UTTARA UNIVERSITY

Image Segmentation using Convolutional Neural Network (CNN) for Automated Vehicle Navigation.

BY

Sanjida Sharmin	**2183081017**	**46 (B DAY)**
Md. Tanzil	**2183081041**	**46 (B DAY)**
Md. Jamiul Islam	**2183081016**	**46 (A DAY)**
Babul Ahamed Sujan	**2183081055**	**46 (B DAY)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF SCIENCE AND ENGINEERING
UTTARA UNIVERSITY

SUMMER 2022

UTTARA UNIVERSITY

Image Segmentation using Convolutional Neural Network (CNN) for Automated Vehicle Navigation.

BY

Sanjida Sharmin	2183081017	46 (B DAY)
Md. Tanzil	2183081041	46 (B DAY)
Md. Jamiul Islam	2183081016	46 (A DAY)
Babul Ahamed Sujan	2183081055	46 (B DAY)

A thesis submitted in partial fulfilment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
School of Science and Engineering
Uttara University, Uttara, Dhaka, Bangladesh

Summer 2022

ABSTRACT

In the current era of technology, new discoveries are coming in front of people. A lot of work is being done on how the earth is suitable for us to live. Examples of which are computers, electricity, space research, etc. In continuation of this, there is a huge demand for autonomous cars at present. If without the help of humans, a car can run on the road in a safe way and people can reach its destination, then it is called a self-driving car. The self-driving car receives images from the road as input while moving on the road. Then the analysis is done. After the analysis, the car decides which way to run, how fast it will run and where to stop. At present, various countries are running experimental activities of self-driving cars. Research is underway on how to make self-driving cars more advanced and safer. In the previous works, many researchers used CNN, R-CNN, faster R-CNN frameworks for developing automated vehicle navigation. However, there are still weaknesses in some places. For example, self-driving cars cannot run properly in bad weather, speed limitations, can't walk properly on more crooked roads etc. In our proposed system we have tried to reduce these weaknesses and gain more accuracy. Now we present a model, where we basically used CNN neural network's extension framework Mask R-CNN. We have specified a road environment and will train the car using Mask R-CNN by segmenting road features with some real-world obstacles. In our model, we will train our data by conv2d layer and max pooling layer, with 3*3 matrixes randomly taken valued kernel. After that, we use a mask R-CNN that can return both the bounding box and a mask for each detected object in an image. After training again and again, our system will recognize the features. So whenever we will give a new input, our system can recognize it by segmenting its features. By this segmenting an autonomous car will make decisions of how to navigate and run safety.

Keywords: *CNN, Neural Network, Computer Vision, Autonomous Cars, Image Segmentations, Bounding box, R-CNN, Mask R-CNN, Conv2D, Kernel*

APPROVAL

I certify that I have supervised this study and read this manuscript. In my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of BSc. in Computer Science and Engineering.

Md. Mijanur Rahman
Supervisor

Uttam Kumar Dey
Co-Supervisor

I certify that I have read this study. In my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of BSc. in Computer Science and Engineering.

Md. Torikur Rahman
Coordinator/Internal Examiner

This thesis was submitted to the Department of Computer Science and Engineering and is accepted as a fulfilment of the requirement for the degree of BSc. in Computer Science and Engineering.

Dr. A. H. M. Saifullah Sadi
Chairman, Dept. of CSE

DECLARATION

We hereby declare that this dissertation is the result of our own investigations, except where otherwise stated. We also declare that it has not been previously or concurrently submitted as a whole for any other degrees at Uttara University or any other institutions. We also declare that the formatting of the manuscript is same as the provided template. We also do not have any objections for the further use of the manuscript as Uttara University has all the rights to update, publish, or conduct further research of the submitted work.

Student Names	Student IDs	Signature	Date
Sanjida Sharmin	2183081017		
Md. Tanzil	2183081041		
Md. Jamiul Islam	2183081016		
Babul Ahamed Sujan	2183081055		

UTTARA UNIVERSITY

DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED RESEARCH

Image Segmentation using Convolutional Neural Network (CNN) for Automated Vehicle Navigation.

We declare that the copyright holders of this dissertation are jointly owned by the students and Uttara University (UU).

1. Any material contained in or derived from this unpublished research may be used by others in their writing with due acknowledgement.

2. UU or its library will have the right to make and transmit copies (print or electronic) for institutional and academic purposes.

3. The UU library will have the right to make, store in a retrieved system and supply copies of this unpublished research if requested by other universities and research libraries.

By signing this form, We acknowledged that we have read and understand the UU Intellectual Property Right and Commercialization policy.

Affirmed by Sanjida Sharmin.

…… ……..

Signature (on behalf of the team) Date

DEDICATION

We dedicate this thesis/report to
our honorable parents, and our younger brother(s)
for their meticulous support, continuous inspiration, and unconditional love
till the very end of this journey.

ACKNOWLEDGEMENTS

Firstly, it is our utmost pleasure to dedicate this work to my dear parents and my family, who granted me the gift of their unwavering belief in our ability to accomplish this goal: thank you for your support and patience.

I wish to express my appreciation and thanks to those who provided their time, effort and support for this project. To the members of my dissertation committee, thank you for sticking with me.

Finally, a special thank to ***Assist. Prof. Md. Torikur Rahman*** for his continuous support, encouragement, and leadership, and for that, I will be forever grateful.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

1.1 OVERVIEW

Autonomous cars have become an interesting discussion in recent years (**Alex, ARSO 2011**). Several major automotive industries such as Tesla, GM and Nissan are trying to become pioneers in autonomous car technology. Several technological approaches are carried out in implementing autonomous cars for recognizing surrounding situations, such as radar, LIDAR, sonar, GPS, and odometer. An automatic control system is used to control navigation based on the data obtained from these sensors. In autonomous driving systems, the task of detecting the vehicle itself is one of the most important prerequisites to autonomous navigation. The advancement of Computer Vision these days has grown up beyond imagination. One of the computer vision tasks performs object detection very effectively compared to earlier methods and this project is to segment the objects like vehicles, persons, road lanes. Instance segmentation an extension of object detection is used to identify the objects in the image by assigning different labels for each instance of the objects belonging to the same class (**E. Shelhamer, 2015**). In the existing systems, CNN is used for segmenting objects for autonomous vehicle. However, there are scopes for further improvement in the accuracy of performance of this system. To get better solution from this problem, this project will highlight the use of CNN deep learning algorithm, for segmenting the surrounding environment in creating the automatic navigation required by autonomous cars more effectively.

1.2 PROBLEM STATEMENTS AND ITS SIGNIFICANCE

With the advent of modern computing and sensing devices major advancement has been done in the field of automatic vehicle navigation. Some of the candidate systems have also been proposed with partial success. In spite of many achievements, some limitations of the systems are also there. Such as:

- Number of crashes of automated cars has been reported. That means auto navigation system is not sophisticated enough to navigate safely in real world.
- Vision algorithm need improvements to cope up with rough weather conditions during heavy fog and snowfall.
- Driverless cars struggle going over bridges or flyovers
- It may not function properly in extremely heavy traffic condition
- It is not suitable for express way due to its speed limitation.

In order to remove the above-mentioned limitations and propose effective and reliable vehicle navigation system, much more efforts are required. In particular, more improvements are required for image analysis techniques used by the navigation system. Image segmentation is crucial step in these image analysis techniques. Now-a-days deep neural network has been seen to demonstrate much success in image classification, as well as segmentation. Therefore, there is a strong motivation to investigate into effectiveness of deep neural network to find improved vision techniques for usage in automatic vehicle navigation.

1.3 RESEARCH OBJECTIVES

The aim of this project is to develop a deep learning system to segment and level a given image into different regions. The proposal involves the following objectives:

To design a convolution deep neural network with appropriate filter and maxpooling layer

- To implement the CNN with appropriate imaging and computing resources.
- To train the implemented CNN system effectively.
- To evaluate the accuracy of the system.

1.4 RESEARCH METHODOLOGY

In this section, overall methodology of the project will be discussed. First, we will highlight the techniques used in the project, followed by construction of neural networks. Finally, our overall approach is presented.

1.4.1 MODEL'S BASIC STRUCTURE

In the same way that a child learns to recognize by looking at the various objects around him, our algorithm has to show millions of pictures, so that our system has a 'generalized' idea of what might happen. After that whenever we give an image, our system can tell what the picture is. We will train our system with some input images through CNN model. In this case during training, we will take input image with labeled region. Then it will go through the convolutional layers and pooling layers and as output initially we will get some error. Our proposed system will calculate the

error by comparing the predicted value with the actual value and then we will adjust our filter values for next step to bring the CNN output closer to the desired output.

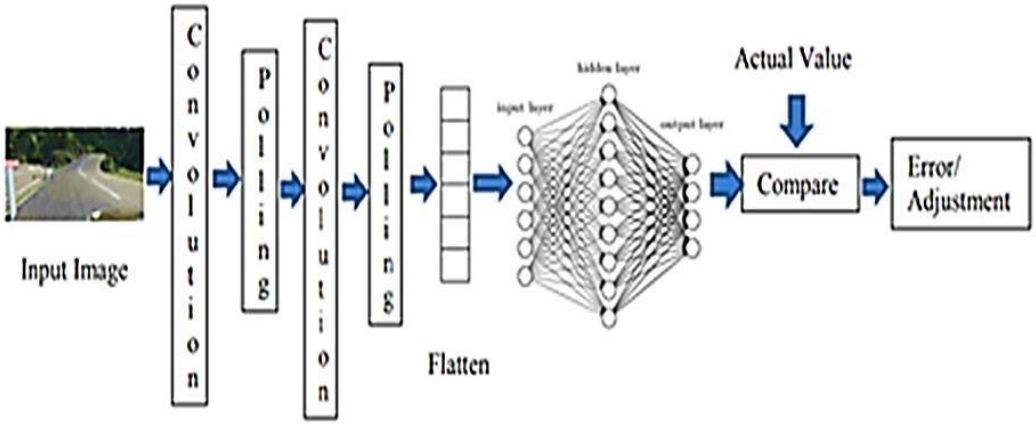

Figure 1.1: CNN Technique for Automated Vehicle Navigation

For this model we have to give an image as input and this input will be sent to convolutional layer and a filter or kernel is used to run over the image. Thus, we get a feature map of image. Then the maxpooling layer operates a small kernel on the image at a fixed stride. It is used to pick the pixel with the highest intensity and discard other pixels. The resultant matrix will be a reduced dimensional matrix of feature image **(Nicolas, 2017).** This helps in reducing the unnecessary sparse cells of image also reduces the training time. After performing the convolutional layer and maxpooling layer several times, we flatten our pooled feature map into a column. And it is connected to the fully-connected layer which gives our segmented images **(Ravi Kaushik, ISSUE 11, NOVEMBER 2019).**

In our proposed model we will use Mask R-CNN, which is an extension of Faster R-CNN. Generally Faster R-CNN creates a bounding box with class label for the objects in an image. But Mask R-CNN additionally generates a binary mask for each object present in the input image. Mask RCNN is compatible with all methods developed for

instance segmentation and object detection and allows people to experiment quickly. It takes image's feature map extract by using CNN model as an input, this part known as backbone model. Then Region Proposal Network (RPN) will run over the feature map in order to get the Regions of Interest (ROI) **(G. Zhu, 2016.).** In that case we often get the bounding boxes, which is misaligned. So RoI Align Network used to wrap the boxes into a fixed dimension. Warped features are then fed into fully connected layers to make classification and boundary box prediction.

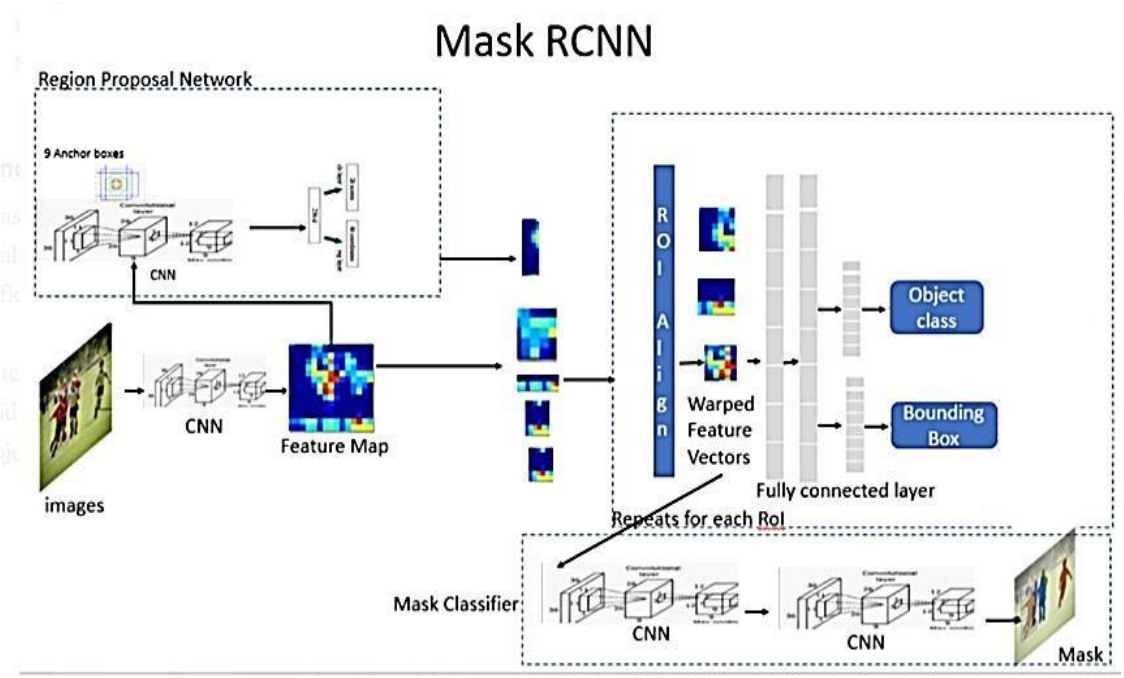

Figure 1.2: Mask R-CNN architecture

Warped features are also fed into Mask classifier, which consists of two CNN's to output a binary mask for each RoI. Mask Classifier allows the network to generate masks for every class without competition among classes. It returns a mask of size 28 X 28 for each region which is then scaled up for inference. The RoI's also gives the boundary box and the objects class name as an output.

1.4.2 FLOW DIAGRAM OF THE ACTIVITIES

Figure 1.1.2 presents the flow diagram of the methodology of the proposed research.

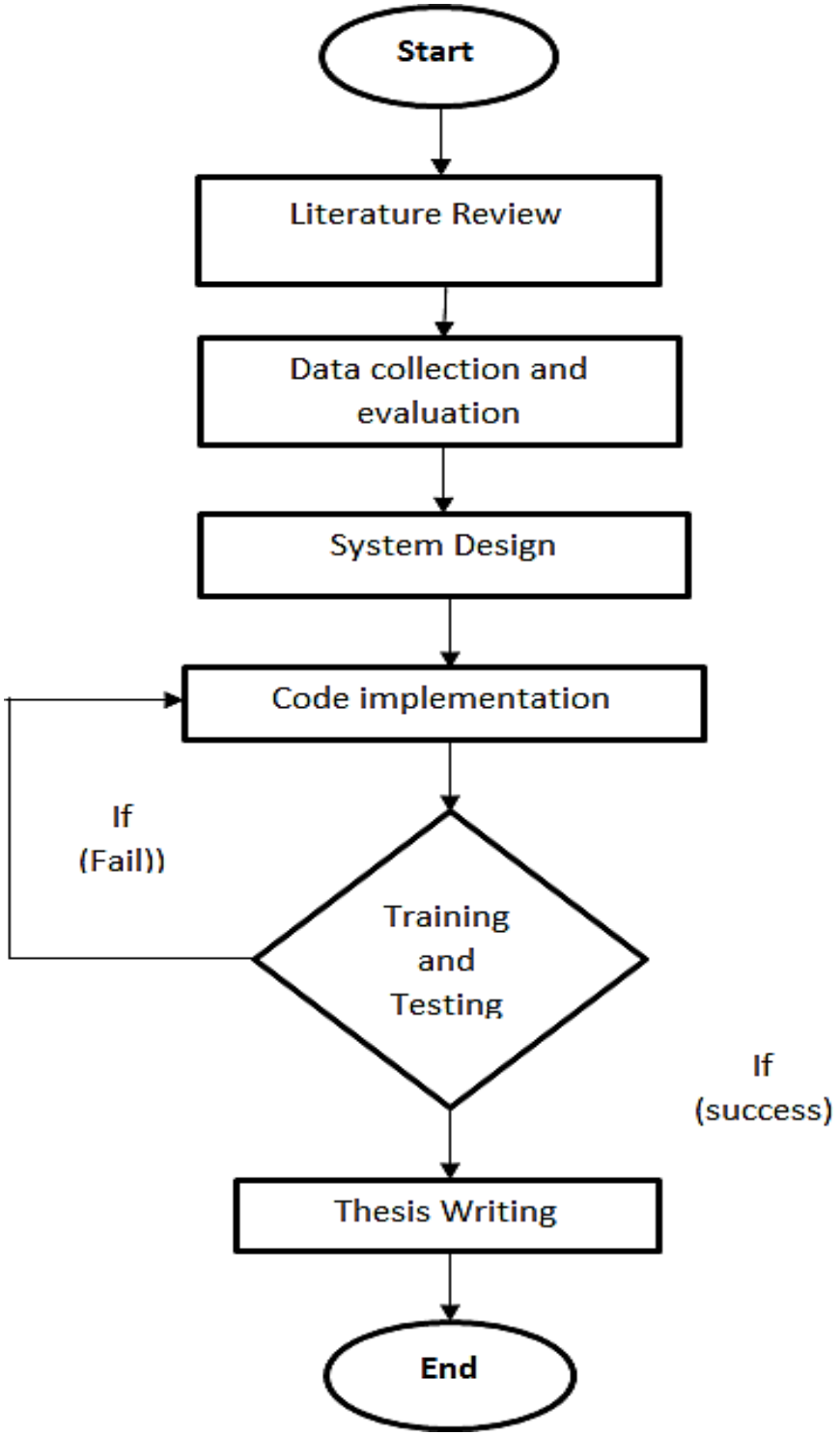

Figure 1.3: Flow diagram of the methodology of the proposed research

1.5 GANTT CHART

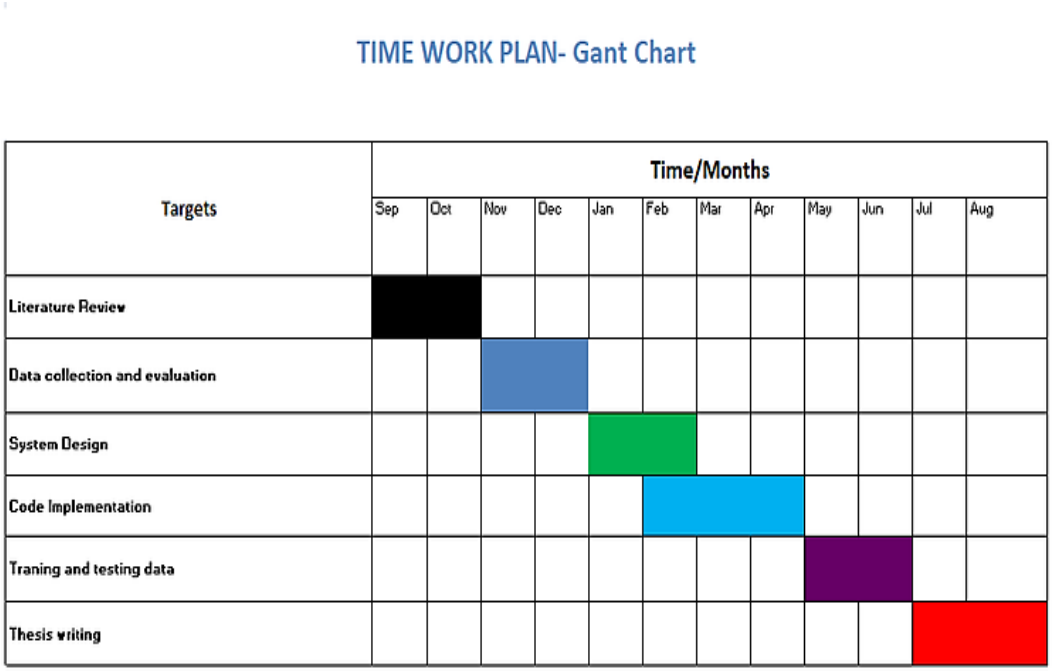

Figure 1.4 Gantt chart of our proposed research activities

1.6 SUMMARY

The overall approach of our project is discussed here. This paper proposes an end-to-end method for training convolutional neural networks for autonomous navigation of a self-driving car. Self-driving car will capture the images and the images will work as an input for the CNN architecture. After performing these layers again and again, finally we will get our segmented image. So, an automated vehicle can detect the object for its navigation purpose.

CHAPTER TWO
LITERATURE REVIEW

2.1 OVERVIEW

Self-driving cars have created a stir in today's world. It can undoubtedly be said that the future technology will revolve around self-driving cars. The new wave of technology is self-driving cars. Research in autonomous navigation was done from as early as the 1900s with the first concept of the automated vehicle revealed by General Motors in 1939 **(Nicolas, 2017).** Although, the accuracy was very low in early research. Besides, it was expensive. Many companies also stopped researching autonomous vehicles. They had different limitations. But nowadays technology has advanced a lot. Autonomous cars have achieved success due to advanced technology. At present, self-driving cars have been launched in various countries. In that context, we tried to improve their models and algorithms. Currently, self-driving car technology is working very successfully as a result of improvements in artificial intelligence, sensor technologies, and cognitive science. Researchers have gotten a step closer to realizing a practical implementation of a self-driving car.

2.2 BACKGROUND STUDY

There are many kinds of CNN algorithms that is used for segmenting image.

2.2.1 CNN (CONVOLUTIONAL NEURAL NETWORK)

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. Convolutional Neural

Networks are the ultimate and basic building blocks for the computer technology task of image segmentation.

CNNs efforts better with data, which has a spatial relationship. The CNN input is usually two-dimensional, a field or matrix **(MA 10609, USA**). But we can also be changed to be one-dimensional, allowing it to progress an internal representation of a one-dimensional order.

- **Convolutional layer:** Convolutional layer helps to abstract the input image as a feature map.
- **Pooling layer:** This layer helps to down sample feature maps by summarizing the presence of features in patches of the feature map.
- **Fully connected layer**: Fully connected layers connect every neuron in one layer to every neuron in another layer.

2.2.2 R-CNN (REGION-BASED CONVOLUTIONAL NEURAL NETWORK)

R-CNN stands for Region-based Convolutional Neural Network. The key concept behind the R-CNN series is region proposals. R-CNNs (Region-based Convolutional Neural Networks) are a family of machine learning models used in computer vision and image processing. Specially designed for object detection, the original goal of any R-CNN is to detect objects in any input image defining boundaries around them.

An input image given to the R-CNN model goes through a mechanism called selective search to extract information about the region of interest. Region of interest can be represented by the rectangle boundaries. Depending on the scenario there can be over 2000 regions of interest. This region of interest goes through CNN to produce

output features. These output features then go through the SVM (support vector machine) classifier to classify the objects presented under a region of interest.

2.2.3 FAST R-CNN

Fast R-CNN is an improved version of R-CNN architectures with two stages:

1. Region Proposal Network (RPN) is simply a Neural Network that proposes multiple objects that are available within a particular image.
2. Fast R-CNN. This extracts features using RoIPool (Region of Interest Pooling) from each candidate box and performs classification and bounding-box regression. RoIPool is an operation for extracting a small feature map from each RoI in detection.

2.2.4 FASTER R-CNN

Faster R-CNN uses a region proposal method to create the sets of regions. Faster R-CNN possesses an extra CNN for gaining the regional proposal, which we call the regional proposal network **(Jannik, 2013).** In the training region, the proposal network takes the feature map as input and outputs region proposals. And these proposals go to the ROI pooling layer for further procedure.

2.2.5 MASK R-CNN

Mask R-CNN is an object detection model based on deep convolutional neural networks (CNN). Mask R-CNN developed by a group of Facebook AI researchers in 2017. Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the

addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object **(Acosta, L., 2016).** Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region ofInterest) in parallel with the existing branch for bounding box recognition.

2.3 RELATED WORKS

One of the earliest reported uses of a neural network for autonomous navigation comes from the research conducted by Pomerleau in 1989 who built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. Training on real road images is logistically difficult, because in order to develop a general representation, the network must be presented with a large number of training exemplaIS depicting roads. Besides, it is difficult to collect data sets in this case. Changes in parameters such as camera orientation would require collecting an entirely new set of road images. To avoid these problem Pomerleau have developed a simulated road generator which produces road images to be used as training exemplars for the network.

Training involves first creating a set of 1200 road snapshots. The first test practices novel artificial road images. After 40 epochs of training on the 1200 simulated road snapshots, the network correctly dictates a turn curving within two units of the correct answer approximately 90% of the novel artificial road snapshots.

In the early 1990s, Carnegie Mellon researcher Dean Pomerleau wrote a Ph.D. thesis describing how neural networks could allow a self-driving vehicle to take in raw images from the road and output steering controls in real time. Pomerleau wasn’t

the only researcher working on self-driving cars, but his use of neural nets proved way more efficient than alternative attempts to manually divide images into “road” and “non-road” categories.

In June 1993, Professor Han Min-Hong in South Korea worked on a self-driving car. He used a Asia Motors to test his car by having it drive around Seoul, accumulating a total of 17 kilometres travelled. Two years later, in 1995, a different car was tested by driving from Seoul to Busan via the Gyeongbu Highway in 1995. As his work was ahead of its time in South Korea, the government focused on heavy industry like steel and shipbuilding at the time

By the early 2000s, there was a huge demand for autonomous vehicles. There are various researches. As a result, autonomous vehicles have been greatly improved. The U.S. Department of Defence’s research arm, DARPA, sponsored a series of challenges to accelerate autonomous cars. In 2004, they held a competition to challenge vehicles to self-navigate 150 miles of desert roadway. But No cars completed the route. In 2007, the challenge simulated a 60-mile long urban environment. And this time four cars completed the route.

By the mid-2010s, several companies (like Ford, Mercedes-Benz, and BMW, as well as rideshare programs like Uber) focused on autonomous vehicles. They want to improve autonomous cars in different ways. It can be said that a competition is created. But the reality was a little harder. That is, autonomous cars were quite difficult to develop and implement in practice. That's why many companies reduce their efforts. In 2020 Uber declared that they were backing down from their efforts at self-driving cars as a result of safety, lawsuits, and a loss of money.

In 2016, NVIDIA released a paper which is a similar idea that benefited from ALVINN. In the paper, they used CNN architecture to extract features from the driving frames. The network was trained using augmented data, which was found to expand the model's performance. Shifted and rotated images were generated from the training set. This method was found to work well in simple real-world scenarios, such as highway lane-following and driving in flat. Several research efforts have been undertaken to build more multipart perception-action models to tackle the numerous of environments, and unpredictable situations usually encountered in urban environments.

Another approach of interest is deep reinforcement learning (RL). RL has been in the limelight only recently regarding autonomous navigation with its success being confirmed by learning of games like Atari and Go by Google DeepMind.

Walpola Perera and Matthias Heinz's research was carried out by driving on highways to find the most accurate Region-based convolutional neural networks (R-CNNs) for objects detection while driving in a vehicle. But they got medium accuracy in this field.

In the 2020s, the first regulations related to automated features appear:

Regulation (EU) 2019/2144 is defined in 2019 and put on from 2022 in the European Union for automated vehicles and for fully automated vehicles.

In June 2020, UNECE WP.29 GRVA established regulation on SAE Level 3.

In October 2020 Tesla released a "beta" form of its "Full Self-Driving" software to a minor group of testers in the United States.

In the 2020s numerous electric, autonomous buses open for public transport are being launched around the world.

2.4 SUMMARY

At the end of this chapter, we can say that a lot of work has been done on self-driving car so far. But our proposed system is to make this system more effective. In earlier systems Object detection for Autonomous Vehicle using YOLOV3 Algorithm was done detect the Object with Bounding box. DNN was used in the existing system for object detection. But it has some disadvantage. Such as less accuracy, time complexity, cannot detect moving vehicle correctly etc. So our proposed system is image segmentation of automated vehicle navigation system using CNN algorithm Mask R-CNN.

CHAPTER THREE
MODELING AND DESIGN

3.1 OVERVIEW

Self-driving car is a complicated system. It is one of the biggest and interesting topics these days, the industry will continue to need outstanding engineers to produce innovative solutions for driving. In this chapter, our objective is to talk about the modeling and designing of our proposed system. Road vehicle detection is an essential part of many intelligent applications, such as speed and number of vehicles and driving assistance systems, automated smart parking systems, automated guidance vehicles and traffic statistics. Automatic vehicle assistance systems aimed at alert vehicle detection, and difficult monitoring of other vehicles about the roadside environment. Our proposed approach is image segmentation using Mask RCNN algorithm, which uses the state of art deep-learning combined with data to scan and detect, segment and classify the objects and to estimate the position of the objects Present around the vehicle and make the decision about how to navigate.

3.2 SYSTEM AT A GLANCE

In our system, at first we collect our data. After that we need to preprocess our data. Because at first convolutional neural network required that all images are the same sized arrays. Also image preprocessing can reduce the run time and could helpful to give better accuracy. Then the images will feed into our model and our model will perform his segmentation and classification operation. So we will get the image's

object as an output and by analyzing these outputs an autonomous vehicle can take decision about his navigation

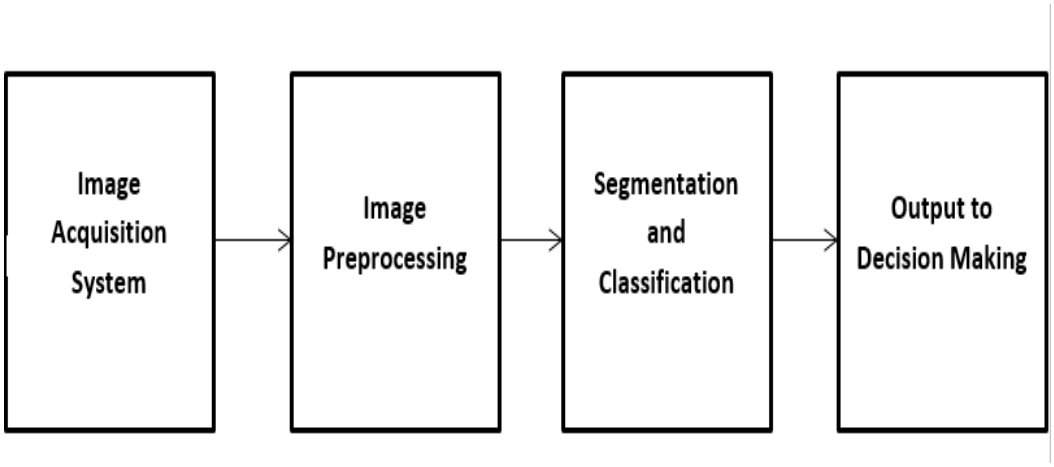

Figure 3.1: Workflow of our system

3.3 SYSTEM DESIGN

In this section design of the proposed system is presented. The components of the proposed system as shown in figure 3.1 are described in details in the following sub sections.

3.3.1 IMAGE ACQUSIATION SYSTEM

We work on different dataset. In our system we are detecting 39 objects, which could be in a road scenario. By doing this, an autonomous vehicle can detect the things around him and could be able to take decision about how to navigate. For now, we are using datasets from online. But when we will test it in real time, the vehicle could take the image around him using a camera.

3.3.2 IMAGE SEGMENTATION

One of the most important operations in Computer Vision is Segmentation. Suppose, you are crossing a road and you see various objects around like vehicles, traffic lights, footpath, zebra-crossing and pedestrians. Now, while crossing the road your eyes instantly analyze each object and process their locations to take decision of whether to cross the road at that moment or not. Now with the help of image segmentation and object identification techniques it is possible for the computers to see the real-world objects and based on their positions they can now take necessary actions **(Garcia-Garcia A, 2018).**

3.3.3 FEATURE EXTRACTION

First of all, our input image will feed into a convolutional neural network. In CNN architecture the Conv2D and the pooling layer will perform again and again. As a result, we will get a feature map of our input image. The entire convolution feature map will pass into the Region Proposal Network (RPN), which is a 3*3 convolution layer **(Ren S, 2015)**. While scanning feature map, we use anchor boxes to bind features to its raw image location. In this step, we get those regions or feature maps which the model predicts contain some object. For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. But these outputs will be in different shapes. So the output from first convolutional layer and the RoI's will pass into the RoI Align who will wrap the features in a fixed dimension. Hence, we get a feature map for each region.

3.3.4 CLASSIFICATION

After feature collection our feature map will pass through two fully connected layers. Each FCN contains 1x1 convolutions that perform the task of fully connected layers (Dense layers). At the end we have use softmax classifier- who will tell us which object is in the image, regression- who will create the bounding box and the mask classifier – who will create the mask pixel wise of the object.

3.4 SUMMARY

Nowadays, technology of self-driving cars become important as it will make the transportation more brilliant and safer. In earlier systems Object detection for Autonomous Vehicle was done detect the Object with Bounding box using DNN. But our proposed approach is combined with data to scan and detect, segment and classify the objects and to estimate the position of the objects that present around the vehicle.

CHAPTER FOUR
SYSTEM SETUP, IMPLEMENTATION AND TESTING

4.1 OVERVIEW

This chapter shows that how we setup and the implementation of self-driving car navigation system using Mask RCNN Algorithm. Mask RCNN allows us to form segments on the pixel level of each object and also separate each object from its background.

4.2 SYSTEM SETUP

4.2.1 GOOGLE COLAB

Google Colab is particularly well suited to machine learning, data analysis and education. It enables anyone to create and execute arbitrary python code through the browser. We can use Google Colab like Jupyter notebooks. They are really convenient because Google Colab hosts them, so we don't use any of our own computer resources to run the notebook. We can also share these notebooks so other people can easily run our code, all with a standard environment since it's not dependent on our own local machines. However, we might need to install some libraries in our environment during initialization.

4.2.2 DATASET

We need a dataset that contains the class, which are presented in a road scenario. So at first we collect some data and labeled them using LabelImg. After labeling the

annotates files are saved in xml format. But for better result we need a big dataset. So we use “Traffic-Sign-Localization-Detection-SSD-Annotated” dataset from the website “Kaggle”.

4.2.3 IMAGE PREPROCESSING

In our proposed algorithm, at first we have use the small dataset that created by us. We collect the images and resize them in the same size. After that we create their XML files using LabelImg. After that we used “Traffic-Sign-Localization-Detection-SSD-Annotated” dataset which is a pre-trained image dataset. The dataset contains images which are already in a same size. And the image and xml file of respected image was in the same folder.

4.3 IMPLEMENTATION

4.3.1 PYTHON LIBRARIES

We used following libraries in our code.

4.3.1.1 SCIKIT-IMAGE

scikit-image is an image processing library that implements algorithms and utilities for use in research, education and industry applications. scikit-image’s simple, well-documented application programming interface (API) makes it ideal for educational use. It provides easy access to a powerful array of image processing functionality. In our model we have used scikit-image==0.16.2.

4.3.1.2 TENSORFLOW

TensorFlow is an end-to-end open source framework for machine learning, deep learning and other statistical and predictive analytics workloads. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. In our model we have used tensorflow version 2 (tensorflow-gpu==2.0.0).

4.3.1.3 KERAS

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. It contains a wealth of developer instructions and documentation. It allows you to define and train neural network models in just a few lines of code. In our model we have used keras==2.3.1

4.3.1.4 OPENEV-PYTHON

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It is just a wrapper around the original C/C++ code. It is normally used for combining best features of both the languages, Performance of C/C++ &

Simplicity of Python. So when you call a function in OpenCV from Python, what actually run is underlying C/C++ source.

4.3.1.5 H5PY

The h5py package is a Pythonic interface to the HDF5 binary data format. HDF5 lets you store huge amounts of numerical data, and easily manipulate that data from NumPy. Thousands of datasets can be stored in a single file, categorized and tagged however you want. In our code we have used h5py==2.10.0.

4.3.1.6 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It contains the necessary tools to manipulate and perform operations on these arrays

4.3.1.7 XML.ETREE

XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree. The xml.etree.ElementTree module implements a simple and efficient API for parsing and creating XML data.

4.3.2 CODE CONTRIBUTION

4.3.2.1 Move XML file into other directory

```
import os

import shutil

sourcepath=r'C:\Users\mdtan\Desktop\Trai
ndatasetforSSD'

sourcefiles = os.listdir(sourcepath)

destinationpath
r'C:\Users\mdtan\Desktop\Final
Dataset\annots'

for file in sourcefiles:

    if file.endswith('.xml'):


shutil.move(os.path.join(sourcepath,file),
os.path.join(destinationpath,file))
```

4.3.2.2 File Rename

```
import os

print(os.getcwd())

for count, f in enumerate(os.listdir()):

    f_name, f_ext = os.path.splitext(f)

    f_name = str(count)



    new_name = f'{f_name}{f_ext}'

    os.rename(f, new_name)
```

4.3.2.1 Read each XML file and collect its object name

```
import xml.etree
import xml.etree.ElementTree as ET
import os
def extract_boxes(filename):
    tree = ET.parse(filename)
    root = tree.getroot()
    name = root.find('.//object/name').text
    return name
namelist = []
for count, f in enumerate(os.listdir()):
    if f == 'z.py':
        break
    name = extract_boxes(f)
    if name not in namelist:
        namelist.append(name)
```

4.4 CLONING MASK R-CNN TF2

At this step we have clone the Mask R-CNN tf2 github repository using "%cd Mask-RCNN-TF2/" command, so that we will be able to use the features of Mask R-CNN in our proposed code. In Mask R-CNN tf2 there is a very important directory named "mrcnn", who holds the source code for the project. It has the following Python files:

4.4.1 __init__.py

Marks the mrcnn folder as a Python library.

4.4.2 model.py

It has the functions and classes for building the layers and the model.

4.4.3 config.py

Holds a class named Config that holds some configuration parameters about the model.

4.4.4 utils.py

Includes some helper functions and classes.

4.4.5 visualize.py

Visualizes the results of the model.

4.4.6 parallel_model.py

Supporting multiple GPUs.

4.5 PREPARE THE DATASET

First we create an instance of the mrcnn.utils.Dataset. Then we override the instance by calling load_dataset() method to load our dataset. It calls the add_class() method to create a class named “dataset”, with our dataset and their ID respectively. We call the add_image() method to add an image to the dataset. The image ID, path, and the path of the annotation file are passed to this method. We call the load_mask() method, who loads the masks for each objects in an image.

4.6 SPLITTING THE DATASET

The load_dataset splits our dataset into training and validation. We keep 11500 data for training and rest of the data for validation. Then in load_dataset() method we pass the dataset_dir parameter, who holds the path of the dataset images and its annotations. We also pass the is_train flag. If this fag is True, then the data is regarded as training data. Otherwise, the data is used for validation or testing. Then we call the prepare() method to prepare the dataset for use. It just creates more attributes about the data like the number of classes, number of images, and more.

4.7 PREPARE THE MODEL CONFIGURATION

Next, we will create our model and load the pre trained weights which we downloaded earlier. A subclass of the mrcnn.config.Config class must be created to hold the model configuration parameters. The number of classes (NUM_CLASSES) is set to 1+39, because the dataset has 39 objects to detect and 1 for the background. The STEPS_PER_EPOCH is 360, which means 360 numbers of steps taken from the generator as soon as one epoch is finished and next epoch has started.

4.8 TRAINING

The next code creates an instance of the mrcnn.model.MaskRCNN class, which builds the architecture of the Mask R-CNN model. The mode parameter is set to 'training' to indicate that the model will be trained. Once the model architecture is created, the weights are loaded using the load_weights() method. After that we train the model using train() method. In the train() method we set the “layers” argument is to “heads”, which means that only the layers at the head of the architecture are trained

and epochs= 8 , which means there will be total 8 passes through the training dataset. Once the model is trained, the trained weights can be saved using the Keras save_weights() method.

4.9 PREDICTION

At first we need to build the Mask R-CNN model architecture. To build the Mask R-CNN model architecture, the mrcnn.model script has a class named MaskRCNN. We create an instance of the mrcnn.model.MaskRCNN class that saved in the model variable. Then we load the weights in the created model using the load_weights() method and pass the path of our dataset's weight "Roadsign_mask_rcnn_trained.h5" as a parameter. Once the model is created and its weights are loaded, next we need to read an image using OpenCV to read an image and reorder its color channels to be RGB, rather than BGR and feed it to the model. Then we call the detect() method, who detect the objects in the image. Also for each input image, the detect() method returns a dictionary that holds information about the detected objects.

4.10 SUMMARY

In this chapter, we have implemented our designed system. It is very important for an automated vehicle to predict the objects around him correctly. Otherwise the vehicle could take wrong decisions and there would be accident in a result. So we have trained our system with a dataset of 39 classes.

CHAPTER FIVE
RESULT ANALYSIS AND BENCHMARKING

5.1 OVERVIEW

There has been a huge interest in self-driving cars lately, which is understandable, given the improvements it is predicted to bring in terms of safety and comfort in transportation. This chapter shows that how our output looks like and benchmarking of our existing model. Also we will analyze the result of our model, so that we could determine the achievements and overcomes of our proposed system.

5.2 RESULT ANALYSIS

A Mask R-CNN model was successfully trained to segmenting the objects. The following figures show us the output of our proposed system. Figure 5.1 shows that our code generate a mask and detect the traffic sign and classify it on “crosswalk” class.

Figure 5.1: Output 1

Figure 5.2: Output 2

Again in figure 5.2, we can see another output. We also can say that in output 2 our code generates a mask around the object and create a boundary box around it and classify it as speed limit. Our object of this project was, we will create a system that will segment the objects that is around a vehicle. Now after analyzing our output we can say that our implemented code is successfully trained and able to segment our targeted objects.

```
360/360 [==============================] - 429s 1s/step - loss: 1.5895 - val_loss: 2.5272
Epoch 2/8
360/360 [==============================] - 311s 864ms/step - loss: 0.9245 - val_loss: 0.7902
Epoch 3/8
360/360 [==============================] - 315s 875ms/step - loss: 0.8945 - val_loss: 1.3788
Epoch 4/8
360/360 [==============================] - 318s 883ms/step - loss: 0.8444 - val_loss: 2.0169
Epoch 5/8
360/360 [==============================] - 318s 883ms/step - loss: 0.8110 - val_loss: 2.6156
Epoch 6/8
360/360 [==============================] - 317s 882ms/step - loss: 0.8060 - val_loss: 1.0234
Epoch 7/8
360/360 [==============================] - 322s 895ms/step - loss: 0.8341 - val_loss: 1.9041
Epoch 8/8
360/360 [==============================] - 324s 900ms/step - loss: 0.7459 - val_loss: 1.8063
```

Figure 5.3: Scenario of training model at per epoch

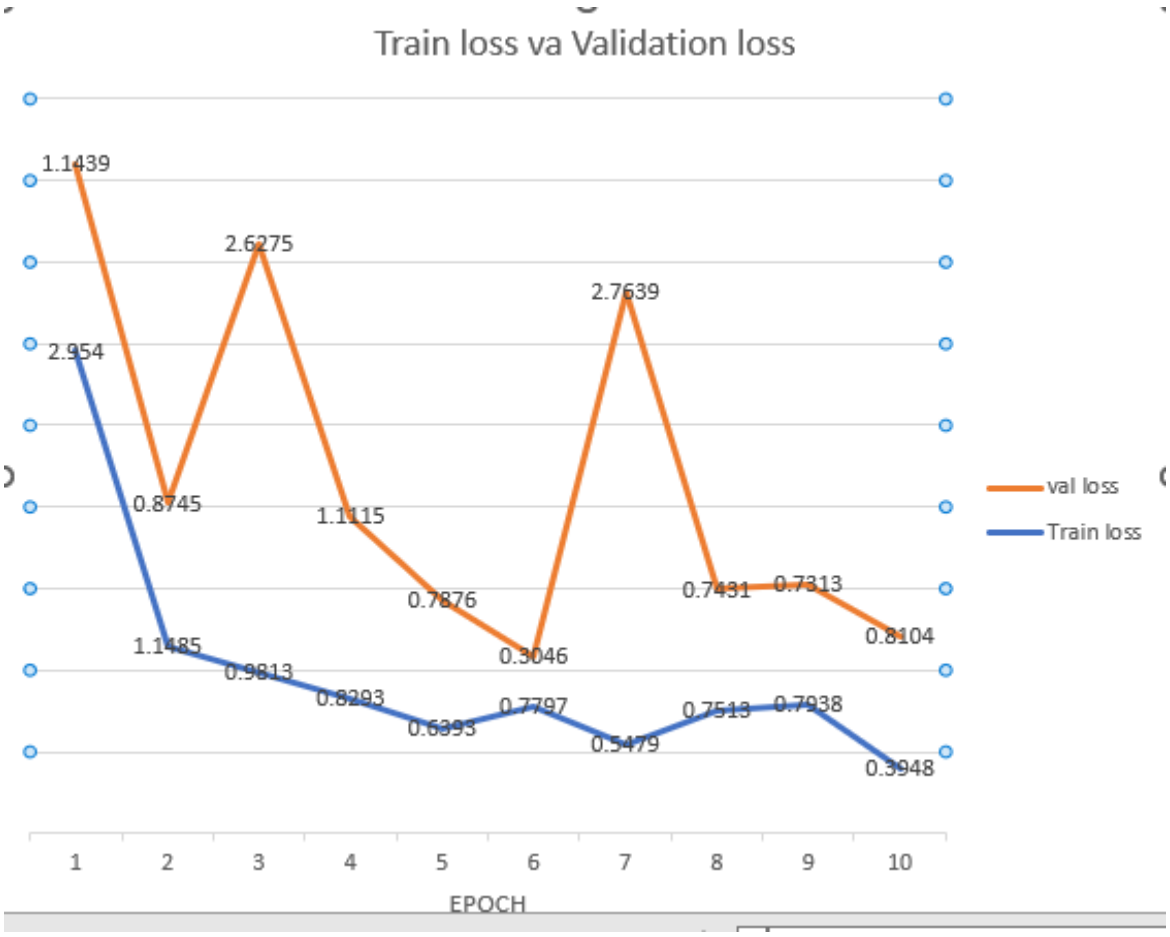

Figure 5.4: Training vs validation loss

Figure 5.3 and 5.4 shows us the model performance. Figure 5.3 shows the training time and loss result at per epoch and 5.4 shows the training loss vs validation loss according to the figure 5.3. From the graph we can see that our validation loss is greater than training loss. In that case we can say that our model is over fitted, which means our model shows better performance during training but during validation its performance is little bit poor. This happens because we use a small dataset for training. Further works will be done to make this model more efficient.

5.3 BENCHMARKING

This chapter presents the integrated output of our designed and implemented system. Initially we want to do segment the objects and also want that after analyzing the output an automated vehicle can immediately take the steering decision. We have researched a lot and at first we have started to work with a dataset that created by our own. As that was a small dataset our system misclassifies the objects. So we need a large set of data. So we searched on google and work with a dataset that has 16000 image data. We train our model using the Mask R-CNN algorithm which is compatible to detect the object more efficiently.

5.4 SUMMARY

A new Mask R-CNN based image segmentation for automated vehicle navigation system is proposed in this paper. The importance of recognizing drivable areas is arguably paramount in the process of developing an autonomous driving solution. As humans, we can clearly process and visualize the road directly in front of us with our eyes and brain. Therefore, if vehicles are to supersede a human's ability to drive,

they need to be able to solve this basic problem. Nevertheless, the problem remains a challenge to researchers. As an area of intense research, there does exist some literature on detecting drivable regions. However, many researchers have utilized high tech equipment like 3D Laser Imaging, Detection and Ranging (LIDAR) scanners as well as traditional radars to create a more holistic picture of the vehicles surroundings to improve their decision making. But the result was not enough efficient. So in this paper we proposed and implemented a system using Mask R-CNN framework for detecting the objects. Mask R-CNN is an extension of faster R-CNN. It can do both object detection and instance segmentation by creating bounding box around the object and generates a mask for the object.

CHAPTER SIX
CONCLUSION AND RECOMMENDATIONS

Artificial Intelligence has become increasingly important in today's world. We are becoming dependent on automation. As a result of this, our time is being saved, as well as the accuracy of the work is increasing. Self-driving car technology has emerged in that context. If the car can maintain proper and safe driving on the road without human assistance, then it is called self-driving car. Various researchers have been researching the development of self-driving car technology at different times. Their tireless efforts have created a niche in the self-driving car world today. It is gradually improving. In that context, we tried to improve the technology model. We hope that our research will play an important role in the development of autonomous vehicles in the future. Of course we have some limitations here. Again there is some merit. But we should know why we took up this project.

The amount of road accidents that occur due to violation of traffic rules will be reduced to a great extent. People often break traffic rules in moments of tension or due to negligence. As a result, road accidents occur at different times. Millions of people die in road accidents every year around the world. If self-driving cars can be improved, then it must comply with traffic laws. As a result, the death rate of road accidents will be greatly reduced. Traffic jams will be reduced to a great extent, because the self-driving car will mainly operate by Artificial Intelligence. So in this case, the car will stop at a certain time at each stop and the car will run at a certain time. As a result, people will wait at the stop at the right time. Since time is being

maintained here, the traffic jam will be reduced to a great extent. Using self-driving cars will greatly reduce fuel consumption. In this case about 4 to 10 percent fuel wastage will be reduced. Also, using self-driving cars will result in 500% increase in lane capacity, 40% reduction in travel time. Self-driving cars will obey speed limits. As a result, the car will not run at extra speed. The number of road accidents will reduce a lot. The Eno Center's study found up to 4.2 million accidents would be prohibited. 21,700 lives would be conserved and more than $400 billion in interrelated costs would be eradicated.

So we realized that the need for autonomous vehicles is very high in today's world. So we conducted our research. Now we will know the outputs of our research.

6.1 RESEARCH OUTCOMES

1. We have designed a system.
2. Our proposed system has been implemented.
3. Developed system can easily detect the road sign.
4. All the objectives of the project have been achieved.

6.2 LIMITATIONS OF THE RESEARCH

1. System works well on dataset which involves a small number of classes. However, accuracy of the system deteriorates if the dataset contains big number of classes.
2. If the image quality is poor, detecting road signs may be a problem
3. Further improvements are required to work with complex images.

6.3 RECOMMENDATIONS

1. The vehicle may have trouble driving properly in bad weather conditions. Further work can be done to make self-driving cars useful in bad weather.
2. Further work can be done to integrate the proposed system with real world automated vehicle navigation system.
3. Proposed system can be enhanced and tested in realistic complex road scenarios.
4. Further efforts are required to address the problem of vehicle driving in difficult scenarios like sharp bends, rough terrain, muddy roads etc.

REFERENCES

Mofeed Turky Rashid, Huda Ameer Zaki, and Rana Jassim Mohammed. “Simulation of autonomous navigation mobile robot system”. In: Journal of Engineering and Sustainable Development 18.4 (2014), pp. 25–38.

Nicolas Gallardo, “Autonomous Decision Making for a Driver-less Car”, 2017.

Naveen S Yeshodara, 2Nikhitha Kishore, “Cloud Based Self Driving Cars”, 2014

K. Mori, H. Fukui, T. Murase, T. Hirakawa, T. Yamashita, H. Fujiyoshi, Visual explanation by attention branch network for end-to-end learning-based self-driving, Proc. of IEEE Intelligent Vehicles Symposium (9-12 June 2019) https://doi.org/10.1109/ IVS.2019.8813900.

International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by http://www.ijert.org

Road Segmentation Using CNN with GRU Yecheng Lyu and Xinming Huang Department of Electrical and Computer Engineering Worcester Polytechnic Institute Worcester, MA 01609, USA

. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

Z. Chen and Z. Chen, “Rbnet: A deep neural network for unified road and road boundary detection,” in International Conference on Neural Information Processing. Springer, 2017, pp. 677–687.

Q. Li, L. Chen, M. Li, S.L. Shaw, A. Nüchter, A sensor-fusion drivable-region and lanedetection system for autonomous vehicle navigation in challenging road scenarios, IEEE Transactions on Vehicular Technology 63 (2) (2013) 540–555.

A. B. Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: a survey,” Machine vision and applications, vol. 25, no. 3, pp. 727–745, 2014.

INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 11,NOVEMBER 2019 ISSN 2277-8616667IJSTR©2019www.ijstr.org Image Segmentation Using Convolutional Neural Network Ravi Kaushik, Shailender Kumar

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks

K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

Mask R-CNN With Pyramid Attention Network for Scene Text Detection DOI: 10.1109/WACV.2019.00086 Conference: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, pages 91–99, 2015.

Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In CVPR, pages 4159–4167, 2016.

Mask RCNN with RESNET50 for Dental Filling Detection Article in International Journal of Advanced Computer Science and Applications · January 2021 DOI: 10.14569/IJACSA.2021.0121079

G. Zhu, Z. Piao and S. C. Kim, "Tooth Detection and Segmentation with Mask R-CNN," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020, pp. 070-072, doi: 10.1109/ICAIIC48513.2020.9065216.

Jannik Fritsch, Tobias Kuhnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on, pages 1693–1700. IEEE, 2013.

Arnay, R., Morales, N., Morell, A., Hernandez-Aceituno, J., Perea, D., Toledo, J. T., Hamilton, A., Sanchez-Medina, J. J., & Acosta, L. (2016). Safe and reliable path planning for the autonomous vehicle verdino. IEEE Intelligent Transportation Systems Magazine, 8, 22–32.

Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Martinez-Gonzalez P, Garcia-Rodriguez J: A survey on deep learning techniques for image and video semantic segmentation. Applied Soft Computing 2018, 70:41-65.

Yurtsever E, Lambert J, Carballo A, Takeda K: A survey of autonomous driving: Common practices and emerging technologies. IEEE Access 2020, 8:58443-58469

Ren S, He K, Girshick R, Sun J: Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:150601497 2015.

Alex Teichman and Sebastian Thrun. “Practical object re cognition in autonomous driving and beyond”. In : Advanced Robotics and its Social Impacts (ARSO), 2011 IEEE Workshop on. IEEE. 2011, pp. 35–38

APPENDIX A
EXAMPLE CODES

CODES FOR PYTHON LIBRARIES INSTALL

```
!apt-get update

!pip3 install scikit-image==0.16.2

!pip3 install opencv-python

!pip3 install tensorflow-gpu==2.0.0

!pip3 install keras==2.3.1

!pip install h5py==2.10.0
```

CODES FOR TRAINING AND VALIDATION

```
import  mrcnn

import mrcnn.utils

import mrcnn.config

import mrcnn.model

import urllib.request

import os

import xml.etree

from numpy import zeros, asarray
```

```
class RoadsignDataset(mrcnn.utils.Dataset):

    def load_dataset(self, dataset_dir, is_train=True):

        self.add_class("dataset", 1, "Speed Limit 30")

        self.add_class("dataset", 2, "Speed Limit 50")

        self.add_class("dataset", 3, "Turn Left")

        self.add_class("dataset", 4, "No Stopping")

        self.add_class("dataset", 5, "Turn Right")

        self.add_class("dataset", 6, "No Waiting")

        self.add_class("dataset", 7, "Go Right or Straight")

        self.add_class("dataset", 8, "Give Way")

        self.add_class("dataset", 9, "Stop")

         self.add_class("dataset", 10, "GreenSignal")

        self.add_class("dataset", 11, "human")

        self.add_class("dataset", 12, "Road Work")

        self.add_class("dataset", 13, "Danger Ahead")

        self.add_class("dataset", 14, "RedSignal")

        self.add_class("dataset", 15, "No Entry")
```

```
self.add_class("dataset", 16, "Speed Limit 80")

self.add_class("dataset", 17, "Traffic Signals Ahead")

self.add_class("dataset", 18, "No Over Taking Trucks")

self.add_class("dataset", 19, "No Over Taking")

self.add_class("dataset", 20, "Slippery Road")

self.add_class("dataset", 21, "Go Straight")

self.add_class("dataset", 22, "Pedestrian")

self.add_class("dataset", 23, "YellowSignal")

self.add_class("dataset", 24, "Cycle Zone")

self.add_class("dataset", 25, "Deer Zone")

self.add_class("dataset", 26, "Speed Limit 70")

self.add_class("dataset", 27, "Speed Limit 60")

self.add_class("dataset", 28, "Speed Limit 120")

self.add_class("dataset", 29, "Speed Limit 100")

self.add_class("dataset", 30, "Huddle Road")

self.add_class("dataset", 31, "Right Curve Ahead")

self.add_class("dataset", 32, "Snow Warning Sign")
```

```
self.add_class("dataset", 33, "Truck Sign")

self.add_class("dataset", 34, "Left Sharp Curve")

self.add_class("dataset", 35, "Right Sharp Curve")

self.add_class("dataset", 36, "End of Right Road (Go straight)")

self.add_class("dataset", 37, "Go Left or Straight")

self.add_class("dataset", 38, "Left Curve Ahead")

self.add_class("dataset", 39, "Car")

 images_dir = dataset_dir + '/images/'

annotations_dir = dataset_dir + '/annots/'



for filename in os.listdir(images_dir):

   image_id = filename[:-4]

   if is_train and int(image_id) >= 11500:

      continue

   if not is_train and int(image_id) < 11500:

      continue

   img_path = images_dir + filename
```

```
        ann_path = annotations_dir + image_id + '.xml'

        self.add_image('dataset', image_id=image_id, path=img_path,
annotation=ann_path,class_ids=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39])



    def load_mask(self, image_id):
        info = self.image_info[image_id]
        path = info['annotation']
        boxes, w, h, name = self.extract_boxes(path)
        masks = zeros([h, w, len(boxes)], dtype='uint8')
        class_ids = list()
        for i in range(len(boxes)):
            box = boxes[i]
            row_s, row_e = box[1], box[3]
            col_s, col_e = box[0], box[2]
            if (name == 'Speed Limit 30'):
                masks[row_s:row_e, col_s:col_e, i] = 1
                class_ids.append(self.class_names.index('Speed Limit 30'))
```

```
elif(name == 'Speed Limit 50'):
    masks[row_s:row_e, col_s:col_e, i] = 2
    class_ids.append(self.class_names.index('Speed Limit 50'))
elif(name == 'Turn Left'):
    masks[row_s:row_e, col_s:col_e, i] = 3
    class_ids.append(self.class_names.index('Turn Left'))
elif(name == 'No Stopping'):
    masks[row_s:row_e, col_s:col_e, i] = 4
    class_ids.append(self.class_names.index('No Stopping'))
elif(name == 'Turn Right'):
    masks[row_s:row_e, col_s:col_e, i] = 5
    class_ids.append(self.class_names.index('Turn Right'))
elif(name == 'No Waiting'):
    masks[row_s:row_e, col_s:col_e, i] = 6
    class_ids.append(self.class_names.index('No Waiting'))
elif(name == 'Go Right or Straight'):
    masks[row_s:row_e, col_s:col_e, i] = 7
```

```
        class_ids.append(self.class_names.index('Go Right or Straight'))
    elif(name == 'Give Way'):
        masks[row_s:row_e, col_s:col_e, i] = 8
        class_ids.append(self.class_names.index('Give Way'))
    elif(name == 'Stop'):
        masks[row_s:row_e, col_s:col_e, i] = 9
        class_ids.append(self.class_names.index('Stop'))
    elif(name == 'GreenSignal'):
        masks[row_s:row_e, col_s:col_e, i] = 10
        class_ids.append(self.class_names.index('GreenSignal'))
    elif(name == 'human'):
        masks[row_s:row_e, col_s:col_e, i] = 11
        class_ids.append(self.class_names.index('human'))
    elif(name == 'Road Work'):
          masks[row_s:row_e, col_s:col_e, i] = 12
        class_ids.append(self.class_names.index('Road Work'))
    elif(name == 'Danger Ahead'):
```

```
            masks[row_s:row_e, col_s:col_e, i] = 13
            class_ids.append(self.class_names.index('Danger Ahead'))
        elif(name == 'RedSignal'):
            masks[row_s:row_e, col_s:col_e, i] = 14
            class_ids.append(self.class_names.index('RedSignal'))
        elif(name == 'No Entry'):
            masks[row_s:row_e, col_s:col_e, i] = 15
            class_ids.append(self.class_names.index('No Entry'))
        elif(name == 'Speed Limit 80'):
            masks[row_s:row_e, col_s:col_e, i] = 16
            class_ids.append(self.class_names.index('Speed Limit 80'))
        elif(name == 'Traffic Signals Ahead'):
            masks[row_s:row_e, col_s:col_e, i] = 17
            class_ids.append(self.class_names.index('Traffic Signals Ahead'))
        elif(name == 'No Over Taking Trucks'):
            masks[row_s:row_e, col_s:col_e, i] = 18
            class_ids.append(self.class_names.index('No Over Taking Trucks'))
```

```
elif(name == 'No Over Taking'):
    masks[row_s:row_e, col_s:col_e, i] = 19
    class_ids.append(self.class_names.index('No Over Taking'))
elif(name == 'Slippery Road'):
    masks[row_s:row_e, col_s:col_e, i] = 20
    class_ids.append(self.class_names.index('Slippery Road'))
elif(name == 'Go Straight'):
    masks[row_s:row_e, col_s:col_e, i] = 21
    class_ids.append(self.class_names.index('Go Straight'))
elif(name == 'Pedestrian'):
    masks[row_s:row_e, col_s:col_e, i] = 22
    class_ids.append(self.class_names.index('Pedestrian'))
elif(name == 'YellowSignal'):
    masks[row_s:row_e, col_s:col_e, i] = 23
    class_ids.append(self.class_names.index('YellowSignal'))
elif(name == 'Cycle Zone'):
    masks[row_s:row_e, col_s:col_e, i] = 24
```

```
            class_ids.append(self.class_names.index('Cycle Zone'))
        elif(name == 'Deer Zone'):
            masks[row_s:row_e, col_s:col_e, i] = 25
            class_ids.append(self.class_names.index('Deer Zone'))
        elif(name == 'Speed Limit 70'):
            masks[row_s:row_e, col_s:col_e, i] = 26
            class_ids.append(self.class_names.index('Speed Limit 70'))
        elif(name == 'Speed Limit 60'):
            masks[row_s:row_e, col_s:col_e, i] = 27
            class_ids.append(self.class_names.index('Speed Limit 60'))
        elif(name == 'Speed Limit 120'):
            masks[row_s:row_e, col_s:col_e, i] = 28
            class_ids.append(self.class_names.index('Speed Limit 120'))
        elif(name == 'Speed Limit 100'):
            masks[row_s:row_e, col_s:col_e, i] = 29
            class_ids.append(self.class_names.index('Speed Limit 100'))
        elif(name == 'Huddle Road'):
```

```
            masks[row_s:row_e, col_s:col_e, i] = 30
            class_ids.append(self.class_names.index('Huddle Road'))
        elif(name == 'Right Curve Ahead'):
             masks[row_s:row_e, col_s:col_e, i] = 31
            class_ids.append(self.class_names.index('Right Curve Ahead'))
        elif(name == 'Snow Warning Sign'):
            masks[row_s:row_e, col_s:col_e, i] = 32
            class_ids.append(self.class_names.index('Snow Warning Sign'))
        elif(name == 'Truck Sign'):
            masks[row_s:row_e, col_s:col_e, i] = 33
            class_ids.append(self.class_names.index('Truck Sign'))
        elif(name == 'Left Sharp Curve'):
            masks[row_s:row_e, col_s:col_e, i] = 34
            class_ids.append(self.class_names.index('Left Sharp Curve'))
        elif(name == 'Right Sharp Curve'):
            masks[row_s:row_e, col_s:col_e, i] = 35
            class_ids.append(self.class_names.index('Right Sharp Curve'))
```

```
        elif(name == 'End of Right Road (Go straight)'):
            masks[row_s:row_e, col_s:col_e, i] = 36
                    class_ids.append(self.class_names.index('End of Right Road (Go straight)'))
        elif(name == 'Go Left or Straight'):
            masks[row_s:row_e, col_s:col_e, i] = 37
            class_ids.append(self.class_names.index('Go Left or Straight'))
        elif(name == 'Left Curve Ahead'):
            masks[row_s:row_e, col_s:col_e, i] = 38
            class_ids.append(self.class_names.index('Left Curve Ahead'))
        elif(name == 'Car'):
            masks[row_s:row_e, col_s:col_e, i] = 39
            class_ids.append(self.class_names.index('Car'))
return masks, asarray(class_ids, dtype='int32')

    # A helper method to extract the bounding boxes from the annotation file
    def extract_boxes(self, filename):
```

```
        tree = xml.etree.ElementTree.parse(filename)
        root = tree.getroot()
        boxes = list()
        for box in root.findall('.//bndbox'):
            xmin = int(box.find('xmin').text)
            ymin = int(box.find('ymin').text)
            xmax = int(box.find('xmax').text)
            ymax = int(box.find('ymax').text)
            coors = [xmin, ymin, xmax, ymax]
            boxes.append(coors)
        width = int(root.find('.//size/width').text)
        height = int(root.find('.//size/height').text)
        name = root.find('.//object/name').text
        return boxes, width, height, name
class RoadsignConfig(mrcnn.config.Config):
    NAME = "Roadsign_cfg"
    GPU_COUNT = 1
```

```
    IMAGES_PER_GPU = 1

    NUM_CLASSES = 1 + 39

    LEARNING_RATE = 0.001

    STEPS_PER_EPOCH = 360

Roadsign_config = RoadsignConfig()



# Train

train_dataset = RoadsignDataset()

train_dataset.load_dataset(dataset_dir='/content/drive/MyDrive/Final        Dataset',
is_train=True)

train_dataset.prepare()



# Validation

validation_dataset = RoadsignDataset()

validation_dataset.load_dataset(dataset_dir='/content/drive/MyDrive/Final
Dataset', is_train=False)

validation_dataset.prepare()

model = mrcnn.model.MaskRCNN(mode='training',
```

```
                    model_dir='./log',

                    config=Roadsign_config)

model.keras_model.summary()

urllib.request.urlretrieve("https://github.com/matterport/Mask_RCNN/releases/do
wnload/v1.0/mask_rcnn_coco.h5", "mask_rcnn_coco.h5")

model.load_weights(filepath='mask_rcnn_coco.h5',

            by_name=True,

                    exclude=["mrcnn_class_logits",  "mrcnn_bbox_fc",  "mrcnn_bbox",
"mrcnn_mask"])

print("Weights loaded!")

model.train(train_dataset=train_dataset,

        val_dataset=validation_dataset,

        learning_rate=Roadsign_config.LEARNING_RATE,

        epochs=8,

        layers='heads')

model_path = 'Roadsign_mask_rcnn_trained.h5'

model.keras_model.save_weights(model_path)
```

CODES FOR TRAINING AND VALIDATION

```
import  mrcnn

import mrcnn.config

import mrcnn.model

import mrcnn.visualize

import cv2

import os

CLASS_NAMES = ['BG','Speed Limit 30', 'Speed Limit 50', 'Turn Left', 'No Stopping',

'Turn Right', 'No Waiting', 'Go Right or Straight', 'Give Way', 'Stop', 'GreenSignal',

'human', 'Road Work', 'Danger Ahead', 'RedSignal', 'No Entry', 'Speed Limit 80',

'Traffic Signals Ahead', 'No Over Taking Trucks', 'No Over Taking', 'Slippery Road', 'Go

Straight', 'Pedestrian', 'YellowSignal', 'Cycle Zone', 'Deer Zone', 'Speed Limit 70',

'Speed Limit 60', 'Speed Limit 120', 'Speed Limit 100', 'Huddle Road', 'Right Curve

Ahead', 'Snow Warning Sign',

'Truck Sign', 'Left Sharp Curve', 'Right Sharp Curve', 'End of Right Road (Go straight)',

'Go Left or Straight', 'Left Curve Ahead','Car',]


class SimpleConfig(mrcnn.config.Config):

   NAME = "coco_inference"

   GPU_COUNT = 1

   IMAGES_PER_GPU = 1

   NUM_CLASSES = len(CLASS_NAMES)

model = mrcnn.model.MaskRCNN(mode="inference",

                        config=SimpleConfig(),
```

```
                    model_dir=os.getcwd())
model.load_weights(filepath="Roadsign_mask_rcnn_trained.h5",
                   by_name=True)
image = cv2.imread("/content/drive/MyDrive/Final Dataset/images/12000.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
r = model.detect([image], verbose=0)
r = r[0]
print(r)
mrcnn.visualize.display_instances(image=image,
                                  boxes=r['rois'],
                                  masks=r['masks'],
                                  class_ids=r['class_ids'],
                                  class_names=CLASS_NAMES,
                                  scores=r['scores'])
```

SPINE TEXT

Spine text should contain:

1) BSc/MSc in CSE

2) 10 blank spaces

3) Title of the project/research

4) UU, Semester & year

Formatting of the texts:

1) Font: Calibri (Body)

2) Bold

3) UPPER CASE

4) Font size: 16 points or less to adjust within the spine.

BSc in CSE Image Segmentation using Convolutional Neural Network (CNN) for Automated Vehicle Navigation UU, SUMMER 2022