FINAL PROJECT REPORT

Submitted in complete fulfillment of the requirements of

EEE F266

ON

Implementation of neural network in autonomous vehicles

BY

Aamer Abdul Rahman
2017AAPS0217U
ECE

Under the guidance of Dr. Shazia Hasan

AT

BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

FINAL PROJECT REPORT

Submitted in complete fulfillment of the requirements of

EEE F266

ON

Implementation of neural network in autonomous vehicles

BY

Aamer Abdul Rahman
2017AAPS0217U ECE

Under the guidance of Dr. Shazia Hasan

AT

BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

Duration: Ongoing Date of Start: 27-JAN-2019

Date of Submission: 25-March-2019

Title of the Project: Implementation of neural networks in autonomous vehicles

Student Name: Aamer Abdul Rahman Student ID: 2017AAPS0217U

Discipline of Student: B.E. (Hons.) in Electronics and Communication Engineering

Name of the Faculty: Dr. Shazia Hasan

Key Words: Neural networks, Convolutional nets, Deep learning

Project Areas: Artificial intelligence, Convolutional neural networks, Machine learning, Computer Vision

Abstract: This project explores the workings of neural networks and computer vision and describes how they are being implemented to achieve to achieve autonomous functioning of a vehicle. Different functions such as lane keeping, pedestrian detection and object tracking are performed and simulated by trained deep neural networks and computer vision. The software used for training of neural networks and simulations is MATLAB. A comparative study is done to understand different functional simulation models via the MATLAB deep learning and automated driving toolbox. The actual implementation of the model is proposed with the help of Simulink.

Signature of Faculty Signature of the Student

Date: Date:

ACKNOWLEDGEMENTS

I would like to express my deepest sense of gratitude and thanks, to my Supervisor **Dr. Shazia Hasan**, Assistant Professor of Electrical and Electronics Engineering Department, BITS Pilani, Dubai campus, United Arab Emirates, for her valuable guidance and encouragement during the course of this Project.

I am grateful to our Director, **Prof. Dr. R.N. Saha,** for his motivation, encouragement and support to pursue my dissertation.

Above all, I thank the Lord for giving me the strength to carry out this work to the best of my abilities and my parents for the constant moral support.

Signature of the Student
Date:

CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 Overview

Operating a vehicle has always been a relatively simple task for the experienced driver. Using vision to observe and measure, applying the gas and brake pedals and turning the wheel is all that was needed. However, when it comes to designing a system that can autonomously operate a vehicle, it becomes complicated. The human brain can extrapolate many more considerations when facing a problem such as this and can obtain a lot of data from as little as the three or four variables. Neural networks are utilized to try to mimic brain function and when trained properly, are able to make sense of the large amounts of data collected and deliver the desired output. The objective is to create a system, that makes use of neural networks, computer vision and fuzzy logic, that is capable of autonomously driving the car to the destination tackling the different criteria's such as lane keeping, pedestrian detection and object tracking.

1.2 Autonomous Vehicles

Autonomous vehicles should be able to operate and guide itself without human intervention. Campbell et al. describes modern autonomous vehicles as vehicles that can sense their local environment, classify different kinds of objects that they detect, can interpret sensory information to identify appropriate navigation paths whilst obeying transportation rules. Such vehicles combine date collected from sensors such as LIDAR, camera, radar and sonar and then software is used to make sense of the data received from the environment to operate, navigate, and drive the vehicle. Automated driving systems have been in the works since the 1920's with the world's first radio-controlled car. Since then significant advances have been made wherein vehicles are able to give appropriate responses to real world situations. Many cars sold on the market are already equipped with mechanisms that bring the vehicle to certain levels of autonomy. Vehicles with partially autonomous systems may require the driver to intervene if the system is not able to process certain environments or situations. On the other hand, fully autonomous vehicles may not even require steering wheels.

1.2.1 Levels of Autonomy

Six levels of autonomy have been outlined by The Society for Automotive Engineers to be used by automakers, supplier and policymakers to determine a system's sophistication.

Level 0 – Zero Automation: The driver completely in control of the vehicle. No autonomous driver aids have been implemented.

Level 1 – Assistance Systems: The driver is still mostly in charge of the vehicle. However, the driver may call upon technology like adaptive cruise control for support. This system determines the distance from the car in front using lasers or radars and adjusts the throttle accordingly to maintain proper distance. A level 1 autonomous vehicle can control either acceleration/braking or steering, but not both simultaneously.

Level 2 – Partial Automation: Level 2 autonomous vehicles are able to control steering and acceleration/breaking at the same time for short periods without driver input. The cars can stay in lane and are able react to warning systems. However, the driver still has to be driving and alert.

Level 3 – Conditional Automation: At this level, the autonomous system is able to handle most of the driving situations. However, when it encounters a situation it can't navigate through, the control is transferred back to the driver.

Level 4 – High Automation: There is a significant leap in functionality from level 3 to level 4. Level 4 vehicles can monitor all roadway activities and perform all critical driving functions for the duration of the trip while operating in operational design domain. However, the driver needs to be aware while travelling and may have to alternate with the system in certain driving conditions.

Level 5 – Full Automation: Level 5 autonomous system functioning is equivalent to a driver in control of the vehicle. The car is able to navigate through any roads in any conditions as a human driver could operate.

1.2.2 Impact

The changes that autonomous vehicles will bring is still largely hypothetical, however there are a plethora of benefits that self-driving cars promises to bring.

Commuters save time – Traffic has always been a major issue that scorns even the most well-planned of cities. Commuters end up wasting a significant amount of time being stuck in traffic jams. Adopting a system of cloud-connected cars will enable the vehicles to effectively communicate with each other. This could alleviate traffic congestion and reduce travel times. Commuters could also use this time for other tasks related to work or entertainment that will lead to a better and more relaxed life.

Safety – Motor vehicle related accidents are a major cause of death and fatality every year. Autonomous vehicles aim to reduce casualties caused by traffic collisions due to human error. According to a recent study, advanced driver assistance systems and self-driving cars may cut traffic vehicle deaths by 90 percent and save 190 billion dollars every year in health care costs.

Environment – Autonomous vehicles will be able to optimize their drive cycles and improve fuel economy. This added to reduced time spent in traffic jams is bound to bring about lower energy consumption and reduced emissions.

1.3 Sensors

An autonomous vehicle must be able to see its environment so that it can recognize where it can and cannot go, identify pedestrians, detect other vehicles or obstacles on the road and handle unexpected situations.

The sensors used can be categorized into two parts, active and passive sensors. Passive sensors can obtain data from the environment without having to send out energy in the form of waves, such as a camera. Active sensors emit waves and receive data upon the information that comes back.

The sensors that will be focused on in this study are camera, LiDAR and radar.

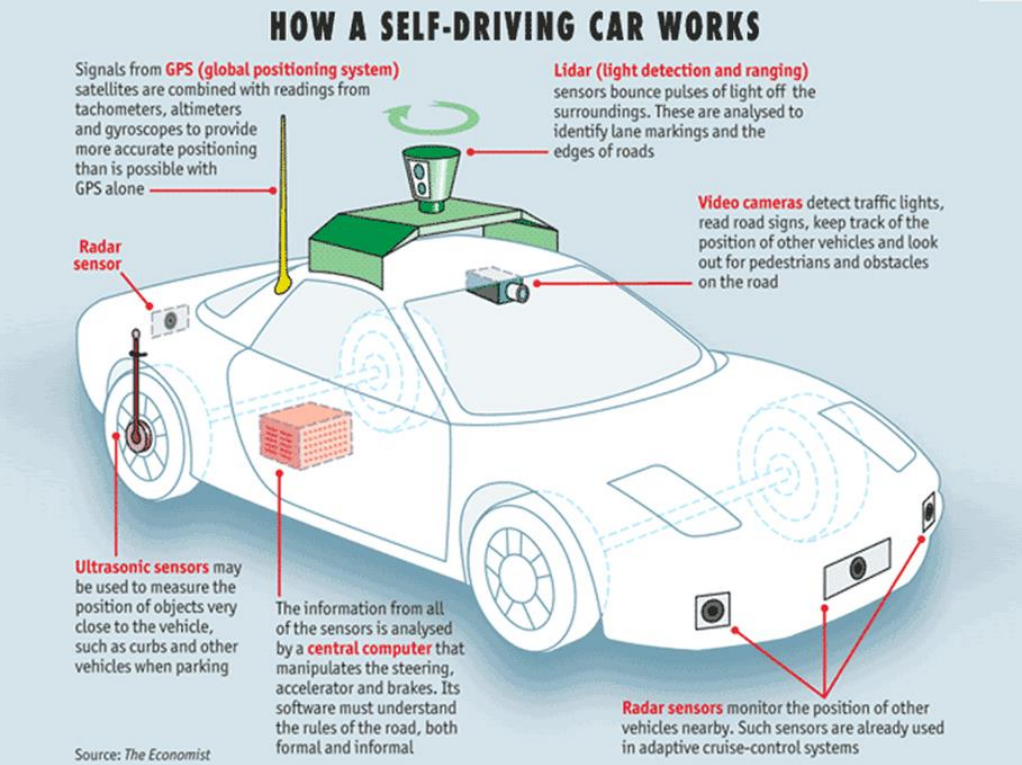


Fig 1.1 Model of an autonomous car

1.3.1 Digital video camera

Cameras can obtain high-resolution images with colour across the full width of its field of view. They efficiently detect and recognize objects and feed the image data to AI-based algorithms for object recognition and classification. With this, operations such as lane keeping, traffic sign detection and pedestrian avoidance becomes possible.

1.3.2 LIDAR

Another sensor often used in autonomous vehicle is the LIDAR system, short for Light Detection and Ranging. They emit laser beams that do not reach eye-harming levels. These beams are reflected by objects in the environment and bounce back to a photodetector. LIDAR sensors provide accurate 3-dimensional data on the surrounding environment, enabling the system to make split second decisions. The data obtained allows the processor to implement object identification, motion vector determination, collision prediction and avoidance strategies. LIDAR sensors are able to detect a 360º view by using a rotating, scanning mirror assembly on the top of the car. The major drawback with using LIDAR sensors is that they are very expensive.

1.3.3 Radar

The LIDAR system is not as effective for close range detection, which is required while parking, lane-changing, or in bumper-to-bumper traffic. For these cases, autonomous vehicles have been fitted with radar sensors that are fit into their front and back bumpers and sides of the vehicle.

1.4 Machine learning

To make sense of all the data collected about the environment by its sensors, deep learning and neural networks are used. Machine learning techniques process the vast data received from the cameras, LIDARs and radars and allow the vehicle to make decisions based on previously neural networks.

Chapter 2
Artificial Neural Networks

2.1 Overview

Influenced by the human brain, neural networks were designed to mimic the way human mind solve problems. The process by which our brain stores and process information occurs by the way of connections present between different neurons and by the varying relative strength of these connections. Without being specifically programmed, neural networks learn to perform tasks by considering examples or training data.

2.1.1 Structure of neural networks

Basic neural networks generally consist of three layers: the input layer, hidden layer and output layer. Information or data enters through the input layer, passes through the hidden layer and then out from the output layer. Each of these layers consist of several neurons and each of these neurons have a number and a formula associated with them called the **threshold value** and **activation function**. Each of the connections between these neurons have a number associated with it called **weights**.

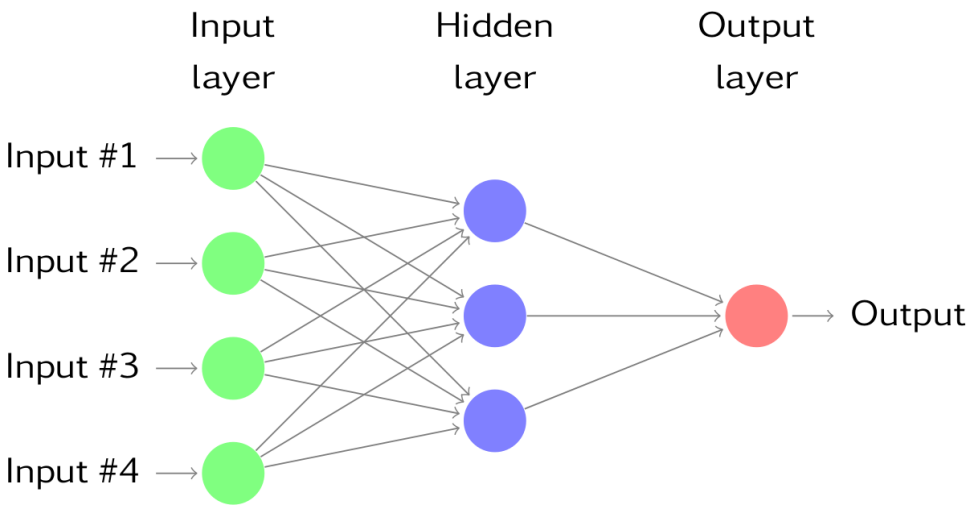


Fig 2.1 Model of a neural network

2.1.2 Training a simple neural network

On training basic neural networks, it is given a set of inputs and the corresponding outputs. The input is run through the neurons of each of the layers in the network where, based on the parameters defined above, the input is transformed and moves to the next layer. The output previously supplied is then compared to the out received at the output layer. Based on how far apart the outputs are, the parameters are adjusted on each of the neurons through the use of algorithms such as gradient descent and back propagation and the actual and produced outputs are brought as close to each other as possible. The neural network is able adjust its threshold values and weights to arrive at the correct output.

For the neural network to learn more complex models for prediction and classification of information that depends of thousands or millions of features, the neural networks needs to be made more complex. This is done by adding more hidden layers or increasing the neuron count per hidden layer.

2.2 Deep Learning

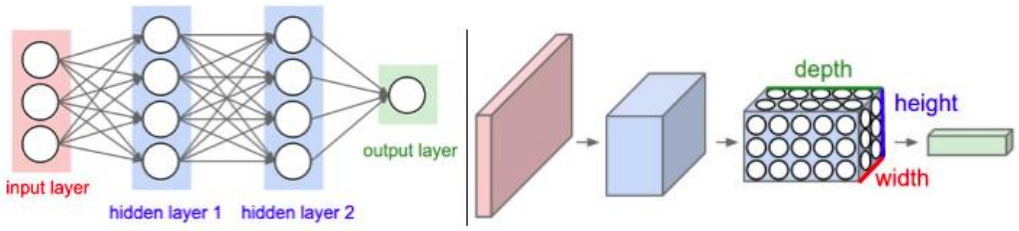
Neural networks that consist of three or more layers, including the input and output layer, are called as deep neural networks and the process of training these networks is known as deep learning. Deep neural networks brings the possibility to solve problems of extreme complexity through prediction and classification in the same manner.

Chapter 3
Convolutional neural networks

3.1 Introduction

The most commonly used type of neural network in deep learning is the convolutional neural network. Convolution is a mathematical term that refers to linear operations between matrixes. This type of neural network consists of layers that are 3 dimensional, the dimensions being width, height and depth. Unlike simple neural networks, the neurons in each layer only connect to a small part of the succeeding layer. The final output is then reduced to a single vector of probability scores, organized along the depth dimension. CNNs also reduce upon the number of parameters as compared to ANN.

Figure 3.1 Regular neural network vs convolutional neural network



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

3.2 Architecture

CNNs are made up of two components:

The Feature extraction component (Hidden layer)
In this stage, the CNN passes the data through a series of convolutions and pooling operations which In this part, the network will perform a series
of convolutions and pooling operations during which the features are detected. This is the part where the network is able to detect different features of the input. For example, if you had a picture of a person, it would be able to detect the person's face, arms and legs.

The Classification component
This component of the CNN consists of fully connected layers that serves as a classifier on top of these extracted features. The image is assigned a probability of what the network predicts it is. This classification layer only accepts one dimensional input. Hence, the 3D data needs to be converted into 1D data

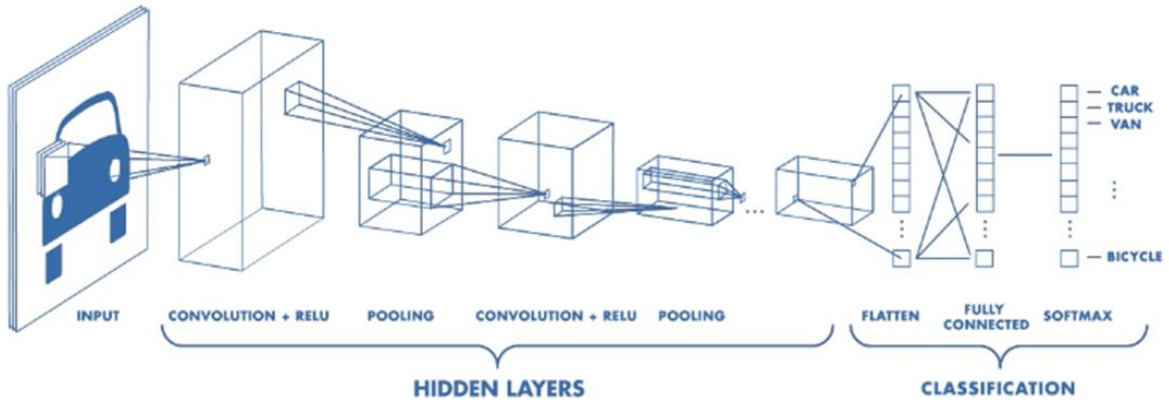


Figure 3.2 Architecture of a convolutional neural network

Chapter 4
Training a deep learning model

4.1 MNIST dataset

Using MATLAB, a convolutional neural network is trained using data from the MNIST data which provided 60,000 pre-labelled images of hand written digits from 0-9. An advantage to using this database was that the network could be trained without the need of investing in an expensive GPU.

Sample →

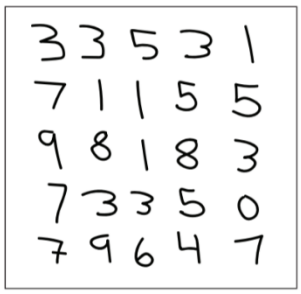


Figure 4.1 Labelled Dataset

4.2 Process

The convolutional neural network is trained through the following iterative steps:

Figure 4.2 Flow chart of training a model

4.3 Training

4.3.1 Attempt 1

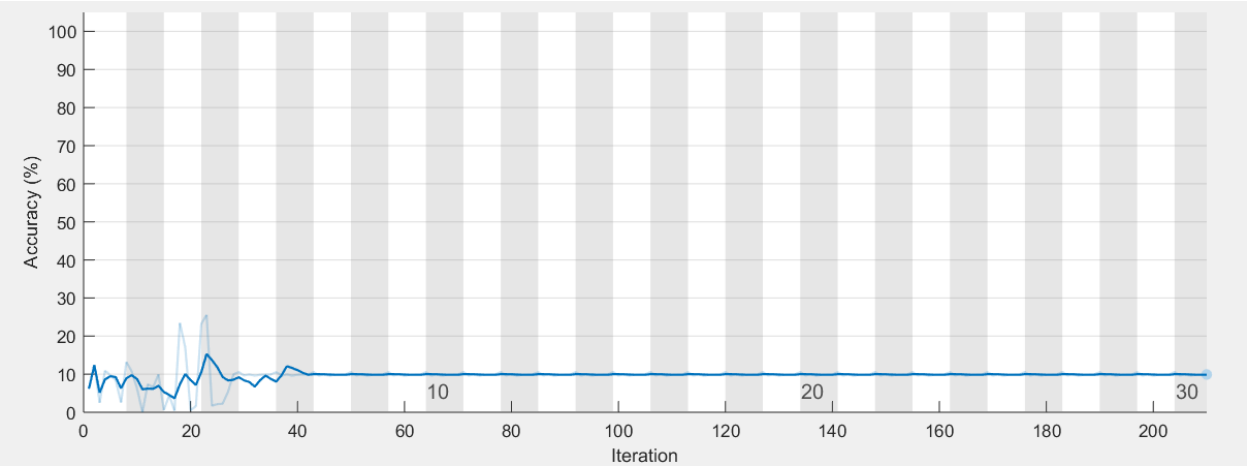


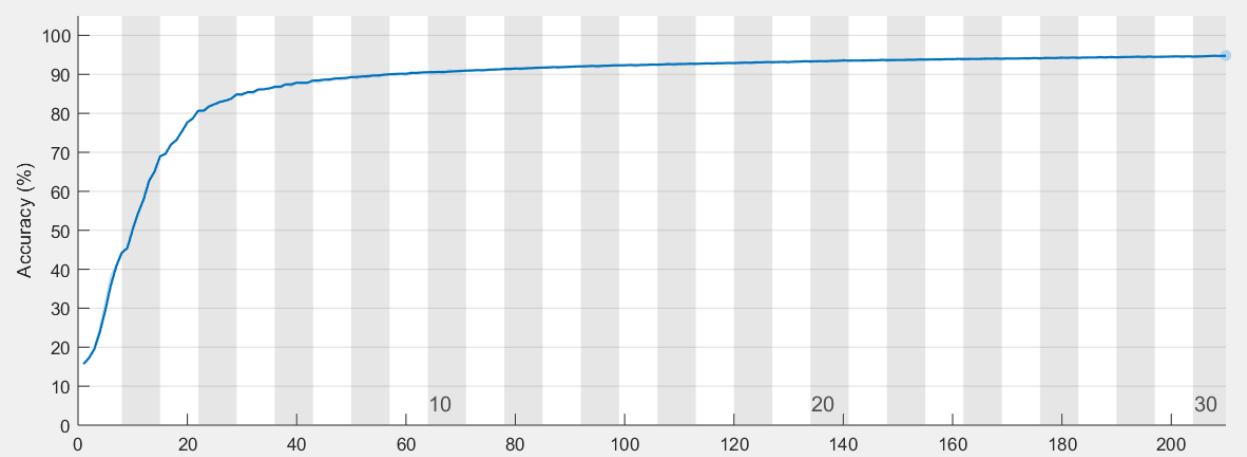
Figure 4.3 Training results of attempt 1

```
layers = [  imageInputLayer([28 28 1])
            convolution2dLayer(5,20)
            reluLayer
            maxPooling2dLayer(2, 'Stride', 2)
            fullyConnectedLayer(10)
            softmaxLayer
            classificationLayer()    ]
```

In this attempt, the learning rate was set to 0.01. The combination of the above layers was used. Using the following parameters, the model was trained to detect the handwritten digits with an accuracy of 10%.

Results	
Validation accuracy:	N/A
Training finished:	Reached final iteration
Training Time	
Start time:	24-Mar-2019 22:40:24
Elapsed time:	8 min 41 sec
Training Cycle	
Epoch:	30 of 30
Iteration:	210 of 210
Iterations per epoch:	7
Maximum iterations:	210
Validation	
Frequency:	N/A
Patience:	N/A
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

4.3.2 Attempt 2

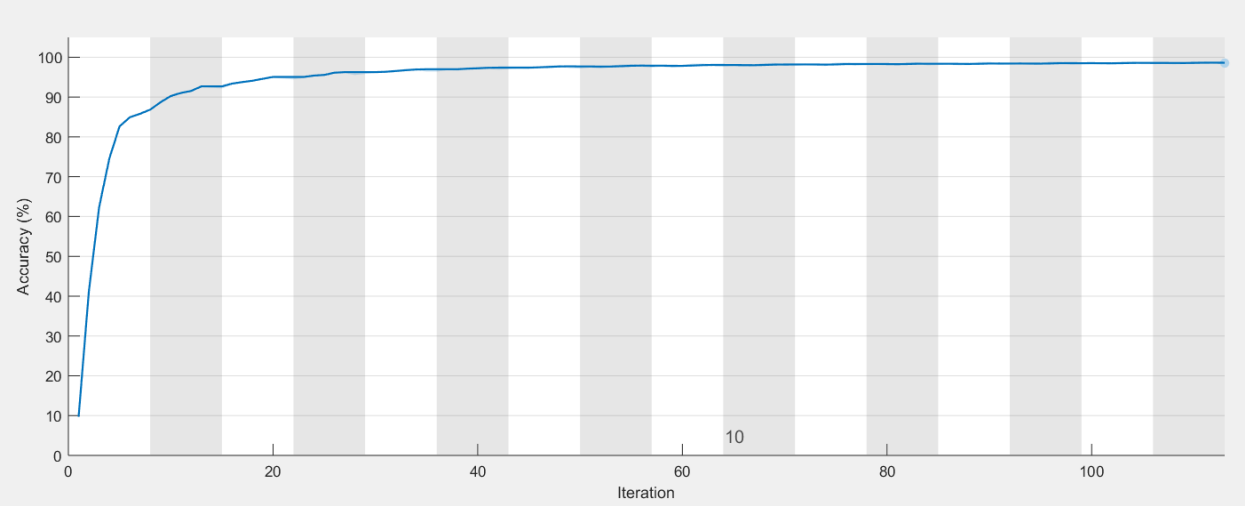


4.4 Training results of attempt 2

In the second attempt, the learning rate was reduced from 0.01 to 0.0001. The results improved vastly from the previous attempt from an accuracy of 10% to an accuracy of 95%.

Results	
Validation accuracy:	N/A
Training finished:	Reached final iteration
Training Time	
Start time:	24-Mar-2019 22:49:07
Elapsed time:	9 min 19 sec
Training Cycle	
Epoch:	30 of 30
Iteration:	210 of 210
Iterations per epoch:	7
Maximum iterations:	210
Validation	
Frequency:	N/A
Patience:	N/A
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

4.3.3 Attempt 3



4.5 Training results of attempt 3

On the third and last attempt, more layers and batch normalization layers were added, making the network deeper and complex, while the learning rate was rolled back to 0.01.

The accuracy of the deep learning model improved from 95% to 99%.

Training Time	
Start time:	24-Mar-2019 22:58:29
Elapsed time:	13 min 21 sec
Training Cycle	
Epoch:	17 of 30
Iterations per epoch:	7
Maximum iterations:	210
Validation	
Frequency:	N/A
Patience:	N/A
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Chapter 5
Training a network to detect vehicles

5.1 Overview

Being able to detect and track cars is necessary for the ego vehicle to operate autonomously. For training the neural network to detect vehicles in still images, faster R-CNN's are utilized.

5.2 CNN vs R-CNN

CNN's are primarily used to classify images, where as R-CNN's, R standing for region, are used to detect objects. Traditional CNN's can classify objects but cannot convey where they are. It is possible to regress bounding boxes using a CNN, but that is only possible for one object at a time. This is because regressing multiple boxes causes interference.

In R-CNN's, one object is focused at a time, so a single object of interest dominates in a given region, this minimizes interference. The regions detected are then resized so that equally sized regions are fed into the CNN for classification and bounding box regression.

5.3 Training the R-CNN

To train this model, a dataset consisting of 295 pre-labelled images of vehicles are used. The images consist of one or more labelled instances of a vehicle.

The detector is trained in four steps. The first and second step train the region proposal and detection networks. The third and fourth steps join the first two networks and creates a single network for detection. The network training options are specified using training options.

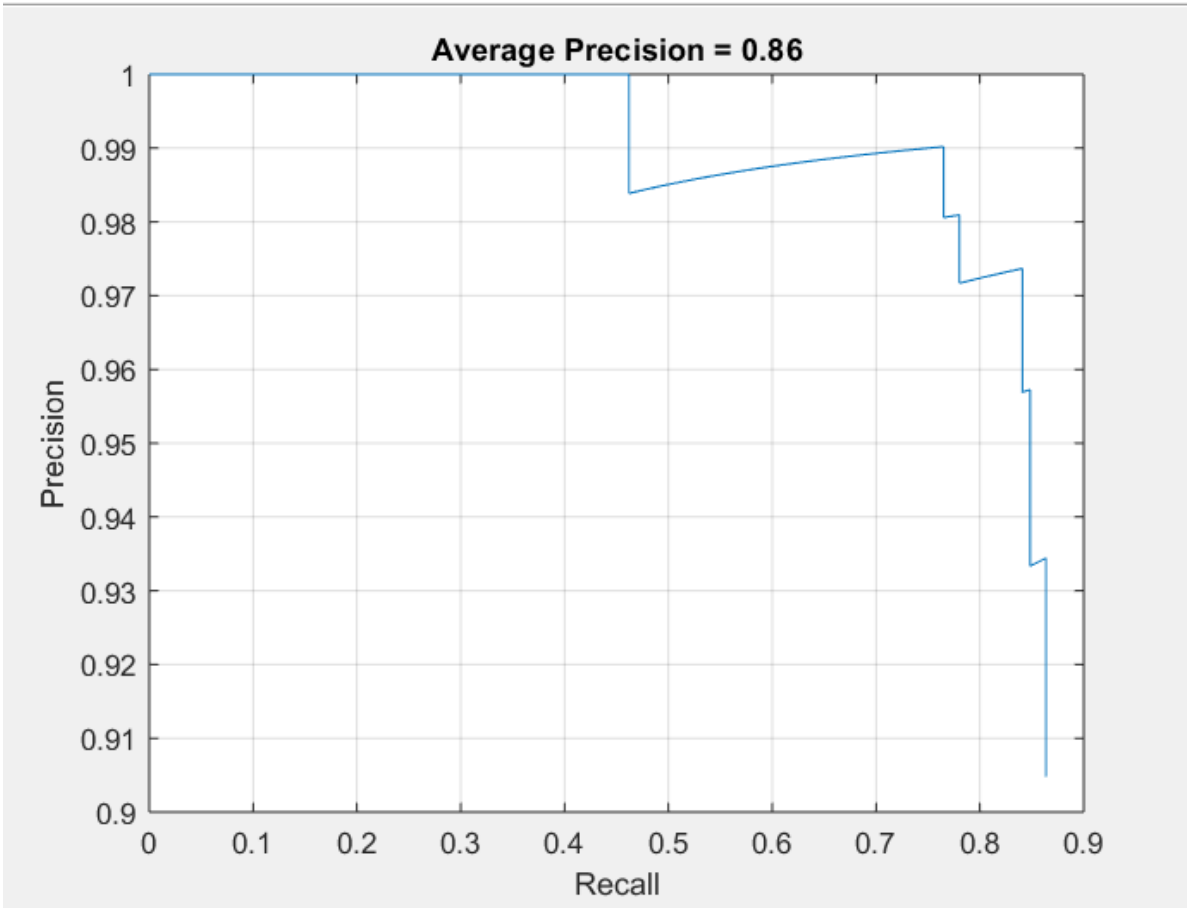
As the dataset consists of images with different sizes, the minibatch size is set to 1. This ensures that these images won't be processed together.

```
% Options for step 1.
options = trainingOptions('sgdm', ...
    'MaxEpochs', 5, ...
    'MiniBatchSize', 1, ...
    'InitialLearnRate', 1e-3, ...
    'CheckpointPath', tempdir);
```

Setting a temporary location for checkpoint path will allow you to pause training of the network and lets you later resume from where you left off. A pretrained CNN called ResNet-50 is used for feature extraction.

On training the R-CNN with the specified parameters, the network was able to detect cars with an average accuracy of 86%. The time taken to complete training totaled to 4 hours.

Fig 5.1 Training results



Chapter 6
Tracking vehicles and pedestrian in video

6.1 Overview

In the real world, vehicles need to be identified and tracked from video input rather than still frames. This is done by identifying the vehicles using the previously trained R-CNN in each frame of the video and constantly updating the tracker with the detection results.

6.2 Workflow

The tracking workflow consists of the following steps:

1) Define camera intrinsics and camera mounting position.
2) Load and configure a pretrained vehicle detector.
3) Set up a multi-object tracker.
4) Run the detector for each video frame.
5) Update the tracker with detection results.
6) Display the tracking results in a video.

In this example, a pretrained ACF vehicle detector is used and configured to incorporate camera information. By default, the detector scans the entire image at multiple scales. By knowing the camera parameters, the detector is configured to detect vehicles on the ground plane only at reasonable scales. The detector only tries to find vehicles at image regions above the ground plane. This can reduce computation and prevent spurious detections. At each time step, the detector is run, the tracker is updated with detection results, and the tracking results are displayed on the video.

6.3 Supporting functions

setupTracker function creates a multiObjectTracker to track multiple objects with Kalman filters. The following functions are considered when multiObjectTracker is created:

FilterInitializationFcn: Sets up models for predicting likely motion and measurements. The objects are bound to have a constant velocity motion in this scenario.

AssignmentThreshold: Used to determine how far the detections can fall from tracks. By default, the value of this parameter is set to 30. If detections that should've been assigned to tracks but aren't, this parameter is increased. If detections that are too far have been assigned to tracks, this value is decreased. This example uses 50.

NumCoastingUpdates: The number of times a track is updated or coasted before being deleted. This parameter has a default value of 5.

ConfirmationParameters: Track is conformed using this parameter. With every unassigned detection, a new track is initialized. As some of these detections may be false, the tracks were initialized as Tentative. The track needs to be detected at least M times in N tracker updates for the track to be confirmed. M and N are determined based on the visibility of the objects. 3 out of 5 updates have been used for this example.

6.4 Tracking vehicles

Using the trained R-CNN and MATLAB's automated driving toolbox, the networks was able to detect and track vehicles from video input.

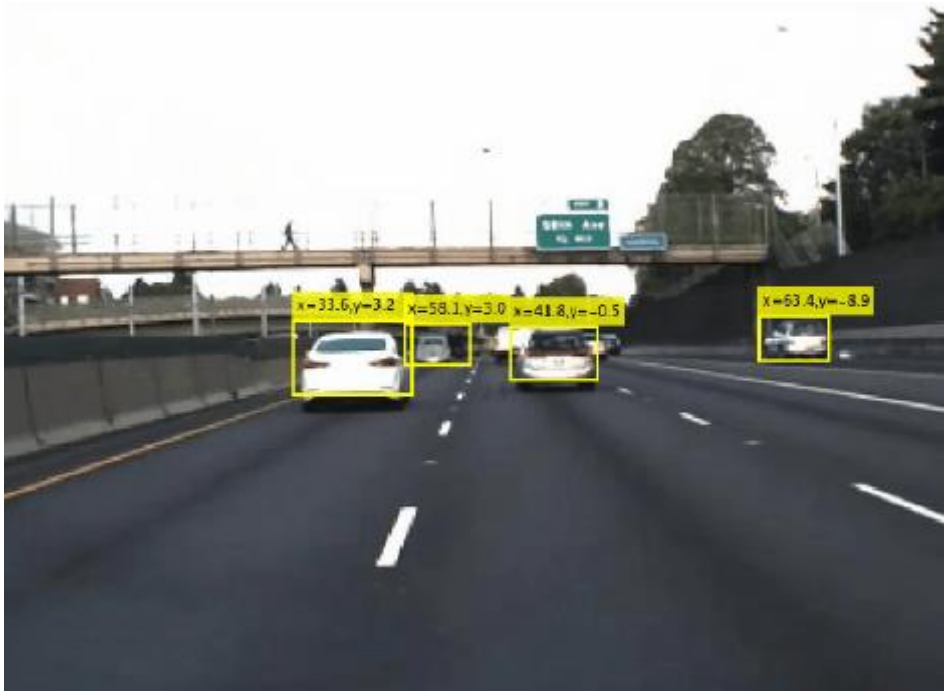


Fig 6.1 Neural network detecting vehicles from video input

6.5 Tracking pedestrians

First, this example computes the cost of assigning every detection to each track by using thebboxOverlapRatio measure. As people move toward or away from the camera, the centroid point alone cannot accurately describe their motion. The cost takes into account the distance on the image plane, and the scale of the bounding boxes. This prevents assigning detections that are far away from the camera to tracks that are closer to the camera, even if the predicted and detected centroids coincide. The choice of this cost function eases the computation without resorting to a more sophisticated dynamic model. The results are stored in an M by N matrix, where M is the number of tracks, and N is the number of detections.

Fig 6.2 Neural network detecting pedestrians from video input

Chapter 7
LIDAR input

7.1 Overview

Lidar sensors report measurements as a point cloud. The lidar data used in this example is recorded from a highway driving scenario. In this example, the recorded data to track vehicles is used with a joint probabilistic data association (JPDA) tracker and an interacting multiple model (IMM) approach.

7.2 Workflow

Lidar sensors report measurements as a point cloud. The lidar data used in this example is recorded from a highway driving scenario. In this example, you use the recorded data to track vehicles with a joint probabilistic data association (JPDA) tracker and an interacting multiple model (IMM) approach. Due to high resolution capabilities of the lidar sensor, each scan from the sensor contains a large number of points, commonly known as a point cloud. This raw data must be preprocessed to extract objects of interest, such as cars, cyclists, and pedestrians. In this example, the point clouds belonging to obstacles are further classified into clusters using the pcsegdist function, and each cluster is converted to a bounding box detection with the following format:

$$[x\ y\ z\ l\ w\ h]$$

The bounding box is fit onto each cluster by using minimum and maximum of coordinates of points in each dimension. The detector is implemented by a supporting class HelperBoundingBoxDetector, which wraps around point cloud segmentation and clustering functionalities. An object of this class accepts a pointCloudinput and returns a list of objectDetection objects with bounding box measurements.

The diagram shows the processes involved in the bounding box detector model and the Computer Vision Toolbox functions used to implement each process. It also shows the properties of the supporting class that control each process.

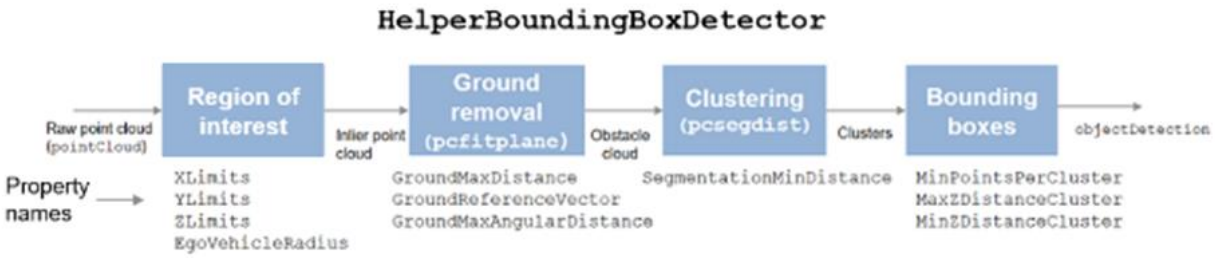


Fig 7.1 Bounding box detector model

7.3 Tracker set up and visualisation

The image below shows the complete workflow to obtain a list of tracks from a pointCloud input.

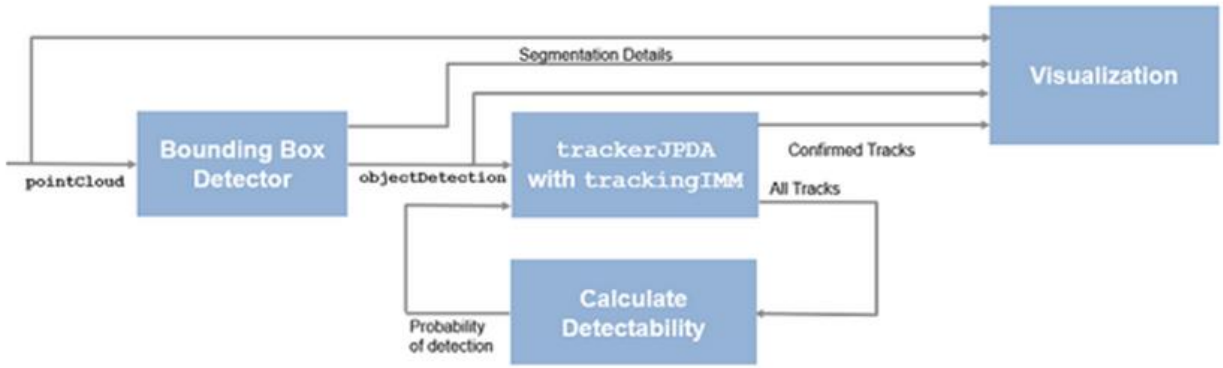


Fig 7.2 Workflow for obtaining tracks from pointCloud input

A joint probabilistic data association tracker (trackerJPDA) coupled with an IMM filter (trackingIMM) is used to track objects in this example. The IMM filter uses a constant velocity and constant turn model and is initialized using the supporting function, helperInitIMMFilter, included with this example. The IMM approach helps a track to switch between motion models and thus achieve good estimation accuracy during events like maneuvering or lane changing. Set the HasDetectableTrackIDsInput property of the tracker astrue, which enables a state-dependent probability of detection to e specified. The detection probability of a track is calculated by the helperCalcDetectability function, listed at the end of this example.

The visualization is divided into these main categories:

1. Lidar Preprocessing and Tracking - This display shows the raw point cloud, segmented ground, and obstacles. It also shows the resulting detections from the detector model and the tracks of vehicles generated by the tracker.

2. Ego Vehicle Display - This display shows the 2-D bird's-eye view of the scenario. It shows the obstacle point cloud, bounding box detections, and the tracks generated by the tracker. For reference, it also displays the image recorded from a camera mounted on the ego vehicle and its field of view.

3. Tracking Details - This display shows the scenario zoomed around the ego vehicle. It also shows finer tracking details, such as error covariance in estimated position of each track and its motion model probabilities, denoted by cv and ct.

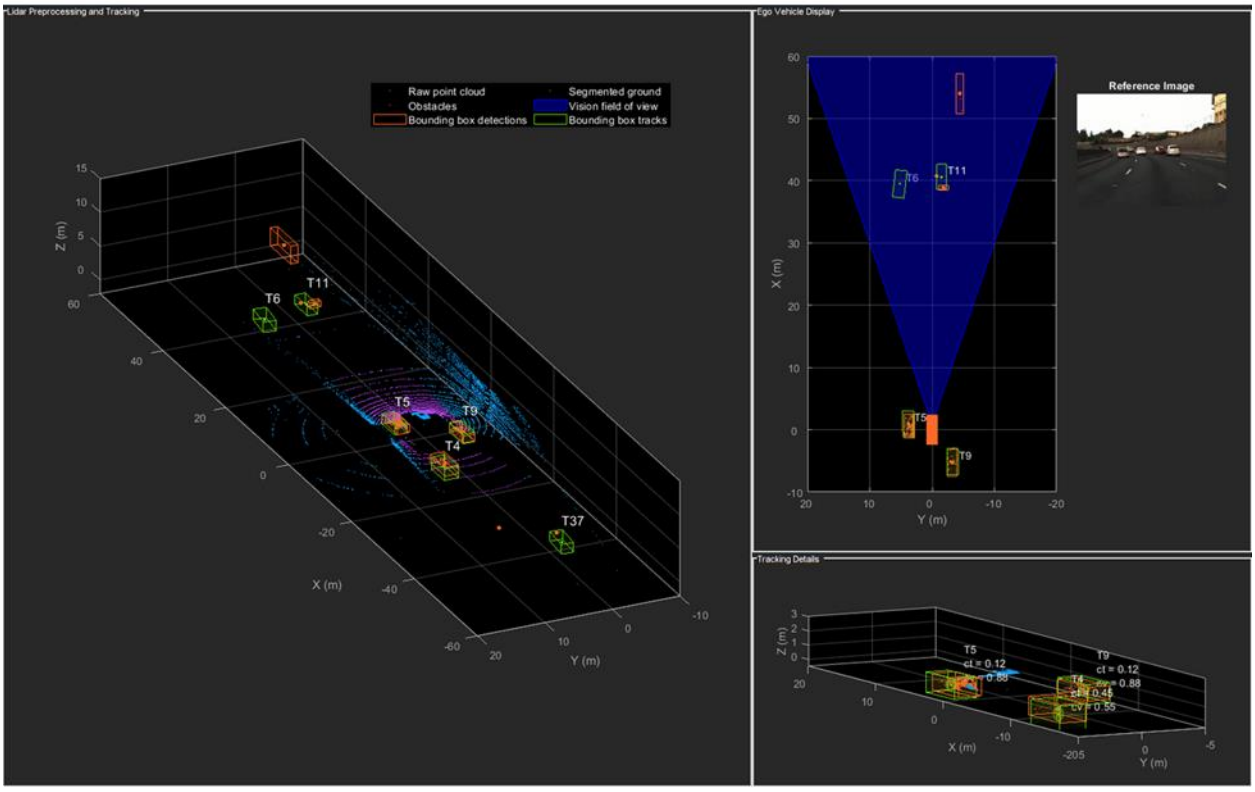


Fig 7.3 Modelling input from LIDAR sensors

Chapter 8
Conclusion and Future Scope

8.1 Conclusion

This paper covers the basic functioning of neural networks and goes in-depth on how deep learning works and how it's implemented for solving different models. Using MATLAB's deep learning toolkit, I was able to train convolutional neural networks to detect and classify images that depicted handwritten digits. Further down the course of this study project, using MATLAB's automated driving toolkit, I trained an R-CNN to identify vehicles and pedestrians in individual frames and then subsequently track them over the length of the video input. To gain further understanding in modelling of data traditionally used in autonomous vehicles, with the help of MATLAB's automated driving toolkit, the input received from LIDAR sensors was processed and modelled to re-create the 360-degree environment of the ego vehicle.

8.2 Future Scope

The models explored and used in this project can be further developed to be implemented in real world models of vehicles in order to achieve certain levels of autonomy. Over the last 50 years, there have been significant advancements in technology that helped speed up the development towards completely autonomous vehicles. Although the future regarding autonomous vehicles remains uncertain in the face of rapidly changing legal and political institutions, there is no doubt that car manufacturers will continue to develop and implement advanced technology as the race towards autonomy progresses.