PHASE 5:

PROJECT REPORT:

PROBLEM DEFENITION AND DESIGN THINKING:

- Develop an efficient and sustainable traffic management system that addresses congestion, enhances safety, minimizes environmental impact, and optimizes

transportation infrastructure utilization within urban areas. The system should incorporate advanced technologies, data analytics, and smart traffic control mechanisms to ensure smooth vehicular flow, prioritize public transport, promote pedestrian safety, and reduce overall commute times for residents and commuters."

- Traffic management is a critical issue in urban areas worldwide. As urban populations continue to grow, the demand for efficient and sustainable transportation systems becomes increasingly imperative. Inadequate traffic management leads to congesΘon, pollution, accidents, and significant economic losses. This problem statement aims to address the challenges and propose solutions for effective traffic management in urban environments.

Inefficient Public Transportation:

- Many cities face challenges in providing reliable and efficient public transportation systems. Inadequate coverage, unreliable schedules, and poor connectivity contribute to a reliance on personal vehicles, exacerbating traffic congestion.

Limited Parking Space:

- A shortage of parking spaces in urban centers not only adds to congestion but also leads to illegal parking.

Safety Concerns:

- High traffic volumes and inadequate enforcement of traffic rules contribute to a high incidence of accidents, leading to injuries, loss of life, and property damage. This poses a significant threat to public safety.

Inadequate Infrastructure:

 Outdated or insufficient road networks, poorly designed intersections, and lack of pedestrian-friendly infrastructure contribute to traffic bottlenecks and unsafe conditions for both motorists and pedestrians.

Lack of Technological Integration: Many cities do not fully leverage modern technologies for traffic management.

Advanced solutions like real-time traffic monitoring, smart traffic signals, and data driven decision-making are underutilized.

Environmental Impact:

 Excessive traffic congestion results in elevated levels of air pollution and greenhouse gas emissions. This not only affects the health of citizens but also contributes to climate change.

INNOVATON:

SYSTEM ARCHITECTURE:

The system follows a layered architecture with four layers :

(i) a sensing layer with active things and sensors, (ii) (ii) a network layer represents the mode of communication and protocols, (iii) (iii) service layer indicates the data analysis and storage, and (iv) (iv) application layer describe the end-user applications. The sensing layer collects vehicle data through the sensors installed on roadsides and the WiFi-based microcontroller transfer the real-time data to the service layer. Several open source cloud IoT platforms

are available to manage connected devices, data storage, and analysis. Thinger. io, which is an open-source IoT platform for integrating data fusion applications acts as a service layer in this study. The end-users receive traffic updates through roadside message display units and dashboards. The physical infrastructures such as sensors and message display units are installed on roadsides at selected road intersections. The message units installed at important road intersections substitutes the smart devices and update drivers on the current traffic scenario. The authorities can also send messages on unusual road incidents along with expected clearance time or alternate route suggestions (if any) to assist emergency vehicle handling. The proposed system aims to generate public value by saving the on-road time of drivers through early warning messages. In summary, the proposed system has the following features:

(i)Appropriate to estimate traffic congestions on collector roads using road occupancy measure

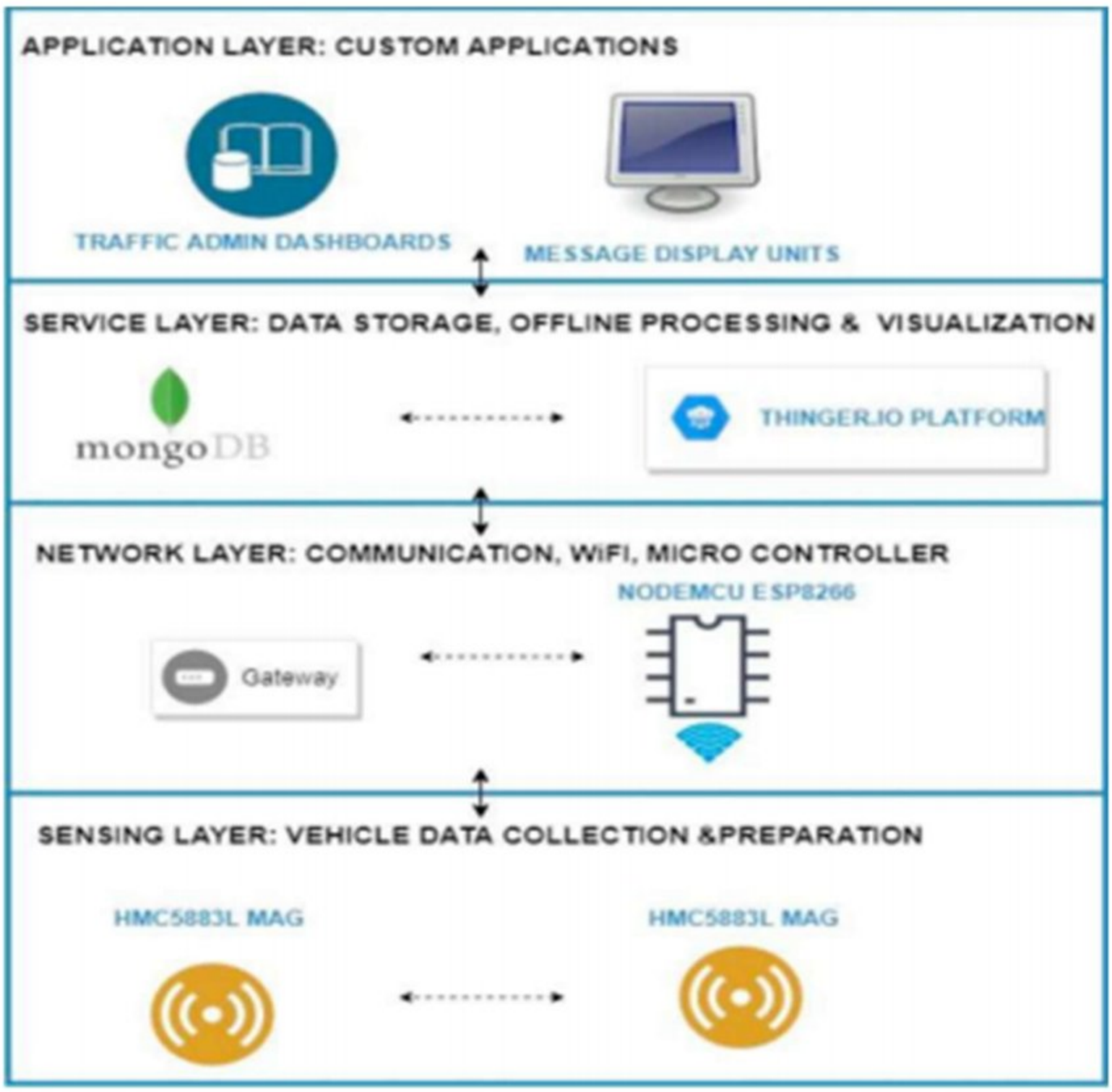
(ii)Update residents on real-time traffic messages through roadside display units

(iii)Monitor the road density of smart campuses especially during peak hours and help to improve mobility

(iv)Assist authorities to broadcast important traffic incident

messages. (v)Provide a real-time dashboard to monitor the traffic updates.

- An IoT based system architecture mostly contains a sensing layer, network layer, service layer, and an application layer . The sensing layer acquires data from the things, the network layer transfers the collected data from devices to the service layer, the service layer controls the devices and analyzes the collected data, and finally, the application layer which indicates the user interface. The layered architecture is presented



Flow chart:

- The geographical map provides the road segment information, intersections, and routes. The maps are processed to load the road information to the database as well as to extract the message board locations. The user

  generated map can be used to find the message board location.

- The road junctions that have more connected road segments are the best locations to display traffic-warning messages. The message board locations are selected based on its exposure to maximize message visibility.

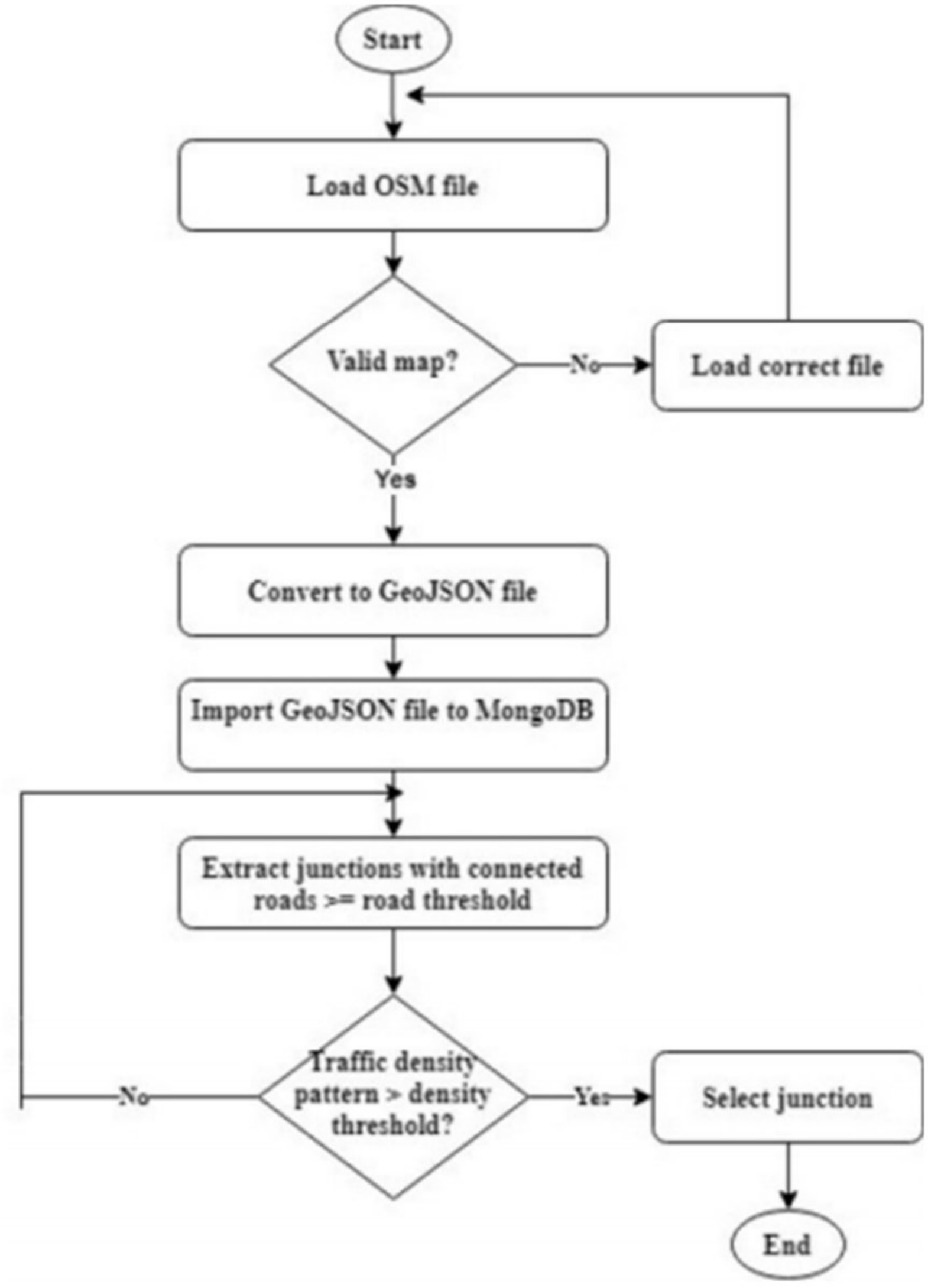
- The message board selection is considered as a maximization problem because the objective is to maximize the visibility of the message.

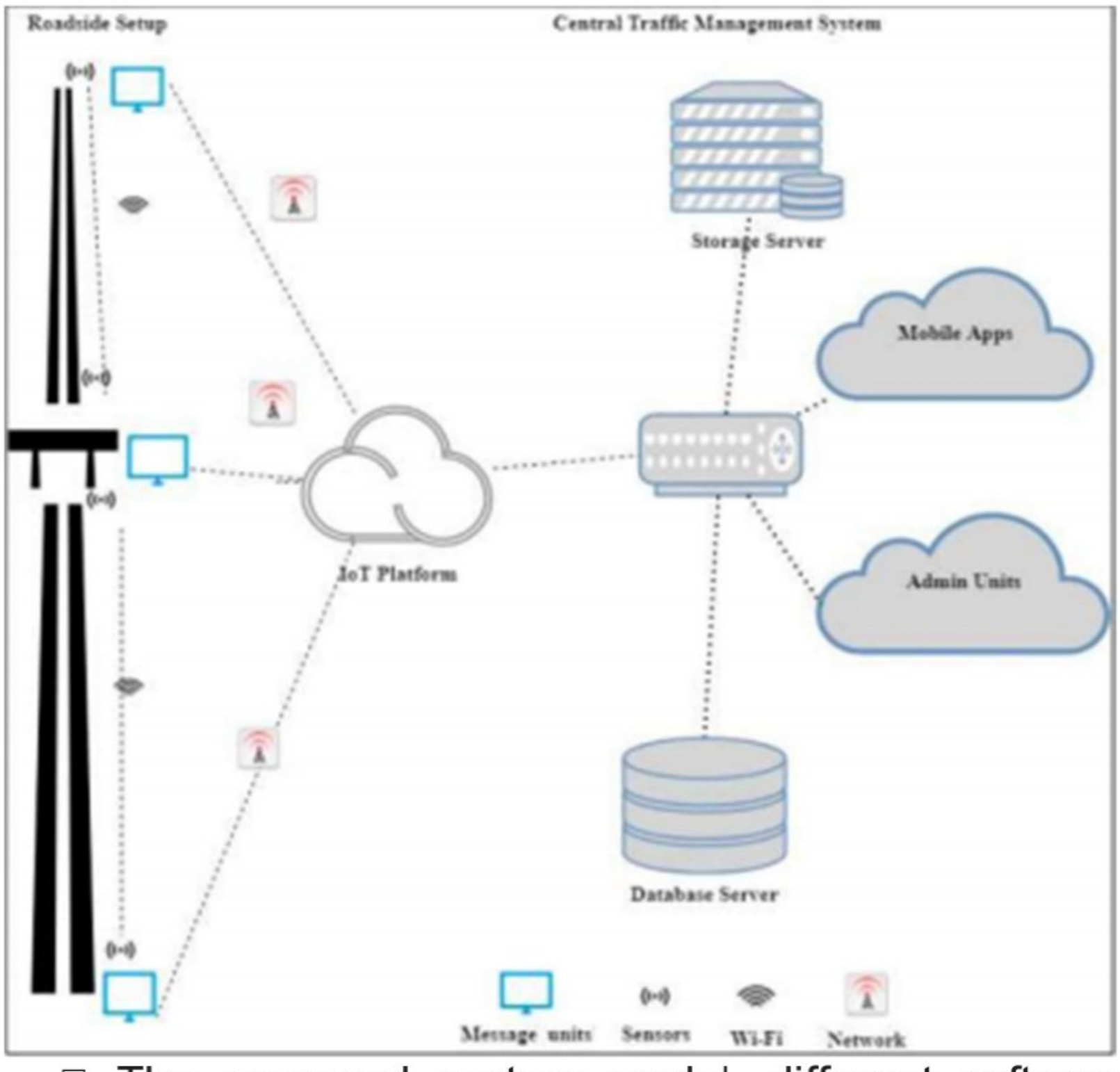
- The idea of billboard advertising can be applied here to maximize the strength of message exposure . Also, the major parking slots in a closed campus can be selected to reach the messages to the maximum.

- The number of connected roads is one parameter that decides the strength of message exposure. Besides, the earlier patterns of traffic density can also be selected while determining the message unit location.
- The process begins with geographical map format

conversion and database loading.

- The second step is to identify the message board locations based on previous traffic density at intersections and the number of connected roads.





- The proposed system model, different software and hardware components required, and algorithms to implement the proposed system. The proposed system communication model is presented in which has components installed at the roadside and a cloud-based central server. The roadside setup includes sensors and message boards. The sensors and boards will be installed between two road segment intersections. The central server includes data storage, cloud services, and interfaces. The components can communicate with each other using WiFi.

DEVELOPMENT PART 1

This Traffic Management System will not only optimize traffic flow but also contribute to a safer and more sustainable urban environment.

By harnessing the power of Python and its extensive libraries, we aim to create a cutting-edge solution that addresses the challenges of modern urban transportation.

1. Obtain the Dataset

- You can find datasets for traffic management from various sources, including government agencies, research institutions, or online data repositories. Make sure the dataset is relevant to your specific problem (e.g., traffic prediction, congestion analysis, etc.).

2. Read and Understand the Data

- Load the dataset into a suitable data structure (e.g., pandas DataFrame in Python). Understand the features (columns) and the meaning of each variable.

3. Data Cleaning

- Handle missing values: Depending on the dataset, you may encounter missing data. Decide whether to impute missing values or remove the corresponding entries.
- Handle duplicates: Check for and remove duplicate records if they exist.
- Handle outliers: Identify and decide whether to remove or transform outliers.

4. Data Exploration and Visualization

- Understand the distribution of variables, correlations, and any interesting patterns in the

data. Visualization tools like matplotlib or seaborn in Python can be helpful.

5. Feature Engineering

- Create new features if necessary. For example, you might extract time-related information (hour, day of the week, etc.) from timestamps, or derive additional features from existing ones.

6. Data Transformation

- Convert categorical variables into numerical format (e.g., one-hot encoding or label encoding).
- Standardize or normalize numerical features if necessary.

7. Splitting the Data

- Divide the dataset into training, validation, and test sets. The training set is used to train the model, the validation set is used for hyperparameter tuning, and the test set is used for evaluating the final model.

8. Scaling

:

- Depending on the algorithm you plan to use, you might need to scale the features (e.g., using StandardScaler in scikit-learn) to ensure they have similar magnitudes.

9. Save Preprocessed Data

:

- Once you've completed the preprocessing steps,

it's a good practice to save the preprocessed data so you can easily load it for future analyses without repeating the entire process.

10. Model Building

- Use the preprocessed data to train a model for your specific traffic management task. This could be a regression model for traffic prediction, a classification model for congestion analysis, or any other relevant machine learning approach.

Importing Libraries

- Pandas – Use to load the data frame in a 2D array format.
- NumPy – NumPy arrays are very fast and can perform large computations in a very short time.
- Matplotlib – This library is used to draw visualizations.
- Sklearn – It contains multiple libraries having pre-implemented functions of model development and evaluation.
- OpenCV – This library mainly focused on image processing and handling.
- Tensorflow – It provides a range of functions to achieve complex functionalities with single lines of code.

PYTHON CODE:

```
import matplotlib.image as mpimg
import os

from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing import
```

```
image_dataset_from_directory  from
tensorflow.keras.preprocessing.image import
ImageDataGenerator,  load_img
from keras.utils.np_utils import to_categorical
from tensorflow.keras.utils import
image_dataset_from_directory  from
tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Conv2D,
MaxPooling2D  from tensorflow.keras.layers import
Activation, Dropout, Flatten, Dense  from
tensorflow.keras.models import Sequential
from keras import layers
from tensorflow import keras
from tensorflow.keras.layers.experimental.preprocessing
import Rescaling  from sklearn.model_selection import
train_test_split

import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
import numpy as np
from glob import glob
import cv2

import warnings
warnings.filterwarnings('ignore')
```

Loading and Extracting the Dataset:

The dataset has 58 classes of Traffic Signs and a label.csv file. The folder is in zip format. To unzip the dataset, we will run the code below.

PYTHON CODE:

```
# Extracting the compressed dataset.
from zipfile import ZipFile data_path
 = '/content/traffic-sign-dataset-classification.zip'
with   ZipFile(data_path, 'r') as zip
 : zip.extractall()
```

Data Visualization

Data visualization means the visualization of data to understand the key features of the dataset.

PYTHON CODE:

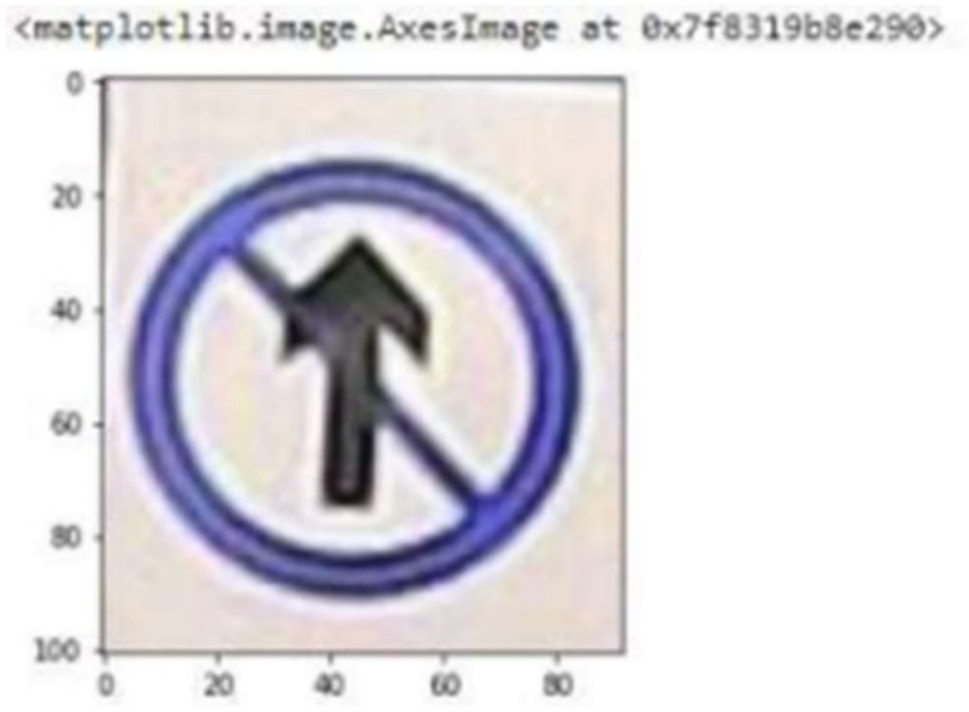
```
# path to the folder containing our dataset
dataset = '../content/traffic_Data/DATA'

# path of label file
labelfile = pd.read_csv('labels.csv')
```

Once we load the dataset, now let's visualize some random images from the different folders. For that, we will use plt.imshow() command.

PYTHON CODE:

```
# Visualize some images from the dataset
img =
cv2.imread("/content/traffic_Data/DATA/10/010_00
11.png")  plt.imshow(img)
```

Output :



ONE MORE IMAGE AS EXAMPLE:

PYTHON CODE:

```
img =
cv2.imread("/content/traffic_Data/DATA/23/023_00
01.png")  plt.imshow(img)
```

Output :

Label File –

This file includes the 58 rows and 2 columns. The columns contains the class id and the name of the symbol. And the rows depicts the 58 different classes id and names.

PYTHON CODE:

```
labelfile.head()
```

Output:

	ClassId	Name
0	0	Speed limit (5km/h)
1	1	Speed limit (15km/h)
2	2	Speed limit (30km/h)
3	3	Speed limit (40km/h)
4	4	Speed limit (50km/h)

Data Preparation for Training

In this section, we will split the dataset into train and val set. Train set will be used to train the model and val set will be use to evaluate the performance of our model.

PYTHON CODE":

```
train_ds =
tf.keras.preprocessing.image_dataset_from_directory(data
set,  validation_split=0.2,
 subset='tr aining',
 image_size =(
 224,  224),
 seed=123,   batch_size =32)
val_ds =
tf.keras.preprocessing.image_dataset_from_directory(dat
aset,  validation_split=0.2,
 subset='vali dation',
 image_size=(   224,  224),
 seed=123,   batch_size=3 2)
```

Output:

Found 4170 files belonging to 58 classes.

Using 3336 files for training.

Found 4170 files belonging to 58 classes.

Using 834 files for validation.

Once we split the dataset, Let's create the list of the class names and print few images along with their class names.

PYTHON CODE:

```
class_numbers = train_ds.class_names
class_names = []
for i in class_numbers:
 class_names.append(labelfile['Name'][int(i)])
```

Let's visualize the train dataset and print 25 images from

the dataset. **PYTHON CODE:**

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
 for i in range(25):
 ax = plt.subplot(5, 5, i + 1)
 plt.imshow(images[i].numpy().astype("uint8"))
 plt.title(class_names[labels[i]])
 plt.axis("off")

plt.show()
```

Output:

Data Augmentation:

Sometimes the data is limited and we the model is not performing well with limited data. For this method, we use Data Augmentation. It is the method to increase the amount and diversity in data. We do not collect new data, rather we transform the already present data.

PYTHON CODE:

```
data_augmentation = tf.keras.Sequential(
 [

tf.keras.layers.experimental.preprocessing.Random
Flip(   "horizontal", input_shape=(224, 224, 3)),

tf.keras.layers.experimental.preprocessing.RandomRotatio
n(0.1),
tf.keras.layers.experimental.preprocessing.RandomZoom(
0.2),
tf.keras.layers.experimental.preprocessing.RandomFlip(
```

```
mode="horizontal_and_vertical")
 ]
)
```

Model Architecture:

The model will contain the following Layers:

 Four Convolutional Layers followed by MaxPooling Layers. The Flatten layer to flatten the output of the convolutional layer. Then we will have three fully connected Dense layers followed by the output of the of Softmax activation function.

PYTHON CODE:

```
model = Sequential()
model.add(data_augmentation)
model.add(Rescaling(1./255))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dense(len(labelfile),
```

activation='softmax')) Let's print the

summary of the model.

PYTHON CODE:

```
model.summary()
```

Output :

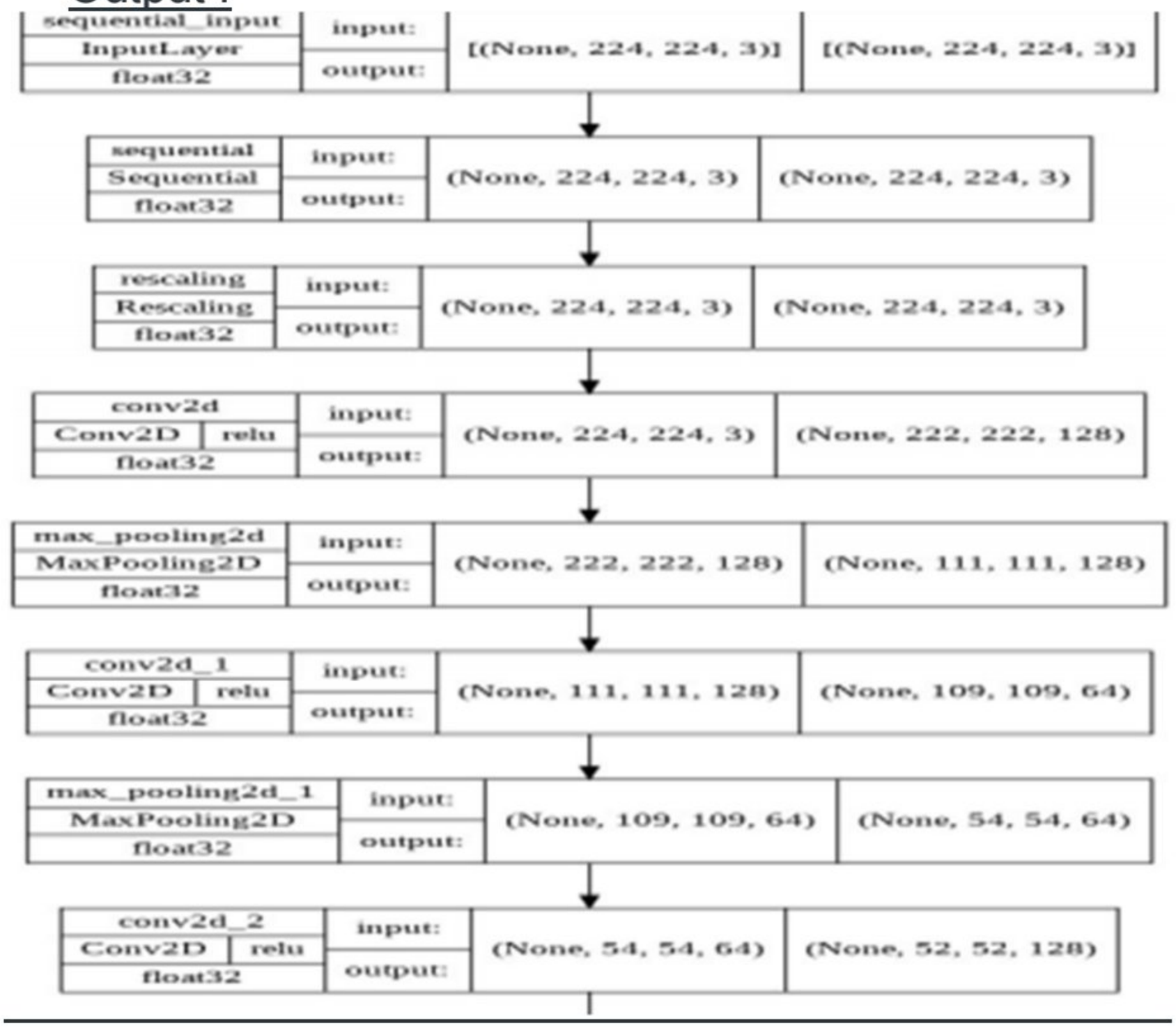
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_2 (Sequential)   (None, 224, 224, 3)       0

 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0

 conv2d_14 (Conv2D)          (None, 222, 222, 128)     3584

 max_pooling2d_12 (MaxPoolin  (None, 111, 111, 128)    0
 g2D)

 conv2d_15 (Conv2D)          (None, 109, 109, 64)      73792

 max_pooling2d_13 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 52, 52, 128)       73856

 max_pooling2d_14 (MaxPoolin  (None, 26, 26, 128)      0
 g2D)

 conv2d_17 (Conv2D)          (None, 24, 24, 256)       295168

 max_pooling2d_15 (MaxPoolin  (None, 12, 12, 256)      0
 g2D)

 flatten_5 (Flatten)         (None, 36864)             0

 dense_9 (Dense)             (None, 64)                2359360

 dropout_5 (Dropout)         (None, 64)                0

 dense_10 (Dense)            (None, 128)               8320

 dense_11 (Dense)            (None, 58)                7482

=================================================================
Total params: 2,821,562
Trainable params: 2,821,562
Non-trainable params: 0
_________________________________________________________________
```

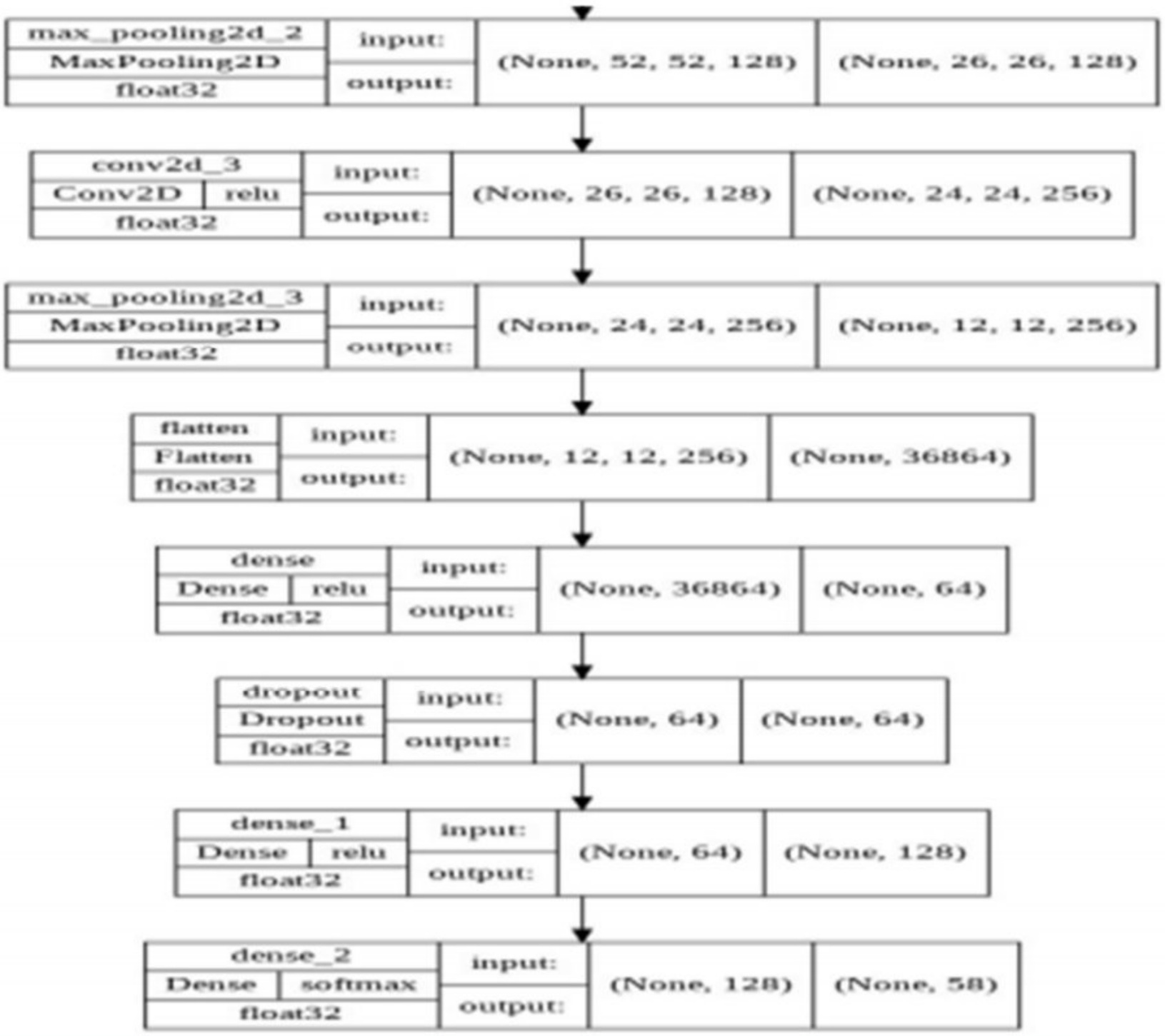
To understand the huge number of parameters and complexity of the model which helps us to achieve a high-performance model let's see the plot_model.

PYTHON CODE:

```
keras.utils.plot_model(
 model,
 show_shapes=True,
 show_dtype=True,
 show_layer_activations=True )
```

Output :





PYTHON CODE:

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),  optimizer='adam',
 metrics=['accuracy'])
```

Model Training

Now we will train our model, the model is working fine on epochs = 50, but you can perform hyperparameter tuning for better results.

We can also use callback function for early stopping the model training. PYTHON CODE:

```
# Set callback functions to early stop training
mycallbacks = [EarlyStopping(monitor='val_loss', patience=5)]  history = model.fit(train_ds,
 validation_data=val_ds,
 epochs=50,
```

```
callbacks=mycallbacks)
```

Output:

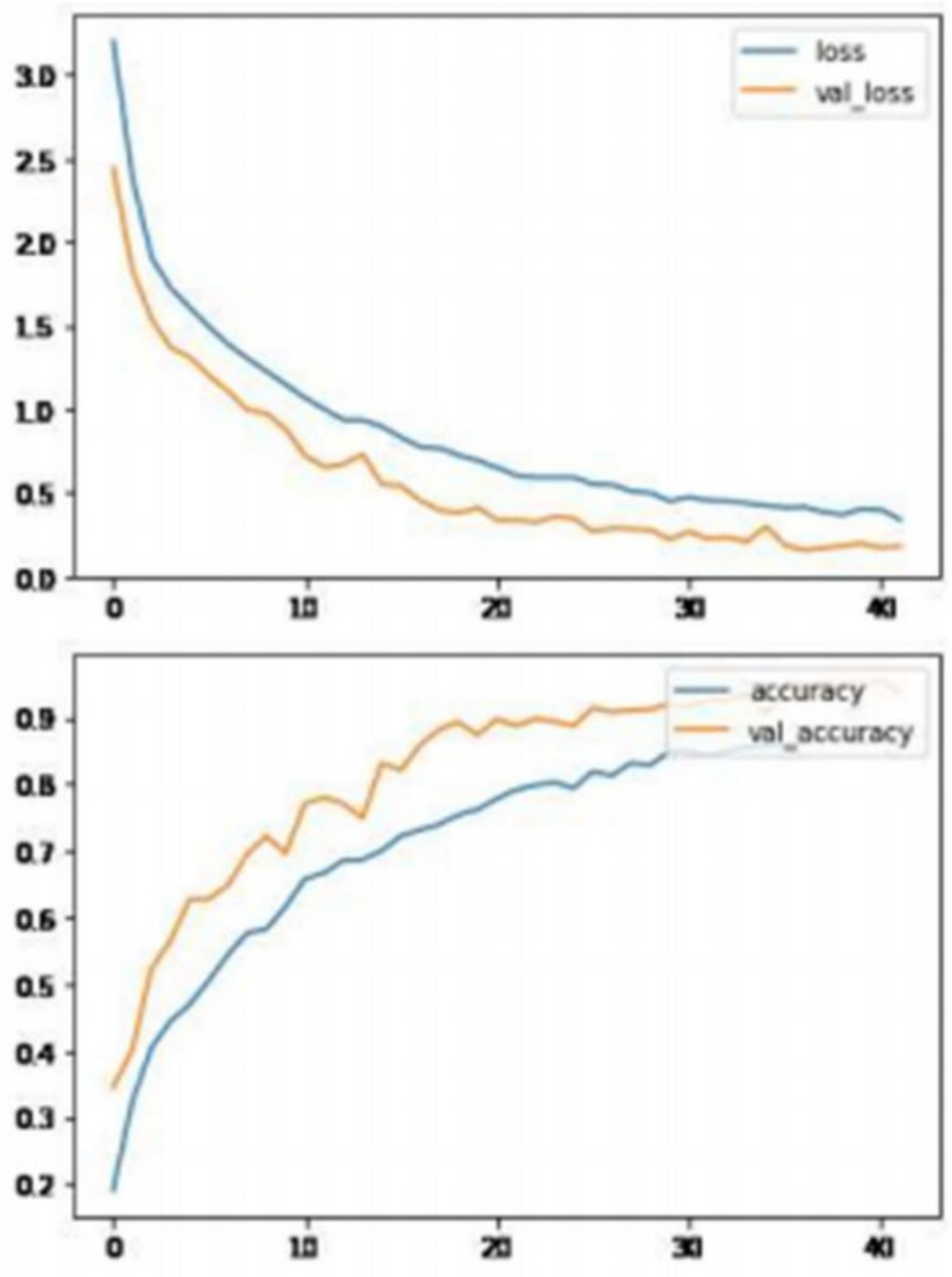
The above image shows the last epochs of the model.

Model Evaluation:

Let's visualize the training and validation accuracy with each epoch. PYTHON CODE:

```
# Loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'], loc='upper right')

# Accuracy
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'], loc='upper right')
```

Output :



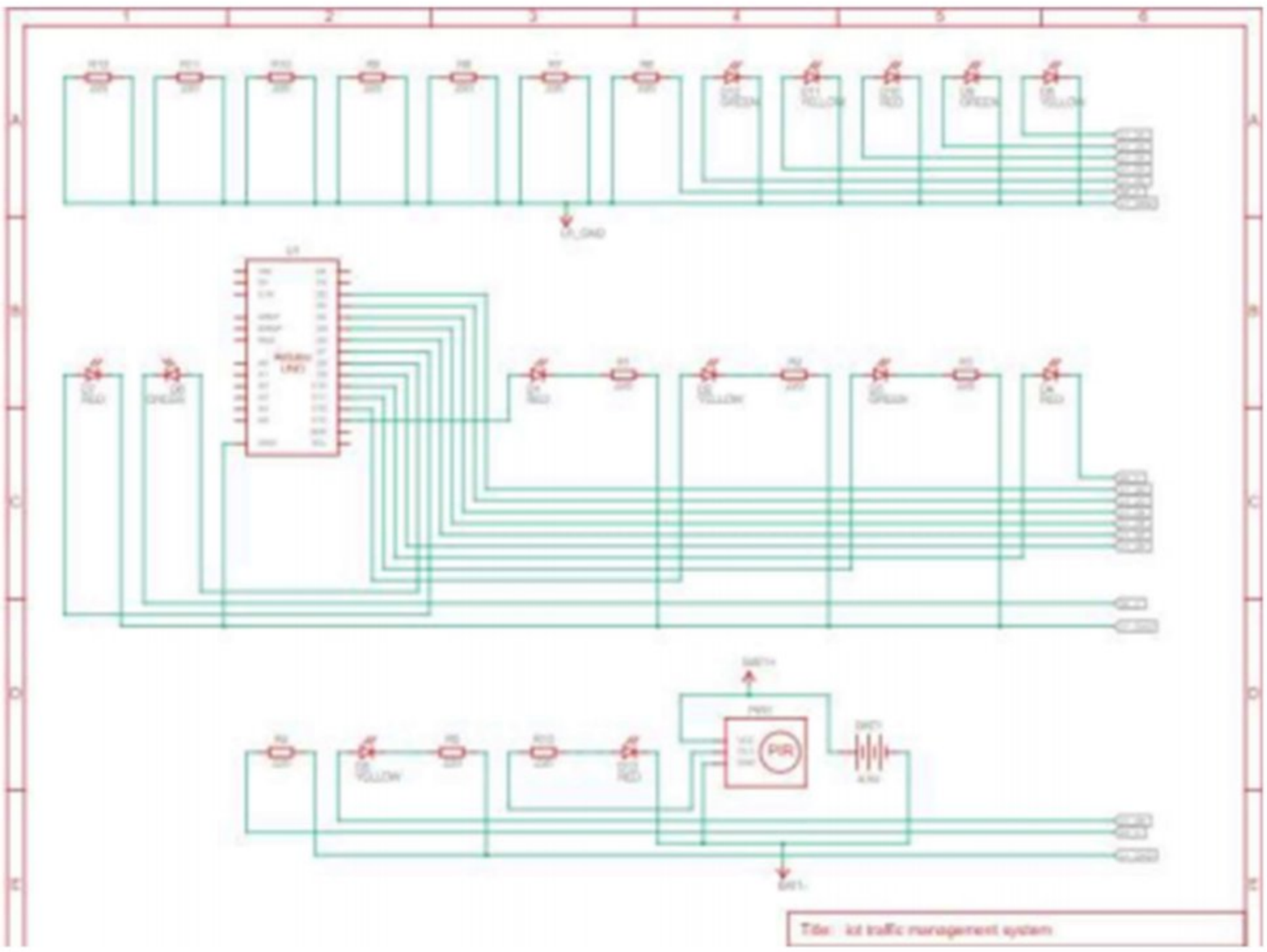
CNN model is performing very well with these layers. Further we can include more traffic and danger sign to warn the drivers.

DEVELOPMENT PART-2

Traffic management using the Internet of Things (IoT) involves the use of interconnected devices and sensors to collect and exchange data related to traffic flow, congestion, and other relevant parameters.

This data can then be analyzed and used to make informed decisions to optimize traffic flow, improve safety, and reduce congestion.

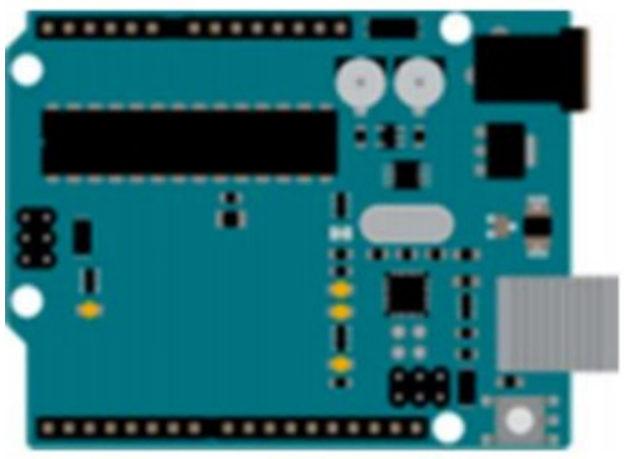
CIRCUIT DIAGRAM:



COMPONENTS REQUIRED:

	Name	Quantity	Component
2	U1	1	Arduino Uno R3
3	D1, D4, D7, D10, D13	5	Red LED
4	D2, D5, D8, D11	4	Yellow LED
5	D3, D6, D9, D12	4	Green LED
6	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13	13	220 Ω Resistor
7	PIR1	1	-46.204919661944814 , -215.21812432083706 , -261.40186756055743 PIR Sensor
8	Bat1	1	3 batteries, AA, no 1.5V Battery

AURDINO UNO:



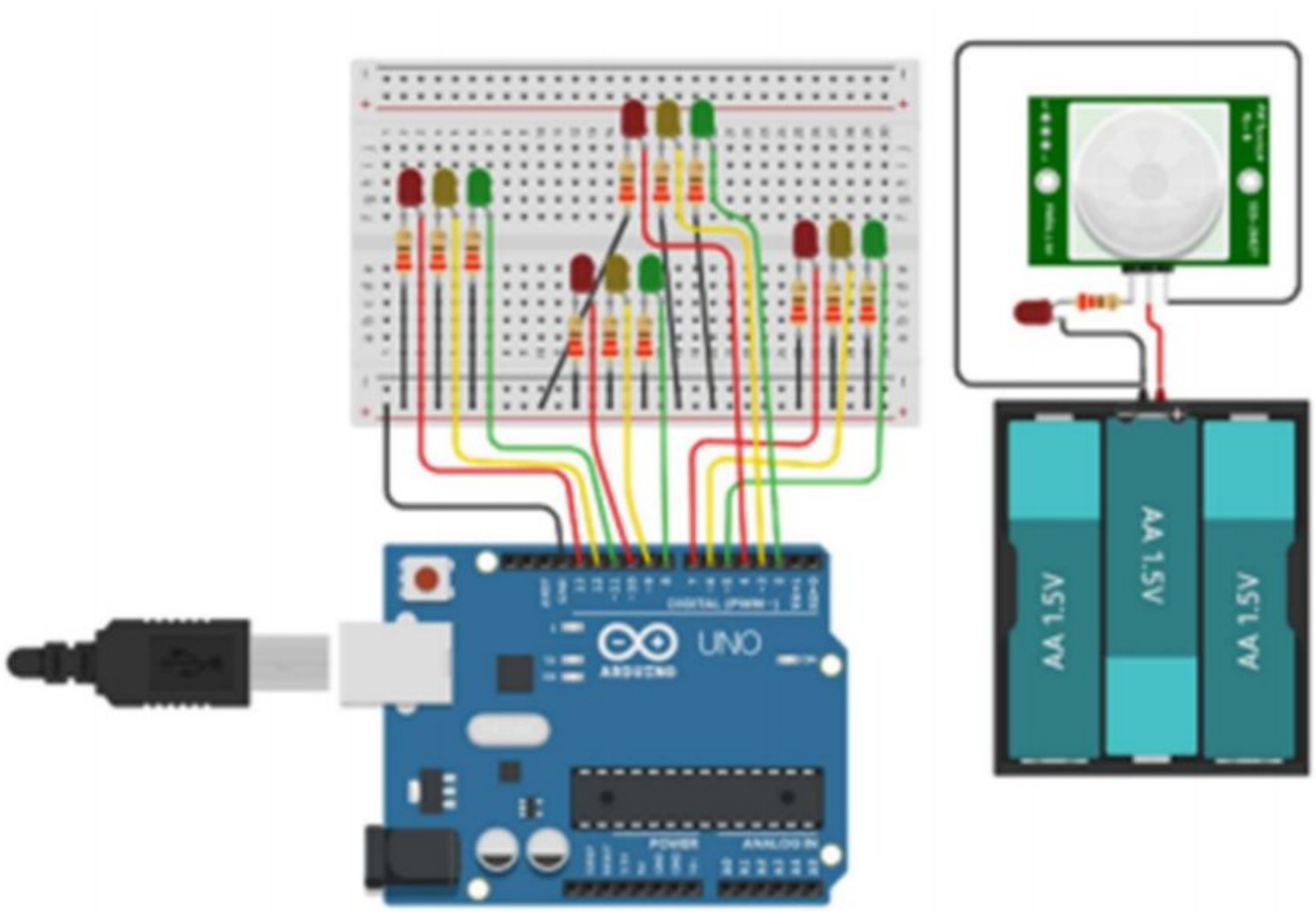
UNO R3

The UNO is the most used and documented board of The Arduino UNO is the best board to get started with electronics and coding. If this is your first experience tinkering with the platform, the UNO is the most robust board the whole Arduino family.

PIR SENSOR:

PIR motion sensors and ultrasonic sensors are placed on each section of the road to detect vehicles to set the duration of traffic lights based on road conditions. The period of the green light time can be determined based on the predetermined road conditions.

AURDINO USING TRAFFIC MANAGEMENT SYSTEM:



:**CODE:**

```
//First of all, we define the pins where we have //connected the LEDs.

int red_1=13; int yellow_1=12; int green_1=11; int

red_2=10; int yellow_2=9; int green_2=8;  int red_3=7;

int yellow_3=6; int green_3=5; int red_4=4; int

yellow_4=3; int green_4=2;


void direction_1_green(void)//green LED of direction 1 will turn ON

{digitalWrite(red_1,LOW); digitalWrite,

(yellow_1,LOW); digitalWrite,(green_1,HIGH);

digitalWrite(red_2,HIGH); digitalWrite,

(yellow_2,LOW); digitalWrite,(green_2,LOW);

digitalWrite(red_3,HIGH); digitalWrite,

(yellow_3,LOW); digitalWrite,(green_3,LOW);
```

digitalWrite(red_4,HIGH); digitalWrite,

(yellow_4,LOW); digitalWrite,(green_4,LOW); }

void direction_2_yellow(void)//yellow LED of direction 2 will turn ON

{digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);

digitalWrite(green_1,LOW);

digitalWrite(red_2,LOW);

digitalWrite(yellow_2,HIGH);

digitalWrite(green_2,LOW);

digitalWrite(red_3,HIGH);

digitalWrite(yellow_3,LOW);

digitalWrite(green_3,LOW);

digitalWrite(red_4,HIGH); digitalWrite(yellow_4,LOW);
digitalWrite(green_4,LOW); }

void direction_2_green(void)//green LED of

direction 2 will turn ON {digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);

digitalWrite(green_1,LOW);

digitalWrite(red_2,LOW);

```
digitalWrite(yellow_2,LOW);

digitalWrite(green_2,HIGH);

digitalWrite(red_3,HIGH);

digitalWrite(yellow_3,LOW);

digitalWrite(green_3,LOW);

digitalWrite(red_4,HIGH);

digitalWrite(yellow_4,LOW);

digitalWrite(green_4,LOW); }


void direction_3_yellow(void)//yellow LED of

direction 3 will turn ON  {digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);

digitalWrite(green_1,LOW);

digitalWrite(red_2,HIGH);

digitalWrite(yellow_2,LOW);

digitalWrite(green_2,LOW);

digitalWrite(red_3,LOW);

digitalWrite(yellow_3,HIGH);

digitalWrite(green_3,LOW);

digitalWrite(red_4,HIGH);
```

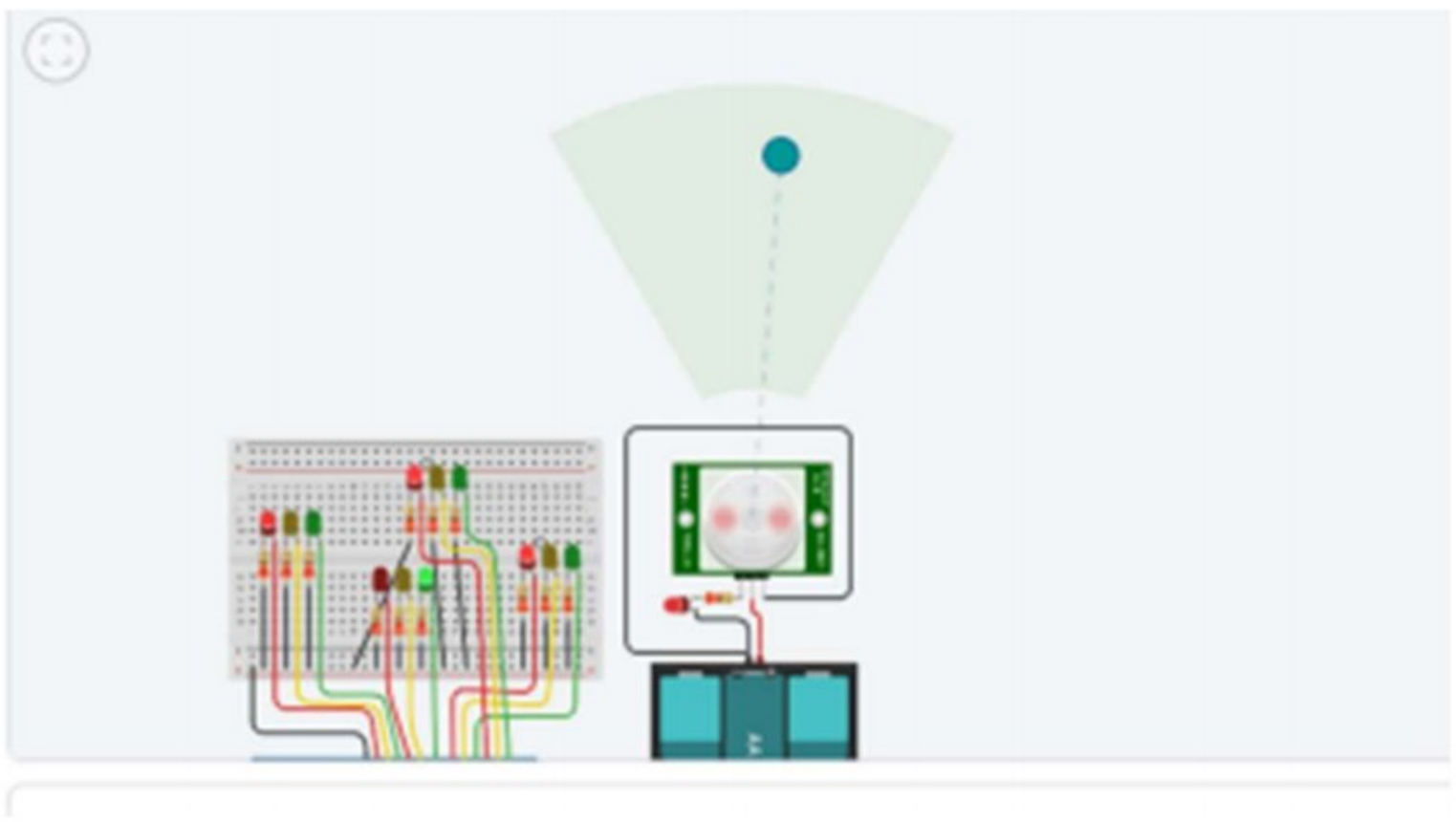
```
digitalWrite(yellow_4,LOW);

digitalWrite(green_4,LOW); }


void direction_3_green(void)//green LED of

direction 3 will turn ON  {digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);

digitalWrite(green_1,LOW);

digitalWrite(red_2,HIGH);

digitalWrite(yellow_2,LOW);

digitalWrite(green_2,LOW);

digitalWrite(red_3,LOW);

digitalWrite(yellow_3,LOW);

digitalWrite(green_3,HIGH);

digitalWrite(red_4,HIGH);

digitalWrite(yellow_4,LOW);

digitalWrite(green_4,LOW); }


void direction_4_yellow(void)//yellow LED of

direction 4 will turn ON  {digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);
```

```
digitalWrite(green_1,LOW);

digitalWrite(red_2,HIGH);

digitalWrite(yellow_2,LOW);

digitalWrite(green_2,LOW);

digitalWrite(red_3,HIGH);

digitalWrite(yellow_3,LOW);

digitalWrite(green_3,LOW);

digitalWrite(red_4,LOW);

digitalWrite(yellow_4,HIGH);

digitalWrite(green_4,LOW); }


void direction_4_green(void)//green LED of

direction 4 will turn ON

{digitalWrite(red_1,HIGH);

digitalWrite(yellow_1,LOW);

digitalWrite(green_1,LOW);

digitalWrite(red_2,HIGH);

digitalWrite(yellow_2,LOW);

digitalWrite(green_2,LOW);

digitalWrite(red_3,HIGH);
```

```
digitalWrite(yellow_3,LOW);

digitalWrite(green_3,LOW);

digitalWrite(red_4,LOW);

digitalWrite(yellow_4,LOW);

digitalWrite(green_4,HIGH); }

void direction_1_yellow(void)//yellow LED of direction 1 will turn ON

{digitalWrite(red_1,LOW);

digitalWrite(yellow_1,HIGH);

digitalWrite(green_1,LOW);

digitalWrite(red_2,HIGH);

digitalWrite(yellow_2,LOW);

digitalWrite(green_2,LOW);

digitalWrite(red_3,HIGH);

digitalWrite(yellow_3,LOW);

digitalWrite(green_3,LOW);

digitalWrite(red_4,HIGH);

digitalWrite(yellow_4,LOW);

digitalWrite(green_4,LOW); }  void setup() { //

Declaring all the LED's as output
```

```
for(int i=2;i<=13;i++) pinMode(i,OUTPUT);
} void loop() //In the loop function, we controlled the signal
one // by one to control the flow of traffic.
{direction_1_green(); delay(5000); direction_2_yellow();
delay(3000);
direction_2_green(); delay(5000); direction_3_yellow(); delay(3000);
direction_3_green(); delay(5000); direction_4_yellow(); delay(3000);
direction_4_green(); delay(5000); direction_1_yellow(); delay(3000); }
```

OUTPUT:



CONCLUSION:

By employing IoT in traffic management, cities can achieve more efficient traffic flow, reduce congestion, lower emissions, and enhance overall safety for both drivers and pedestrians. It's essential to consider privacy and security measures when implementing such systems to protect sensitive data and ensure the integrity of the traffic management