AI BASED INTELLIGENT TRAFFIC LIGHT CONTROL SYSTEM USING CNN

A PROJECT REPORT

Submitted by

MOULI SHANKAR R (211419105081)

SARAN B (211419105128)

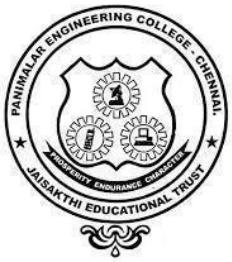
SUNDAR AKAASH V (211419105148)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"AI BASED INTELLIGENT TRAFFIC LIGHT CONTROL SYSTEM USING CNN"** is the bonafide work "**MOULI SHANKAR R [211419105081] SARAN B [211419105128] SUNDAR AKAASH [2114191050148]"** who carried out the project work under my supervision.

SIGNATURE

Dr. S. SELVI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF EEE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLE,
CHENNAI - 600123

SIGNATURE

Mr. P. HARIRAMAKRISHNAN M.E, Ph.D.
ASSISTANT PROFESSOR
DEPARTMENT OF EEE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLE,
CHENNAI - 600123

Submitted for End Semester Project Viva Voice held on ..... at

Panimalar Engineering College, Chennai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Our sincere thanks to our Honourable Founder and Chairman, **Dr. JEPPIAAR, M.A., B.L., Ph.D.,** for his sincere endeavour in educating us in his premier institution.

We would like to express our deep gratitude to our beloved **Secretary and Correspondent, Dr.P. CHINNADURAI, M.A., M.Phil. Ph.D.** for his enthusiastic motivation which inspired us a lot in completing this project and our sincere thanks to our directors **Mrs.C. VIJAYA RAJESWARI, Dr.C. SAKTHI KUMAR, M.E., Ph. D** and **Dr. SARANYASREE SAKTHIKUMAR, B.E, M.B.A, Ph. D** for providing us with the necessary facilities for the completion of this project.

We would like to express thanks to our Principal, **Dr. K. Mani M.E., Ph.D.,** for having extended his guidance and cooperation.

We would also like to thank our **Head of the Department**, **Dr.S. Selvi, M.E., Ph.D., Professor and Head, Department of Electrical and Electronics Engineering** for her encouragement.

Personally, we thank our Guide **Mr. P. HARIRAMAKRISHNAN M.E, Ph.D. in Department of Electrical and Electronics engineering** for the persistent motivation and supportfor this project, who at all times was the mentor of germination of the project from a small idea.

We express our sincere thanks to the project coordinators **Dr.S. Deepa & Dr.N. MANOJ KUMAR, M.E., Ph.D., in Department of Electrical and Electronics Engineering** for the Valuable suggestions from time to time at every stage of our project.

Finally, we would like to take this opportunity to thank our family members, faculty and non-teaching staff members of our department, friends, well-wishers who have helped us for the successful completion of our project.

ABSTRACT:

Road traffic control has been around for a long time to guarantee the safety of vehicles and pedestrians. However, emergency vehicle of Ambulances are stuck in the Traffic and saving patients who is in the ambulance are very challenging in the heavy traffic. There is no system that detects and classifies Emergency vehicle from the road traffic in real time. To solve this issue, the following study proposes the training of a machine learning model for detection and classification of Emergency vehicle on the road using the road images. The model of Convolutional Neural network (CNN) is trained to detect and classify the Emergency vehicle. From the real world road image datasets. Our proposed CNN model based on the Xception architecture using ADAM optimizer is better for classification. The proposed method is superior to the existing literature, indicating that it can be used to quickly and accurately detect Ambulance in the roads.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVATIONS

CNN	CONVOLUTIONAL NEURAL NETWORK
VGG	VISUAL GEOMETRY GROUP
PYPI	PYTHON PACKAGE INDEX
PYGUI	PYTHON GRAPHICAL USER INTERFACE
VENV	PYTHON VIRTUAL ENVIRONMENT
LCD	LIQUID CRYSTAL DISPLAY
LED	LIGHT EMITTING DIOD
PWM	PULSE WIDTH MODULATION
IOT	INTERNET OF THINGS
SOC	SYSTEMON CHIP

CHAPTER 1

INTRODUCTION:

Machine Learning (ML) is one of the most important and popular emerging branches these days as it are a part of Artificial Intelligence (AI). In recent times, machine learning becomes an essential and upcoming research area for transportation engineering, especially in traffic prediction. Traffic congestion affects the country's economy directly or indirectly by its means. Traffic congestion also takes people's valuable time, cost of fuel every single day. As traffic congestion is a major problem for all classes in society, there has to be a small-scale traffic prediction for the people's sake of living their lives without frustration or tension. For ensuring the country's economic growth, the road user's ease is required in the first place. And in emergency vehicle role traffic causes the main difficulties in saving life due to traffic condition. This is possible only when the traffic flow is smooth. To deal with this, Emergency Vehicle and Traffic prediction is needed so that we can estimate or predict the future traffic to some extent. In addition to the country's economy. The government is also investing in the intelligent transportation system (ITS) to solve these issues. The plot of this research paper is to find different machine learning algorithms and speculating the models by utilizing python3. The goal of traffic flow prediction is to predict the Ambulance and traffic to the users as soon as possible. So, this research can be helpful to predict traffic. Machine learning is usually done using anaconda software but in this paper, I have used the python program using command prompt window which is much easier than the usual way of detecting the traffic and also the emergency vehicle of Ambulance.

CHAPTER-2

LITERATURE SURVEY

PAPER 1:TRAFFIC FLOW PREDICTION FOR SMART TRAFFIC LIGHTS USING MACHINE LEARNING ALGORITHMS

AUTHORS: ALFONSO NAVARRO-ESPINOZA, OSCAR ROBERTO LOPEZ-BONILLA .

YEAR: 2022

ABSTRACT:

Nowadays, many cities have problems with traffic congestion at certain peak hours, which produces more pollution, noise and stress for citizens. Neural networks (NN) and machine-learning (ML) approaches are increasingly used to solve real-world problems, overcoming analytical and statistical methods, due to their ability to deal with dynamic behavior over time and with a large number of parameters in massive data. In this paper, machine-learning (ML) and deep-learning (DL) algorithms are proposed for predicting traffic flow at an intersection, thus laying the groundwork for adaptive traffic control, either by remote control of traffic lights or by applying an algorithm that adjusts the timing according to the predicted flow. Therefore, this work only focuses on traffic flow prediction. Two public datasets are used to train, validate and test the proposed ML and DL models. The first one contains the number of vehicles sampled every five minutes at six intersections for 56 days using different sensors. For this research, four of the six intersections are used to train the ML and DL models. The Multilayer Perceptron Neural Network (MLP-NN) obtained better results (R-Squared and EV score of 0.93) and took less training time, followed closely by Gradient Boosting then Recurrent Neural Networks (RNNs), with good metrics results but the longer training time, and finally

Random Forest, Linear Regression and Stochastic Gradient. All ML and DL algorithms scored good performance metrics, indicating that they are feasible for implementation on smart traffic light controller.

PAPER 2: AN EDGE TRAFFIC FLOW DETECTION SCHEME BASED ON DEEP LEARNING IN AN INTELLIGENT TRANSPORTATION SYSTEM

AUTHORS:CHEN CHEN, BIN LIU SHAOHUA WAN, PENG QIAO, QINGQI P .

YEAR: 2020

ABSTRACT:

An intelligent transportation system (ITS) plays an important role in public transport management, security and other issues. Traffic flow detection is an important part of the ITS. Based on the real-time acquisition of urban road traffic flow information, anITS provides intelligent guidance for relieving traffic jams and reducing environmental pollution. The traffic flow detection in an usually adopts the cloud computing mode. The edge of the network will transmit all the captured video to the cloud computing center. However, the increasing traffic monitoring has brought great challenges to the storage, communication and processing of traditional transportation systems based on cloud computing. To address this issue, a traffic flow detection scheme based on deep learning on the edge node is proposed in this article. First, we propose a vehicle detection algorithm based on the YOLOv3 (You Only Look Once) model trained with a great volume of traffic data. We pruned the model to ensure its efficiency on the edge equipment. After that, the DeepSORT (Deep Simple Online and Real-time Tracking) algorithm is optimized by retraining the feature extractor for multi object vehicle tracking. Then, we propose a real-time

vehicle tracking counter for vehicles that combines the vehicle detection and vehicle tracking algorithms to realize the detection of traffic flow. Finally, the vehicle detection network and multiple-object tracking network are migrated and deployed on the edge device Jetson TX2 platform, and we verify the correctness and efficiency of our framework. The test results indicate that our model can efficiently detect the traffic flow with an average processing speed of 37.9 FPS (frames per second) and an average accuracy of 92.0% on the edge device.

PAPER 3: INTELLIGENT OBJECT DETECTION IN AUTONOMOUS VEHICLES USING CNN ALGORITHM

AUTHORS: R.DEEPIKA, N.LILLY, P DIVIJA.

YEAR: 2022

ABSTRACT:

In recent years, the growth of usage of automobile vehicles in the transport system is increasing. In near future autonomous vehicles will play vital role in modern intelligent transport systems. The success of intelligent transport is depending on the object detection accuracy. Autonomous vehicles (AV) detect the object based on the information or data collected through the sensors. These AV are designed to overcome the challenges of accident, security using advanced driving assistant system (ADAS). The ADAS uses sensors to perceive the surrounding environment. To implement of smart vehicle transport system, several machine learning algorithms have been developed to detect object accurately with a faster detection rate. But still there are several bottlenecks in the detection accuracy and detection rate. The efficiency of the detection algorithm is based speed of detection objects. Hence we proposed a novel hybrid deep learning algorithms using CNN for object detection which provides

higher efficiency and performance compared to the other existing machine learning algorithms. Our experiment results show that the proposed algorithm efficiently identifies static and dynamic objects using CNN with the highest accuracy rate.

PAPER 4: A PERFORMANCE MODELING AND ANALYSIS OF A NOVEL VEHICULAR TRAFFIC FLOW PREDICTION SYSTEM USING A HYBRID MACHINE LEARNING-BASED MODEL

AUTHORS: AZZEDINE BOUKERCHE, JIAHAO WANG .

YEAR:2020

ABSTRACT:

Traffic prediction on the road, as a vital part of the Intelligent Transportation System (ITS) has attracted much attention recently. It is always one of the hot topics about how to implement an efficient, robust, and accurate vehicular traffic prediction system. With the help of Machine Learning-based (ML) methods, especially Deep Learning-based (DL) methods, the accuracy of the prediction model is increased. However, we also noticed that there are still many open challenges under ML-based vehicular traffic prediction model real-world implementation. Firstly, the time consumption for training DL model is relatively large when compared to parametric models, such as ARIMA, SARIMA. Second, it is still a hot topic for road traffic prediction that how to capture the spacial relationship between road detectors, which is affected by the geographic correlation, as well as the time change. The last but not the least, it is important for us to implement the prediction system into the real world; meanwhile, we should find a way to make use of the advanced technology applied in ITS to improve the prediction system itself. In this paper, we focus on improving the features of the prediction model, which can be helpful for

implementing the model in the real world. We present a new hybrid deep learning model by using Graph Convolutional Network (GCN) and the deep aggregation structure (i.e., the sequence to sequence structure) of Gated Recurrent Unit (GRU). Meanwhile, in order to solve the real-world prediction problem, i.e., the online prediction task, we present a new online prediction strategy by using refinement learning. In order to further improve the model's accuracy and efficiency when applied to ITS, we make use of an efficient parallel training strategy while taking advantage of the vehicular cloud structure.

PAPER 5: A SURVEY OF TRAFFIC PREDICTION: FROM SPATIO - TEMPORAL DATA TO INTELLIGENT TRANSPORTATION

AUTHORS: HAITAO YUAN & GUOLIANG LI

YEAR: 2021

ABSTRACT:

Intelligent transportation (e.g., intelligent traffic light) makes our travel more convenient and efficient. With the development of mobile Internet and position technologies, it is reasonable to collect spatio-temporal data and then leverage these data to achieve the goal of intelligent transportation, and here, traffic prediction plays an important role. In this paper, we provide a comprehensive survey on traffic prediction, which is from the spatio-temporal data layer to the intelligent transportation application layer. At first, we split the whole research scope into four parts from bottom to up, where the four parts are, respectively, spatio-temporal data, preprocessing, traffic prediction and traffic application. Later, we review existing work on the four parts. First, we summarize traffic data into five types according to their difference on spatial and temporal

dimensions. Second, we focus on four significant data preprocessing techniques: map-matching, data cleaning, data storage and data compression. Third, we focus on three kinds of traffic prediction problems (i.e., classification, generation and estimation/forecasting). In particular, we summarize the challenges and discuss how existing methods address these challenges. Fourth, we list five typical traffic applications. Lastly, we provide emerging research challenges and opportunities. We believe that the survey can help the practitioners to understand existing traffic prediction problems and methods, which can further encourage them to solve their intelligent transportation applications.

CHAPTER-3

METHODOLOGY

3.1 EXISITNG SYSTEM:

The goal of automated surveillance and monitoring systems is to remove the need of human labor for simple vision based tasks that can be performed by a computer or an automated system. The applications of computer vision systems have also been applied in various public areas such as roads, airports and retail areas. One such application of vision systems is in the task of monitoring and analyzing scenes of road traffic, with particular interest in monitoring highways and intersection. Such a system is required for effective real time traffic management systems that can detect changes in traffic characteristics in a timely manner allowing regulators and authorities the ability to quickly respond to traffic situations.

3.2 DISADVANTAGES:

- Need Large Manpower to monitor traffic and make the way for ambulance .
- Challenging to save the Patients in ambulance .
- Wastage of time .

3.3 PROPOSED SYSTEM:

This method represents a deep CNN was used to classify images of road vehicles into 4 classes like Low, Medium, High and No Traffic. They show that using deep CNNs for Traffic type classification achieves state of the art results. Currently, CNN is the best and most powerful image processing technique with great learning ability largely due to the use of several feature extraction phases (hidden layers) that can automatically learn representations from the data. the increase in a massive amount of dataset has accelerated the research in CNNs, and in recent times a very inspiring deep CNN architectures are stated. Convolutional Neural Network (CNN) model learning the mapping of the input images to their labeled classes and show good generalization to the test dataset. Experimental results on real-world datasets have shown that the proposed CNN method is effective for Traffic detection, and can perform the task with an High accuracy on four target traffic classes.

3.4 ADVANTAGES:

- Emergency vehicle will easily reach hospital without stuck in the traffic
- Low Manpower
- High Accuracy level

3.5 SYSTEM CONFIGURATION:

3.5.1 H/W SYSTEM CONFIGURATION:

- Processor - i3, i5, i10, AMD Processor
- RAM - 4 Gb
- Hard Disk - 1 TB

3.5.2 S/W SYSTEM CONFIGURATION:

- Operating System - Windows 8/10
- Language - Python(3.10) Version.
- Server - GUI

3.6 BLOCK DIAGRAM:

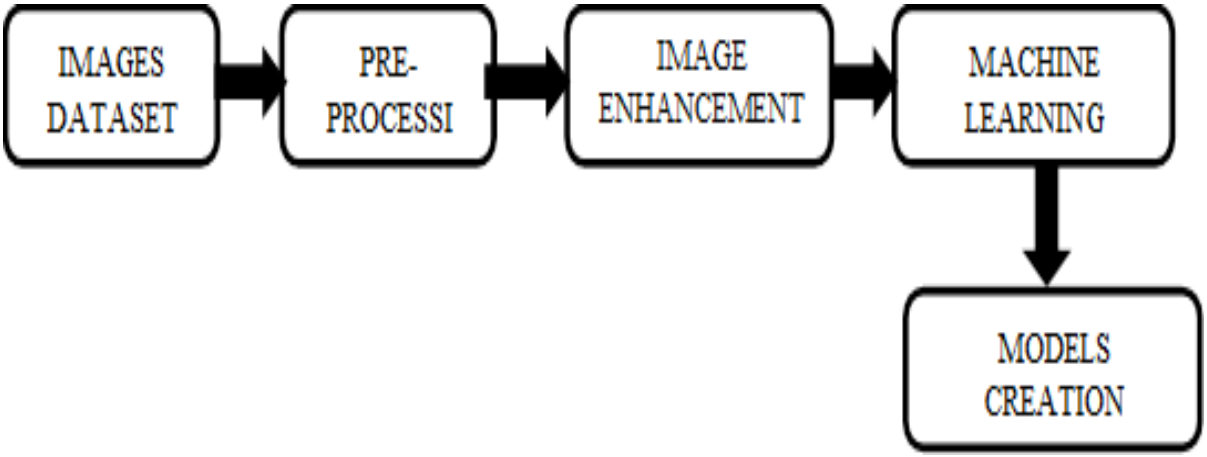


Fig 1 Model creation

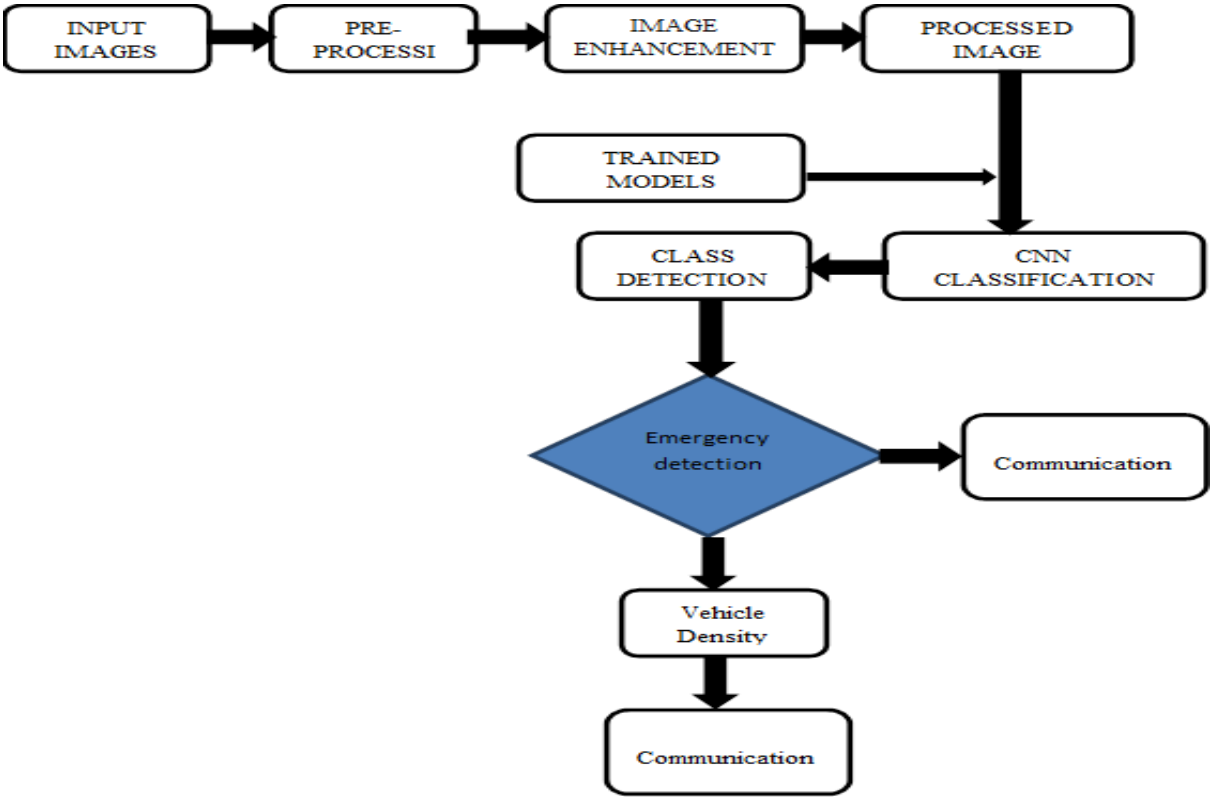


Fig 2. Emergency detection

CHAPTER-4

MODULE DISCRIPTION

4.1 MODULES LIST:

- Image Acquisition
- Image Pre-Processing
- Image Segmentation
- Feature Extraction
- Emergency Vehicle & Traffic Classification
- Convolutional Neural Network

4.2 IMAGE ACQUISITION:

The Real time Road Images of dataset acquisition has been used to implement the proposed methods. This method is used to design for extraction of Traffic with accuracy and composed number of stages is including image capturing, edge detection, and classification of Traffic Detection and Emergency Vehicle Detection.

4.3 IMAGE PRE-PROCESSING:

In this module, we are performing some basic operation on image to get proper image for processing. In this module, we are perform certain operation like gray-scale conversion, filtering, sharpening, smoothing, edging, and image segmentation to get proper and clean image. Preprocessing step enhances the

quality of the images by eliminating noise. The Gray scale images, kind of black-and-white or gray monochrome images, are composed exclusively of shades of gray. Gray scale images can be measuring the intensity of light at each pixel. The Filtering operation is performed on the image to increase the smoothness, sharpness as well as edge enhancement. In sharpening filter is used to enhancement the images in sharpening and to enhance detail that has been blurred. Smoothing filter is used to reduce the noise. It has used many different algorithms. Edging is a technique of finding and identifying sharpness presented in an image.

4.4 IMAGE SEGMENTATION:

Image Segmentation is an important step in domain of computer vision based on emerging applications including imaging, video surveillance and many more. The image segmentation is a step of processing which is used threshold method to segment the Road image level to binary image. Segmentation means partitioning the digital images into multiple parts of segments or objects. Segmentation is a process of grouping the pixels that have similar attributes. Is used to locate the objects and boundaries in images. Basically, the segmentation process performed to extract important features from the image for further analysis.

4.5 FEATURE EXTRACTION:

In this module, we are performing some more operation on segmented image. In this module we will perform feature extraction operation to get all detailed information about Road Traffic image. Feature Extraction and reduction has been playing a vital role for Emergency Vehicle & Traffic Detection into their relevant categories in the field of computer vision and machine learning. The

major issue behind feature extraction is to compute the most active or robust features for classification, which produced an efficient performance. The Feature extraction is used related to dimensionality reduction.

4.6 EMERGENCY VEHICLE & TRAFFIC CLASSIFICATION:

In this module, we are performing classification techniques with help of deep learning algorithm to determine Road Traffic condition and Detect Ambulance. The Road Traffic classification is the final step of the proposed approach that is used to identify the Emergency vehicle detection and the type of Traffic like Low Traffic, Medium Traffic and High Traffic. After features are extracted and selected the classification step using CNN is performed on the resulted feature vector. Classification is performed by using training phase and testing phase of CNN structure.

4.7 CONVOLUTIONAL NEURAL NETWORK:

The name of ‒Convolutional Neural Network‖ performs the mathematical operation called convolution. Convolution is a specialized kind of linear operation. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, Convolutional networks are simply neural networks that use convolution in matrix multiplication in at least one of their layers. ConvNets have been successful in Identifying faces, objects, and Traffic detection. A convolutional neural network consists of an input layer, output layer, as well as multiple hidden layers. CNN which is feed forward neural network and is widely used for image recognition and classification. The Convolutional neural layers convolve the input and pass its result output to the next layer. CNNs are regularized versions of multilayer perceptron’s. The

Multilayer perceptron’s are the fully connected networks each one neuron is connected to all other neurons in next layer. The "fully-connected" network means over fitting data.

4.8 PROPOSED SYSTEM ALGORITHM:

4.8.1 CNN ALGORITHM :

Image classification involves the extraction of features from the image to observe some patterns in the dataset. Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.

For example, if we have a 50 X 50 image of a cat, and we want to train our traditional ANN on that image to classify it into a dog or a cat the trainable parameters become –

(50*50) * 100 image pixels multiplied by hidden layer + 100 bias + 2 * 100 output neurons + 2 bias = 2,50,30

4.8.2 EXAMPLES:

Filters help us exploit the spatial locality of a particular image by enforcing a local connectivity pattern between neurons.

Convolution basically means a pointwise multiplication of two functions to produce

a third function. Here one function is our image pixels matrix and another is our filter. We slide the filter over the image and get the dot product of the two matrices. The resulting matrix is called an ‒Activation Map‖ or ‒Feature Map‖.

STEP 1:

Choose a dataset of your interest or you can also create your own image dataset for solving your own image classification problem. An easy place to choose a dataset is on kaggle.com.

The dataset I'm going with can be found here.

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil.

Here are all the libraries that we would require and the code for importing them.

STEP 2:

Preparing our dataset for training will involve assigning paths and creating categories(labels), resizing our images.

Resizing images into 200 X 200

STEP 3:

Training is an array that will contain image pixel values and the index at which the image in the CATEGORIES list.

STEP 4:

Shuffle the Dataset .

STEP 5:

Assigning Labels and Features .

STEP 6:

Normalising x and converting labels to categorical data .

STEP 7:

Split X and Y for use in CNN

STEP 8:

Define, compile and train the CNN Model

STEP 9:

Accuracy and Score of model

function XCOMPRESSCU(*pCurCU)

M <= FastCUMope(PO,QP)

if M 4 SPLIT then

C2n <=CHECKINTRA(pCurCU)

else

C2n <= ∞

end if

if M != HOMO and Dcur<Dmax then

Cn<= 0

for i = 0 to 3 do

pSubCUi<= pointer to SubCU

CN <= CN+ XCompressCU(pSubCUi)

end for

else

CN <= ∞

end if

CHECKBESTMODE(C2N, CN)

end function

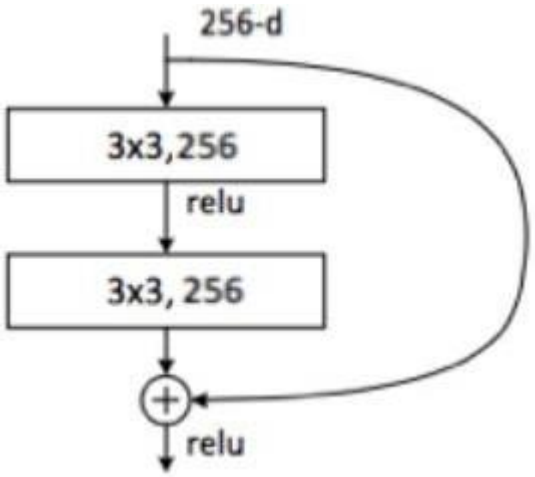
4.9 RESIDUAL NETWORKS (RESNET) IN KERAS:

Very deep neural networks are hard to train as they are more prone to vanishing or exploding gradients. To solve this problem, the activation unit from a layer could be fed directly to a deeper layer of the network, which is termed as a skip connection.

This forms the basis of residual networks or ResNets. This post will introduce the basics the residual networks before implementing one in Keras.

4.9.1 RESIDUAL BLOCK :

A building block of a ResNet is called a residual block or identity block. A residual block is simply when the activation of a layer is fast-forwarded to a deeper layer in the neural network.



As you can see in the image above, the activation from a previous layer is being added to the activation of a deeper layer in the network.

In theory, the training error should monotonically decrease as more layers are added to a neural network. In practice however, for a traditional neural network, it will reach a point where the training error will start increasing.

ResNets do not suffer from this problem. The training error will keep decreasing as more layers are added to the network. In fact, ResNets have made it possible to train networks with more than 100 layers, even reaching 1000 layers.

Now, let's build a ResNet with 50 layers for image classification using Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

In this case, we will use TensorFlow as the backend. Of course, feel free to grab the entire notebook and make all the necessary imports before starting.

Step 1: Define the identity block

Step 2: Convolution block

Step 3: Build the model

Step 4: Training

Step 5: Print the model summary

4.9.2 PSEUDO CODE :

Input: The raw 1D sensor signal (S) with size of 5625

Output: Graylevel image (Im) with size of 125 x 45

1: count = 1;

2: for i=1 to 125 do

3: for j=1 to 45 do

4: Im(i,j) = S(count);

5: count = count + 1;

6: end for j

7: end for i

8: Normalize Im by using min-max normalization.

4.10 VGG16 IMPLEMENTATION IN KERAS :

VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.

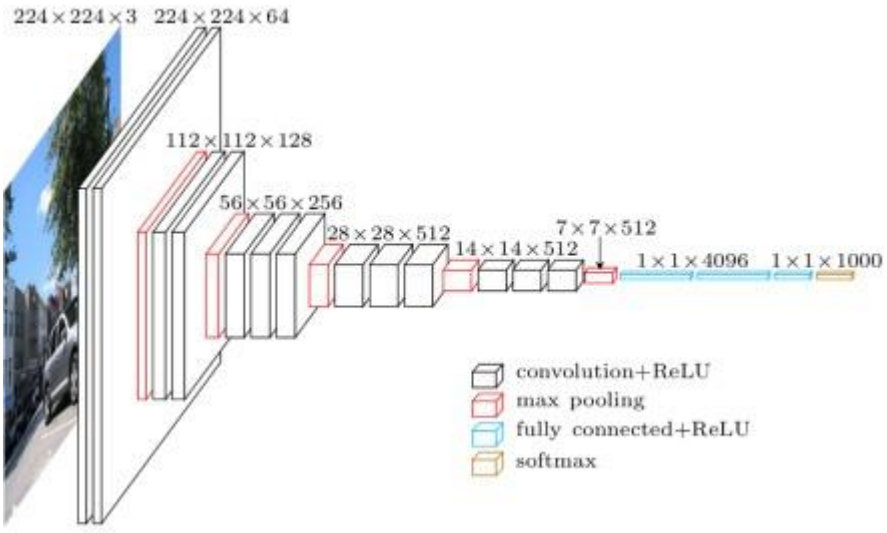


Fig 3. VGG16 Architecture

Here I first import all the libraries which i will need to implement VGG16. I will be using Sequential method as I am creating a sequential model. Sequential model means that all the layers of the model will be arranged in sequence. Here I have imported Image Data Generator from keraspreprocessing.

The objective of Image Data Generator is to import data with labels easily into the model. It is a very useful class as it has many function to rescale, rotate, zoom, flip etc. The most useful thing about this class is that it doesn’t affect the data stored on the disk. This class alters the data on the go while passing it to the model.

```
function CONV(i, [Bufz)
Co <= 0
while Co <Odo
if Co, = 0 and i = 0 then
        c<=0
READBIAS()
READKERNEL(Co ~ Co + 31, KC)
        end if
        || PPMACConv(K., [Bufr)
        || PREFETCHKERNEL(C, + 32 ~ C, + 63, K.)
        C <= C
end while
end function

function CONV(TX, Ty, Ci)
TX <= 0, Ty <= 0,Ci<= 0
```

```
while Ty < Y do

whileTx< X do

whileCi< I do

ifTx =0 and Ty = 0 then

 c<=0

READTILE(I  Buf., Tx, Ty, Ci)

end if

|| Convop(Ci, [Buf.)

|| PREFETCHTILE(IBufc., Tx, Ty, Ci + 1

C <= c

end while

end while

        end while

end function
```

4.11MICE IMPUTATION :

The fancyimpute package offers various robust machine learning models for imputing missing values. You can explore the complete list of imputers from the detailed documentation. Here, we will use IterativeImputer or popularly called MICE for imputing missing values.

The IterativeImputer performs multiple regressions on random samples of the data and aggregates for imputing the missing values. You will use the diabetes DataFrame for performing this imputation.

- Import IterativeImputer from fancyimpute.
- Copy diabetes to diabetes_mice_imputed.
- Create an IterativeImputer() object and assign it to mice_imputer.
- Impute the diabetes DataFrame.

4.12 UML DIAGRAM:

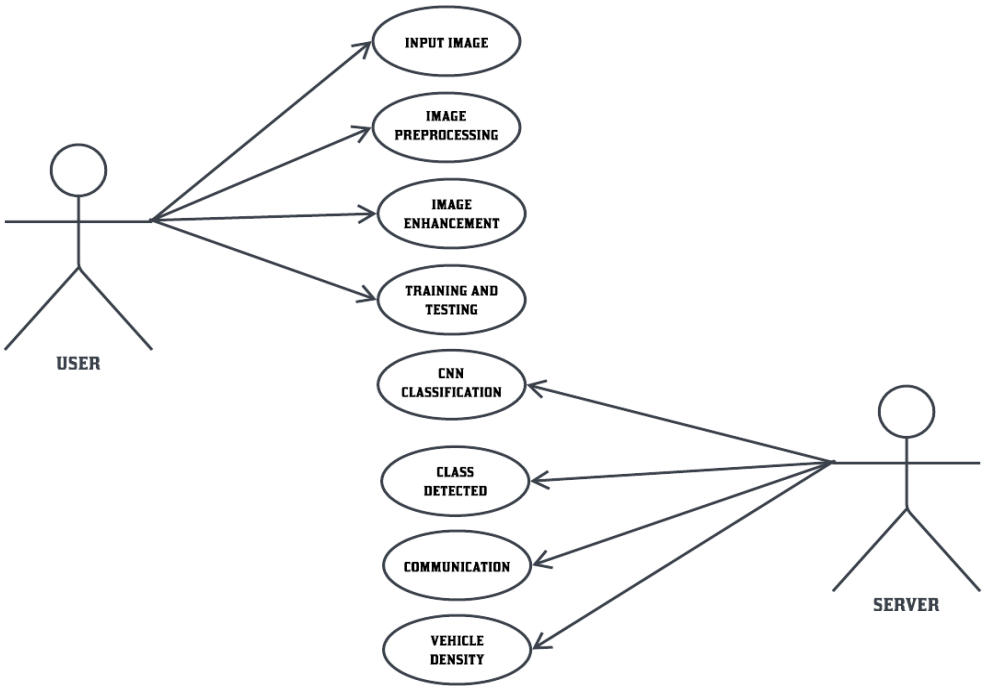


Fig 4. UML Diagram

4.13 ACTIVITY DIAGRAM:

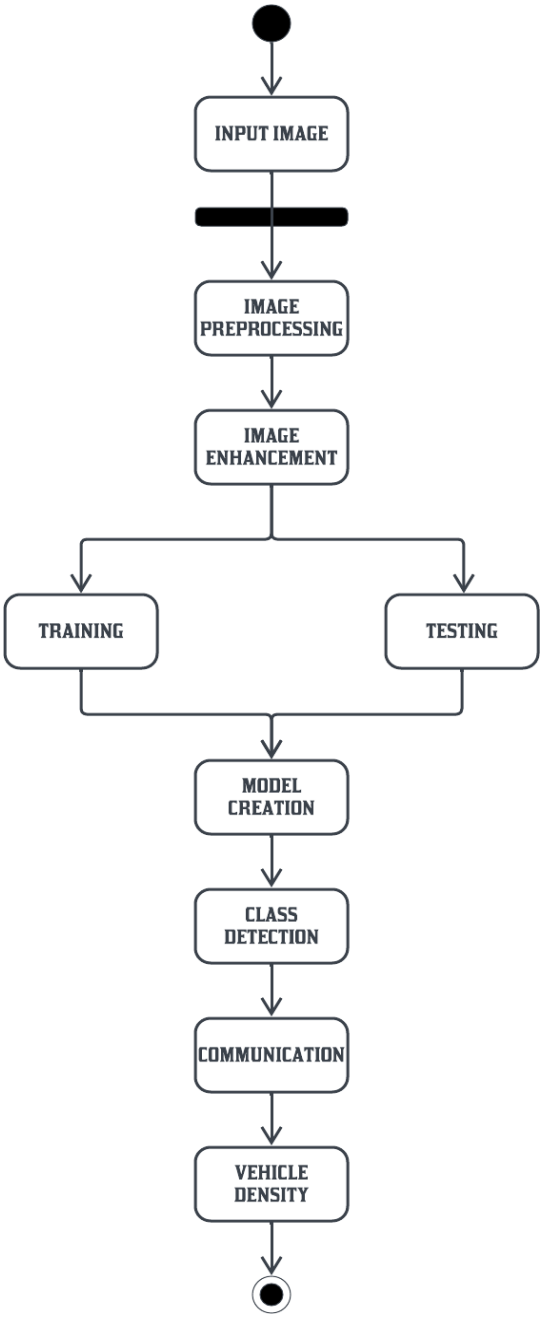


Fig 5 Activity Diagram

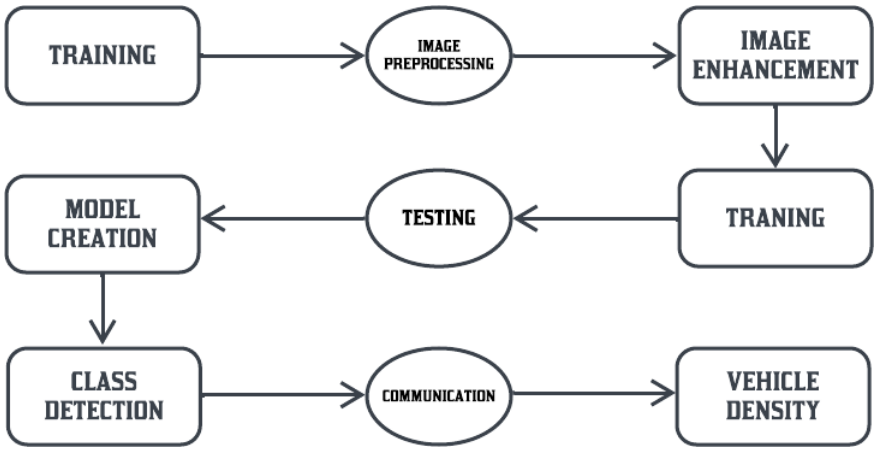
4.14 DATA FLOW DIAGRAM:

- **LEVEL 0:**

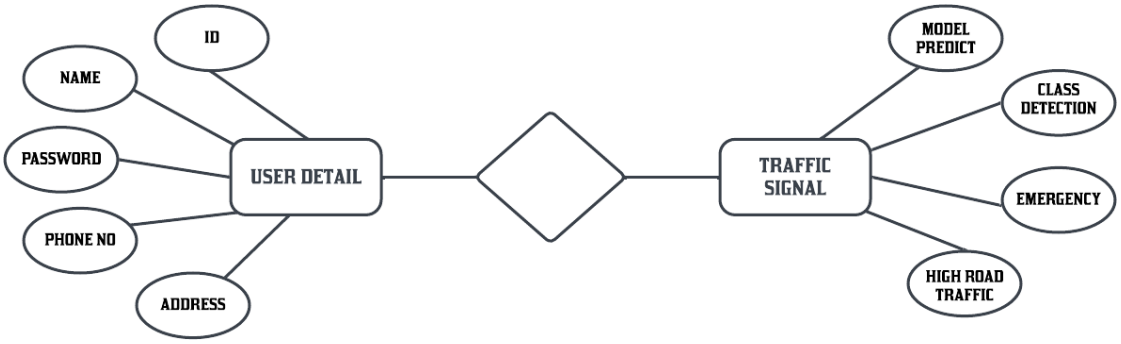
- **LEVEL 1:**

- **LEVEL 2:**

4.15 OVERALL DIAGRAM:



4.15.1 ER-DIAGRAM:



CHAPTER 5

SOFTWARE REQUIREMENTS

5.1 PYTHON :

Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

- Web Development
- Software Development
- Game Development
- AI & ML
- Data Analytics

Every Programming language serves some purpose or use-case according to a domain. for eg, Javascript is the most popular language amongst web developers as it gives the developer the power to handle applications via different frameworks like react, vue, angular which are used to build beautiful User Interfaces. Similarly, they have pros and cons at the same time. so if we consider python it is general-purpose which means it is widely used in every domain the reason is it’s very simple to understand, scalable because of which the speed of development is so fast. Now you get the idea why besides learning python it doesn’t require any programming background so that’s why it’s popular amongst developers as well. Python has simpler syntax similar to the English language and also the syntax allows developers to write programs with

fewer lines of code. Since it is open-source there are many libraries available that make developers' jobs easy ultimately results in high productivity. They can easily focus on business logic and Its demanding skills in the digital era where information is available in large data sets.

Now in the era of the digital world, there is a lot of information available on the internet that might confuse us believe me.what we can do is follow the documentation which is a good start point. Once we are familiar with concepts or terminology we can dive deeper into this.I hope now you guys are excited to get started right so you might be wondering where we can start coding right so there are a lot of options available in markets. we can use any IDE we are comfortable with but for those who are new to the programming world I am listing some of IDE's below for python:

1) Visual Studio: https://visualstudio.microsoft.com/

2) PyCharm: https://www.jetbrains.com/pycharm/

3) Spyder: https://www.spyder-ide.org/

4) Atom: https://atom.io/

5) Google Colab: https://research.google.com/colaboratory/

5.2EXAMPLES:

NASA (National Aeronautics and Space Agency): One of Nasa's Shuttle Support Contractors, United Space Alliance developed a Workflow Automation

System (WAS) which is fast. Internal Resources Within critical project stated that:

–Python allows us to tackle the complexity of programs like the WAS without getting bogged down in the language‖.

Nasa also published a website (https://code.nasa.gov/) where there are 400 open source projects which use python.

NETFLIX: There are various projects in Netflix which use python as follow:

- Central Alert Gateway
- Chaos Gorilla
- Security Monkey
- Chronos

Amongst all projects, Regional failover is the project they have as the system decreases outage time from 45 minutes to 7 minutes with no additional cost.

INSTAGRAM: Instagram also uses python extensively. They have built a photo-sharing social platform using Django which is a web framework for python. Also, they are able to successfully upgrade their framework without any technical challenges.

5.3APPLICATIONS OF PYTHON PROGRAMMING:

1) **Web Development**: Python offers different frameworks for web development like Django, Pyramid, Flask. This framework is known for security, flexibility, scalability.

2) **Game Development:** PySoy and PyGame are two python libraries that are used for game development

3) **Artificial Intelligence and Machine Learning:** There is a large number of open-source libraries which can be used while developing AI/ML applications.

4) **Desktop GUI:** Desktop GUI offers many toolkits and frameworks using which we can build desktop applications.PyQt, PyGtk, PyGUI are some of the GUI frameworks.

5.4 PYTHON STANDARD LIBRARY:

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages,

so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

5.5PYTHON PACKAGE :

To understand Python packages, we'll briefly look at scripts and modules. A ‒script‖ is something you execute in the shell to accomplish a defined task. To write a script, you'd type your code into your favorite text editor and save it with the .py extension. You can then use the python command in a terminal to execute your script.

A module on the other hand is a Python program that you import, either in interactive mode or into your other programs. ‒Module‖ is really an umbrella term for reusable code.

A Python package usually consists of several modules. Physically, a package is a folder containing modules and maybe other folders that themselves may contain more folders and modules. Conceptually, it's a namespace. This simply means that a package's modules are bound together by a package name, by which they may be referenced.

Circling back to our earlier definition of a module as reusable, importable code, we note that every package is a module — but not every module is a package. A package folder usually contains one file named_init_.py that basically tells Python: ‒Hey, this directory is a package!‖ The init file may be empty, or it may contain code to be executed upon package initialization.

You've probably come across the term –library‖ as well. For Python, a library isn't as clearly defined as a package or a module, but a good rule of thumb is that whenever a package has been published, it may be referred to as a library.

5.6 USEAGE OF PYTHON PACKAGE:

We've mentioned namespaces, publishing packages and importing modules. If any of these terms or concepts aren't entirely clear to you, we've got you! In this section, we'll cover everything you'll need to really grasp the pipeline of using Python packages in your code.

Importing a Python Package

We'll import a package using the **import** statement:

```
>>> import <some_package>
```

Let's assume that we haven't yet installed any packages. Python comes with a big collection of pre-installed packages known as the Python Standard Library. It includes tools for a range of use cases, such as text processing and doing math. Let's import the latter:

```
>>> import math
```

You might think of an import statement as a search trigger for a module. Searches are strictly organized: At first, Python looks for a module in the

cache, then in the standard library and finally in a list of paths. This list may be accessed after importing sys (another standard library module).

```
>>> import sys
>>> sys.path
['', '/home/monty/python/lib/python3.7',
'/home/monty/.local/lib/python3.7']
```

The sys.path command returns all the directories in which Python will try to find a package. It may happen that you've downloaded a package but when you try importing it, you get an error:

```
>>> import gensim
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named gensim
```

In such cases, check whether your imported package has been placed in one of Python's search paths. If it hasn't, you can always expand your list of search paths:

```
>>> sys.path.append('/home/monty/gensim-package')
```

At that point, the interpreter will have more than one more location to look for packages after receiving an **import** statement.

Namespaces and Aliasing

When we had imported the math module, we initialized the math namespace. This means that we can now refer to functions and classes from the math module by way of ‒dot notation‖:

```
>>> import math
>>> math.factorial(3)
6
>>> math.log(1)
0.0
```

Assume that we were only interested in our math module's factorial function, and that we're also tired of using dot notation. In that case, we can proceed as follows:

```
>>> from math import factorial
```

If you'd like to import multiple resources from the same source, you can simply comma-separate them in the import statement:

```
>>> from math import factorial, log
```

There is, however, always a small risk that your variables will clash with other variables in your namespace. What if one of the variables in your code was named log, too? It would overwrite the log function, causing bugs. To avoid that, it's better to import the package as we did before. If you want to save typing time, you can alias your package to give it a shorter name:

```
>>> import math as m
>>> m.factorial(3)
```

Aliasing is a pretty common technique. Some packages have commonly used aliases: For instance, the numerical computation library NumPy is almost always imported as –np.‖

Another option is to import all a module's resources into your namespace:

```
>>> from math import *
```

However, this method poses serious risk since you usually don't know all the names contained in a package, increasing the likelihood of your variables being overwritten. It's for this reason that most seasoned Python programmers will discourage use of the wildcard * in imports. Also, as the Zen of Python states, –namespaces are one honking great idea!‖

How to Install a Python Package

How about packages that are not part of the standard library? The official repository for finding and downloading such third-party packages is the Python Package Index, usually referred to simply as PyPI. To install packages from PyPI, use the package installer pip:

```
$ pip install gensim
```

pip can install Python packages from any source, not just PyPI. If you installed Python using Anaconda or Miniconda, you can also use the conda command to install Python packages.

```
$ conda install gensim
```

While conda is very easy to use, it's not as versatile as pip. So if you cannot install a package using conda, you can always try pip instead.

Reloading a Module

If you're programming in interactive mode, and you change a module's script, these changes won't be imported, even if you issue another import statement. In such case, you'll want to use the reload() function from the importlib library:

```
>>> import importlib
>>> importlib.reload(>some_module<)
```

How to Create Your Own Python Package

Packaging your code for further use doesn't necessarily mean you'll want it published to PyPI. Maybe you just want to share it with a friend, or reuse it yourself. Whatever your aim, there are several files that you should include in your project. We've already mentioned the_init_.py file.

Another important file is setup.py. Using the setuptools package, this file provides detailed information about your project and lists all dependencies — packages required by your code to run properly.

Publishing to PyPI is beyond the scope of this introductory tutorial. But if you do have a package for distribution, your project should include two more files: a README.md written in Markdown, and a license. Check out the official Python Packaging User Guide (PyPUG) if you want to know more.

5.7 INSTALLATION OF PYTHON PACKAGE:

This section covers the basics of how to install Python packages.

It's important to note that the term ‑package‖ in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a distribution). It does not to refer to the kind of package that you import in your Python source code (i.e. a container of modules). It is common in the Python community to refer to a distribution using the term ‑package‖. Using the term ‑distribution‖ is often not preferred, because it can easily be confused with a Linux distribution, or another larger software distribution like Python itself.

This section describes the steps to follow before installing other Python packages.

Ensure you can run Python from the command line

Before you go any further, make sure you have Python and that the expected version is available from your command line. You can check this by running:

- Unix/macOS

```
python3 --version
```

Windows

You should get some output like `Python 3.6.3`. If you do not have Python, please install the latest 3.x version from python.org or refer to the Installing Python section of the Hitchhiker's Guide to Python.

Note

If you're using an enhanced shell like IPython or the Jupyter notebook, you can run system commands like those in this tutorial by prefacing them with a `!` character:

```
In [1]: import sys
!{sys.executable} --version
Python 3.6.3
```

It's recommended to write `{sys.executable}` rather than plain `python` in order to ensure that commands are run in the Python installation matching the currently running notebook (which may not be the same Python installation that the `python` command refers to).

Note

Due to the way most Linux distributions are handling the Python 3 migration, Linux users using the system Python without creating a virtual environment first

should replace the `python` command in this tutorial with `python3` and the `python -m pip` command with `python3 -m pip --user`. Do *not* run any of the commands in this tutorial with `sudo`: if you get a permissions error, come back to the section on creating virtual environments, set one up, and then continue with the tutorial as written.

Ensure you can run pip from the command line

Additionally, you'll need to make sure you have pip available. You can check this by running:

Unix/macOS

```
python3 -m pip --version
```

Windows

If you installed Python from source, with an installer from python.org, or via Homebrew you should already have pip. If you're on Linux and installed using your OS package manager, you may have to install pip separately, see Installing pip/setuptools/wheel with Linux Package Managers.

If `pip` isn't already installed, then first try to bootstrap it from the standard library:

Unix/macOS

```
python3 -m ensurepip --default-pip
```

Windows

If that still doesn't allow you to run `python -m pip`:

- Securely Download get-pip.py 1
- Run `python get-pip.py`. 2 This will install or upgrade pip. Additionally, it will install setuptools and wheel if they're not installed already.

  Warning

  Be cautious if you're using a Python install that's managed by your operating system or another package manager. get-pip.py does not coordinate with those tools, and may leave your system in an inconsistent state. You can use `python get-pip.py --prefix=/usr/local/` to install in `/usr/local` which is designed for locally-installed software.

Ensure pip, setuptools, and wheel are up to date

While `pip` alone is sufficient to install from pre-built binary archives, up to date copies of the `setuptools` and `wheel` projects are useful to ensure you can also install from source archives:

Unix/macOS

```
python3 -m pip install --upgrade pip setuptools wheel
```

Windows

Optionally, create a virtual environment

See section below for details, but here's the basic venv 3 command to use on a typical Linux system:

Unix/macOS

```
python3 -m venvtutorial_env
sourcetutorial_env/bin/activate
```

Windows

This will create a new virtual environment in the tutorial env subdirectory, and configure the current shell to use it as the default python environment.

Creating Virtual Environments

Python –Virtual Environments‖ allow Python packages to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see Installingstand alone command line tools.

Imagine you have an application that needs version 1 of LibFoo, but another application requires version 2. How can you use both these applications? If you install everything into /usr/lib/python3.6/site-packages (or whatever your platform's standard location is), it's easy to end up in a situation where you unintentionally upgrade an application that shouldn't be upgraded.

Or more generally, what if you want to install an application and leave it be? If an application works, any change in its libraries or the versions of those libraries can break the application.

Also, what if you can't install packages into the global site-packages directory? For instance, on a shared host.

In all these cases, virtual environments can help you. They have their own installation directories and they don't share libraries with other virtual environments.

Currently, there are two common tools for creating Python virtual environments:

- venv is available by default in Python 3.3 and later, and installs pip and setuptools into created virtual environments in Python 3.4 and later.
- virtualenv needs to be installed separately, but supports Python 2.7+ and Python 3.3+, and pip, setuptools and wheel are always installed into created virtual environments by default (regardless of Python version).

The basic usage is like so:

Using venv:

Unix/macOS

```
python3 -m venv<DIR>
source<DIR>/bin/activate
```

Windows

Using virtualenv:

- Unix/macOS

```
python3 -m virtualenv<DIR>
source<DIR>/bin/activate
```

- Windows

For more information, see the venv docs or the virtualenv docs.

The use of **source** under Unix shells ensures that the virtual environment's variables are set within the current shell, and not in a subprocess (which then disappears, having no useful effect).

In both of the above cases, Windows users should _not_ use the **source** command, but should rather run the **activate** script directly from the command shell like so:

```
<DIR>\Scripts\activate
```

Managing multiple virtual environments directly can become tedious, so the dependency management tutorial introduces a higher level tool, Pipenv, that automatically manages a separate virtual environment for each project and application that you work on.

Use pip for Installing

pip is the recommended installer. Below, we'll cover the most common usage scenarios. For more detail, see the pip docs, which includes a complete Reference Guide.

Installing from PyPI

The most common usage of pip is to install from the Python Package Index using a requirement specifier. Generally speaking, a requirement specifier is composed of a project name followed by an optional version specifier. **PEP 440** contains a **full specification** of the currently supported specifiers. Below are some examples.

To install the latest version of "SomeProject":

- Unix/macOS

```
python3 -m pip install "SomeProject"
```

- Windows

To install a specific version:

- Unix/macOS

```
python3 -m pip install "SomeProject==1.4"
```

- Windows

To install greater than or equal to one version and less than another:

Unix/macOS

```
python3 -m pip install "SomeProject>=1,<2"
```

Windows

To install a version that's **"compatible"** with a certain version: 4

Unix/macOS

```
python3 -m pip install "SomeProject~=1.4.2"
```

Windows

In this case, this means to install any version "==1.4.*" version that's also ">=1.4.2".

Source Distributions vs Wheels

pip can install from either Source Distributions (sdist) or Wheels, but if both are present on PyPI, pip will prefer a compatible wheel. You can override pip`s default behavior by e.g. using its –no-binary option.

Wheels are a pre-built distribution format that provides faster installation compared to Source Distributions (sdist), especially when a project contains compiled extensions.

If pip does not find a wheel to install, it will locally build a wheel and cache it for future installs, instead of rebuilding the source distribution in the future.

Upgrading packages

Upgrade an already installed SomeProject to the latest from PyPI.

- Unix/macOS

```
python3 -m pip install --upgrade SomeProject
```

- Windows

Installing to the User Site

To install packages that are isolated to the current user, use the --user flag:

- Unix/macOS

```
python3 -m pip install --user SomeProject
```

- Windows

For more information see the User Installs section from the pip docs.

Note that the --user flag has no effect when inside a virtual environment - all installation commands will affect the virtual environment.

If SomeProject defines any command-line scripts or console entry points, --user will cause them to be installed inside the user base binary directory, which may or may not already be present in your shell's PATH. (Starting in version

10, pip displays a warning when installing any scripts to a directory outside PATH.) If the scripts are not available in your shell after installation, you'll need to add the directory to your PATH:

- On Linux and macOS you can find the user base binary directory by running `python -m site --user-base` and adding `bin` to the end. For example, this will typically print `~/.local` (with `~` expanded to the absolute path to your home directory) so you'll need to add `~/.local/bin` to your `PATH`. You can set your `PATH` permanently by modifying `~/.profile`.
- On Windows you can find the user base binary directory by running `py -m site --user-site` and replacing `site-packages` with `Scripts`. For example, this could return `C:\Users\Username\AppData\Roaming\Python36\site-packages` so you would need to set your `PATH` to include `C:\Users\Username\AppData\Roaming\Python36\Scripts`. You can set your user `PATH` permanently in the Control Panel. You may need to log out for the `PATH` changes to take effect.

Requirements files

Install a list of requirements specified in a Requirements File.

Unix/macOS

```
python3 -m pip install -r requirements.txt
```

Windows

Installing from VCS

Install a project from VCS in ‒editable‖ mode. For a full breakdown of the syntax, see pip’s section on VCS Support.

Unix/macOS

```
python3 -m pip install -e git+https://git.repo/some_pkg.git#egg=SomeProject # from git
python3 -m pip install -e hg+https://hg.repo/some_pkg#egg=SomeProject # from mercurial
python3 -m pip install -e svn+svn://svn.repo/some_pkg/trunk/#egg=SomeProject# from svn
python3 -m pip install -e git+https://git.repo/some_pkg.git@feature#egg=SomeProject  # from a branch
```

Windows

Installing from other Indexes

Install from an alternate index

Unix/macOS

```
python3 -m pip install --index-url http://my.package.repo/simple/ SomeProject
```

Windows

Search an additional index during install, in addition to PyPI

Unix/macOS

```
python3 -m pip install --extra-index-url http://my.package.repo/simple SomeProject
```

Windows

Installing from a local src tree

Installing from local src in Development Mode, i.e. in such a way that the project appears to be installed, but yet is still editable from the src tree.

Unix/macOS

```
python3 -m pip install -e <path>
```

Windows

You can also install normally from src

Unix/macOS

```
python3 -m pip install <path>
```

Windows

Installing from local archives

Install a particular source archive file.

Unix/macOS

```
python3 -m pip install ./downloads/SomeProject-1.0.4.tar.gz
```

Windows

Install from a local directory containing archives (and don’t check PyPI)

Unix/macOS

```
python3 -m pip install --no-index --find-links=file:///local/dir/ SomeProject
python3 -m pip install --no-index --find-links=/local/dir/ SomeProject
python3 -m pip install --no-index --find-links=relative/dir/ SomeProject
```

Windows

Installing from other sources

To install from other data sources (for example Amazon S3 storage) you can create a helper application that presents the data in a **PEP 503** compliant index format, and use the `--extra-index-url` flag to direct pip to use that index.

```
./s3helper --port=7777
python -m pip install --extra-index-url http://localhost:7777 SomeProject
```

Installing Prereleases

Find pre-release and development versions, in addition to stable versions. By default, pip only finds stable versions.

- Unix/macOS

```
python3 -m pip install --pre SomeProject
```

- Windows

Installing Setuptools ‑Extras‖

Install setuptools extras.

- Unix/macOS

```
python3 -m pip install SomePackage[PDF]
python3 -m pip install SomePackage[PDF]==3.0
python3 -m pip install -e .[PDF]# editable project in current directory
```

CHAPTER 6

SOFTWARE TESTING

6.1 UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 INTEGRATION TESTING:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.3 UNCTIONAL TEST:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

6.4 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.5TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.6 SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods

that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6.7 PROTOTYPE CODING :

```
importtensorflow as tf
fromkeras_preprocessing.image import ImageDataGenerator
fromkeras_preprocessing import image
importnumpy as np
importeasygui
fromkeras.models import load_model
importos
#import serial
importtkinter as tk
fromtkinter import *
fromtkinter import messagebox
from PIL import Image, ImageTk
import serial
import time



my_w = tk.Tk()
sw=my_w.winfo_screenwidth()
```

```
sh=my_w.winfo_screenheight()
w=sw-10
print(sw,sh)
my_w.geometry('%dx%d' %(w,sh))
my_w.title('Traffic Detetction')
my_font1=('times', 18, 'bold')

bg = ImageTk.PhotoImage(file='background2.png')
bgLabel = Label(my_w, image=bg)
bgLabel.place(x=0, y=0)
bgLabel.pack(fill=BOTH,expand=YES)



xl=sw/4
x1=20
print(x1)
x2=(xl*1)+x1
x2=int(x2)
print(x2)
x3=(xl*2)+x1
x3=int(x3)
print(x3)
x4=(xl*3)+x1
x4=int(x4)
print(x4)
```

```
#l1=tk.Label(my_w,text='Lane
1',width=20,font=my_font1,bg='#FFFF40',fg='black',)
#l1.place(x=x1, y=100, width=250)
#l2=tk.Label(my_w,text='Lane
2',width=20,font=my_font1,bg='#FFFF40',fg='black',)
#l2.place(x=x2, y=100, width=250)
#l3=tk.Label(my_w,text='Lane
3',width=20,font=my_font1,bg='#FFFF40',fg='black',)
#l3.place(x=x3, y=100, width=250)
#l4=tk.Label(my_w,text='Lane
4',width=20,font=my_font1,bg='#FFFF40',fg='black',)
#l4.place(x=x4, y=100, width=250)


b1 = tk.Button(my_w, text='Upload Images',
width=20,command=lambda:result(),font=my_font1,
activebackground='#22228B', bg='black',fg='yellow')
b1.place(x=x1,y=470, width=250, height=40)
b2 = tk.Button(my_w, text='Upload Images',
width=20,command=lambda:result1(),font=my_font1,
activebackground='#22228B', bg='black',fg='yellow')
b2.place(x=x2,y=470, width=250, height=40)
b3 = tk.Button(my_w, text='Upload Images',
width=20,command=lambda:result2(),
font=my_font1,activebackground='#22228B', bg='black',fg='yellow')
b3.place(x=x3,y=470, width=250, height=40)
b4 = tk.Button(my_w, text='Upload Images',
width=20,command=lambda:result3(),font=my_font1,
activebackground='#22228B', bg='black',fg='yellow')
```

```
b4.place(x=x4,y=470, width=250, height=40)

print(tf.__version__)

lb1 = tk.Button(text = ' ')
lb1.config(bg = 'black')
lb1.place(x=140,y=58,height=40,width=30)

lb2 = tk.Button(text = ' ')
lb2.config(bg = 'black')
lb2.place(x=390,y=58,height=40,width=30)

lb3 = tk.Button(text = ' ')
lb3.config(bg = 'black')
lb3.place(x=640,y=58, height=40,width=30)

lb4 = tk.Button(text = ' ')
lb4.config(bg = 'black')
lb4.place(x=890,y=58,height=40,width=30)

get_result=[]

defshowresult():
solution=''
print(get_result)
print(len(get_result))
try:
iflen(get_result)==4:
if(get_result[0]=="No Traffic"):
```

```
        l1="d"
elif(get_result[0]=="Low Traffic"):
        l1="c"
elif(get_result[0]=="Medium Traffic"):
        l1="b"
elif(get_result[0]=="High Traffic"):
        l1="a"

if(get_result[1]=="No Traffic"):
        l2="d"
elif(get_result[1]=="Low Traffic"):
        l2="c"
elif(get_result[1]=="Medium Traffic"):
        l2="b"
elif(get_result[1]=="High Traffic"):
        l2="a"

if(get_result[2]=="No Traffic"):
        l3="d"
elif(get_result[2]=="Low Traffic"):
        l3="c"
elif(get_result[2]=="Medium Traffic"):
        l3="b"
elif(get_result[2]=="High Traffic"):
        l3="a"

if(get_result[3]=="No Traffic"):
        l4="d"
elif(get_result[3]=="Low Traffic"):
```

```
        l4="c"
elif(get_result[3]=="Medium Traffic"):
        l4="b"
elif(get_result[3]=="High Traffic"):
        l4="a"

print(l1,l2,l3,l4)
print("_____")
        l1=str(l1)
        l2=str(l2)
        l3=str(l3)
        l4=str(l4)
        l1=bytes(l1,'utf-8')
        l2=bytes(l2,'utf-8')
        l3=bytes(l3,'utf-8')
        l4=bytes(l4,'utf-8')

print("_____")
        '''
SerialObj = serial.Serial('COM3')
SerialObj.baudrate = 9600
SerialObj.bytesize = 8
SerialObj.parity   ='N'
SerialObj.stopbits = 1
SerialObj.write(l1)
time.sleep(1)
SerialObj.write(l2)
time.sleep(1)
SerialObj.write(l3)
```

```
time.sleep(1)
SerialObj.write(l4)
time.sleep(1)



SerialObj.close()'"

        #else:
#  pass

else:
messagebox.showerror('Error','Upload All Files')

except Exception as e:
pass

titleLabel = Label(my_w, text='Traffic Detetction', font=('italic', 22, 'bold '),
bg='black',
fg='white', )
titleLabel.place(x=0, y=5, width=sw, height=50)

show_result=Button(my_w,text="Show Result", font='italic 17
bold',fg='red',bg='#FFFF45',command=showresult)
show_result.place(x=xl+120, y=580, width=220)

model1 = load_model('model/Class1/model_Class1.h5')
model2 = load_model('model/Class2/model_Class2.h5')
model3 = load_model('model/Class3/model_Class3.h5')
```

```
def result():
try:
get_result.pop(0)
finally:
filename =upload_file1()
    test_image2 = image.load_img(filename, target_size = (64, 64))
    test_image2 = image.img_to_array(test_image2)
    test_image2 = np.expand_dims(test_image2, axis = 0)
    # cnn prediction on the test image
    result1 = model1.predict(test_image2)
print(result1)
    result2 = model2.predict(test_image2)
print(result2)
    result3 = model3.predict(test_image2)
print(result3)


if result1[0][0] == 0:
        prediction2="No Traffic"
elif result2[0][0]==0:
        prediction2="Low Traffic"
elif result3[0][0]==0:
        prediction2="Medium Traffic"
else:
        prediction2="High Traffic"
print(prediction2)
prediction=prediction2
get_result.insert(0,prediction)
return filename
```

```
def result1():
try:
get_result.pop(1)
finally:
    filename1 =upload_file2()
    test_image2 = image.load_img(filename1, target_size = (64, 64))
    test_image2 = image.img_to_array(test_image2)
    test_image2 = np.expand_dims(test_image2, axis = 0)
    result1 = model1.predict(test_image2)
print(result1)
    result2 = model2.predict(test_image2)
print(result2)
    result3 = model3.predict(test_image2)
print(result3)

if result1[0][0] == 0:
        prediction2="No Traffic"
elif result2[0][0]==0:
        prediction2="Low Traffic"
elif result3[0][0]==0:
        prediction2="Medium Traffic"
else:
        prediction2="High Traffic"

print(prediction2)
prediction=prediction2
get_result.insert(1,prediction)
return filename1
```

```
def result2():
try:
get_result.pop(2)
finally:
    filename2 =upload_file3()
    test_image2 = image.load_img(filename2, target_size = (64, 64))
    test_image2 = image.img_to_array(test_image2)
    test_image2 = np.expand_dims(test_image2, axis = 0)
    result1 = model1.predict(test_image2)
print(result1)
    result2 = model2.predict(test_image2)
print(result2)
    result3 = model3.predict(test_image2)
print(result3)
if result1[0][0] == 0:
        prediction2="No Traffic"
elif result2[0][0]==0:
        prediction2="Low Traffic"
elif result3[0][0]==0:
        prediction2="Medium Traffic"
else:
        prediction2="High Traffic"
print(prediction2)
prediction=prediction2
get_result.insert(2,prediction)
return filename2

def result3():
```

```
try:
get_result.pop(3)
finally:
    filename3 =upload_file4()
    test_image2 = image.load_img(filename3, target_size = (64, 64))
    test_image2 = image.img_to_array(test_image2)
    test_image2 = np.expand_dims(test_image2, axis = 0)
    result1 = model1.predict(test_image2)
print(result1)
    result2 = model2.predict(test_image2)
print(result2)
    result3 = model3.predict(test_image2)
print(result3)

if result1[0][0] == 0:
        prediction2="No Traffic"
elif result2[0][0]==0:
        prediction2="Low Traffic"
elif result3[0][0]==0:
        prediction2="Medium Traffic"
else:
        prediction2="High Traffic"

print(prediction2)
prediction=prediction2
get_result.insert(3,prediction)
return filename3

def upload_file1():
```

```
filename=easygui.fileopenbox()
img=Image.open(filename) # read the image file
img=img.resize((200,140)) # new width & height
img=ImageTk.PhotoImage(img)
    e1 =tk.Label(my_w)
    e1.place(x=x1, y=180, width=240, height=250)
    e1.image = img
e1['image']=img
return filename

def upload_file2():
filename=easygui.fileopenbox()

img=Image.open(filename) # read the image file
img=img.resize((200,140)) # new width & height
img=ImageTk.PhotoImage(img)
    e1 =tk.Label(my_w)
    e1.place(x=x2, y=180, width=240, height=250)
    e1.image = img
e1['image']=img
return filename

def upload_file3():
filename=easygui.fileopenbox()
img=Image.open(filename) # read the image file
img=img.resize((200,140)) # new width & height
img=ImageTk.PhotoImage(img)
    e1 =tk.Label(my_w)
    e1.place(x=x3, y=180, width=240, height=250)
```

```
    e1.image = img
e1['image']=img
return filename

def upload_file4():
filename=easygui.fileopenbox()
img=Image.open(filename) # read the image file
img=img.resize((200,140)) # new width & height
img=ImageTk.PhotoImage(img)
    e1 =tk.Label(my_w)
    e1.place(x=x4, y=180, width=240, height=250)
    e1.image = img
e1['image']=img
return filename
my_w.mainloop()
```

6.8 SIMULATION OUTPUT:

HOME PAGE:

Fig 7 Simulation output homepage

TRAFFIC DETECTION:

CHAPTER 7

CONCLUSION:

In this study, we used transfer learning to develop a CNN model for automatic Traffic detection using Road images. Transfer learning uses weights from networks previously trained on millions of data. The proposed study implements four different transfer learning models with different optimizers (ADAM, SGD, RMSprop), and extensive experiments were performed on the datasets with the largest number of Road images currently available. For this models, the features are extracted using transfer learning, and three dense layers along with the softmax layer are used for classification purposes. The proposed deep TL models shows fast learning by using the Adam optimizer and the dropout method avoids the problem of over fitting. In future work, the performance of the system can still be improved by using larger data sets and using other deep learning techniques.

7.1 FUTURE ENHANCEMENTS:

In the future, the system are often further improved using more factors that affect traffic management using other methods like deep learning, artificial neural network, and even big data. The Ambulance can use this technique to seek out which route would be easiest to achieve on destination. The system can help in suggesting the users with their choice of search and also it can help to find the simplest choice where traffic isn't in any crowded environment. Many forecasting methods have already been applied in road traffic jam forecasting.

CHAPTER 8

REFERENCES:

[1] Guo, K., Hu, Y., Qian, Z., Liu, H., Zhang, K., Sun, Y., & Yin, B. (2020). Optimized graph convolution recurrent neural network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 22(2), 1138-1149.

[2] Xu, H., & Jiang, C. (2020). Deep belief network-based support vector regression method for traffic flow forecasting. Neural Computing and Applications, 32(7), 2027-2036.

[3] Zhao, F., Zeng, G. Q., & Lu, K. D. (2019). EnLSTM-WPEO: Short-term TFP by ensemble LSTM, NNCT weight integration, and population extremal optimization. IEEE Transactions on Vehicular Technology, 69(1), 101-113.

[4] Peng, H., Du, B., Liu, M., Liu, M., Ji, S., Wang, S., ...& He, L. (2021). Dynamic graph

convolutional network for long-term TFP with reinforcement learning. Information Sciences, 578, (1) 401-416.

[5] B. Hussain, M. K. Afzal, S. Ahmad and A. M. Mostafa. (2021) . Intelligent TFP Using Optimized GRU Model. IEEE Access, 9(1), 100736-100746

[6] Ma, Q., Huang, G. H., &Ullah, S. (2020). A Multi-Parameter Chaotic Fusion Approach for Traffic Flow Forecasting. IEEE Access, 8(1), 222774-222781.

[7] Wang, Z., Su, X., & Ding, Z. (2020). Long-term traffic prediction based on lstm encoder-decoder architecture. IEEE Transactions on Intelligent Transportation Systems, 22(10), 6561-6571.

[8] Kumar, B. R., Chikkakrishna, N. K., &Tallam, T. (2020). Short Term Predictions of Traffic Flow Characteristics using ML Techniques. In 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA) 4(1),1504-1508.

[9]Jia, T., & Yan, P. (2020). Predicting citywide road traffic flow using deep spatiotemporal neural networks.IEEE Transactions on Intelligent Transportation Systems, 22(5), 3101-3111.

[10]Qu, W., Li, J., Yang, L., Li, D., Liu, S., Zhao, Q., & Qi, Y. (2020). Short-term intersection Traffic flow forecasting. Sustainability, 12(19), 1-13.