

A NOVEL LEARNING DEEP Q NETWORK FOR DENSE TRAFFIC CONTROL

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

KORRAPATI PREMCHAND, KURNI JAGADISH (38110265, 38110281)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL – 2020



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of KORRAPATI PREMCHAND, KURNI JAGADISH (38110265, 38110281) who carried out the project entitled **"A NOVEL LEARNING DEEP Q NETWORK FOR DENSE TRAFFIC CONTROL"** under my supervision from November 2019 to April 2020.

Internal Guide
Dr.V.Maria Anu, M.E, Ph.D

Head of the Department

Dr.Vigneshwari.S, M.E., Ph.D.

Dr.Lakshmanan.L, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **P.CHANAKYA REDDY** and **A.NIKHIL**, hereby declare that the Project Report entitled **A NOVEL LEARNING DEEP Q NETWORK FOR DENSE TRAFFIC CONTROL** done by me under the guidance of **Dr.V.Maria Anu, M.E, Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer Science and Engineering.

DATE:

PLACE :Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Sathyabama Institute of Science and Technology management** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D. and Dr.L.Lakshmanan M.E., Ph.D. , Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.Maria Anu, M.E, Ph.D.**,for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project .

ABSTRACT

Road intersection management is done through traffic lights, whose inefficient control causes numerous problems, such as long delay of travelers and huge waste of energy. Even worse, it may also contribute to vehicular accidents. Existing traffic light control either deploys fixed programs without considering real-time traffic or considering the traffic to a very limited degree. The Existing model is composed of several components to improve the performance, such as dueling network, target network, double Q-learning network, and prioritized experience replay. We evaluate our model via simulation in the Simulation of Urban MObility (SUMO) in a vehicular network, and the simulation results show the efficiency of our model in controlling traffic lights. In the proposed system design of a Dual Learning approach using CNN and KNN algorithms. The proposed system is more keen in predicting the accurate classification and finding out the density of the traffic in 4 way roads through thr pre-determined sample dataset. Based on the training and testing of the proposed DL model the traffic light controls will be appropriately changed.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 CAUSES	1
	1.2 ECONOMIC THEORIES	2
	1.3 CLASSIFICATION	2
	1.4 TRAFFIC CONGESTION HAS NEGATIVE EFFECTS	3
	1.5 EMERGENCIES	3
	1.6 URBAN PLANNING AND DESIGN	4
2	LITERATURE SURVEY	5
3	DESIGN METHODOLOGY	
	3.1 THE MATLAB SYSTEM	8
	3.1.1 The Matlab Language	8
	3.1.2 The Matlab Working Environment	8
	3.1.3 Handle Graphics	8
	3.1.4 The Matlab Library	9
	3.1.5 The Matlab API	9
	3.2 OVERVIEW OF PYTHON	9
	3.2.1 Python	9
	3.2.2 Why Python	10
	3.3 NEURAL NETWORKS	10
	3.3.1 How Neural Networks Work	10
	3.3.2 Neural Networks Working Principle	11
	3.3.3 Techniques with Neural Networks	11

3.3.4 Typical Neural Networks Architecture	11
3.4 SUPERVISED LEARNING	11
3.4.1 Deep Learning	12
3.4.2 Pattern Recognition	12
3.4.3 Unsupervised Learning	12
3.4.4 Clustering	13
3.5 SHALLOW NEURAL NETWORKS WITH MATLAB	13
3.6 SYSTEM ARCHITECTURE	14
3.6.1 Convolutional Neural Networks	14
3.6.2 Knn	16
3.6.3 Feature Extraction	16
3.6.4 Feature Selection	17
3.6.5 Workflow for Neural Networks	17
3.6.6 Classification and Clustering	17
3.6.7 Preprocessing and Postprocessing	18
3.7 OVERALL ARCHITECTURE	19
3.7.1 Solution to Improve Generalization	20
3.7.2 Simulink Support	20
4. RESULT AND DISCUSSION	
4.1 RESULTS	22
4.2 DISCUSSION	22
4.2.1 Write and Run Test Scripts	22
4.2.2 Function Based Unit Tests	22
4.2.3 Performance Testing Framework	23
4.2.3.1 Measure Performance of Code	23
4.2.3.2 Measured Estimate of Framework	23
4.3 CODE GENERATION AND DEPLOYMENT	23
5. SUMMARY AND CONCLUSION	
5.1 SUMMARY	25

5.2 CONCLUSION AND FUTURE WORK	25
REFERENCES	26
APPENDIX	
a. SOURCE CODE	29
b. SCREENSHOTS	32
c. PAPER	36

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
3.1	Typical Neural Network Architecture	11
3.2	System Architecture	14
3.3	Convolutional Neural Network	15
3.4	Overall Architecture	19

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
KNN	K-Nearest Neighbour
SUMO	Simulation of Urban Mobility

CHAPTER 1

INTRODUCTION

Traffic congestion is a condition on transport that is characterised by slower speeds, longer trip times, and increased vehicular queueing. Traffic congestion on urban road networks has become increasingly problematic since the 1950s. When traffic demand is great enough that the interaction between vehicles slows the speed of the traffic stream, this results in some congestion. While congestion is a possibility for any mode of transportation, this article will focus on automobile congestion on public roads.

As demand approaches the capacity of a road (or of the intersections along the road), extreme traffic congestion sets in. When vehicles are fully stopped for periods of time, this is known as a traffic jam or (informally) a traffic snarl-up. Traffic congestion can lead to drivers becoming frustrated and engaging in road rage.

Mathematically, congestion is usually looked at as the number of vehicles that pass through a point in a window of time, or a flow. Congestion flow lends itself to principles of fluid dynamics.

1.1 Causes:

Traffic congestion occurs when a volume of traffic or modal split generates demand for space greater than the available street capacity, this point is commonly termed saturation. There are a number of specific circumstances which cause or aggravate congestion, most of them reduce the capacity of a road at a given point or over a certain length, or increase the number of vehicles required for a given volume of people or goods. About half of U.S. traffic congestion is recurring, and is attributed to sheer weight of traffic, most of the rest is attributed to traffic incidents, road work and weather events. In terms of traffic operation, rainfall reduces traffic capacity and operating speeds, thereby resulting in greater congestion and road network productivity loss.

Traffic research still cannot fully predict under which conditions a "traffic jam" (as opposed to heavy, but smoothly flowing traffic) may suddenly occur. It has been found that individual incidents (such as accidents or even a single car braking heavily in a previously smooth flow) may cause ripple effects (a cascading failure)

which then spread out and create a sustained traffic jam when, otherwise, normal flow might have continued for some time longer.

1.2 Economic Theories:

India's economic growth has resulted in a massive increase in the number of private vehicles on its roads overwhelming the transport infrastructure. Shown here is a traffic jam in Delhi.

Congested roads can be seen as an example of the tragedy of the commons. Because roads in most places are free at the point of usage, there is little financial incentive for drivers not to over-use them, up to the point where traffic collapses into a jam, when demand becomes limited by opportunity cost. Privatization of highways and road pricing have both been proposed as measures that may reduce congestion through economic incentives and disincentives. Congestion can also happen due to non-recurring highway incidents, such as a crash or roadworks, which may reduce the road's capacity below normal levels.

Rapid economic growth in China has resulted in a massive increase in the number of private vehicles in its major cities. Shown here is a traffic jam downtown Haikou City, Hainan Province.

1.3 Classification:

Qualitative classification of traffic is often done in the form of a six letter A-F level of service (LOS) scale defined in the Highway Capacity Manual, a US document used (or used as a basis for national guidelines) worldwide. These levels are used by transportation engineers as a shorthand and to describe traffic levels to the lay public. For instance, while the percent time spent following a slower-moving vehicle figures into the LOS for a rural two-lane road, the LOS at an urban intersection incorporates such measurements as the number of drivers forced to wait through more than one signal cycle.

Traffic congestion occurs in time and space, i.e., it is a spatiotemporal process. Therefore, another classification schema of traffic congestion is associated with some common spatiotemporal features of traffic congestion found in measured traffic data. Common spatiotemporal empirical features of traffic congestion are those features, which are qualitatively the same for different highways in different countries

measured during years of traffic observations. Common features of traffic congestion are independent on weather, road conditions and road infrastructure, vehicular technology, driver characteristics, day time, etc. Examples of common features of traffic congestion are the features [J] and [S] for, respectively, the wide moving jam and synchronized flow traffic phases found in Kerner's three-phase traffic theory. The common features of traffic congestion can be reconstructed in space and time with the use of the ASDA and FOTO models.

1.4 Traffic congestion has several negative effects:

Wasting time of motorists and passengers ("opportunity cost"). As a non-productive activity for most people, congestion reduces regional economic health.

Delays, which may result in late arrival for employment, meetings, and education, resulting in lost business, disciplinary action or other personal losses.

Inability to forecast travel time accurately, leading to drivers allocating more time to travel "just in case", and less time on productive activities.

Wasted fuel increasing air pollution and carbon dioxide emissions owing to increased idling, acceleration and braking.

Wear and tear on vehicles as a result of idling in traffic and frequent acceleration and braking, leading to more frequent repairs and replacements.

Stressed and frustrated motorists, encouraging road rage and reduced health of motorists

1.5 Emergencies:

Blocked traffic may interfere with the passage of emergency vehicles traveling to their destinations where they are urgently needed.

Spillover effect from congested main arteries to secondary roads and side streets as alternative routes are attempted ('rat running'), which may affect neighborhood amenity and real estate prices.

Higher chance of collisions due to tight spacing and constant stopping-and-going.

1.6 Urban planning and design:

City planning and urban design practices can have a huge impact on levels of future traffic congestion, though they are of limited relevance for short-term change.

Grid plans including fused grid road network geometry, rather than tree-like network topology which branches into cul-de-sacs (which reduce local traffic, but increase total distances driven and discourage walking by reducing connectivity). This avoids concentration of traffic on a small number of arterial roads and allows more trips to be made without a car.

CHAPTER 2

LITERATURE SURVEY

[1].Recognition of the necessity for connected and automated vehicles (CAVs) is gaining momentum. CAVs can improve both transportation network efficiency and safety through control algorithms that can harmonically use all existing information to coordinate the vehicles. This study addresses the problem of optimally coordinating CAVs at merging roadways to achieve a smooth traffic flow without stop-and-go driving. We present an optimization framework and an analytical, closed-form solution that allows online coordination of vehicles at the merging zone. The effectiveness of the efficiency of the proposed solution is validated through simulation and it is shown that coordination of 12vehicles can reduce fuel consumption and travel time significantly.

[2].The emergence of traffic and subsequently traffic congestion in urban road networks are increasing worldwide with the growing number of vehicles. In the worst case, traffic congestion results in excess delays, reduced safety and increased environmental pollution. As a result, to proffer effective solutions to the aforementioned congestion problems, a vast amount of literature offered fixed-time or traffic-response control strategies, each focusing on modelling traffic light controlled intersections. In this paper, a Timed Coloured Petri Net (TCPN) formalism is employed to model and simulate a multi-phase traffic light controlled intersection with an associated fixed signal timing plan using T-type junction located in Federal Capital Territory, Abuja, Nigeria, as a case study. Besides, the developed model is simulated via CPN tools while its validation is carried out on the basis of real traffic data measured by detectors positioned at each input link of the T-type junction under consideration.

[3].Steady-state traffic flow on a ring road with up- and down-slopes is investigated using a semi-discrete model. By exploiting the relations between the semi-discrete and continuum models, a steady-state solution is uniquely determined for a given total number of vehicles on a ring road. The solution is exact and always stable with respect to the first-order continuum model, whereas it is a good approximation with respect to the semi-discrete model provided that the involved equilibrium constant states are linearly stable. In other cases, the instability of one or more equilibria

could trigger stop-and-go waves propagating in certain sections of road or throughout the ring road. The indicated results are reasonable and thus physically significant for better understanding of real-world traffic flow on inhomogeneous roads, such as those with junctions or bottlenecks.

[4].With the success of deep learning in artificial intelligence, more and more researchers use deep learning to solve transportation problems. A cooperative traffic light control system based on reinforcement learning is proposed. The authors propose to cluster vehicles and use a linear function to approximate the Q values; however, only the queue information is used in the states. The complexity in a traffic road system can not be actually presented by such limited information. With the development of deep learning and reinforcement learning, they are combined together as deep reinforcement learning to estimate the Q value. Some researchers have applied deep reinforcement learning to control the wireless communication but the systems cannot be directly applied in traffic light control scenarios due to different actions and states. Game-based method is proposed in to manage the wireless communication in Unmanned Aerial Vehicles (UAVs) and a Q learning method is proposed into efficiently allocate communication resources in multiple UAs.

[5].Our problem is to optimize the efficiency of the intersection usage by dynamically changing every phase's duration of a traffic light via learning from historical experiences. The duration of a phase should be extended strategically if there are more vehicles in that direction. In this paper, we build a deep Q learning network to learn the timing strategy of every phase to optimize the traffic management. Our network self updates by continuously receiving states and rewards from the environment. The left side shows the structure in a traffic light. The traffic light first gathers road traffic information via a vehicular network or other tools, which is presented by the dashed purple lines in the figure. The traffic light processes the data to obtain the road traffic's state and reward, which has been assumed in many previous studies. The traffic light chooses an action based on the current state and reward using a deep neural network shown in the right side. The left side is the reinforcement learning part and the right side is the deep learning part.

CHAPTER 3

DESIGN METHODOLOGY

SOFTWARE REQUIREMENTS:

- **MATLAB R2018a**
- **Python**

HARDWARE REQUIREMENTS:

- **PC, Pentium 4 processor, 1 GB RAM, CPU 3.06 GHz**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System:

3.1 The MATLAB system consists of five main parts:

3.1.1 The MATLAB language:

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

3.1.2 The MATLAB working environment:

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

3.1.3 Handle Graphics:

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands

that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

3.1.4 The MATLAB mathematical function library:

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

3.1.5 The MATLAB Application Program Interface (API):

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

3.2 OVERVIEW OF THE PLATFORM:

3.2.1 Python:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

3.2.2 Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

3.3 NEURAL NETWORKS:

A neural network is a computing model whose layered structure resembles the networked structure of neurons in the brain, with layers of connected nodes. A neural network can learn from data—so it can be trained to recognize patterns, classify data, and forecast future events.

A neural network breaks down your input into layers of abstraction. It can be trained over many examples to recognize patterns in speech or images, for example, just as the human brain does. Its behavior is defined by the way its individual elements are connected and by the strength, or weights, of those connections. These weights are automatically adjusted during training according to a specified learning rule until the neural network performs the desired task correctly.

Neural networks are especially well suited to perform pattern recognition to identify and classify objects or signals in speech, vision, and control systems. They can also be used for performing time-series prediction and modeling.

3.3.1 Here are just a few examples of how neural networks are used:

- Power companies accurately forecast the load on their electric grids to

ensure reliability and optimize efficiency of the power generators they operate.

- ATMs reliably accept bank deposits by reading the account number and deposit amount on your check.
- Pathologists rely on cancer detection applications to guide them in classifying tumors as benign or malignant, based on uniformity of cell size, clump thickness, mitosis, and other factors.

3.3.2 Neural Networks Working Principle:

A neural network combines several processing layers, using simple elements operating in parallel and inspired by biological nervous systems. It consists of an input layer, one or more hidden layers, and an output layer. The layers are interconnected via nodes, or neurons, with each layer using the output of the previous layer as its input.

3.3.3 Techniques Used with Neural Networks:

Common machine learning techniques for designing neural network applications include supervised and unsupervised learning, classification, regression, pattern recognition, and clustering.

3.3.4 Typical neural network architecture:

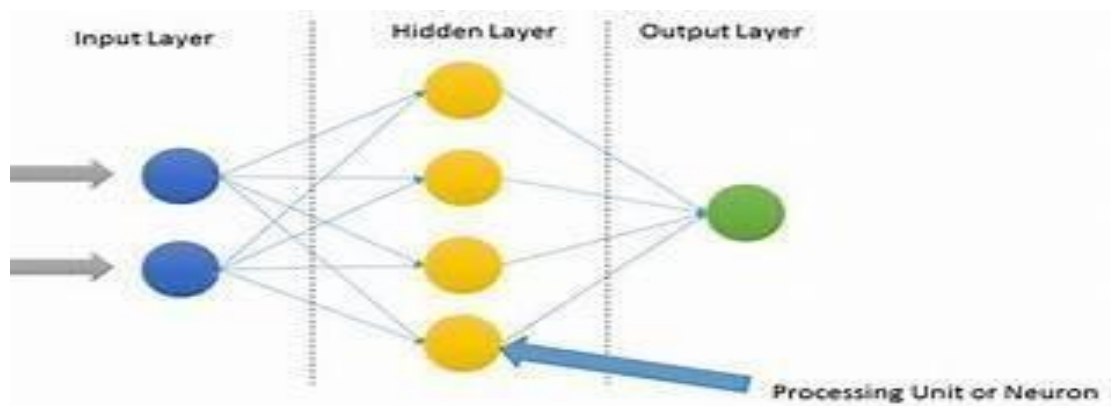


Fig 3.1: Typical neural network architecture

3.4 Supervised Learning:

Supervised neural networks are trained to produce desired outputs in response to sample inputs, making them particularly well suited for modeling and controlling dynamic systems, classifying noisy data, and predicting future

events. Deep Learning Toolbox™ includes four types of supervised networks: feedforward, radial basis, dynamic, and learning vector quantization.

3.4.1 Deep Learning:

Neural networks that operate on two or three layers of connected neuron layers are known as shallow neural networks. Deep learning networks can have many layers, even hundreds. Both are machine learning techniques that learn directly from input data.

Deep learning is getting lots of attention, and for good reason. It's achieving results that were not possible before.

Deep learning is especially well suited to complex identification applications such as face recognition, text translation, and voice recognition. It's also a key technology used in advanced driver assistance systems and tasks including lane classification and traffic sign recognition.

3.4.2 Pattern Recognition:

Pattern recognition is an important component of neural network applications in computer vision, radar processing, speech recognition, and text classification. It works by classifying input data into objects or classes based on key features, using either supervised or unsupervised classification.

For example, in computer vision, supervised pattern recognition techniques are used for optical character recognition (OCR), face detection, face recognition, object detection, and object classification. In image processing and computer vision, unsupervised pattern recognition techniques are used for object detection and image segmentation.

3.4.3 Unsupervised Learning:

Unsupervised neural networks are trained by letting the neural network continually adjust itself to new inputs. They are used to draw inferences from data sets consisting of input data without labeled responses. You can use them to discover natural distributions, categories, and category relationships within data.

Deep Learning Toolbox includes two types unsupervised networks: competitive layers and self-organizing maps.

3.4.4 Clustering:

Clustering is an unsupervised learning approach in which neural networks can be used for exploratory data analysis to find hidden patterns or groupings in data. This process involves grouping data by similarity. Applications for cluster analysis include gene sequence analysis, market research, and object recognition.

3.5 Developing Shallow Neural Networks with MATLAB:

With tools and functions for managing large data sets, MATLAB® offers specialized toolboxes for working with machine learning, neural networks, deep learning, computer vision, and automated driving.

With just a few lines of code, MATLAB lets you develop neural networks without being an expert. Get started quickly, create and visualize models, and deploy models to servers and embedded devices.

With MATLAB, you can integrate results into your existing applications. MATLAB automates deploying your neural network models on enterprise systems, clusters, clouds, and embedded devices.

Workflow for Neural Network Design

Getting Started with Deep Learning Toolbox (4:20)

Wine Classification with Neural Net Pattern Recognition App (3:36)

3.6 SYSTEM ARCHITECTURE:

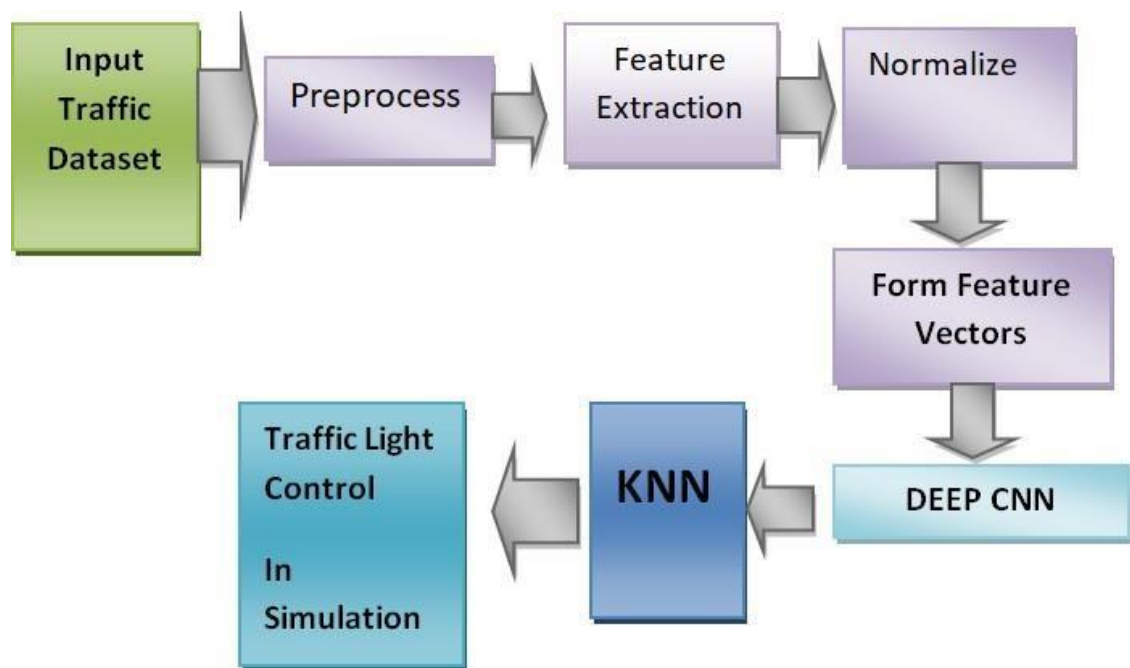


Fig 3.2 : SYSTEM ARCHITECTURE

3.6.1 Convolutional Neural Network:

A convolutional neural network consists of an input and an output layer, as well as multiple [hidden layers](#). The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a multiplication or other [dot product](#).

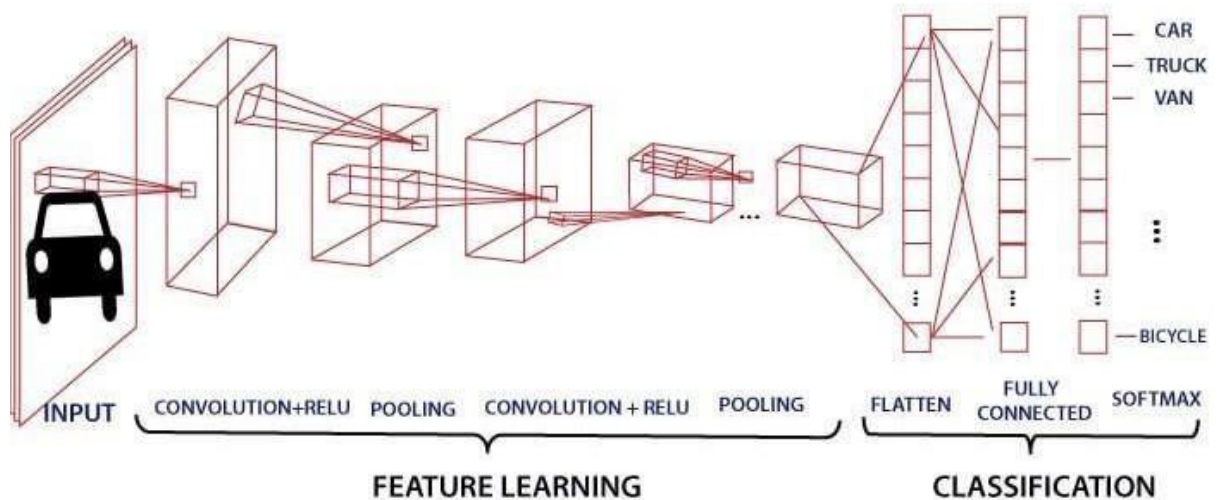


Fig 3.3 : Convolutional Neural Network

CNN help in running neural networks directly on images and are more efficient and accurate than many of the deep neural networks. ConvNet models are easy and faster to train on images comparatively to the other models.

One of the limitations of the CNN model is that they cannot be trained on a different dimension of images. So, it is mandatory to have same dimension images in the dataset.

We'll check the dimension of all the images of the dataset so that we can process the images into having similar dimensions. In this dataset, the images have a very dynamic range of dimensions from $16 \times 16 \times 3$ to $128 \times 128 \times 3$ hence cannot be passed directly to the ConvNet model.

We need to compress or interpolate the images to a single dimension. Not, to compress much of the data and not to stretch the image too much we need to decide the dimension which is in between and keep the image data mostly accurate. I've decided to use dimension $64 \times 64 \times 3$.

3.6.2 KNN:

In [pattern recognition](#), the **k-nearest neighbors algorithm (k-NN)** is a [non-parametric](#) method used for [classification](#) and [regression](#).^[1] In both cases, the input consists of the k closest training examples in the [feature space](#).

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.^[2]

The neighbors are taken from a set of objects for which the class (for k -NN classification) or the object property value (for k -NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

3.6.3 Feature extraction:

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying k-NN algorithm on the transformed data in feature space.

3.6.4 FEATURE SELECTION:

Feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons: • Simplification of models to make them easier to interpret by researchers/users. • Shorter training times. • To avoid the curse of dimensionality. • Enhanced generalization by reducing over fitting by reducing over fitting.

3.6.5 Typical Workflow for Designing Neural Networks:

Each neural network application is unique, but developing the network typically follows these steps:

1. Access and prepare your data
2. Create the neural network
3. Configure the network's inputs and outputs
4. Tune the network parameters (the weights and biases) to optimize performance
5. Train the network
6. Validate the network's results
7. Integrate the network into a production system

3.6.6 Classification and Clustering of Shallow Networks:

MATLAB and Deep Learning Toolbox provide command-line functions and apps for creating, training, and simulating shallow neural networks. The apps make it easy to develop neural networks for tasks such as classification, regression (including time-series regression), and clustering. After creating your networks in these tools, you can automatically generate MATLAB code to capture your work and automate tasks.

3.6.7 Preprocessing, Postprocessing, and Improving Your Network:

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Datagathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: -100), impossible data combinations missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. Often, data pre-processing is the most important phase of a machine learning project, especially in computations. The product of data pre-processing is the final training set.

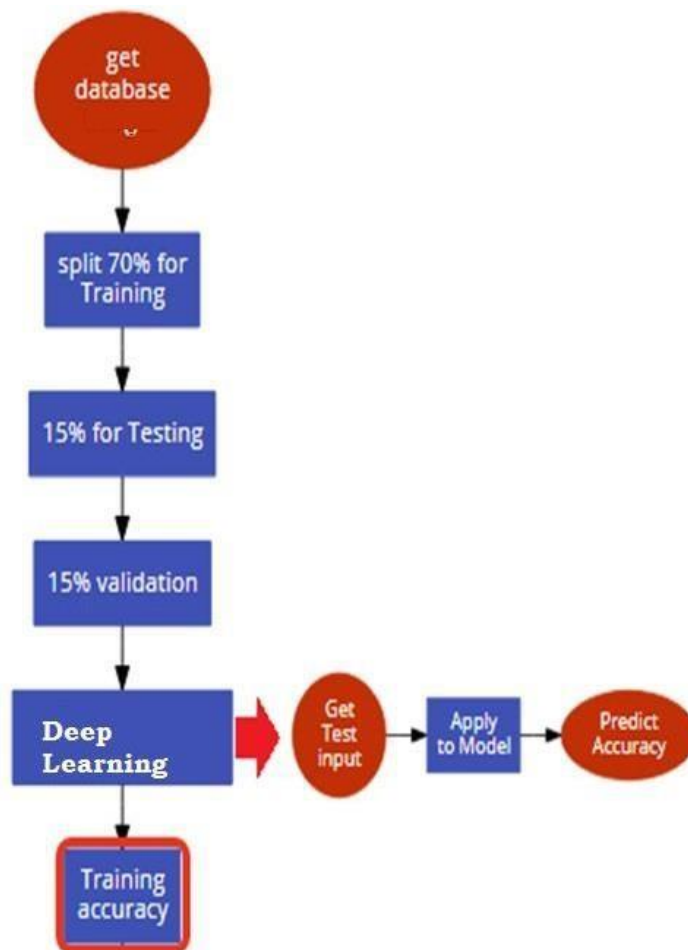
Preprocessing the network inputs and targets improves the efficiency of shallow neural network training. Postprocessing enables detailed analysis of network performance. MATLAB and Simulink® provide tools to help you:

- Reduce the dimensions of input vectors using principal component analysis
- Perform regression analysis between the network response and the corresponding targets
- Scale inputs and targets so they fall in the range [-1,1]
- Normalize the mean and standard deviation of the training data set
- Use automated data preprocessing and data division when creating your networks

Improving the network's ability to generalize helps prevent overfitting, a common problem in neural network design. Overfitting occurs when a network has memorized the training set but has not learned to generalize to new inputs. Overfitting produces a relatively small error on the training set but a much larger error when new data is presented to the network.

3.7 OVERALL ARCHITECTURE:

Our deep learning architecture is illustrated in above figure. The current state and the tentative actions are fed to the primary convolutional neural network to choose the most rewarding action. The current state and action along with the next state and received reward are stored into the memory as a fourtuple hs, a, r, s_0 . The data in the memory are selected by the prioritized experience replay to generate mini-batches and they are used to update the primary neural network's parameters. The target network θ^- is a separate neural network to increase stability during the learning. We use the double DQN and dueling DQN to reduce the possible overestimation and improve performance. Through this way, the approximating function can be trained and the Q value at every state to every action can be calculated. The optimal policy can then be obtained by choosing the action with the max Q value.



W

Fig 3.4 : OVERALL ARCHITECTURE

3.7.1 Two solutions to improve generalization include:

- Regularization modifies the network's performance function (the measure of error that the training process minimizes). By including the sizes of the weights and biases, regularization produces a network that performs well with the training data and exhibits smoother behavior when presented with new data.
- Early stopping uses two different data sets: the training set, to update the weights and biases, and the validation set, to stop training when the network begins to overfit the data.

Postprocessing plots for analyzing network performance, including mean squared error validation performance for successive training epochs (top left), error histogram (top right), and confusion matrices (bottom) for training, validation, and test phases.

3.7.2 Simulink Support:

Deep Learning Toolbox provides a set of blocks for building shallow neural networks in Simulink. All blocks are compatible with Simulink Coder™. These blocks are divided into four libraries:

- Transfer function blocks, which take a net input vector and generate a corresponding output vector
- Net input function blocks, which take any number of weighted input vectors, weight-layer output vectors, and bias vectors, and return a net input vector
- Weight function blocks, which apply a neuron's weight vector to an input vector (or a layer output vector) to get a weighted input value for a neuron
- Data preprocessing blocks, which map input and output data into the ranges best suited for the neural network to handle directly

Alternatively, you can create and train your networks in the MATLAB environment and automatically generate network simulation blocks for use

with Simulink. This approach also enables you to view your networks graphically.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 RESULTS:

To evaluate the performance of the proposed method, we compared it against a baseline traffic controller, a controller that gives an equal fixed time to each phase of the intersection. We ran SUMO-GUI simulator for the suggested model and compared the average total cumulative delay and average queue length achieved to the baseline. The performance comparison of the leaning agent regarding average cumulative delay time and average queue length metrics, respectively, to the baseline, while the agent is following the learning policy over time.

Our model's objective is to maximize the defined reward, which is to reduce the cumulative delay of all vehicles. Thus, we evaluate the performance of our model using the following two metrics: cumulative reward and average waiting time. The cumulative reward is measured by adding up the rewards of all cycles in every episode within one hour period. The average waiting time is measured by dividing the total waiting time by the number of vehicles in an episode.

4.2 DISCUSSION:

4.2.1 Write and run test scripts:

Write script-based tests to check that the outputs of MATLAB® scripts, functions, or classes are as you expect. For example, you can use the `assert` function to test for actual output values that match expected values. Or you can test that the output variables have the correct size and type. To run your test scripts use the `runtests` function.

4.2.2 Function-Based Unit Tests:

Write tests using qualifications; customize test runner; select and run tests

Write function-based tests to check that the outputs of MATLAB® scripts, functions, or classes are as you expect. You can use a full library of qualification functions to produce four different types of test failures.

4.2.3 Performance Testing Framework:

4.2.3.1 Measure performance of MATLAB® code:

You can use the MATLAB performance testing framework to measure the performance of your MATLAB code. The framework includes performance measurement-oriented features such as running your code several times to warm it up and accounting for noise in the measurements.

The performance test interface leverages the script, function, and class-based unit testing interfaces. Therefore, you can perform qualifications within your performance tests to ensure correct functional behavior while measuring code performance. Also, you can run your performance tests as standard regression tests to ensure that code changes do not break performance tests.

The performance test interface leverages the script, function, and class-based unit testing interfaces. You can perform qualifications within your performance tests to ensure correct functional behavior while measuring code performance. Also, you can run your performance tests as standard regression tests to ensure that code changes do not break performance tests.

4.2.3.2 Measured estimate of the framework overhead:

Class-based deriving from `matlab.perftest.TestCase` and using `startMeasuring` and `stopMeasuring` methods Code between calls to `startMeasuring` and `stopMeasuring` in each method tagged with the `Test` attribute.

4.3 Code Generation and Deployment:

Using Deep Learning Toolbox with MATLAB Coder™, GPU Coder™, and MATLAB Compiler™, you can deploy trained networks to embedded systems, or integrate them with a wide range of production environments. You can use MATLAB Coder to generate C and C++ code for your trained network, which enables you to simulate a trained network on PC hardware and then deploy the network to embedded systems. You can use MATLAB Compiler and

MATLAB Compiler SDK™ to deploy trained networks as C/C++ shared libraries, Microsoft®. NET assemblies, Java® classes, and Python®packages from MATLAB programs. You can also train a network model in the deployed application or a component.

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 SUMMARY:

In this work, we applied a deep reinforcement learning algorithm to traffic signal control problem in order to find optimal control policies of signaling, just by using raw visual input data of the traffic simulator snapshots. Our approach has led to promising results and showed it can find more stable control policies compared to previous work of using deep reinforcement learning in traffic light optimization. In our work, we propose to solve the traffic light control problem using a deep reinforcement learning model. The traffic information is gathered from vehicular networks or a camera. The states are two-dimension values with the vehicles' position and speed information. The actions are modeled as a Markov decision process and the rewards are the cumulative waiting time difference between two cycles.

5.2 CONCLUSION AND FUTURE WORK:

To handle the complex traffic scenario in our problem, we propose a double dueling deep Q network (3DQN) with prioritized experience replay. The model can learn a good policy under both the rush hours and normal traffic flow rates. It can reduce over 20% of the average waiting timing from the starting training. In our work, we developed and tested the proposed method in a small application, extending the work for more complex traffic simulations, for instance considering many intersections and multiple agents to control each intersection, using multi-agent learning techniques to handle coordination problem between agents would be a direction for future research. The proposed model also outperforms others in learning speed, which is shown in extensive simulation in SUMO and Tensorflow.

REFERENCES:

- [1] R. S. Sutton, A. G. Barto et al., Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.
- [2] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “Sumo – simulation of urban mobility: An overview,” in SIMUL 2011, S. . U. of Oslo Aida Omerovic, R. I. R. T. P. D. A. Simoni, and R. I. R. T. P. G. Bobashev, Eds. ThinkMind, October 2011. [Online]. Available: <https://elib.dlr.de/71460/>
- [3] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-gamestarcraft-ii/>, 2019.
- [4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke et al., “Qt-opt: Scalable deep reinforcement learning for visionbased robotic manipulation,” arXiv preprint arXiv: 1806.10293, 2018.
- [5] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, “A survey on reinforcement learning models and algorithms for traffic signal control,” ACM Computing Surveys (CSUR), vol. 50, no. 3, p. 34, 2017.
- [6] W. Genders and S. Razavi, “Evaluating reinforcement learning state representations for adaptive traffic signal control,” Procedia computer science, vol. 130, pp. 26–33, 2018.
- [7] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, “Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network,” arXiv preprint arXiv:1705.02755, 2017.

- [8] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," arXiv preprint arXiv:1611.01142, 2016.
- [9] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, 2017.
- [10] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.
- [11] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2496–2505.
- [12] D. Krajzewicz, G. Hertkorn, C. Rossel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.
- [13] L. A. Rodegerdts, B. Nevers, B. Robinson, J. Ringert, P. Koonce, J. Bansen, T. Nguyen, J. McGill, D. Stewart, J. Suggett et al., "Signalized intersections: informational guide," Tech. Rep., 2004.
- [14] R. Dowling, "Traffic analysis toolbox volume vi: Definition, interpretation, and calculation of traffic analysis tools measures of effectiveness, Tech. Rep., 2007.
- [15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992. [16] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.
- [16] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-difference learning with function approximation," in *Advances in neural information processing systems*, 1997, pp. 1075–1081.
- [17] L.-J. LIN, "Reinforcement learning for robots using neural networks," Ph.D. thesis, Carnegie Mellon University, 1993.

[18] P. Koonce and L. Rodegerdts, "Traffic signal timing manual." United States. Federal Highway Administration, Tech. Rep., 2008

APPENDIX:

a. SOURCE CODE:

```
clc;

clear all;

close all;

warning off;


% Read the Wind Dataset

[f p]=uigetfile('*.xls');

[ndata, text, alldata] = xlsread([p f]);

% [ndata, text, alldata] = xlsread('traffic1.xlsx');

% % % % [ndata, text, alldata] = xlsread('traffic.xls');


ndataX=ndata(1:6000);

TrafficDataSet=ndata(:,4);

TrafficDataSet=TrafficDataSet(1:1000);

%*****

% Input

xa=TrafficDataSet(1:100);

[r c]=size(xa);

xa=reshape(xa,[c r]);
```

```
yda=TrafficDataSet(101:200);
```

```
yea=TrafficDataSet(201:300);
```

```
ya=[yda yea];
```

```
[r1 c1]=size(ya);
```

```
ya=reshape(ya,[c1 r1]);
```

```
[ppa]=SelfOrg_Model(ya);
```

```
BusCount = alldata(:,2);
```

```
TwoWheeler = alldata(:,3);
```

```
CarCount = alldata(:,4);
```

```
OtherVehicles = alldata(:,5);
```

```
AllVeh=alldata(:,9);
```

```
CapFactor= AllVeh(1:400)
```

```
for i=1:400
```

```
    if i>1
```

```
        capF(i)=cell2mat(CapFactor(i))
```

```
    end
```

```
end
```

```
[predictedLabels]=callRCnn(capF);
```

```
Predicted_BusCount = BusCount(predictedLabels);
```

```
Predicted_TwoWheeler = TwoWheeler(predictedLabels);
```

```

Predicted_CarCount = CarCount(predictedLabels);

Predicted_OtherVehicles =OtherVehicles(predictedLabels);


Total_Vehicles_detected=(cell2mat(Predicted_BusCount)+cell2mat(Predicted
_TwoWheeler)+cell2mat(Predicted_CarCount)+cell2mat(Predicted_OtherVehi
cles));

pc=num2str(Total_Vehicles_detected);


msgbox(['Traffic Congestion =' pc ]);

if (Total_Vehicles_detected >50)

    msgbox('Congestion is TOO HIGH');

end


if ((Total_Vehicles_detected <50) && (Total_Vehicles_detected >30))

    msgbox('Congestion is Medium');

end

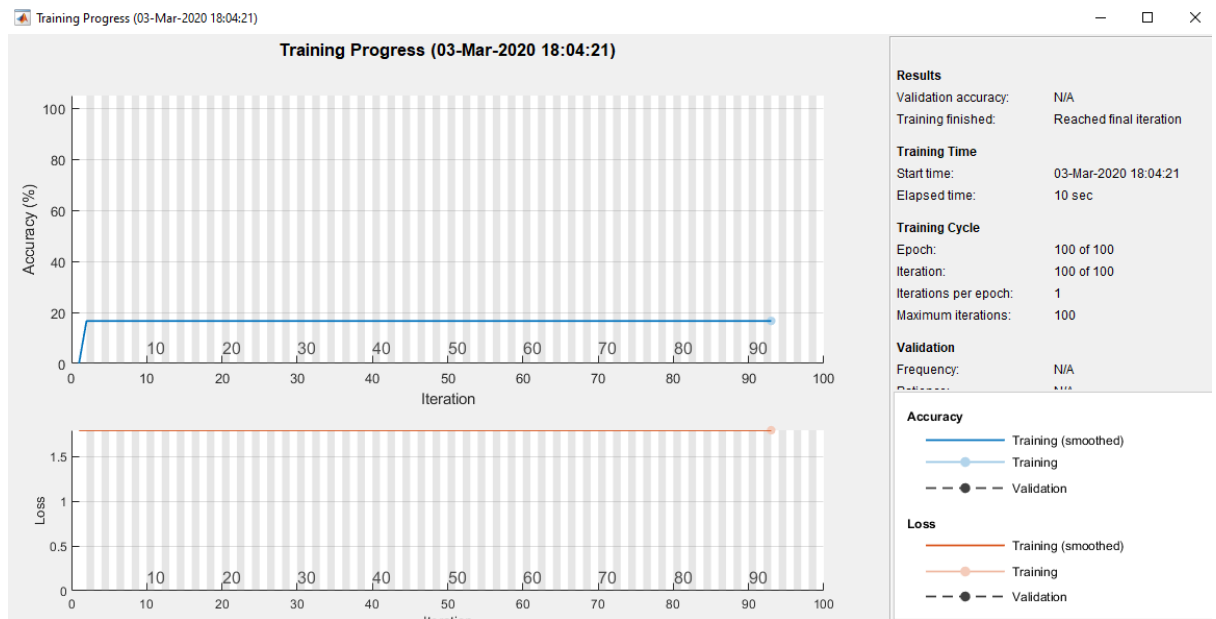

if ((Total_Vehicles_detected <29))

    msgbox('NORMAL - TRAFFIC');

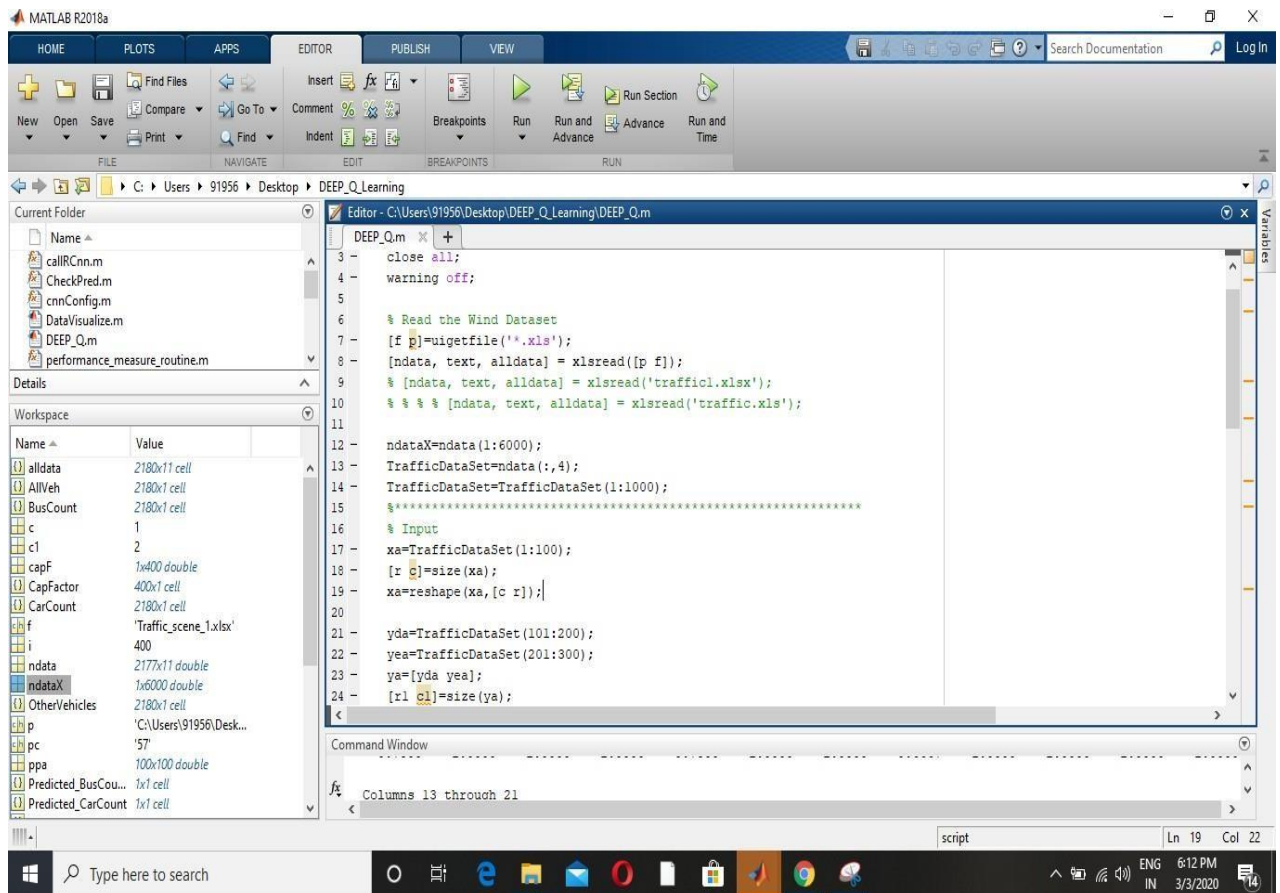
end

```

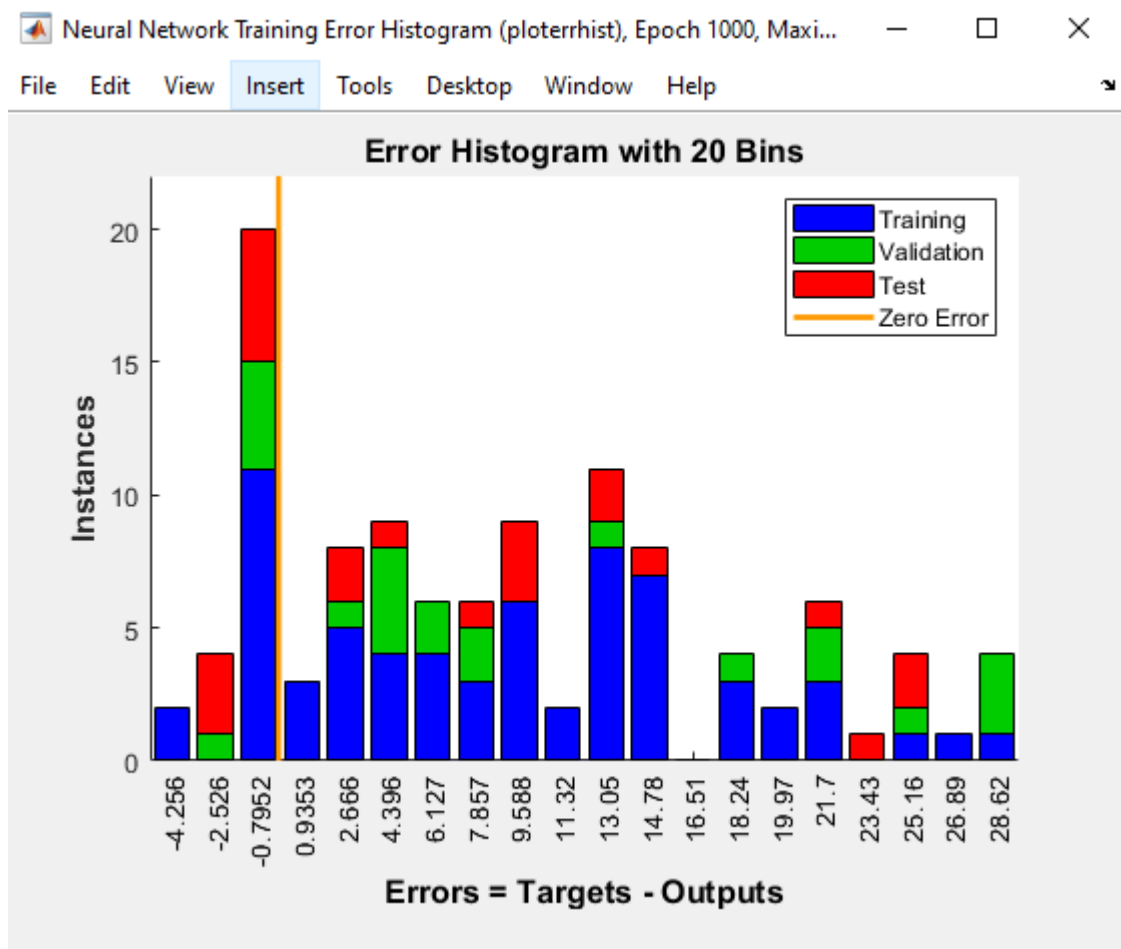
b. SCREENSHOTS:



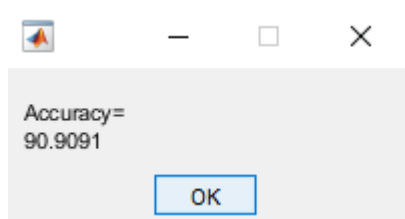
The above screenshot gives the brief idea about how our loss and accuracy varies within training of data.



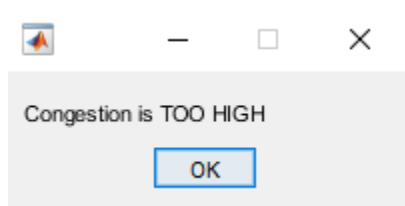
The above screenshot gives the information about our code and the platform which is used.



The above screenshot is a histogram which tells us relation of errors and instances.



We calculate the accuracy to evaluate the performance of our model.



With the help of our algorithm we simulate the traffic signal and we make a decision based on the simulation.