A PROJECT REPORT
ON

LICENSE PLATE RECOGNITION USING DEEP LEARNING METHODS

Submitted to
UNIVERSITY OF PUNE

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN ELECTRONICS AND TELECOMMUNICATION ENGINEERING

BY

MIHIR INGOLE	B150433054
ANANGSA BISWAS	B150433007
SAMRUDDHI DIDAKE	B150433037
AGNIJ DAVE	B150433033

UNDER THE GUIDANCE OF
Dr.Mrs.P.S.Deshpande

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING
Sinhgad Accademy Of Engineering
LOCATION IN PUNE, PUNE - 411048

2018-2019

AFFILIATED TO
UNIVERSITY OF PUNE

A PROJECT REPORT

ON

LICENSE PLATE RECOGNITION USING DEEP LEARNING METHODS

Submitted to

UNIVERSITY OF PUNE

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN ELCTRONICS AND TELECOMMUNICATION ENGINEERING

BY

MIHIR INGOLE	B150433054
ANANGSA BISWAS	B150433007
SAMRUDDHI DIDAKE	B150433037
AGNIJ DAVE	B150433033

UNDER THE GUIDANCE OF

Dr.Mrs.P.S.Deshpande

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Sinhgad Academy Of Engineering

2018-2019

AFFILIATED TO

UNIVERSITY OF PUNE

Sinhgad Academy Of Engineering

Department of Electronics And Telecommunication Engineering LOCATION IN PUNE, Pune 411048

Sinhgad Institutes

CERTIFICATE

This is certify that the project entitled

LICENSE PLATE RECOGNITION USING DEEP LEARNING METHODS

submitted by

MIHIR INGOLE	B150433054
ANANGSA BISWAS	B150433007
SAMRUDDHI DIDAKE	B150433037
AGNIJ DAVE	B150433033

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Electronics And Telecommunication Engineering) at Sinhgad Academy Of Engineering, Pune under the University of Pune. This work is done during year 2018-2019, under our guidance.

Date: / /

(Dr. Mrs.P.S.Deshpande)
Project Guide

(Prof.S.S.Shaha)
Project Coordinator

(Dr. K.M.Gaikwad)
HOD, ENTC

(Dr. K. P. Patil)
Principal

External Examiner

Acknowledgements

We are profoundly grateful to Dr. Mrs.P.S.Deshpande for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards Dr. K.P.Patil, Princi-pal, Sinhgad Academy Of Engineering, Dr. K.M.Gaikwad, Head of Department of Electronics And Telecommunication Engineering and Prof.S.S.Shaha, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

MIHIR INGOLE
ANANGSA BISWAS
SAMRUDDHI DIDAKE
AGNIJ DAVE

ABSTRACT

License Plate detection and recognition is one of the crucial methods for supervising the vehicle movements. Several studies have been carried out over the years on this concept, despite that theres possibility of enhancement as there are various factors which governs the characteristics of the image captured. Here we present an automatic system for LPs detection and recognition based on deep learning approach. It comprises of three parts as detection, extraction and optical character recognition of the LPs. A faster R-CNN Model is used for detecting LPs and drawing bound-ary boxes around it. Once a LP is detected in an image it is further extracted using Pytesseract which is an Optical Character Recognition (OCR) tool for python. That is, it will recognize and read the text embedded in images. Here TensorFlow’s Object Detection API is used to train the object detection classifier on Windows 10. This technology can be implemented in parking lots and toll booth management systems. It can also be used in Law Enforcement Systems and Traffic Control.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 OVERVIEW

Automatic License Plate Recognition (ALPR) helps to identify vehicle license plates in an efficient manner without the need for major human resources and has become more and more important in the recent years. There are a growing number of cars on the roads and all of them have license plates. The rapid development in digital image processing technology has also made it possible to detect and identify license plates at a fast rate.

Identification of vehicles is useful for many different operators. It can be used by government agencies to find cars that are involved in crime, look up if annual fees are paid or identify persons who violate the traffic rules. U.S., Japan, Germany, Italy, U.K and France are all countries that have successfully applied ALPR in their traffic management. Several private operators may also benefit from ALPR systems.

This project will present an alternative approach by using deep learning to automatically recognize license plates. Deep learning techniques do not use hand-engineered features, but automatically select the features themselves. The strongest deep learning methods involve Convolutional Neural Networks (CNNs). CNNs hierarchical neural networks are based on sparse connections and weight sharing giving them an immense representational capacity and high learning potential.

There are various algorithms that can be used for object detection. However the Faster R-CNN family provides better accuracy as compared to other algorithms like Single Shot Detector(SSD) and Region based Fully Convolutional Network (R-FCN).[5] Faster RCNN is the modified version of Fast RCNN. We have used the deep learning library Tensorflow[1] for our implementation. The main reason for using tensor flow is because it is an open source library and allows developers to create large scale neural networks with many layers.

1.2 MOTIVATION

The main motivation of this project is to to take into consideration the techno-logical advancements of this generation and implement it for efficient management and supervising of vehicles. This system tackles the problem of mismanagement and excess time wastage incurred due to involvement of manual labor. Furthermore, it provides information about the vehicles entering an area which is valuable to the concerned authorities. This ultimately helps in cost reduction and time management along with proper supervision in various sectors as compared to the old approach.

1.3 PROBLEM DEFINITION

To develop an ALPR system using deep learning techniques and evaluate to what extent it meets the prediction accuracy requirements of the ALPR task.

1.4 OBJECTIVES

The objectives of our project are:

1. To understand and implement the working of CNNs for image processing.
2. To successfully detect a LP in an image and extract the LP number from the image.
3. To compare few object detection algorithms and implement the most suitable model for the application.

Chapter 2

SCOPE OF THE PROJECT

The ALPR problem is a complex task with many variations. Images are taken by a color camera (RGB format). Furthermore, nations have different standards for li-cense plates allowing different background colors or distracting patterns on the plate. The number of letters, numbers and spaces, their position and the font used are other variations.

The scope of this project is to be able to detect license plates appearing on the roads. This means that the most important part of the system is to be able to detect a license plate with high accuracy. An ideal example of a license plate is shown in Figure 1.1. However stylish plates do not follow the standardized pattern of license plates. It is also within the scope of this project as they appear on the roads. The problem can be solved by including many variations of LPs in the training and testing dataset.

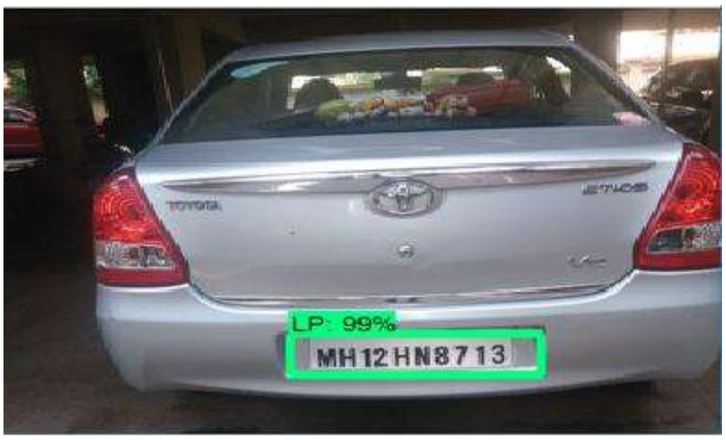


Figure 2.1: Example of Ideal License Plate Detection

Even though most cases will be of standard license plates, there are a number of variations causing challenges when detecting and recognizing these plates listed below.

1. Background color: While is most commonly used as background color. How-ever, special vehicles may have other background colors.

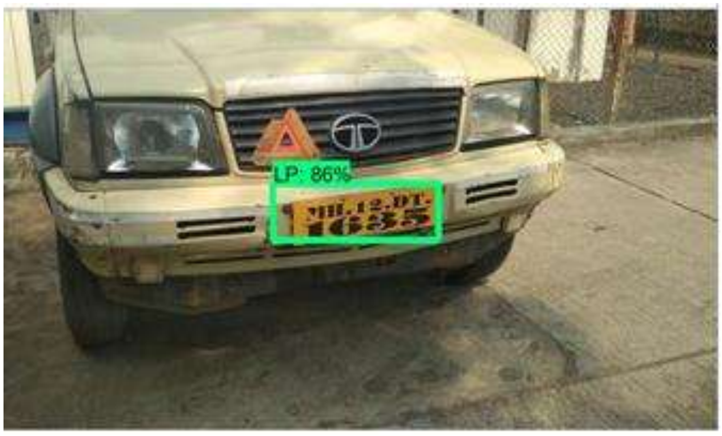


Figure 2.2: Example of Yellow License Plate Detection

Since the image of the license plate is not captured perfectly, a number of other variations to the plate may appear. These are listed below.

2. Location: The plate may be located in different parts of the image

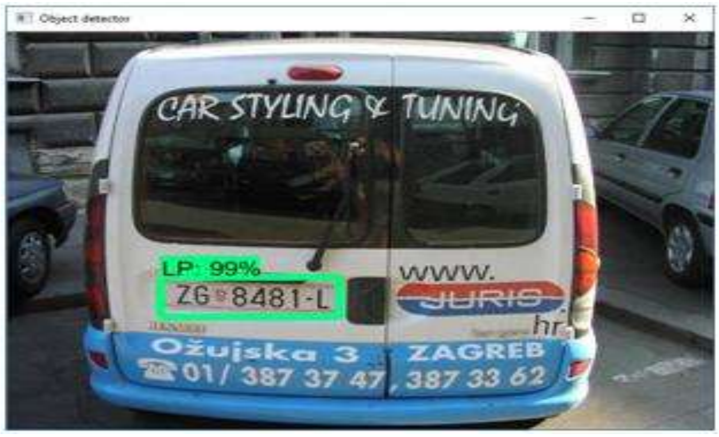


Figure 2.3: Example of Challenging condition for LP Detection

3. Size: The size of the plate compared to the full captured image differs based on camera distance and zooming.

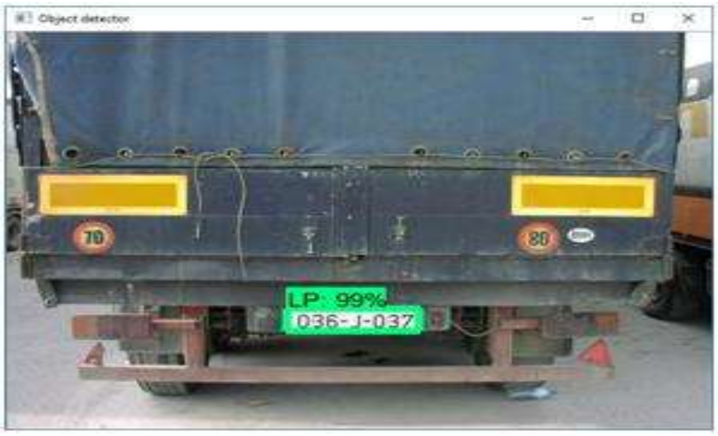


Figure 2.4: Example of Challenging condition for LP Detection

4. Rotation: The plate may be tilted based on the angle the image is taken.

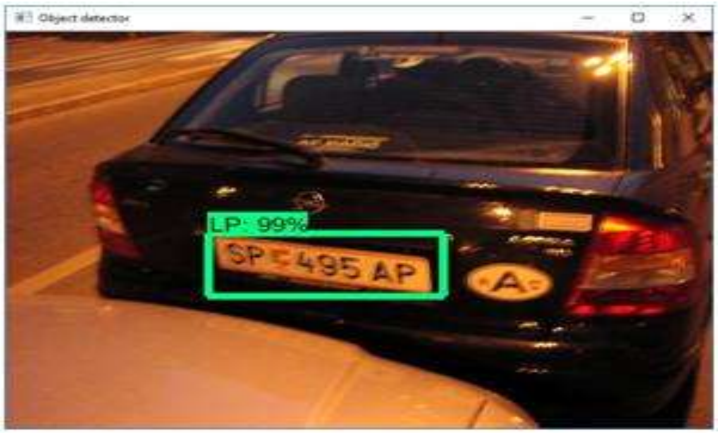


Figure 2.5: Example of rotated LP being Detected

5. Occlusion: The plate may be obscured by additional noise, such as dirt or snow.

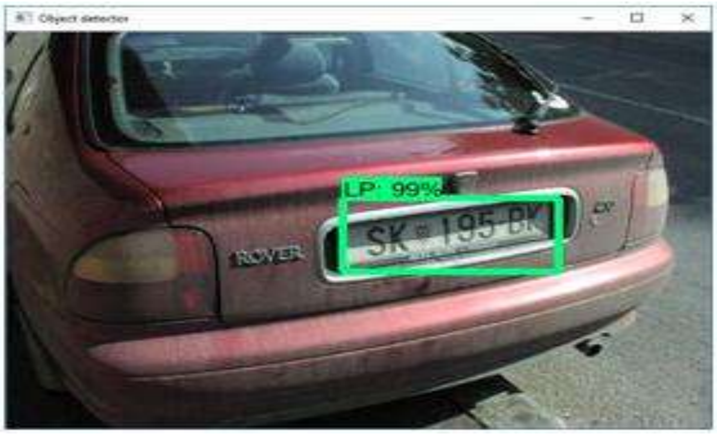


Figure 2.6: Example of dirty LP being Detected

6. Quantity: An image may have captured none, one or multiple plates.

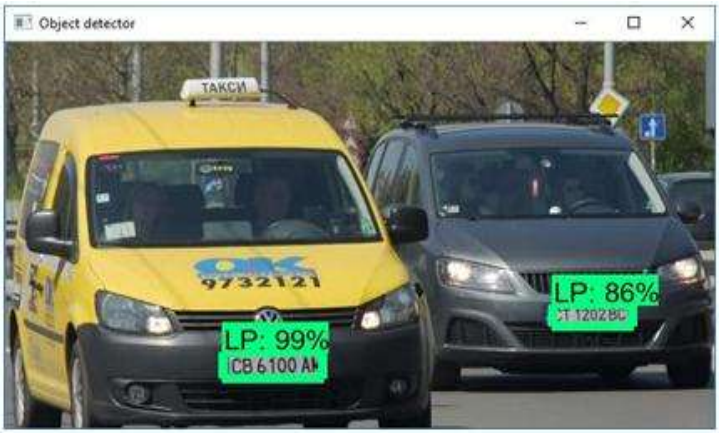


Figure 2.7: Example of multiple LPs being Detected

7. Miscellaneous: The plate may contain screws, frames and stickers.

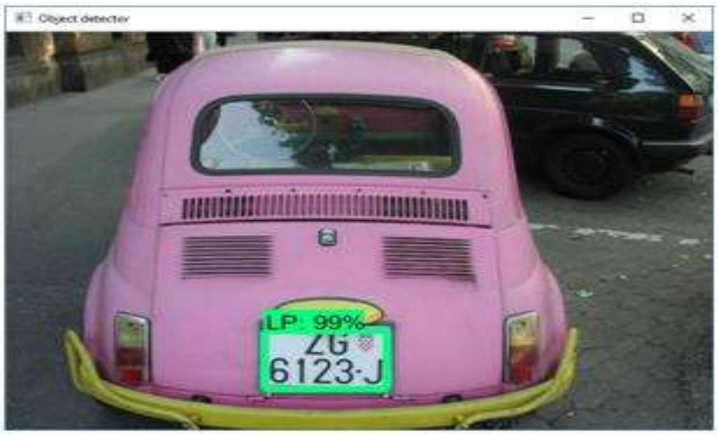


Figure 2.8: Example of LP with sticker

Lastly, a number of environmental variations given by the following list may also be a challenge.

8. Illumination: The illumination may differ based on environmental light, cam-era light or vehicle light.

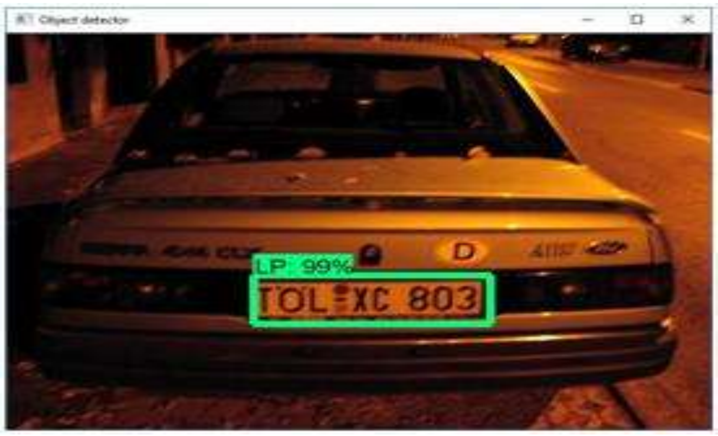


Figure 2.9: Example of LP detection in low light

9. Shadows: Objects close by may cast shadows on the license plate making the illumination on part of the plate differ from the rest.

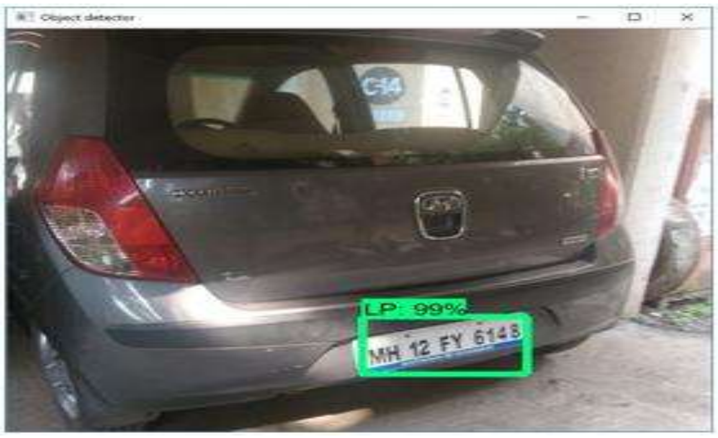


Figure 2.10: Example of LP being partly illuminated by sunlight

Chapter 3

LITERATURE REVIEW

Various image processing techniques can be used to enhance the captured image and thereby increase the recognition rate [1]. Image normalization, de-nosing, filtering, histogram equalization, image resizing and cropping and accurate face detection are certain techniques to enhance image quality and improve recognition rate. Image preprocessing is done prior to extraction of features from the image. Especially when images are of different size, have lighting variations, noisy images shows a greater impact on recognition rate. In such cases image has to be preprocessed ap-propriately. This approach has also shown good results even without preprocessing prior to the feature extraction.

In [2] image processing is used for facial recognition. A biometric is observed data of a human that allows the identity of that person to be determined. Examples of biometrics actively being investigated are DNA, shape of the ear, faces, fingerprints, hand geometry, irises and pattern of keystrokes on a keyboard, signature and speech. In verification also referred to as authentication the algorithm is presented with a biometric image and a claimed identity of the person. The algorithm either accepts or rejects the claim. Or the algorithm can return a coincidence measure of the valid-ity of the claim. Recognition from irises is based on wavelet features derived from the texture patterns of the iris. Like fingerprints the irises are phonotypical features are a function of the interaction of genetics environment and development whereas faces are genotypical features are primarily genetically inherited.

Tremendous progress has been made in image recognition, primarily due to the availability of large-scale annotated datasets and the recent revival of deep convolu-tional neural networks (CNN). For data-driven learning, large-scale well-annotated datasets with representative data distribution characteristics are crucial to learning more accurate or generalizable models. In [3], they have exploited three important, but previously under-studied factors of employing deep convolutional neural networks to computeraided detection problems. Particularly, we explore and eval-uate different CNN architectures varying in width (ranging from 5 thousand to 160 million parameters) and depth (various numbers of layers), describe the effects of varying dataset scale and spatial image context on performance, and discuss when and why transfer learning from pretrained ImageNet CNN models can be valuable.

In [4] new hybrid method of feature extraction is presented in which total 91 fea-tures are used for recognition. These features are combination of structural or geo-metric features, regional features; gradient features and distance transform features. In existing case, previously combination of structural features, regional features and moment features were used, however there is more time required for structural fea-tures extraction. In this paper structural features are optimized by using Universe of discourse over input segmented image, which can speed up the tasks of 81 feature extraction. In addition to this, proposed feature vector delivers better accuracy as compared to existing method. As proposed research area is online handwritten char-acter recognition on Devanagari script, efficient, faster and robust feature extraction method is presented. The most commonly used feature extraction methods are Gra-dient features, structural features, regional features, projection histograms, Zernike moments, zoning etc.

Conversion of given input data in to set of features are known as Feature Extrac-tion. In machine learning, Feature Extraction begins with the initial set of consistent data and develops the borrowed values also called as features, expected for being de-scriptive and non-redundant, simplifies the consequent learning and observed steps. The process by which a different selective feature is attained from those accessible input data. Classification is done by using the different group of features. This is the task which is used to achieve attributes which are peculiar. It is a mechanism for extracting consistent data from the image. A dimensionality contraction approach that identifies a decreased set of features that are the sequence of the initial ones. The process of obtaining relevant attributes that encloses with-in the given input data. Feature Extraction is about converting training data and establish it with extra features in order to make machine learning algorithms much adequate. [5] presents a technique, for finding the impact of automatic feature extraction and distribution that uses in Deep Learning such as Convolutional Neural Network (CNN).

A lot of researches has been done in developing new classifiers in particular classifiers which can learn and adapt to new conditions with minimal parameters/model changes. However, the performance of a classifier with a set of parameters can perform better in one application but may perform extremely poor in other real-world applications which leads the researchers to move and develop new methods which can perform better across different datasets. [6] presents a research methodology to identify the impact of automatic feature extraction and classification used in deep learning such as CNN. An approach has been proposed to systematically analyze the classification accuracy of CNN, image, and feature based traditional MLPs. CNN with automatic feature extraction was firstly evaluated on a well established MNIST dataset and then a traditional Multi-Layer Perceptron (MLP) with full image, and a manual feature extraction were evaluated on the same benchmark dataset.

Vehicle detection methods have been proposed to identify the object as a vehicle. It can be used for multiview vehicle tracking and verification. Convolution neural networks (CNN) has been proposed that shows considerably high image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). CNN can also be applied for object detection, such as R-CNN, which shows a great improvement on object detection accuracy compared with the conventional feature-based detectors. [7] presents a simple and fast method by modifying fast R-CNN for vehicle detection and localization. The CNN and max pooling provide sparse object representation of the ROI for the following fully connected layer to classify the ROI.

In [8], a Deep CNN based human target detection and recognition scheme is pro-posed. First, a New Region Proposal Network (NRPN) method was introduced to improve the VGG model and a new VGG model was built on the basis of current Fast-RCNN algorithm. And then, training the model with supervised training. Fi-nally, using the trained model to finish the detection and recognition. Deep Learning method is applied to human feature extraction and human detection in the context of low recognition rate of HOG+SVM algorithm in complex background. A NRPN model was added on the basis of Fast-RCNN algorithm and the database were ex-tended based on the Daimler and INRIA.

In [9],focuses on the algorithmic modification of Faster RCNN. A preprocessing method is integrated to faster RCNN to improve training and detection speed. The preprocessing pipeline reduces the region of interest which results in a reduced number of pixels to be processes by faster RCNN. The preprocessing pipeline is based on the Sobel edge detection and Hough transform with an extracted rectangular region as the end result. This improves the processing speed of the Faster RCNN. A vehicle detection method is proposed that reduces a region of interest by preprocessing an image before it is processed by faster RCNN. The results show that preprocessing an image into a reduced rectangular portion of the original image improves training speed. The image is processed faster as a result of the rectangular cropped image based on lane detection.

According to [10] classical object detection based on sliding windows, multi-scale and/or cascade approaches have shown successful results on several applications. Recently, new approaches based on deep learning outperformed the classical methods in various applications. Region-based convolutional network (RCNN) is one of the state art detection methods. The conventional Faster-RCNN is composed of 5 main parts: a deep fully convolutional network, region proposal network, ROI pool-ing and fully connected networks, bounding box regressor, and classifier. Through the deep fully convolutional and region proposal networks, a lot of object candi-dates are proposed and the candidate regions (proposals) are normalized through ROI Pooling layer. Then, fully connected layers extract good features to conduct classification and regression. The proposed RFRCNN is applied to human and li-cense plate detection, and as a result, it outperforms Faster-RCNN on both detection performance and preciseness of detected bounding box.

The variations of the plate types or environments cause challenges in the detection and recognition of license plates. [11] presents a comprehensive survey on exist-ing ALPR techniques by categorizing them according to the features used in each stage. Comparisons of them in terms of pros, cons, recognition results, and process-ing speed are addressed.

Also [12] describes the image preprocessing algorithm, license plate location algorithm, license plate segmentation algorithm and character recognition algorithm. The entire algorithms uses edge detection, image erosion operation, image clustering, rand transform, morphological processing, window searching and BPNN (back propagation neural network) technologies etc. Especially, this paper detail describes the processing of the training model generated by tool of BPNN. The training model is the key of the character recognition algorithm about license plate. The image preprocessing algorithm, license plate location algorithm, license plate segmentation al-gorithm not only are prepared for generating training mode, but also are prepared for character recognition algorithm. The training model is generated using the character segmentation images by BPNN. The training mode is the key of entire algorithms in this paper, the tool of BPNN is the important method. Training model is involving in parameters configuration of neural network function.

In [13] we compare the training and testing times, with more focus on the testing times of the different model combinations and what difference we obtain with dif-ferent hyperparameter tuning. The fastest model combination was SSD MobileNet trained on a low resolution image of 300, the slowest combination was Faster R-CNN Inception Resnet trained on a resolution of 600.

Chapter 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

The figure below shows the system architecture of the project.

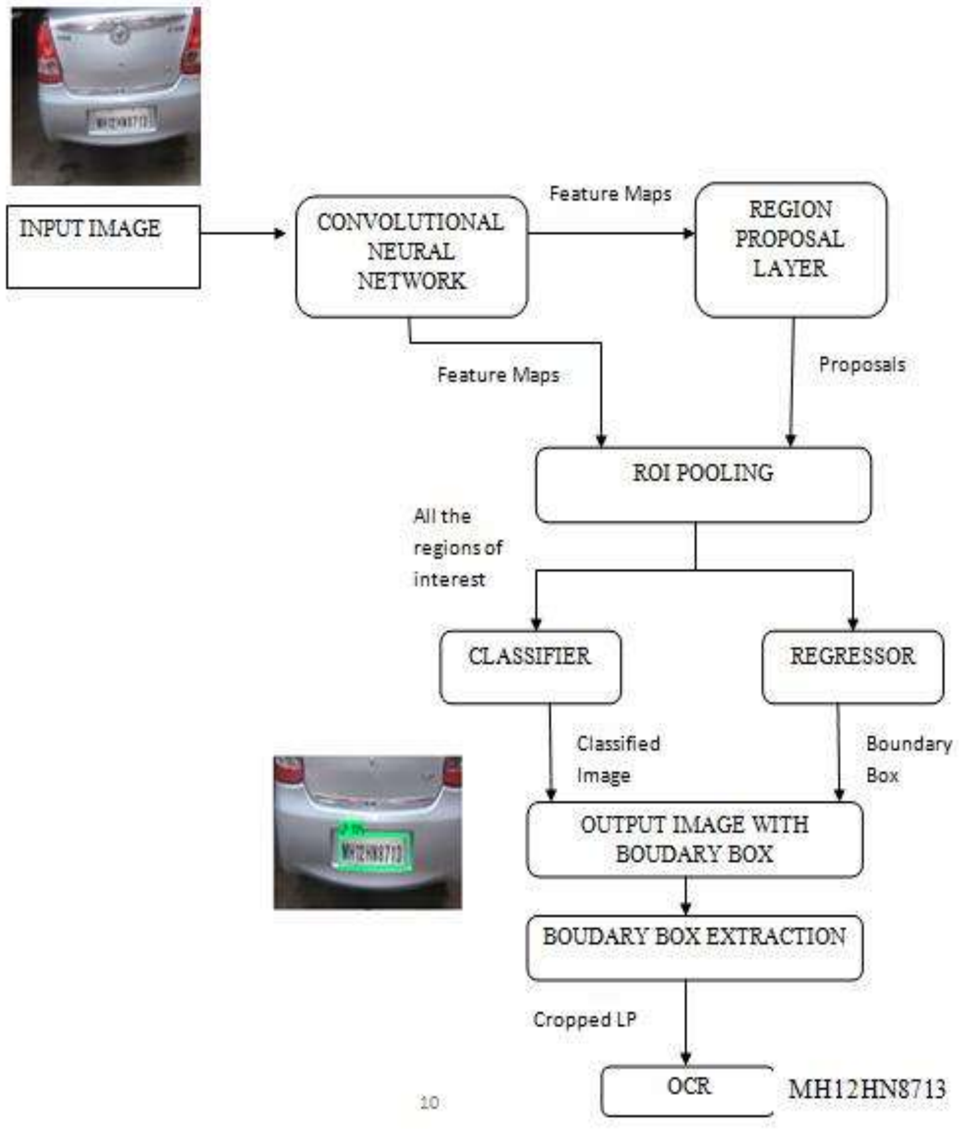


Figure 4.1: System Architecture

How It Works?

- Run the image through the CNN to get a feature map.
- Run the Activation Map through a separate network, called the Region Proposal Network (RPN), that outputs interesting boxes/regions.
- For the interesting boxes/regions from RPN use several fully connected layer to output class + bounding box coordinates.
- Extract the bounding box in the image that represents the LP.
- Finally convert the cropped image into characters or numbers using a method called Optical Character Recognition.

4.2 SYSTEM FLOWCHART

The working of the system can be divided into four parts:

Input Image- The image can be taken by any digital camera or CCTV footage. It is taken in RGB format.

LP Detection-A pre-trained detection model Faster RCNN Inception V2 has been used to detect the LP inside the image. It uses a faster R-CNN for object detection. First the captured image is passed through a CNN to obtain its fea-ture maps. The feature maps are then sent through region proposal networks to obtain region of interests/proposals. These proposals are then sent to the Region of Interest Pooling(ROIP) to bring down all the proposals to the same size. Finally, these proposals are passed to a fully connected layer in order to classify and predict the bounding boxes for the LP.

LP Extraction- As shown in Fig 4.1 the LP is highlighted by the bounding box. Using tensorflow and OpenCV we calculate the position of the bounding box. Hence we can extract the LP by cropping everything in the image except the bounding box.

Optical Character Recognition(OCR)- Here Pytesseract acts as a tool to con-vert the image into a string and hence extract the text embedded in the image. Tesseract is an optical character recognition engine. In this project Tesseract is used with python bindings so that the programming can be done in Python language.

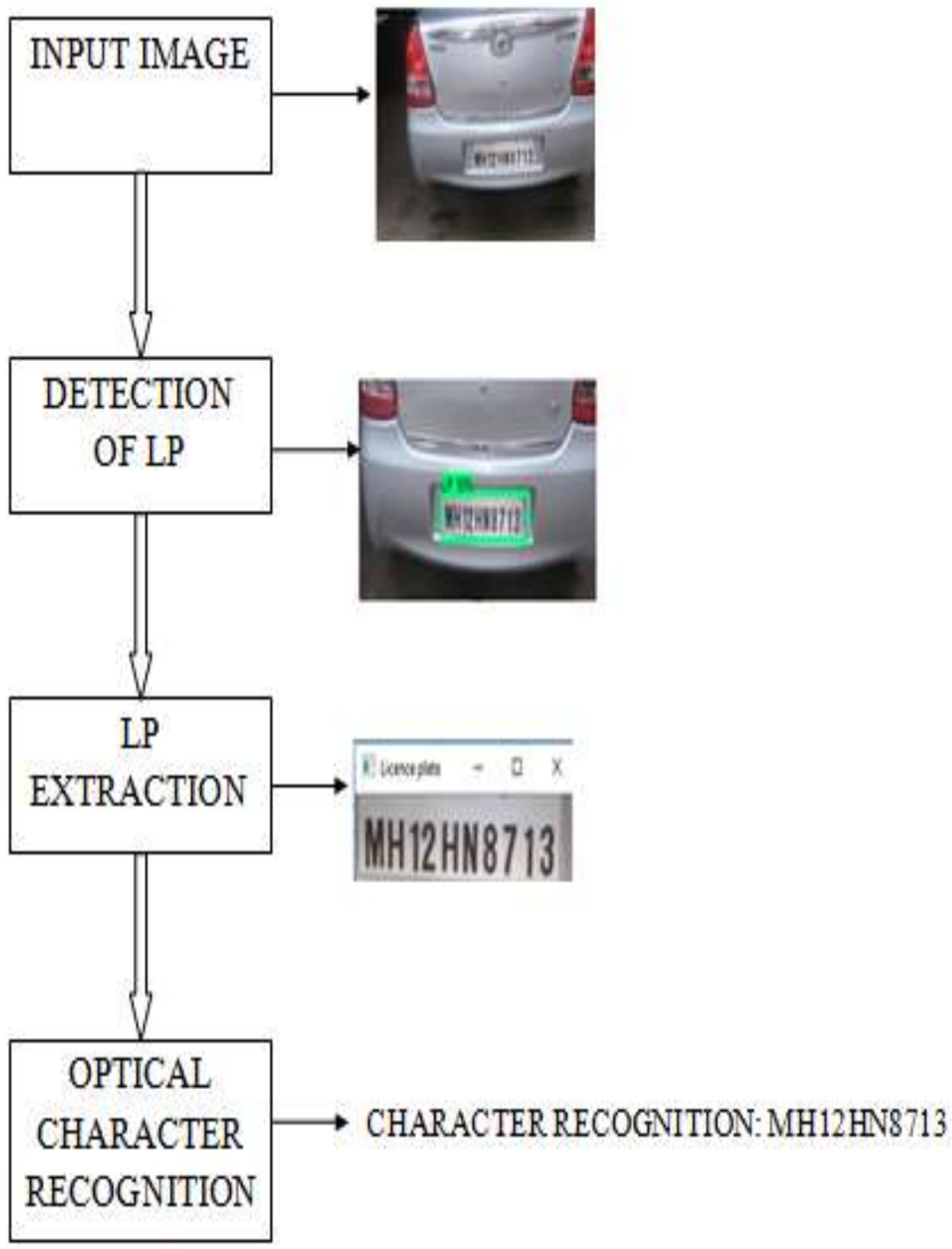

INPUT IMAGE
DETECTION OF LP
LP EXTRACTION
Licence plate
MH12HN8713
OPTICAL CHARACTER RECOGNITION
CHARACTER RECOGNITION: MH12HN8713

Figure 4.2: System Flowchart

4.3 PROJECT SETUP

4.3.1 Hardware

The environmental setup used for this project is a computer running on Windows 10. The computer specifications are Intel Core i3 2.3 GHz CPU, 4GB RAM and Nvidia GeForce 940MX 4GB GPU.

4.3.2 Software

The following list of software have been installed and used for this project.

1. Anaconda Version 3
The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. It is used for creating a virtual environment for creating the project. It comes with Python 3.7 version.

2. TensorFlow GPU
TensorFlow is an end-to-end platform that makes it easy for you to build and deploy ML models. It requires CUDA v9.0 and cuDNN v7.0 .

3. Pillow
The Python Imaging Library supports a wide variety of file formats. Over 30 different file formats can be identified and read by the library like BMP, GIF, ICO, EPS,etc. In our project we have used JPEG and PNG formats for images.

4. LXML
It is a library used for processing XML and HTML in the Python language.

5. Spyder
Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language.

6. Matplolib
Matplotlib is a plotting library for the Python programming language. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advan-

tage of being free and open-source. It is used to generate 2D plots and histograms in the project.

7. Open-CV

OpenCV (Open Source Computer Vision Library) is an open source computer vi-sion and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications. OpenCV library is used to perform the preprocessing steps on the image (eg. Conversion of RGB to grayscale, binariza-tion,etc).

8. NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. Since the images are represented in the form of matrices it is used to implement various computations and mathemat-ical functions on these images.

9. PyTesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and read the text embedded in images. Python-tesseract is a wrap-per for Google’s Tesseract-OCR Engine. It has been used in this project to recognize the characters on the LP.

10. LabelImg

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format. It is used for labeling the data before training the model.

Chapter 5

COMPARISON OF METHODOLOGIES

5.1 CONVOLUTION NEURAL NETWORK(CNN)

In neural networks, Convolution neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detec-tions, recognition faces etc., are some of the areas where CNNs are widely used.
There is an input image that were working with. We perform a series convolution + pooling operations, followed by a number of fully connected layers. If we are performing multiclass classification the output is softmax.

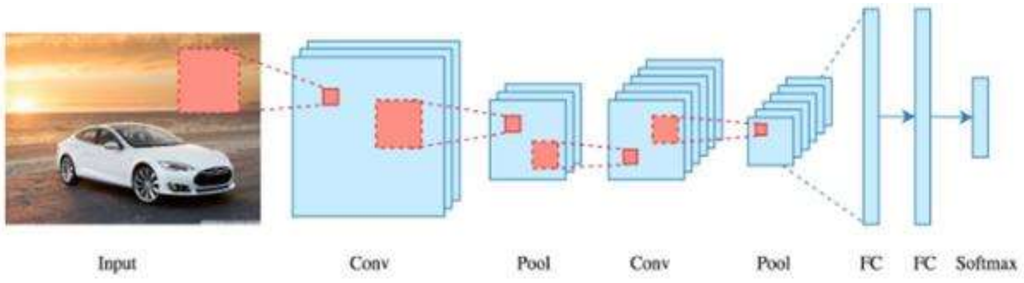


Figure 5.1: CNN Architecture

5.1.1 Object Detection using CNN

In this approach, we pass an image to the neural network and it is then sent through various convolutions and pooling layers. Finally, we get the output in the form of the objects class.
1. First, we take an image as input.
2. Then we divide the image into various regions.
3. We will then consider each region as a separate image.
4. Pass all these regions (images) to the CNN and classify them into various classes.
5. Once we have divided each region into its corresponding class, we can combine

all these regions to get the original image with the detected objects.

The problem with using this approach is that the objects in the image can have dif-ferent aspect ratios and spatial locations. The shapes of the objects might also be different (happens a lot in real-life use cases).
As a result of these factors, we would require a very large number of regions result-ing in a huge amount of computational time. So to solve this problem and reduce the number of regions, we can use region-based CNN, which selects the regions using a proposal method.

5.2 REGION-BASED CONVOLUTION NEURAL NETWORK(R-CNN)

Instead of working on a massive number of regions, the RCNN algorithm pro-poses a bunch of boxes in the image and checks if any of these boxes contain any object. RCNN uses selective search to extract these boxes from an image (these boxes are called regions).

There are basically four regions that form an object: varying scales, colors, textures, and enclosure. Selective search identifies these patterns in the image and based on that, proposes various regions. Here is how selective search works:

It first takes an image as input.

Then, it generates initial sub-segmentations so that we have multiple regions from this image.

The technique then combines the similar regions to form a larger region (based on color similarity, texture similarity, size similarity, and shape compatibility).

Finally, these regions then produce the final object locations (Region of Inter-est).

5.2.1 Object Detection Using R-CNN

The steps followed in RCNN to detect objects are as follows:

1. We first take a pre-trained convolutional neural network.
2. Then, this model is retrained. We train the last layer of the network based on the number of classes that need to be detected.
3. The third step is to get the Region of Interest for each image. We then reshape all these regions so that they can match the CNN input size.
4. After getting the regions, we train SVM to classify objects and background. For each class, we train one binary SVM.
5. Finally, we train a linear regression model to generate tighter bounding boxes for each identified object in the image.

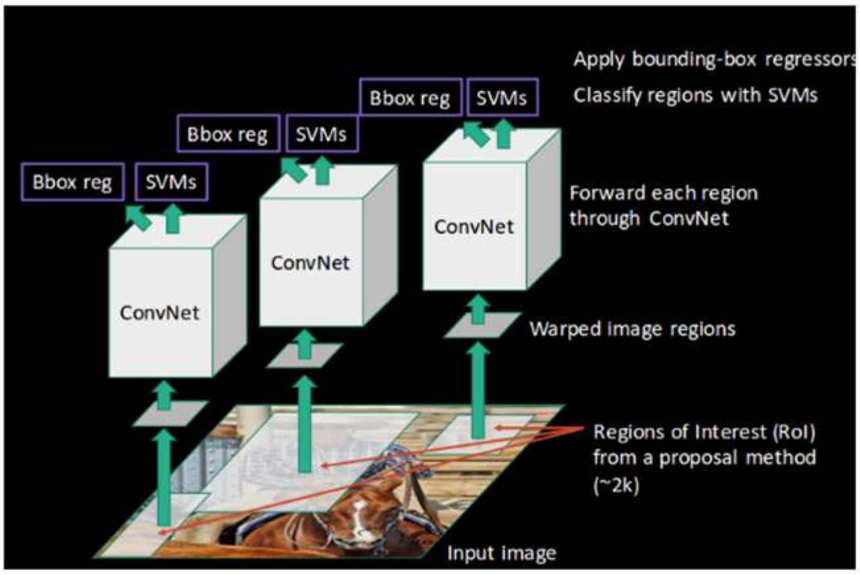

Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Figure 5.2: Working of RCNN

But this technique comes with its own limitations:

Extracting 2,000 regions for each image based on selective search.

Extracting features using CNN for every image region. Suppose we have N images, then the number of CNN features will be N*2,000.

The entire process of object detection using RCNN has three models:

1. CNN for feature extraction
2. Linear SVM classifier for identifying objects
3. Regression model for tightening the bounding boxes.

All these processes combine to make RCNN very slow.

5.2.2 Object Detection Using Fast R-CNN

Instead of running a CNN 2,000 times per image, we can run it just once per image and get all the regions of interest (regions containing some object). In Fast RCNN, we feed the input image to the CNN, which in turn generates the convolu-tional feature maps. Using these maps, the regions of proposals are extracted. We then use a RoI pooling layer to reshape all the proposed regions into a fixed size, so that it can be fed into a fully connected network. It follows the below steps for object detection:

1. We take an image as an input. This image is passed to a Convolutional Network (ConvNet) which in turns generates the Regions of Interest.

2. A RoI pooling layer is applied on all of these regions to reshape them as per the input of the ConvNet. Then, each region is passed on to a fully connected network.
3. A softmax layer is used on top of the fully connected network to output classes. Along with the softmax layer, a linear regression layer is also used parallely to out-put bounding box coordinates for predicted classes.
So, instead of using three different models (like in RCNN), Fast RCNN uses a single model which extracts features from the regions, divides them into different classes, and returns the boundary boxes for the identified classes simultaneously.

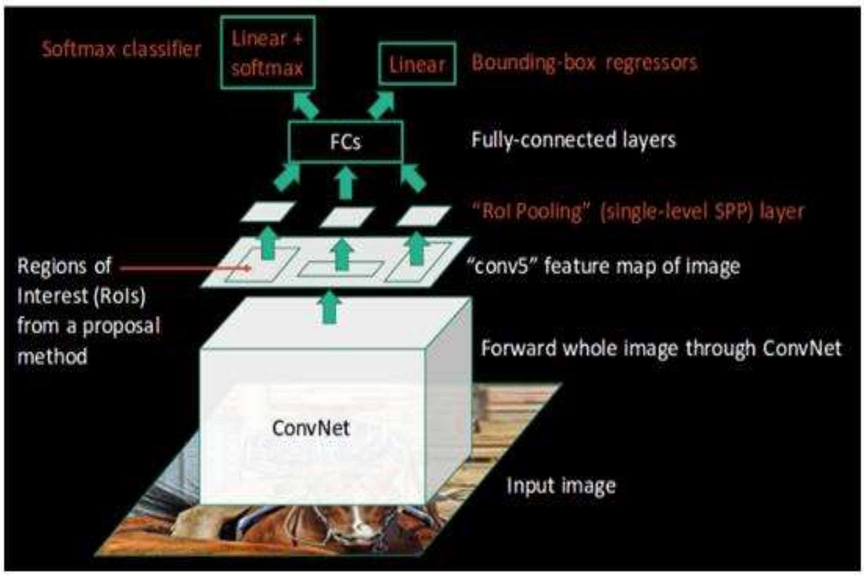

Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Figure 5.3: Working of Fast RCNN

But even Fast RCNN has certain problem areas. It also uses selective search as a proposal method to find the Regions of Interest, which is a slow and time consum-ing process.

5.2.3 Object Detection Using Faster R-CNN

Faster RCNN uses Region Proposal Network(RPN) for generating Regions of In-terest instead of selective search. RPN takes image feature maps as an input and generates a set of object proposals, each with an objectness score as output.
Faster RCNN takes the feature maps from CNN and passes them on to the Region Proposal Network. RPN uses a sliding window over these feature maps, and at each window, it generates k Anchor boxes of different shapes and sizes.

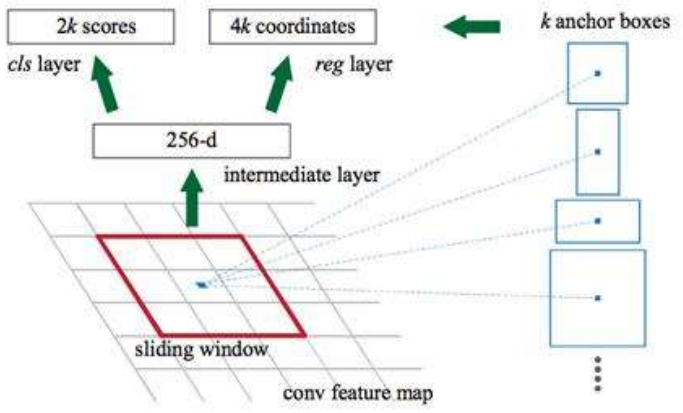


Figure 5.4: Working of RPN

Anchor boxes are fixed sized boundary boxes that are placed throughout the image and have different shapes and sizes. For each anchor, RPN predicts two things:

The first is the probability that an anchor is an object (it does not consider which class the object belongs to)

Second is the bounding box regressor for adjusting the anchors to better fit the object

The below steps are followed in the Faster RCNN approach:
1. We take an image as input and pass it to the ConvNet which returns the feature map for that image.
2. Region proposal network is applied on these feature maps. This returns the object proposals along with their objectness score.
3. A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size.
4. Finally, the proposals are passed to a fully connected layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.

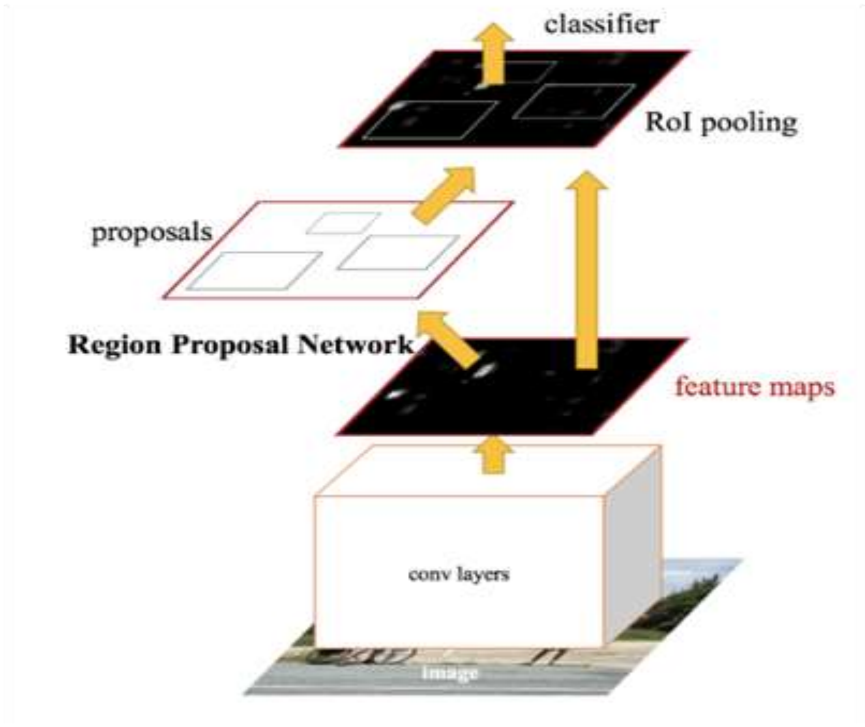


Figure 5.5: Faster R-CNN is a single, unified network for object detection

5.2.4 Summary

Algorithm	Features	Limitations
CNN	Divides the image into multiple regions and then classify each region into various classes.	Needs a lot of regions to predict accurately and hence high computation time.
R-CNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.
Fast R-CNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together	Selective search is slow and hence computation time is still high.
Faster R-CNN	Replaces the selective search method with region proposal network which made the algorithm much faster	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

Chapter 6

WORKING

6.1 WORKING OF CNN

Convolutional neural networks (CNNs) are biologically inspired neural networks first proposed by LeCun . Based on early work by Hubel and Wiesel on the cats vi-sual cortex, we know that the visual cortex has cells with small sensitive sub-regions, called local receptive fields. The cells work as local filters over the input space and respond differently based on their cell types. With supervised training on a large number of examples, the CNN may learn complex patterns mapped to high dimen-sional useful features. CNN has seen much success within the field of computer vision. Object detection, object recognition, optical character recognition , face de-tection are all tasks where using CNNs have achieved state-of-the-art results. Nor-mally the raw pixels of an image are given as input to a CNN. Consequently, CNNs require minimal amounts of preprocessing, which is a huge advantage to other ma-chine learning techniques. The drawback to this is that CNNs usually are extremely computationally intensive for large data-sets. Without preprocessing, we need more perceptrons to see patterns. This raises the time per training sample and the num-ber of epochs to reach convergence. Thus, it is in some cases beneficial to do some preprocessing to reduce the computation time though it may reduce the theoretical accuracy of the network.

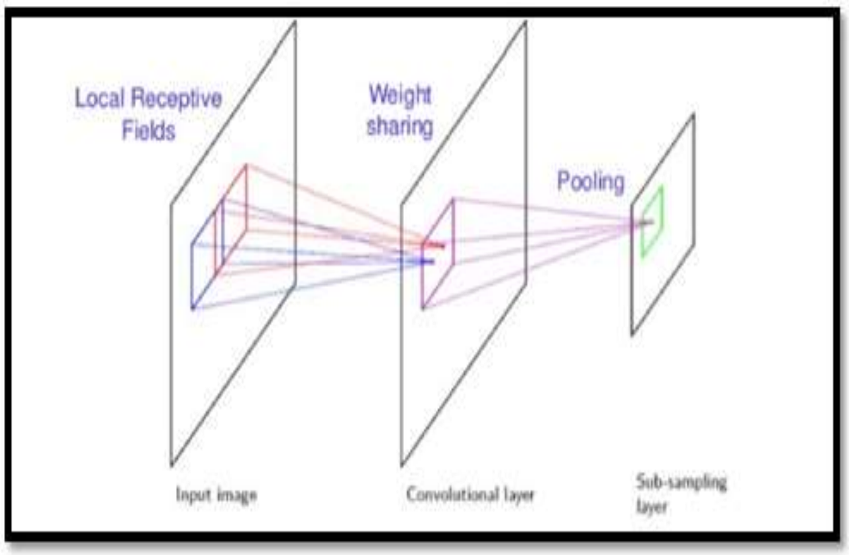


Figure 6.1: Brief overview of how the properties are connected

In general, there are three properties that makes the CNN different from stan-dard feed-forward neural networks: local receptive fields, shared weights and pool-ing.

6.1.1 Local receptive fields

a fully connected neural network is shown. All neurons in one layer are con-nected to all neurons in the subsequent layer. This is not a computational problem for a small network, but for each extra neuron added to the network, the number of connections increases exponentially. Consequently, it is computationally extremely hard to train large fully connected neural networks. However, in visual problems, it is normally not necessary to use a fully connected neural network. Pixels in an image are normally highly correlated to adjacent pixels and less correlated to pixels elsewhere in the image. This fact is exploited in CNNs. A neuron, representing a pixel, may for instance only depend on the window of n-by-n adjacent pixels sur-rounding it and will not have connections to any other neurons. This is often called the receptive field of the neuron, as it works exactly as the receptive fields in the vi-sual cortex. The receptive field is local because it only accounts for pixels nearby as shown in Figure 4.1. The projectile from the input image to the convolutional layer represents the connections and the squared n-by-n areas in the input image represent the pixels accounted for in the computation of the new neuron in the convolutional layer. When we look at the next neuron in the convolutional layer, the n-by-n win-dow of pixels slides c pixels to the right (or down). The c is called the stride length.

Stride length 1 is the most common, but other stride lengths works too.

6.1.2 Shared weights

We recall that each neuron computes the weighted linear combination of its in-puts. This calculation was based on the weight matrix W and the bias vector b. In feed-forward neural networks there are no restrictions on what weights and biases are used based on which neuron is computed. If desired, weights and biases can be defined differently for every neuron. CNNs differs that all weights and biases are shared and used independently of which n-by-n pixel window is evaluated. The shared weights and biases for a layer may be defined as kernels, also called filters. When the n-by-n window is evaluated, it is convolved with respect to the filter. The result is a convolutional response map which is given as input to the next layer. By using the same filter for all parts of the image, it is effectively detecting the same fea-tures indifferent to location in the image. If a CNN learns to respond to eyes on the left part of a face, the filter will also respond to eyes on the right part of a face. This property is called translation invariance. Filters may also learn other invariances as rotation invariance, indifference to the rotation of an object. Another advantage of shared weights is that it massively reduces the number of parameters used by the CNN. This gives more effective training and reduces overfitting.

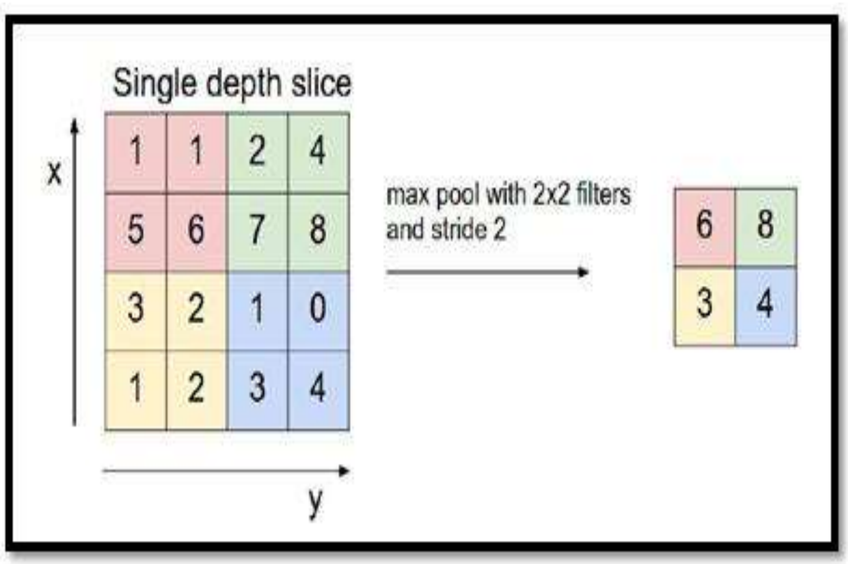


Figure 6.2: Max pooling with a stride of 2 and filter size 2-by-2.

6.1.3 Pooling

After the convolution step it is normal to use a pooling function on the resulting convolutional response map. The pooling function works by sampling sub-regions of the convolutional response map to a smaller response map. Maximum value in each sub-region in the convolutional response map. In Figure 4.2, an example of max pooling with a 4-by-4 response map is shown. It uses a 2-by-2 filter and the stride is 2. The filter is the area being considered by each pooled response and the stride is the number of pixels the filter will slide each step. The resulting pooled response map in Figure 4.2 is downsampled to a 2-by-2 matrix. Another commonly used pooling function is average pooling. It works in the same way as max pooling, except that it returns the average value of each sub-region instead of the maximum value. The benefits of using pooling are that it reduces the amount of parameters and also slightly regularizes the CNN. In the recent years, people have been more critical to the use of pooling layers and have proposed to discard it in favor of convolutional layers using larger stride . However, pooling layers are still used in most CNNs.

6.2 FASTER R-CNN INCEPTION V2 MODEL

It is a pre-trained detection model on the COCO(Common Objects in Context) dataset. Typically it runs at a speed of 58ms per 600x600 image (including all pre and postprocessing). These timings were performed using an Nvidia GeForce GTX TITAN X card on a desktop GPU. The detector performance on subset of the COCO validation set as measured by the dataset-specific mAP measures upto 28mAP.

The below architecture explains the sequence of operations performed once an im-age got its feature matrix from CNN layer.

Our object detection system is composed of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions.

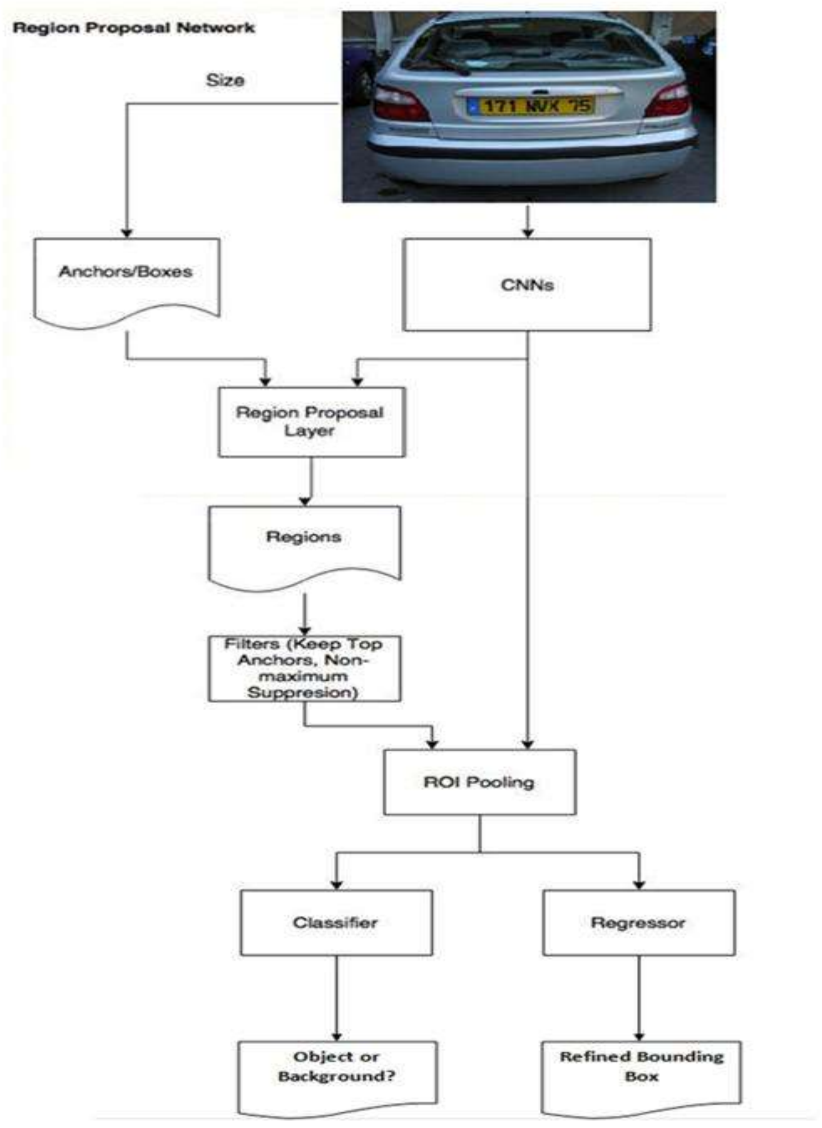


Figure 6.3: Architecture of Faster RCNN Inception V2 Model

6.2.1 Working of RPN

After getting a set of convolutional feature maps (feature matrix) on the last con-volutional layer, a sliding window is run spatially on these feature maps. The size of sliding window is nn (here 33). For each sliding window, a set of 9 anchors are generated which all have the same center (xa, ya) but with 3 different aspect ratios and 3 different scales as shown below.

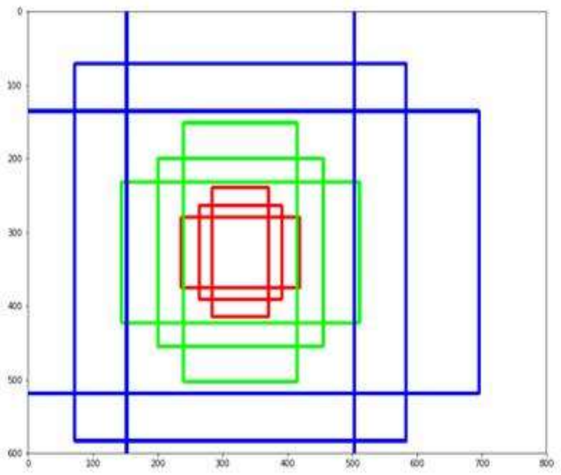


Figure 6.4: Set of anchors generated by RPN

Three colors represent three scales or sizes: 128x128, 256x256 and 512x512. The three boxes of each color have height width ratios 1:1, 1:2 and 2:1 respectively. Furthermore, for each of these anchors, a value p is computed which indicates how much these anchors overlap with the ground-truth bounding boxes (GTBox).

$$IoU = \frac{Anchor \cap GTBox}{Anchor \cup GTBox}$$

Figure 6.5

where IOU is Intersection over Union.

$$\text{IoU} = \frac{\text{A} \cap \text{Gt}}{\text{A} \cup \text{Gt}} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$

Figure 6.6

Finally, the 33 spatial features extracted from those convolution feature maps are fed to a smaller network which has two tasks: classification and regression.

The output of regressor determines a predicted bounding-box (x,y,w,h), the output of classification subnetwork is a probability p indicating whether the predicted box contains an object (1) or it is from background (0 for no object).

6.2.2 Working of Region of Interest Pooling layer (ROIP)

After RPN, we get proposed regions with different sizes. Different sized regions mean different sized CNN feature maps. Its not easy to make an efficient structure to work on features with different sizes. ROI Pooling can simplify the problem by reducing the feature maps into the same size.
The result is that from a list of rectangles with different sizes we can quickly get a list of corresponding feature maps with a fixed size.
If there are multiple object proposals on the frame (and usually therell be a lot of them), we can still use the same input feature map for all of them. Since computing the convolutions at early stages of processing is very expensive, this approach can save us a lot of time. Hence it improves the processing speed of the system.

6.2.3 Final Layer R-CNN

Region-based convolutional neural network (R-CNN) is the final step in Faster R-CNNs pipeline. After getting a convolutional feature map from the image, using it to get object proposals with the RPN and finally extracting features for each of those proposals (via ROIP), we finally need to use these features for classification. R-CNN tries to mimic the final stages of classification CNNs where a fully-connected layer is used to output a score for each possible object class. R-CNN has two different goals:
1. Classify proposals into one of the classes, plus a background class (for removing bad proposals).
2. Improve the adjusting of the bounding boxes for the proposal according to the predicted class.

6.2.4 Results of LP Detection

The object detector takes a few minutes to initialize and then displays a window showing the LP detected (using a bounding box) in the image along with a probabil-ity of the object being an LP.

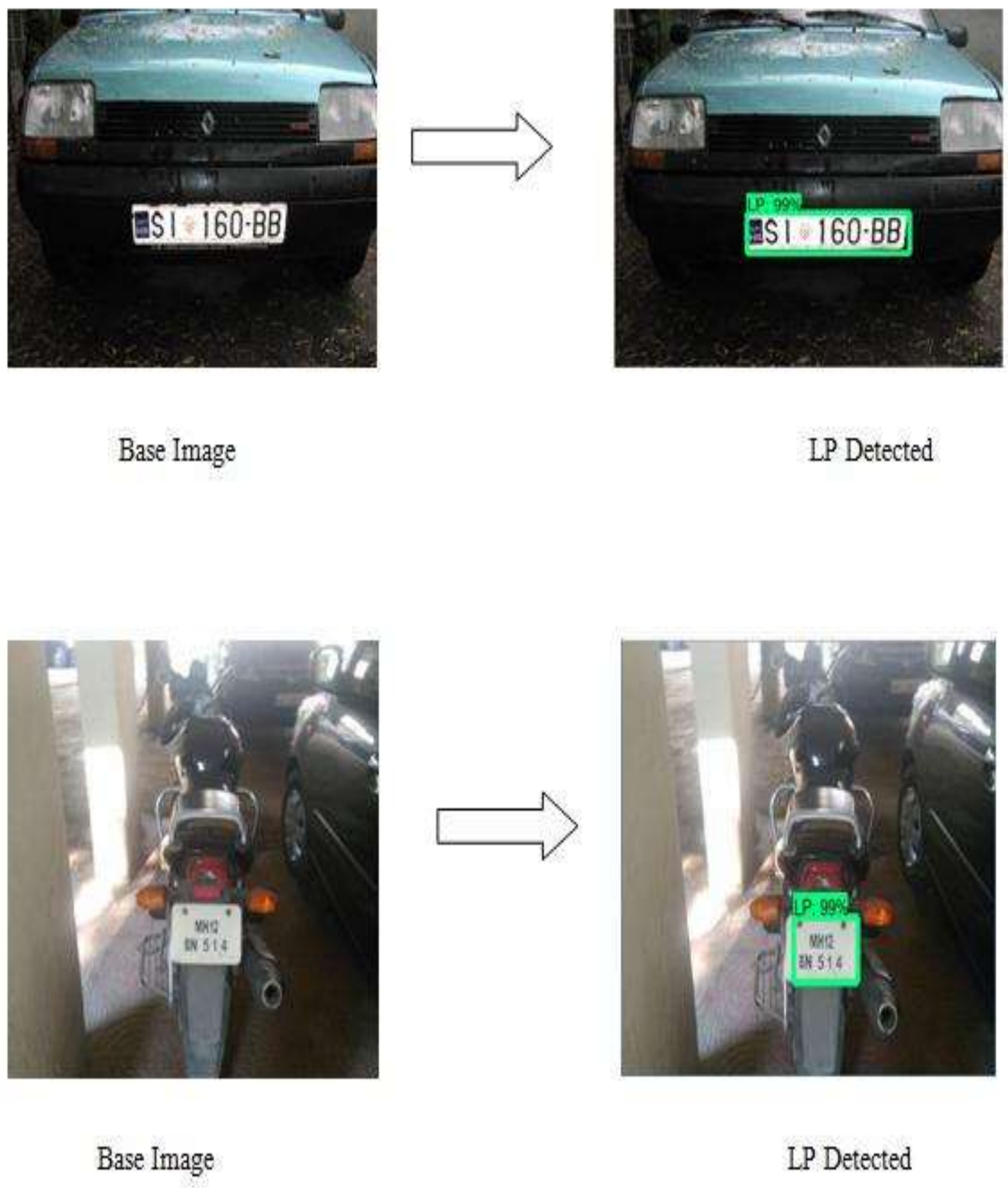


Figure 6.7: Results of LP Detection

6.3 DATASET

We have used a dataset of over 219 images of cars (front view and rear view) to train the model. The dataset comprises of various types of cars with their LPs, the images being taken from different angles and at different lighting conditions. We obtained the dataset from Kaggle which is a renowned website where one can easily obtain a public dataset.

The dataset is divided into two parts that is training and testing dataset.

The training dataset consists of 180 images. The images were resized to dimensions 640 x 480 and the LP from the images were labeled using LabelImg tool. The labels were stored as .xml files along with the images. These labels are then converted to .csv file which is used to train the model.The testing dataset consists of 39 images.

The testing dataset is used to validate the model that has been trained.

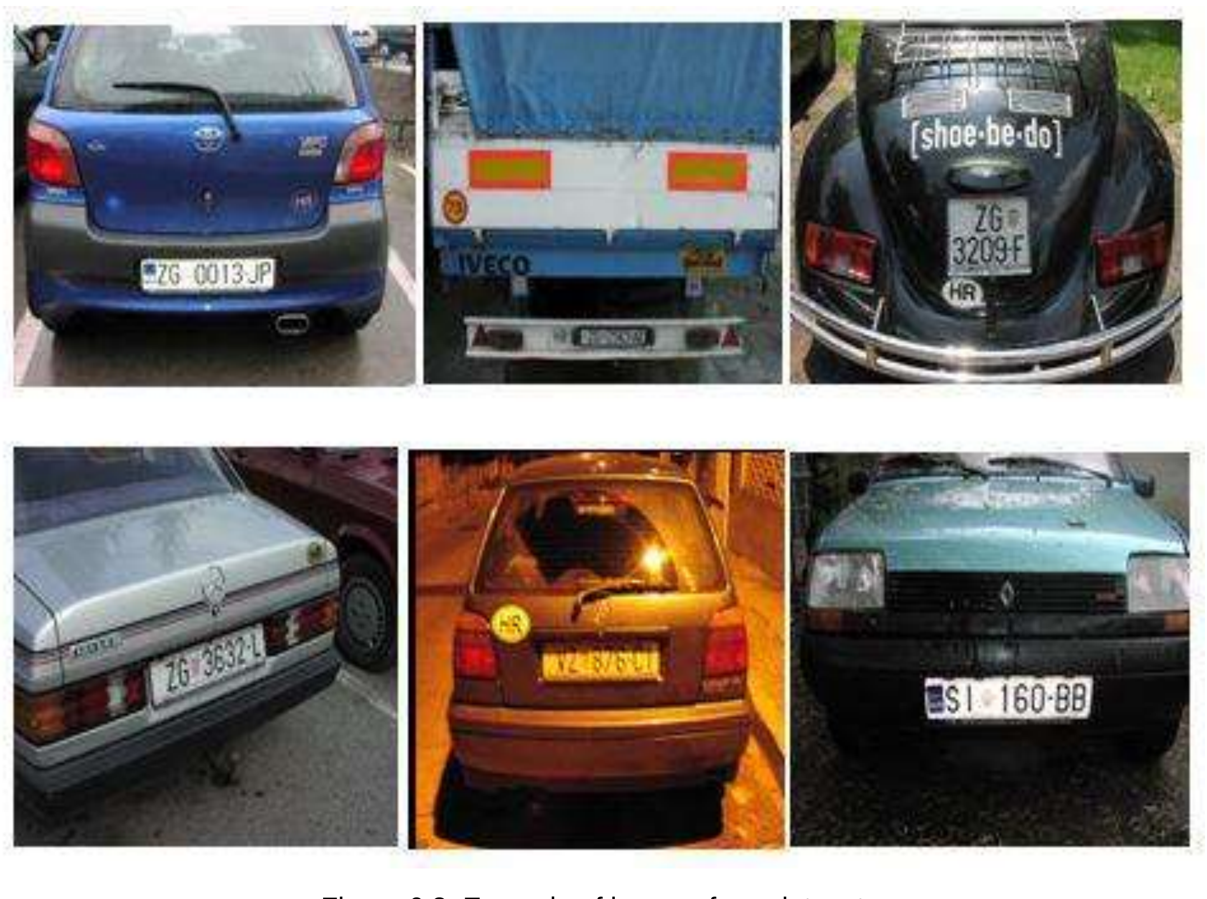


Figure 6.8: Example of images from dataset

6.4 LICENSE PLATE EXTRACTION

Once the LP has been detected in an image the next step is to extract the LP from the image in order to perform optical character recognition (OCR) on it.

Algorithm for LP Extraction:

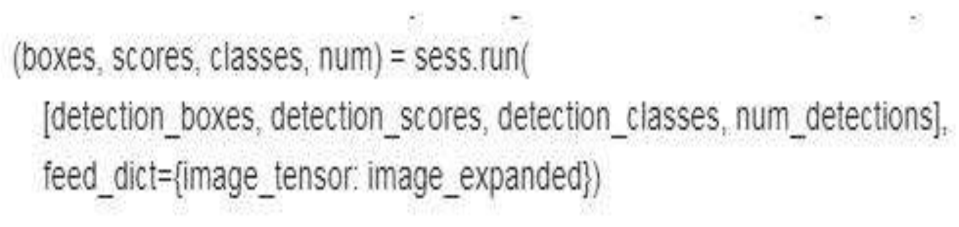
1. Load image using OpenCV.
2. Expand image dimensions to have shape: [1, None, None, 3] i.e. a single-column array, where each item in the column has the pixel RGB value using numpy.
3. Get the tensor which stores the values of the detection boxes with the help of detectiongraph function.
4. Run the tensorflow session with graph initialized as detectiongraph command.

4. Run the tensorflow session with graph initialized as detectiongraph command.

5. Get the values of Ymin, Ymax, Xmin, Xmax by manipulating the detection box array.
6. Get the values of width and height of the input image.
7. Multiply the minimum and maximum values of X and Y dimensions to the width and height of the image and store them in variables xminn, xmaxx, yminn, ymaxx (these values define the bounding box).
8. Extract the bounding box by using the function tf.image.crop to bounding box.
9. Run the tensorflow Session() command on the extracted image.
10. Display the extracted image.

The detection graph stores all the units of computation and units of data that flow during the operation.

The tensorflow session command encapsulates the environment which contains the operation objects which are contained in the Graph() function and evaluates the ten-sor objects.



```
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})
```

Figure 6.9

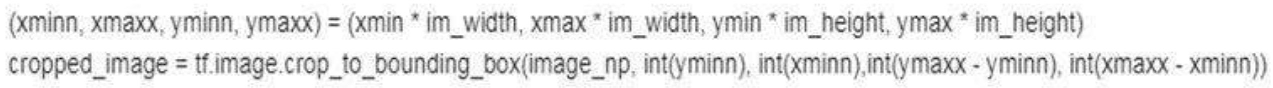
The tf.image.crop to bounding box function crops an image to a specified bound-ing box. This operation cuts a rectangular part out of the image. The top left corner of the returned image is at offset height, offset width and its lower-right corner is at offset height + target height, offset width + target width.

1.offset height: Vertical coordinate of the top-left corner of the result in the input.

2.offset width: Horizontal coordinate of the top-left corner of the result in the input.

3.target height: Height of the result.

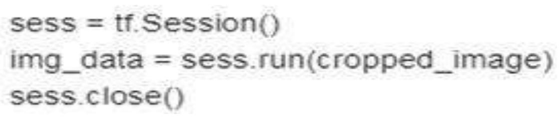
4.target width: Width of the result.



```
(xminn, xmaxx, yminn, ymaxx) = (xmin * im_width, xmax * im_width, ymin * im_height, ymax * im_height)
cropped_image = tf.image.crop_to_bounding_box(image_np, int(yminn), int(xminn),int(ymaxx - yminn), int(xmaxx - xminn))
```

Figure 6.10

We run the session command on the cropped image to evaluate its tensor and save the output as imgdata.



```
sess = tf.Session()
img_data = sess.run(cropped_image)
sess.close()
```

Figure 6.11

Hence we get the following result.

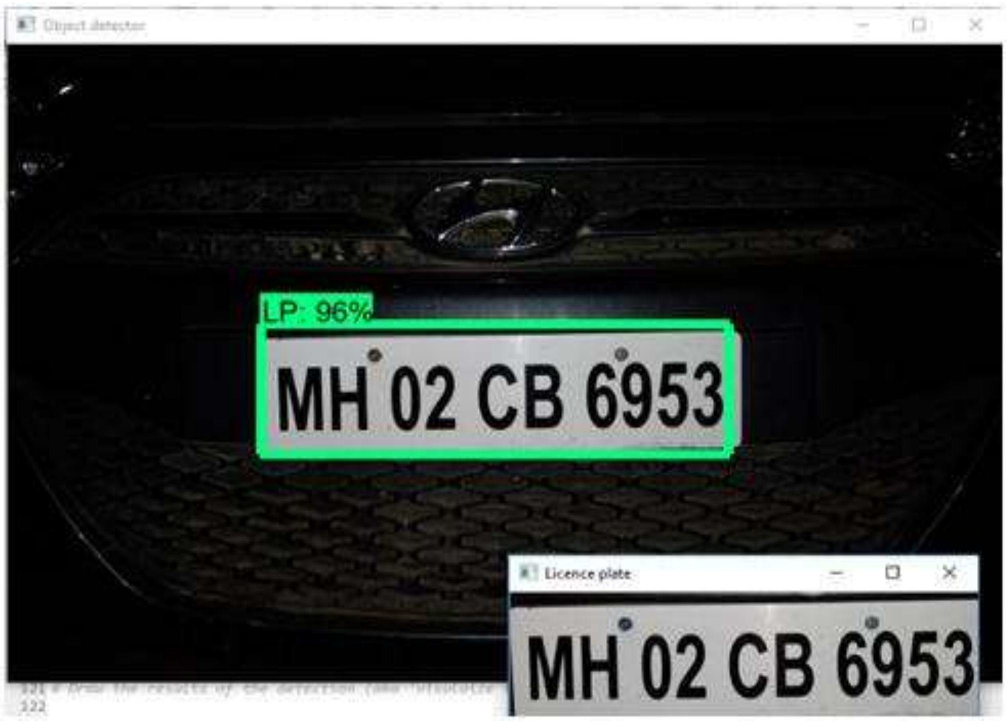


Figure 6.12: LP Extraction

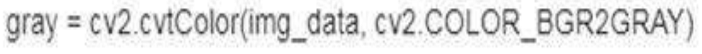
6.5 OPTICAL CHARACTER RECOGNITION

OCR is the final step of the project. We perform some pre-processing steps to the extracted image and then use the Tesseract library to perform OCR.

Algorithm for OCR:
1. Take the extracted image as input.
2. Convert the RGB image to grayscale.
3. Convert the grayscale image to binary by using the Threshold function with an extra flag which uses the Otsu binarization.
4. Use the Median Blur filter to eliminate the noise in the binary image.
5. Convert the image pixel values to string value using the tesseract library.
6. Store the string value in text format.
7. Display the stored value on the console.

6.5.1 Convert RGB to grayscale

A grayscale image is one in which the value of each pixel is a single sample repre-senting only an amount of light, that is, it carries only intensity information. Images of this sort, also known as black-and-white or gray monochrome, are composed ex-clusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest. This pixel depth allows 256 different intensities (i.e., shades of gray) to be recorded, and also simplifies computation as each pixel sample can be accessed individually as one full byte. Since the LP has already been detected and extracted color will now be considered as noise.



```
gray = cv2.cvtColor(img_data, cv2.COLOR_BGR2GRAY)
```

Figure 6.13: RGB to grayscale conversion

The conversion is performed by importing the .cvtColor function from the OpenCV library.

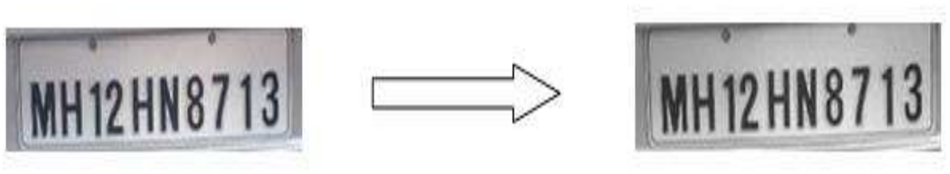


Figure 6.14: RGB to grayscale conversion

6.5.2 OTSU Binarization

A binary image is a digital image that has only two possible values for each pixel. In gray scale image the intensity values of the pixels vary from 0 to 255. But in Binary image the pixel value is either 0 or 1. Binarization makes it easier for the model to distinguish between the characters and the background.
In Otsu’s method we exhaustively search for the threshold that minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Figure 6.15

Weights (omega 0 and 1) are the probabilities of the two classes separated by a threshold t ,and sigma(0 and 1) are variances of these two classes.

```
gray = cv2.threshold(gray, 0, 255,cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
```

Figure 6.16

This operation is also performed by importing the threshold function from the OpenCV library.

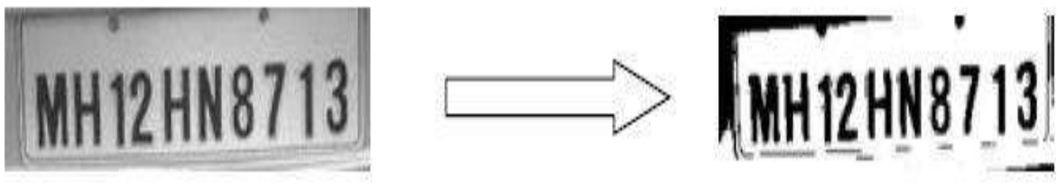
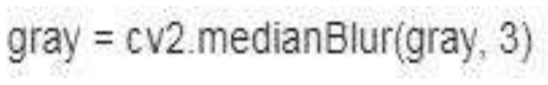


Figure 6.17: Binarization using the Otsu method

6.5.3 Median Blur Filter

The Median Filter is a non-linear digital filtering technique used to remove noise from the image. It preserves the edges of the image while removing noise. The main idea of the median filter is to run through the image entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the ”window”, which slides, entry by entry, over the entire image.

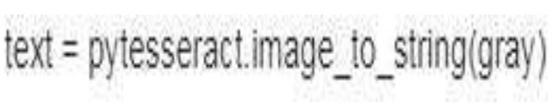


```
gray = cv2.medianBlur(gray, 3)
```

Figure 6.18

6.5.4 Tesseract Library

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and read the text embedded in images. Python-tesseract is a wrap-per for Google’s Tesseract-OCR Engine.



```
text = pytesseract.image_to_string(gray)
```

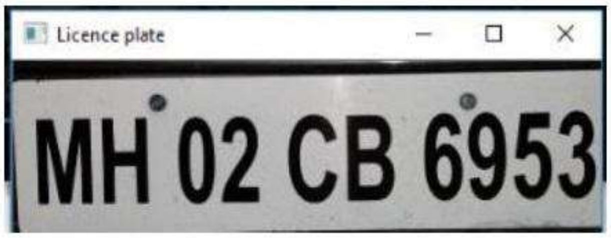
Figure 6.19

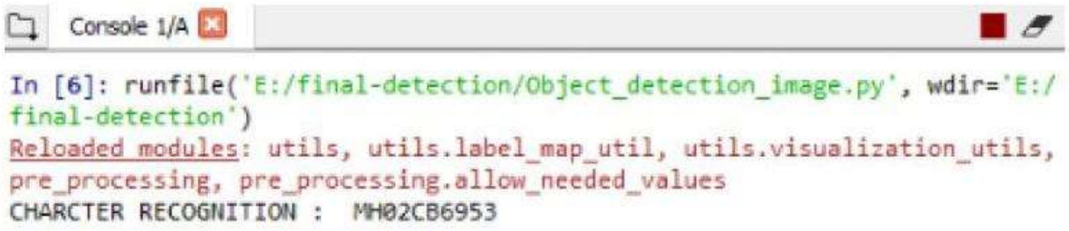
It can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp (We have used jpeg and png formats for images).
The imagetostring function returns the result of a Tesseract OCR run on the image and converts it to string and prints the recognized text.

How Tesseract Works:
1. The first step is a connected component analysis in which outlines of the compo-nents are stored.
2. Outlines are gathered together, purely by nesting, into Blobs.
3. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text.
4. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells.
5. Proportional text is broken into words using definite spaces and fuzzy spaces.
6. Recognition then proceeds as a two-pass process.

In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the image. Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the image, a second pass is run over the page.
In second pass, words that were not recognized well enough are recognized again. A

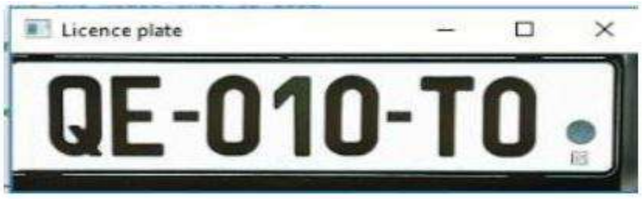
final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small-cap text.

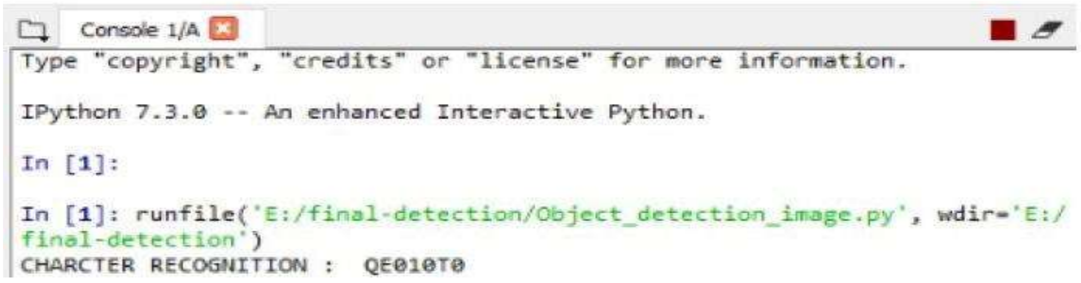




```
In [6]: runfile('E:/final-detection/Object_detection_image.py', wdir='E:/
final-detection')
Reloaded modules: utils, utils.label_map_util, utils.visualization_utils,
pre_processing, pre_processing.allow_needed_values
CHARCTER RECOGNITION :  MH02CB6953
```

Figure 6.20: OCR on extracted LP





```
Type "copyright", "credits" or "license" for more information.

IPython 7.3.0 -- An enhanced Interactive Python.

In [1]:

In [1]: runfile('E:/final-detection/Object_detection_image.py', wdir='E:/
final-detection')
CHARCTER RECOGNITION :  QE010T0
```

Figure 6.21: OCR on extracted LP

Chapter 7

RESULTS

The trained model we obtained after training using 180 images is tested with 39 images, as the entire dataset of 216 is divided into 80 train images and approx. 20 test image,is bound to give an accuracy based on loss as described in the graph in fig below.

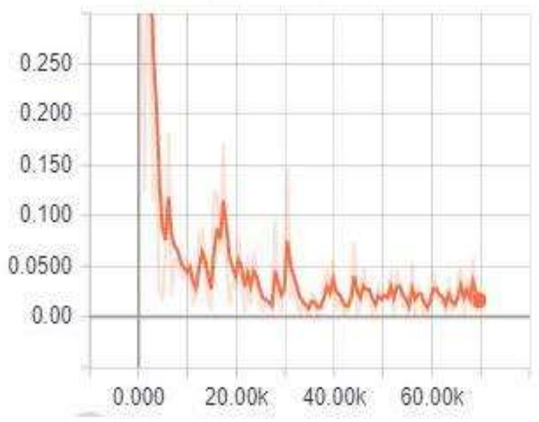
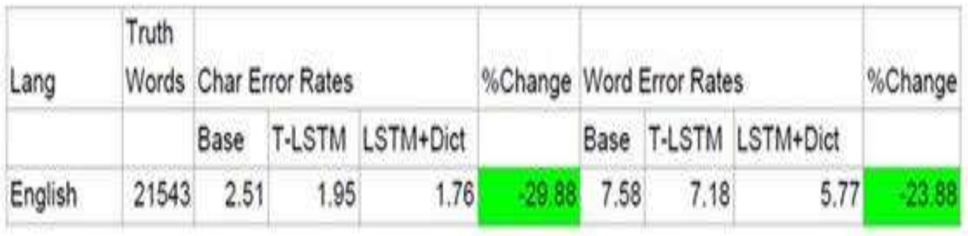


Figure 7.1: Total loss vs Epochs

The calculated training and validation accuracy are:
Train Accuracy:: 0.9764634601043997
Test Accuracy :: 0.7933284397683713
Tesseracts (used for optical character recognition) performance accuracy is described in the table below.



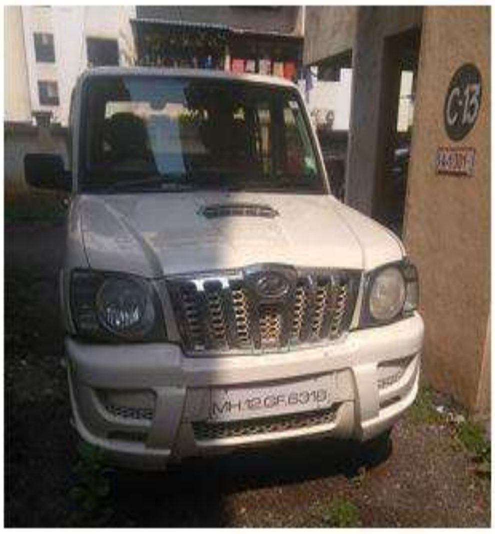
Lang	Truth Words	Char Error Rates			%Change	Word Error Rates			%Change
		Base	T-LSTM	LSTM+Dict		Base	T-LSTM	LSTM+Dict	
English	21543	2.51	1.95	1.76	-29.88	7.58	7.18	5.77	-23.88

Figure 7.2: Tesseract Performance Accuracy

Chapter 8

LIMITATIONS

To achieve‘ such high processing speeds, a relatively modern GPU is required. This is not problematic for real-world applications with ALPR systems running on desk-top computers or other large stationary devices, but real-world applications with ALPR systems running on mobile devices would struggle to achieve the processing speeds claimed.

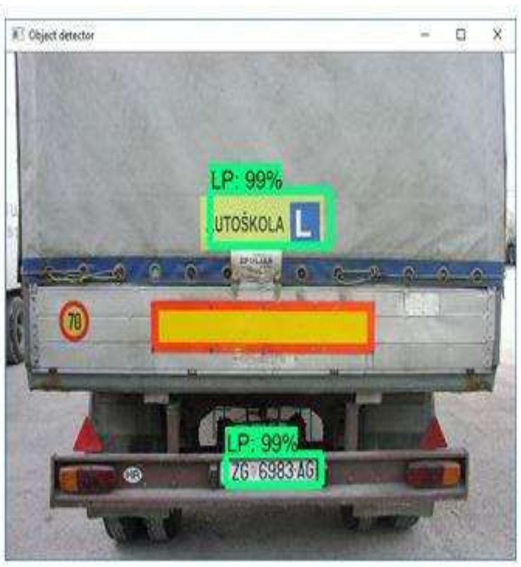


System unable to detect LP due to excess of sunlight incident on the LP.

Figure 8.1: Examples of Faulty Detection

However, the newest mobile devices do have integrated GPUs which may be powerful enough to run the proposed method within the speed requirements for real-time video streams.

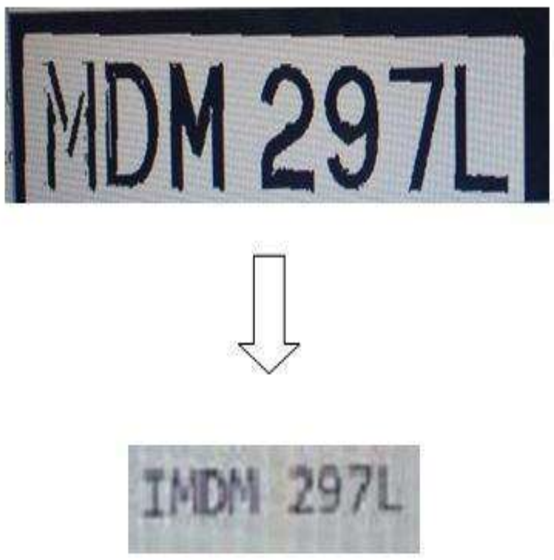
Even though the system shows high accuracy for most of the images fed to the system there are some cases where the system is unable to detect the LP or shows faulty results.



System detects an extra I due to darkened border of the LP. (Bad Picture Quality)

Figure 8.2: Examples of Faulty Detection

Also during OCR the system may mistake a symbol for an alphabet or wrongly interpret an alphabet or digit if it is written in an unusual (not standard) manner. Though the number plates have to follow a standard font and it is highly discour-aged to change the font of text on an LP however there are a number of vehicles having stylish patterns and symbols on their LP.



System detects two LPs in this image. It mistakes a rectangular sign for an LP.

Figure 8.3: Example of Faulty OCR

Chapter 9

APPLICATIONS

1.Parking Lot Management Systems: This system can be installed at the entrance or exit of parking lots to record all the cars entering the parking lot. This data can be used in malls, offices and other institutions to keep track of all vehicles entering its premises.

2.Toll Booth Management Systems: This sytem does not require any human in-tervention and hence can easily record the license numbers for large number of ve-hicles passing through the toll plaza without any errors. This will make it easy for law enforcers to track any vehicle travelling on the highway.

3.Traffic Control: This system can be installed at signals and accident prone ar-eas. It can keep track of vehicles that break the law or cause accidents. It can also keep track of all the vehicles which frequently pass by and hence update traffic con-ditions.

Chapter 10

CONCLUSION

Using digital image processing there are multiple methods to detect an object in an image. In this project we have used a deep learning approach to detect an LP and further obtain its characters in a text format from an image. The Convolutional Neural Network is a deep learning algorithm which can take in an input image, as-sign importance(weights/bias) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required is much lower as compared to other classification algorithms. Also it automatically learns its filter-s/characteristics.

We have looked into multiple R-CNN based approaches for detection. The faster R-CNN is faster than its predecessors and provides high accuracy as compared to other classification algorithms. Hence we have chosen the faster R-CNN Inception Version 2 model to implement the project. Not only does the model provide very high accuracy for a small size of training data but it also has good generalization capability.

The OCR part works very well with Pytesseract. It works really well for recognizing black text on a white background which is an ideal case for LPs. We have used the GPU version of Tensorflow to provide a proper framework and implement object detection algorithms. The experimental results demonstrate that the model has good recognition result.

Chapter 11

FUTURE WORK

Possible directions for further experimentation and development are given in the list below.

1. Train and test the proposed method on other datasets. The proposed method was only tested on the test set of parked cars. Even though there are no universal datasets for license plates, the proposed method may be tested on semi-universal datasets such as the AOLP dataset to further compare its performance to existing methods.

2. Complete and further develop the video functionality. It is possible to use the proposed method to process video streams, but only functionality for the license plate detection was implemented due to time constraints. To make this functionality more useful, character detection should be performed to give a final prediction.

3. Investigate possibility for a mobile system. Although the proposed method is computationally complex, we believe it would be possible to port it to modern mobile devices with sufficient processing speed. Most modern mobile devices have integrated GPUs available and by downsizing the system finding a trade-off between prediction accuracy and processing speed, we believe an ALPR system using the proposed method would work well on mobile devices.

References

[1] Krishna Dharavath, , G. Amarnath, , Fazal A. Talukdar, and Rabul H. Laskar, Impact of Image Preprocessing on Face Recognition, 2014 International Con-ference on Communication and Signal Processing. ;

[2] P. Jonathon Phillips and R. Michael McCabe , Rama Chellapa, BIOMETRIC IMAGE PROCESSING AND RECOGNITION on 9th European Signal Process-ing Conference,1998. ;

[3] Hoo-Chang Shin, , Holger R. Roth, Mingchen Gao, Le Lu, , Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, Ronald M. Summers on Deep Convo-lutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning published by IEEE Transaction on Medical imaging, May 2016. ;

[4] Mrs.Saniya Ansari, Dr.Udaysingh Sutar Optimized and Efficient Feature Ex-traction Method for Devanagari Handwritten Character Recognition , 2015 IEEE Conference on Information Processing . ;

[5] Suresh Dara and Priyanka Tumma ,Feature Extraction By Using Deep Learn-ing, 2018, IEEE Conference on Electronics, Communication and Aerospace Technology. ;

[6] Fatma Shaheen, Brijesh Verma and Md Asafuddoula Impact of Automatic Fea-ture Extraction in Deep Learning Architecture, 2016 IEEE Conference on Dig-ital Image Computing: Techniques and Applications. ;

[7] Shih-Chung Hsu, Chung-Lin Huang, Cheng-Hung Chuang,Tai-Chung, Vehicle Detection using Simplified Fast R-CNN , 2018 IEEE Conference on Advanced Image Technology. ;

[8] Lei Quan, Dong Pei, Binbin Wang, Wenbin Ruan , Research on Human Target Recognition Algorithm of Home Service Robot Based on Fast-RCNN , 2017, 10th IEEE Conference on Intelligent Computation Technology and Automation.
;

[9] Mduduzi Manana, Chunling Tu and Pius Adewale Owolawi Preprocessed Faster RCNN for Vehicle Detection , 2018 IEEE Conference on Intelligent and Innovative Computing Applications . ;

[10] Myung-Cheol Roh, Ju-young Lee ,Refining Faster-RCNN for Accurate Object Detection, 15th IAPR International Conference on Machine Vision Applica-tions,May2017. ;

[11] Shan Du, , Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy, Auto-matic License Plate Recognition (ALPR): A State of the Art Review, volume 4,2012. ;

[12] Wang Naiguo, Zhu Xiangwei, Zhang Jian , License Plate Segmentation and Recognition of Chinese Vehicle Based on BPNN , 2016 12th IEEE Conference on Computational Intelligence and Security. ;

[13] Nikhil Yadav , Utkarsh Binay, Comparative Study of Object Detection Algo-rithms, International Research Journal of Engineering and Technology (IR-JET),Nov 2017. ;