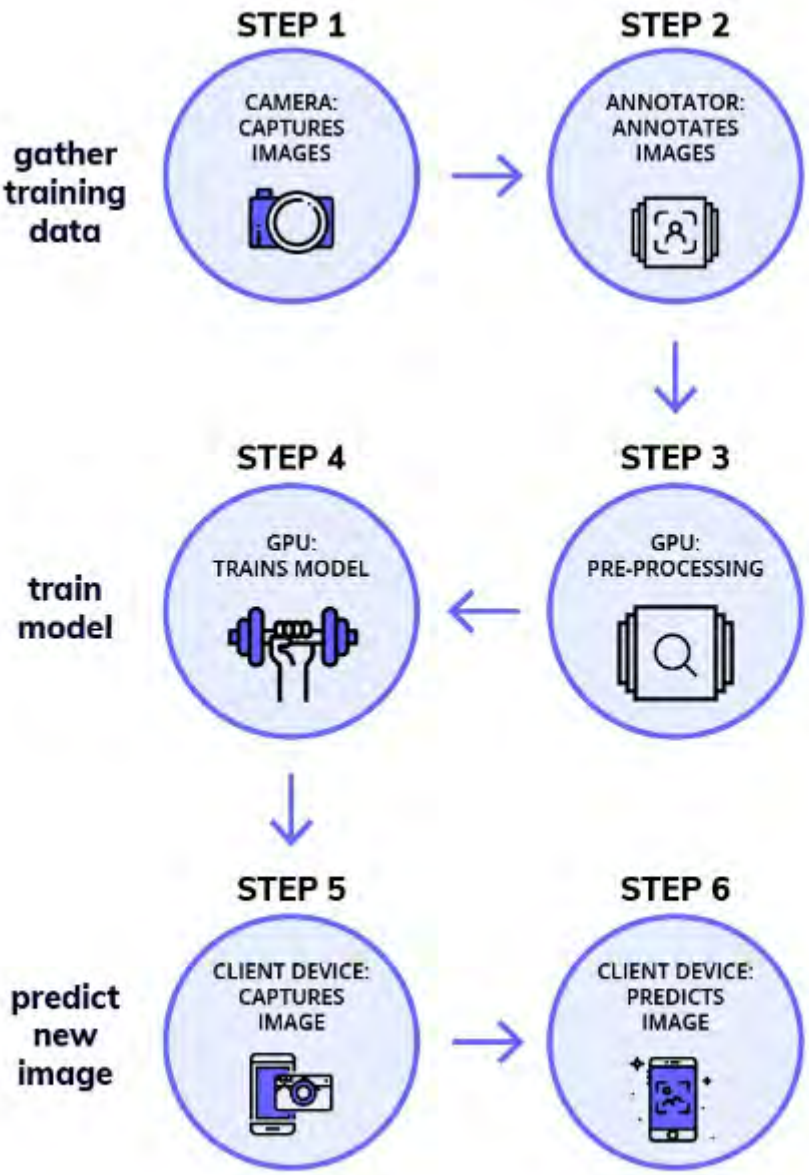


Figure 4.21: Steps for object detection

Figure 4.22: Number plate from a bike

Figure 4.23: Number plate segmentation

Chapter 5

Experiment and Results

After training and all we have tested our system that we have developed. The object detection checkpoint which is a IPYNB file, we run it with Jupyter. Before that we put the images we want to test in the Test images folder.
We do this twice as we have used 2 models for our system. Though the accuracy is different for the 2 models but the difference is minimum. At first we tried SSD Mobilenet v2 model. For this it took around 6 hours for approximately 45k steps. The accuracy for the helmet was 90 percent, human was 55 percent, bike was 80 percent and number plate was 95 percent. This model is quite light and we can use this trained model even in mobile devices.
For Faster RCNN inception v2 it took more computations but the accuracy we got was slightly better as it is more heavyweight than SSD Mobilenet. The accuracy we got for this was 92 percent for the helmet, 58 percent for the human, 81 percent for bikes and 96 percent for number plates.
After running the test images file result we got was pretty good. Here is an image we have tested -

Figure 5.1: output image with one biker

In this image we can see that the result is really good in detecting the parts. Now let's move on to an image which has multiple people in it.

Figure 5.2: output image with multiple bikers

In this image we can see that it has detected the parts well too. But in the right side it couldn't detect a human which is a loss. Here is another image where there are multiple people -

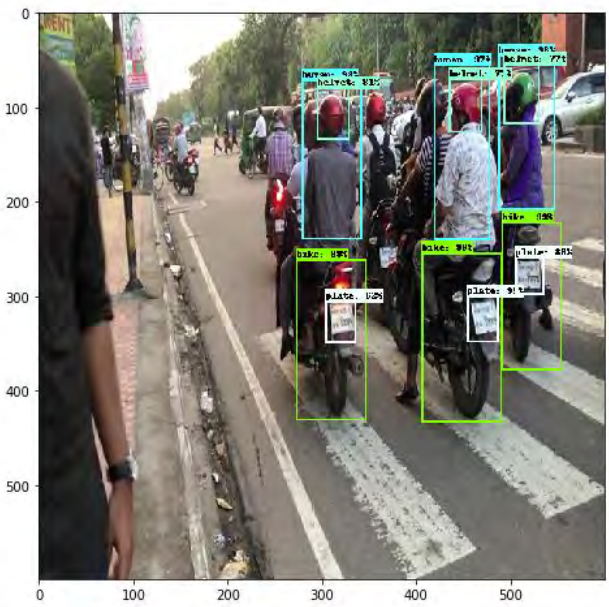


Figure 5.3: output image with congested bikers

In this image we can see that it has perfectly detected all the bikers with all parts visible. Here the loss is close to zero.
Now we have seen that not all the license plates are digital printed. There are some which are hand painted. It was a challenge for the system to detect those. Our model successfully detected hand written number plates too.

Figure 5.4: output image with handwritten plate

In this image we can see that the model has detected a number plate which is hand written.
Our model is slightly overfitted. The accuracy we got from the 2 models is quite good. Our model successfully gave a higher helmet and number plate detection accuracy which was our main goal.

Models	helmet	human	num plate	bike
SSD M.net V2	90 percent	55 percent	95 percent	80 percent
F-RCNN Inc. V2	92 percent	58 percent	96 percent	81 percent

Table 5.1: Helmet detection accuracy for 2 models

Everytime we got bikers without helmet we crop the number plate and save it as an image.

Figure 5.5: Number plate from our images

Then we process further and recognize the number with the help of Tesseract. So the overall process of number plate recognition looks like Figure 5.6.
Tesseract is a open source library. With the use of Tesseract we can recognize the numbers from plates. But it's still not that good in terms of accuracy. So whenever we fail to recognize a number plate, we send the number plate again for recognition and the have to repeat the whole process again.[14]

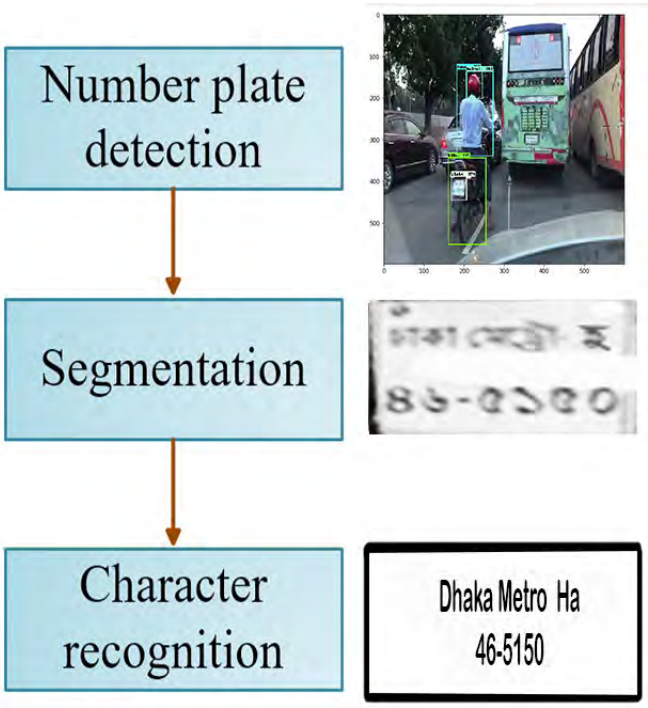


Figure 5.6: Number plate recognition steps

5.1 Analysis

Now that we have experimented the data with our system, comparison and overall graphs about the models will make it more clear. At first, for object detection we have used Tensorflow Library. It has an huge advantage than other libraries as shown here -

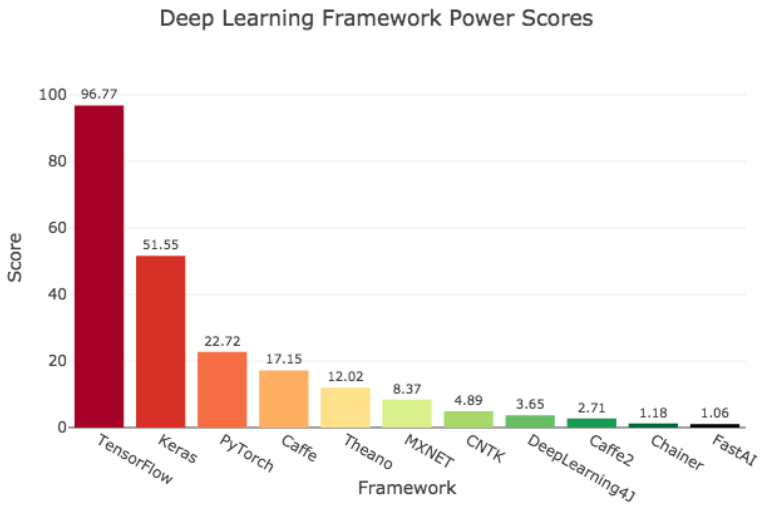


Figure 5.7: Tensorflow advantage over other deep learning frameworks

At first we have used the SSD Mobilenet V2 model of Tensorflow. It is really lightweight. It's iteration time per step takes a bit less time than other models. The accuracy we got from the training is shown here -

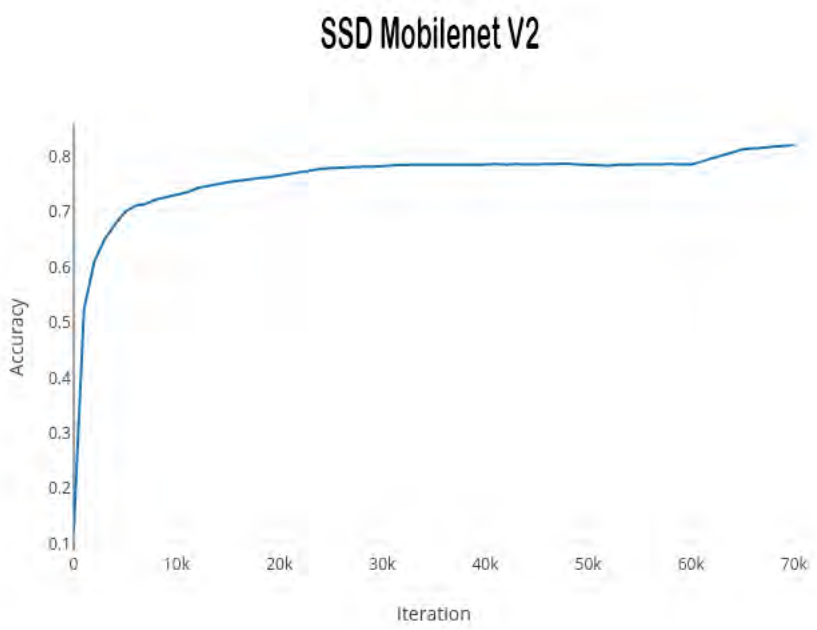


Figure 5.8: SSD Mobilenet V2 Accuracy

Then we have used the Faster RCNN Inception V2 model. It had a slight increase in accuracy as shown below -

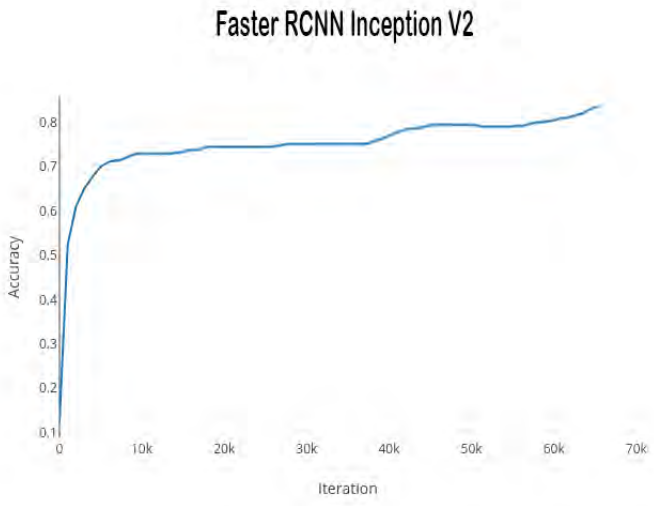


Figure 5.9: Faster RCNN Inception V2 Accuracy

It shows that it has resemblance with SSD Mobilenet V2 and has the almost same accuracy. At first the accuracy starts rising. After continuous iterations it reaches a point where the accuracy doesn't vary so much and stay at a stable phase. At the point where the iteration count reaches 45000 we have stopped the training as the accuracy doesn't vary anymore at that position.
When the training occurs we have to take a look at the loss function continuously and depending on that we carry on the iteration process. When the loss function reaches a minimum number and it doesn't go much less than that we stop iterations. Here is the loss function graph of SSD Mobilenet V2 -

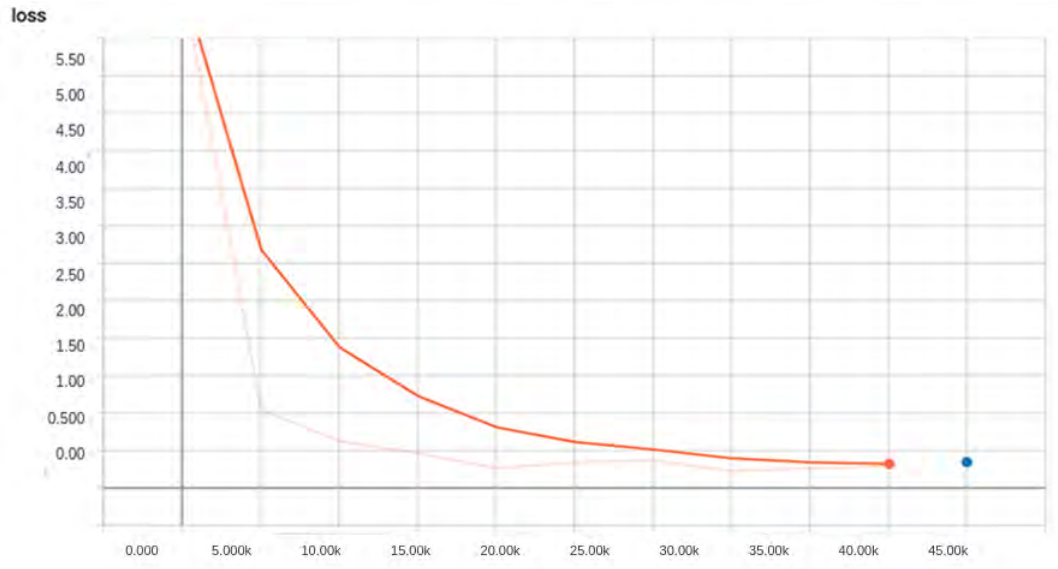


Figure 5.10: SSD Mobilenet V2 loss

Now let's take a look at the loss function of Faster RCNN Inception V2 -

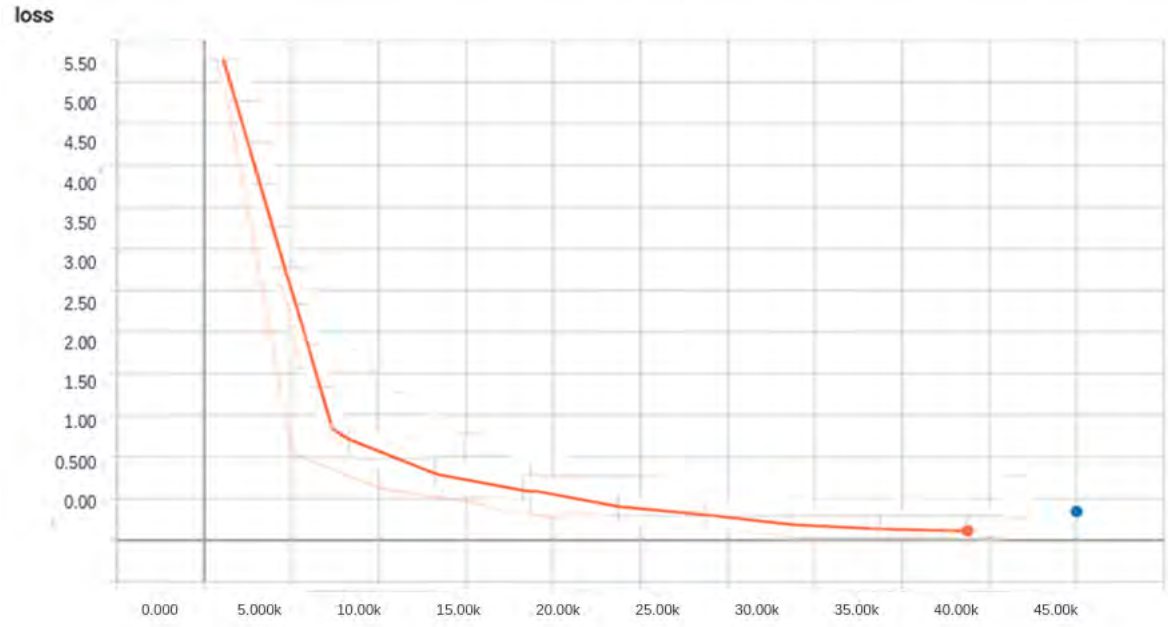


Figure 5.11: Faster RCNN Inception V2 loss

Now let's take a look at the heat map of our annotated images if we convert those into heat maps. Heat map shows us that where it's red are the classes we have annotated. In the figure 5.12 we can see that there are 3 red spots in the map. In this case the human has no helmet. That's why the red parts are for human, number plate and bike.
Now in the figure 5.13 there are 3 bikers. If we see the heatmap we can see that there are 3 different parts with each having their own red parts. In the middle one there is no helmet on the biker so there is no red part on top of human. But in the

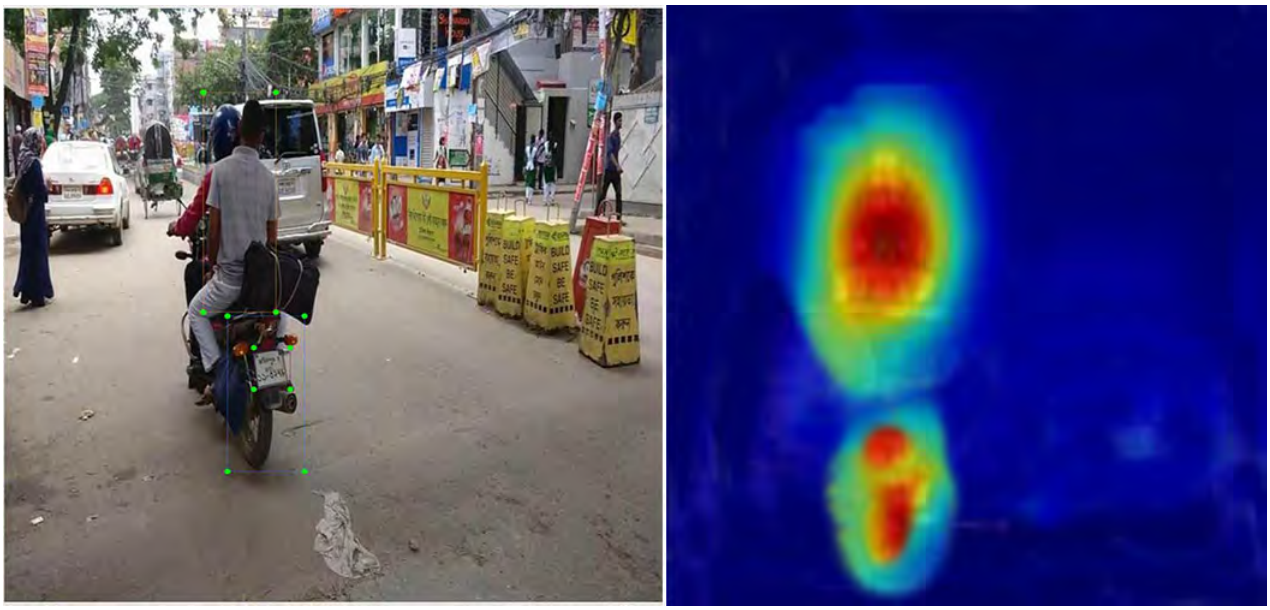


Figure 5.12: Heatmap of annotated image

other 2 parts there are helmets on bikers which seen clearly by the heatmap.

Figure 5.13: Heatmap of annotated image with multiple bikers

Now let's move on to the number plate part. We have done the number plate recognition with Tesseract. Machines learn by means of a loss function. It's a method of evaluating how well specific algorithm models the given data. If predictions deviates too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction. In this article we will go through several loss functions and their applications in the domain of machine/deep learning. In the figure 5.14 the graph shows the loss function graph of number plate recognition.
At the same time the accuracy gets higher with time. Here is the graph which shows the accuracy is getting high with time. The number plate recognition system is not

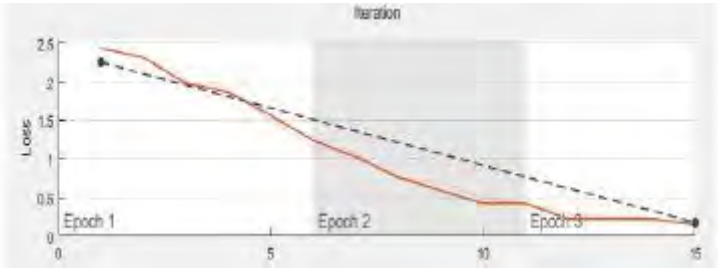


Figure 5.14: Number plate loss function

that accurate still. So to make it accurate the system does overfitting. Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study.

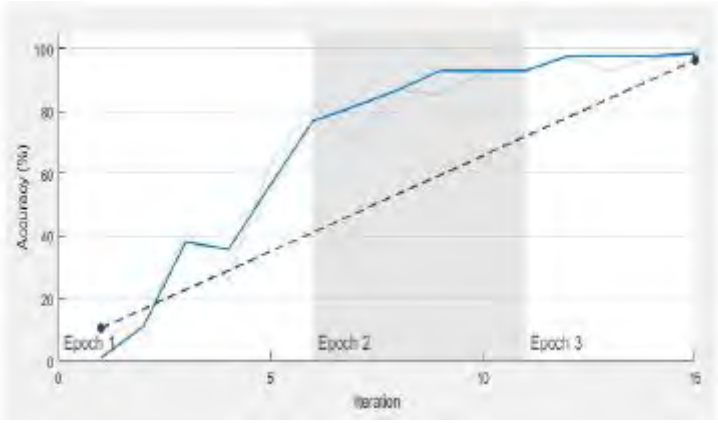


Figure 5.15: Number plate accuracy

Here is a mAP comparison between the 2 models SSD Mobilenet V2 and Faster RCNN Inception V2 -

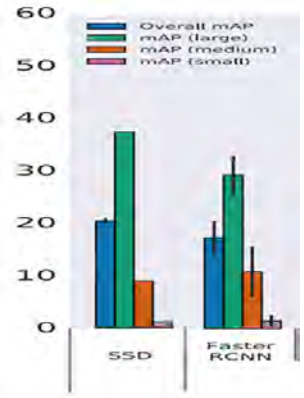


Figure 5.16: mAP comparison

Chapter 6

Conclusion

We cannot stop the road accidents but we have the power to minimize it. Law for safety measures like wearing helmet has already been established. Only we need proper monitoring to enforce the law accurately. Our system provides a ready solution for the this scenario. We can see there are CCTV cameras all over the streets. If we properly place them in different angles from where we can get the footage which can be used to detect bikers without helmets then we can easily identify the law offenders by detecting their number plate. We can easily get their registration number from their number plate and this information can be used for further punishment/fine. Our goal is to minimize the major part of road accidents happening at present which is bike accidents. If we can apply our system properly then we can save more lives than we can think. From our experiment with deep learning algorithms it's safe to say that the day is not too far when we can use the detection system for smart surveillance in real time and safety awareness will definitely increase.

References

[1] S. Correspondent, *Growing number of motorbikes to blame, nischa survey finds*, Aug. 2019. [Online]. Available: https://www.thedailystar.net/country/road-accident-in-bangladesh-2018-survey-blames-rising-of-bikers-1641166.

[2] J. E. Espinosa Oviedo *et al.*, "Detection and tracking of motorcycles in urban environments by using video sequences with high level of oclussion", PhD thesis, Universidad Nacional de Colombia-Sede Medellı´n.

[3] K. Dahiya, D. Singh, and C. K. Mohan, "Automatic detection of bike-riders without helmet using surveillance videos in real-time", in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016, pp. 3046–3051.

[4] G. R. Jadhav and K. J. Karande, "Automatic vehicle number plate recognition for vehicle parking management system", *Computer Engineering and Intelligent Systems, ISSN*, pp. 2222–1719,

[5] J. Bagade, M. Kamble, K. Pardeshi, B. Punjabi, and R. Singh, "Automatic number plate recognition system: Machine learning approach", *IOSR Journal of Computer Engineering*, vol. 1, pp. 34–39, 2013.

[6] C. Vishnu, D. Singh, C. K. Mohan, and S. Babu, "Detection of motorcyclists without helmet in videos using convolutional neural network", in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 3036–3041.

[7] R. Silva, K. Aires, T. Santos, K. Abdala, R. Veras, and A. Soares, "Automatic detection of motorcyclists without helmet", in *2013 XXXIX Latin American Computing Conference (CLEI)*, IEEE, 2013, pp. 1–7.

[8] eshant sah, *Convolution -in deep learning*, Jan. 2019. [Online]. Available: https://medium.com/@eshant.sah/convolution-in-deep-learning-62eb08b42070.

[9] M. Hollemans, *Mobilenet version 2*, Aug. 2019. [Online]. Available: https://machinethink.net/blog/mobilenet-v2/.

[10] G. Yadam, *Object detection using tensorflow and coco pre-trained models*, Nov. 2018. [Online]. Available: https://medium.com/object-detection-using-tensorflow-and-coco-pre/object-detection-using-tensorflow-and-coco-pre-trained-models-5d8386019a8.

[11] R. Smith, "An overview of the tesseract ocr engine", in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, IEEE, vol. 2, 2007, pp. 629–633.

[12] tuorialspoint, *Opencv - overview*, 2019. [Online]. Available: https://www.tutorialspoint.com/opencv/opencv_overview.

[13] github, *Tensorflow*, 2019. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

[14] M. T. Shahed, M. R. I. Udoy, B. Saha, A. I. Khan, and S. Subrina, "Automatic bengali number plate reader", in *TENCON 2017-2018 IEEE Region 10 Conference*, IEEE, 2017, pp. 1364–1368.