PROJECT TITLE

A MINOR PROJECT REPORT
ROAD VISION
Submitted by

SK.MD.Hussain Basha[RA2011026010070]
VAIBHOV SHARMA [RA2011026010106]
R. Parkavi [RA2011026010119]

Under the guidance of

DR.M.S. Abirami

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE & ENGINEERING
of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M.Nagar, Kattankulathur, Chengalpattu District

JUNE 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Course Project Report titled **"TRAFFIC SIGN CLASSIFICATION"** is the bonafide work done by **SK.MD. Hussain Basha (RA2011026010070)** , **Vaibhov Sharma (RA2011026010106)** , **R.Parkavi (RA2011026010119)** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

[illegible] 11/5/23

SIGNATURE
Faculty In-Charge
DR.M.S.ABRAMI
Assistant Professor
Department of computational intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

[illegible]

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head ,
Department of Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

ABSTRACT

This project aims to develop a machine learning model that can accurately identify traffic signs in real-world images. The model utilizes a convolutional neural network (CNN) architecture to extract features from the input images and classify them into different traffic sign categories. The dataset used for training and testing the model is the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which consists of more than 50,000 labeled images of 43 different traffic sign classes. The performance of the model is evaluated based on its accuracy in correctly classifying new, unseen images. The results show that the proposed model achieves high accuracy in traffic sign classification and has the potential to be deployed in real-world applications for improving road safety.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AES	Advanced Encryption Standard
ANN	Artificial Neural Network
CSS	Cascading Style Sheet
CV	Computer Vision
DB	Data Base
DNA	Deoxyribo Neucleic Acid
SQL	Structured Query Language
SVM	Support Vector Machine
UI	User Interface

CHAPTER 1

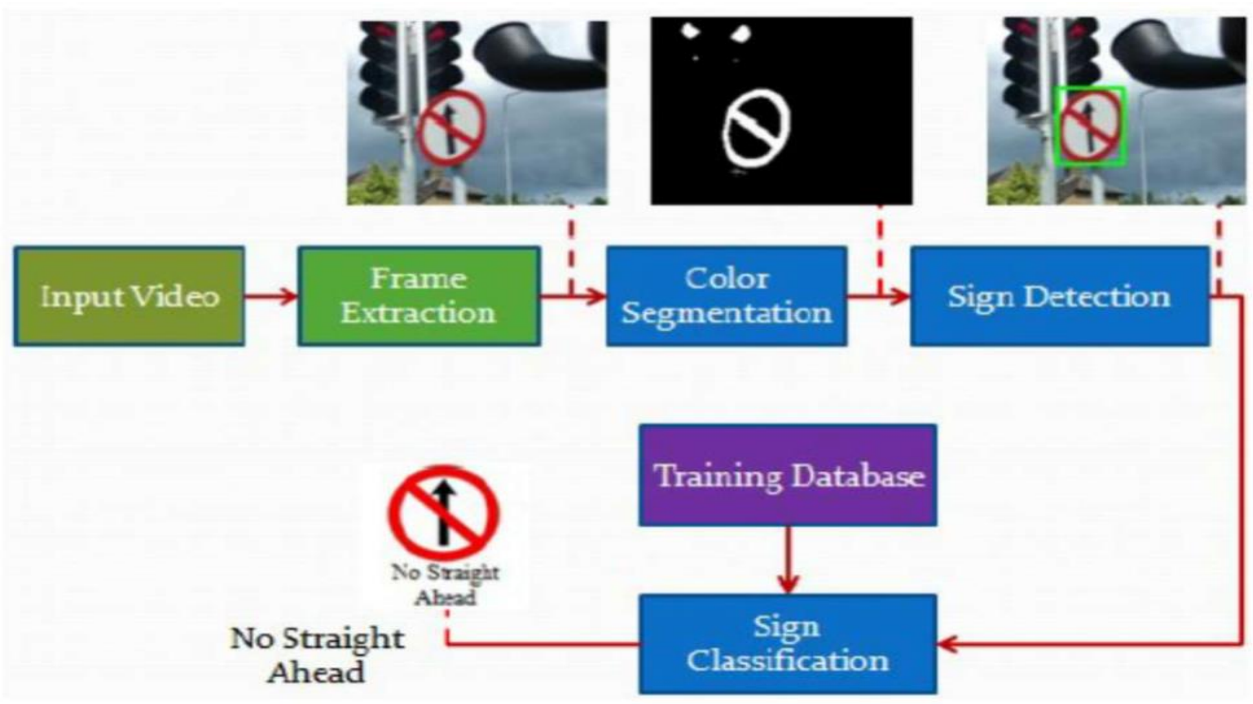
INTRODUCTION

Traffic sign classification using machine learning is a project that aims to automatically identify the traffic signs on the road, especially in real-time scenarios. The project involves training a machine learning model with a dataset of various traffic signs such as stop signs, speed limit signs, no entry signs, etc., to enable it to recognize and classify new traffic signs that are encountered.

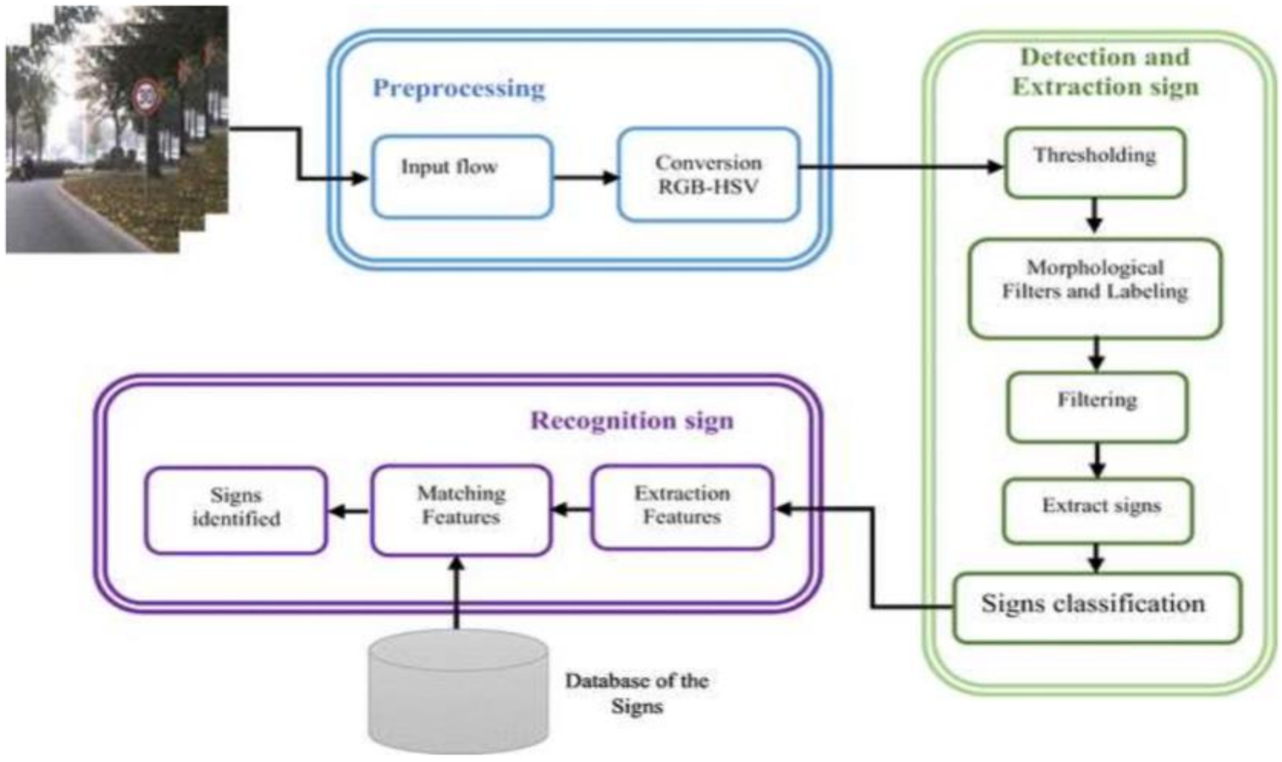
The main goal of this project is to increase road safety by automating the process of identifying traffic signs. By utilizing computer vision techniques and machine learning algorithms, we can develop an intelligent system for detecting and classifying different types of traffic signs accurately.

This project has practical applications in autonomous driving technology, where self-driving cars need to recognize and interpret traffic signs efficiently. Additionally, it can also be used in advanced driver assistance systems (ADAS) to alert drivers when they miss important traffic signs.

Overall, this project is an excellent opportunity to learn about computer vision, machine learning, and their real-world applications in creating safer roads for everyone.



Input Video
Frame Extraction
Color Segmentation
Sign Detection
Training Database
No Straight Ahead
Sign Classification
No Straight Ahead



Preprocessing
Input flow
Conversion RGB-HSV
Detection and Extraction sign
Thresholding
Morphological Filters and Labeling
Filtering
Extract signs
Signs classification
Recognition sign
Signs identified
Matching Features
Extraction Features
Database of the Signs

LITERATURE SURVEY

Overview of Traffic Sign Classification: The literature survey should begin with an overview of traffic sign classification, its importance, and its applications. This section should include information about the various types of traffic signs, the challenges in their detection and recognition, and the potential benefits of using machine learning for classification.

Traditional Approaches for Traffic Sign Detection and Recognition: The literature survey should also cover traditional approaches for traffic sign detection and recognition, such as template matching, feature-based methods, and hybrid approaches. This section should provid insights into the limitations of these methods and the need for more advancedtechniques.

Machine Learning Techniques: The literature survey should extensively cover machine learning techniques applied to traffic sign classification tasks. Supervised, unsupervised, and reinforcement learning approaches can be discussed, along with their strengths and weaknesses.

Deep Learning Techniques and Architectures: Given the current state-of-the-art in traffic sign classification, the literature survey should focus on deep learning techniques such as Convolutional Neural Networks (CNNs) and architectures like YOLO (You Only Look Once), SSD (Single Shot Detector), Faster R-CNN, and others.

Datasets for Traffic Sign Classification: The literature survey should highlight relevant datasets used for training and evaluating traffic sign classification models. Popular datasets such as German Traffic Sign Recognition Benchmark (GTSRB), Belgium Traffic Sig Benchmark (BTSC), and others can be discussed.

Evaluation Metrics for Traffic Sign Classification: The literature survey should also cover evaluation metrics used to assess the performance of traffic sign classification models,including accuracy, precision, recall, FI score, and others.

By conducting a comprehensive literature survey, you can gain insights into the latest techniques, identify research gaps, and develop a better understanding of the scope of the project.

SYSTEM ARCHITECTURE AND DESIGN

1) Python: Python is a popular programming language for machine learning and deep learning tasks, and many machine learning libraries are available in Python.

2)Machine Learning Libraries: There are many machine learning libraries available in Python such as Scikit-learn, Tensor Flow, Keras, PyTorch which can be used for building machine learning models.

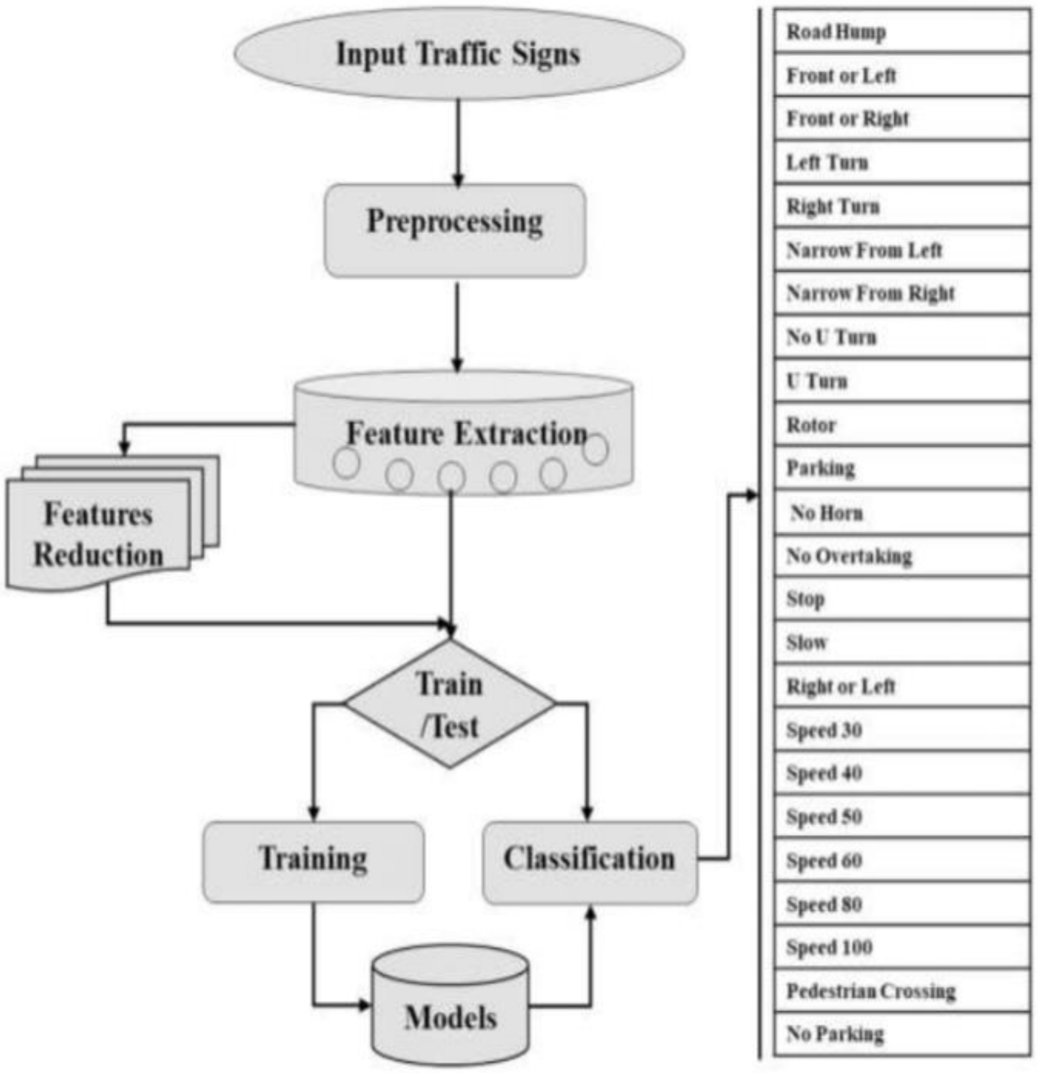
3)0penCV: OpenCV is an open-source computer vision library that can be used for image processing tasks such as pre-processing images before training a model or detecting traffic signs in real-time.

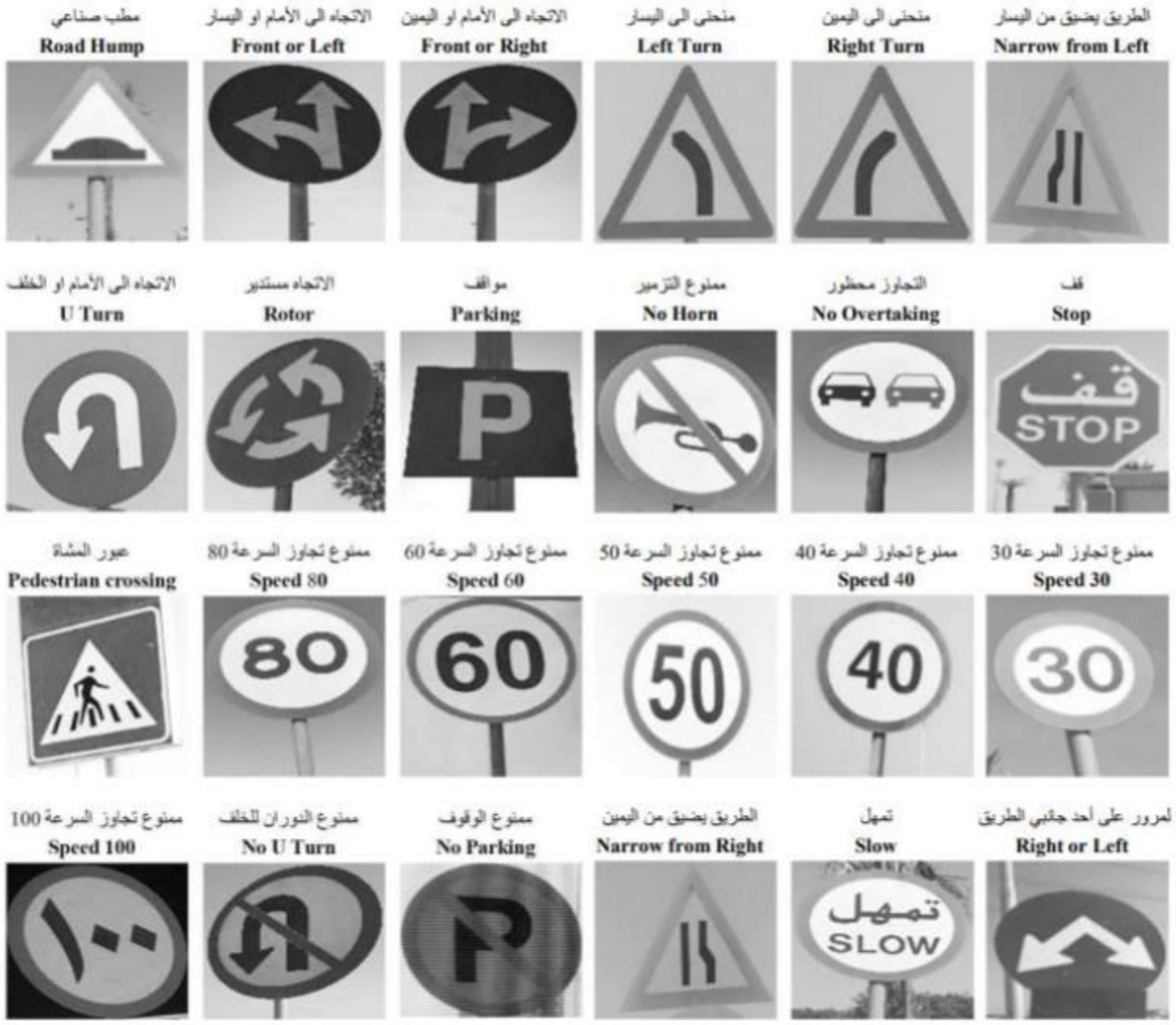
4)Jupter Notebook: Jupter Notebook is an interactive notebook environment that allows you to run code and visualize data. It is often used for prototyping and testing machine learning models.

5)Git: Git is a version control system that allows you to track changes to your code and collaborate with others on a project.

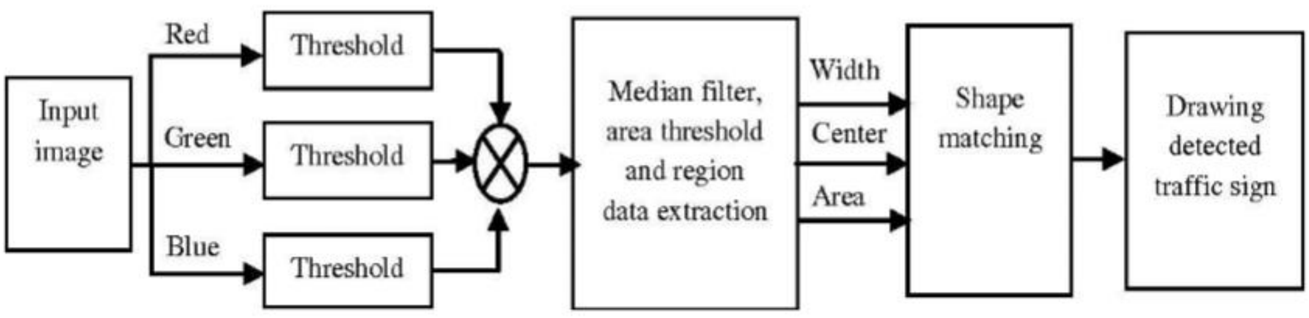
6)IDEs: Integrated Development Environments (IDEs) like PyCharm, Visual Studio Code, or Spyder can help streamline development tasks such as debugging, code completion, and running tests.

7)Operating System: The choice of operating system will depend on the specific machine learning library you choose and personal preference. Most libraries are compatible with Windows, Linux, or macOS.


Input Traffic Signs
Preprocessing
Feature Extraction
Features Reduction
Train /Test
Training
Classification
Models
Road Hump
Front or Left
Front or Right
Left Turn
Right Turn
Narrow From Left
Narrow From Right
No U Turn
U Turn
Rotor
Parking
No Horn
No Overtaking
Stop
Slow
Right or Left
Speed 30
Speed 40
Speed 50
Speed 60
Speed 80
Speed 100
Pedestrian Crossing
No Parking



مطب صناعي
Road Hump
الاتجاه الى الأمام او اليسار
Front or Left
الاتجاه الى الأمام او اليمين
Front or Right
منحنى الى اليسار
Left Turn
منحنى الى اليمين
Right Turn
الطريق يضيق من اليسار
Narrow from Left
الاتجاه الى الأمام او الخلف
U Turn
الاتجاه مستدير
Rotor
مواقف
Parking
ممنوع التزمير
No Horn
التجاوز محظور
No Overtaking
قف
Stop
عبور المشاة
Pedestrian crossing
ممنوع تجاوز السرعة 80
Speed 80
ممنوع تجاوز السرعة 60
Speed 60
ممنوع تجاوز السرعة 50
Speed 50
ممنوع تجاوز السرعة 40
Speed 40
ممنوع تجاوز السرعة 30
Speed 30
ممنوع تجاوز السرعة 100
Speed 100
ممنوع الدوران للخلف
No U Turn
ممنوع الوقوف
No Parking
الطريق يضيق من اليمين
Narrow from Right
تمهل
Slow
لمرور على أحد جانبي الطريق
Right or Left



Input image
Red
Green
Blue
Threshold
Threshold
Threshold
Median filter, area threshold and region data extraction
Width
Center
Area
Shape matching
Drawing detected traffic sign

MODULES

Data acquisition: This involves collecting images of traffic signs from various sources. The images can be collected using cameras or downloaded from online datasets.

Data pre-processing: This involves cleaning and preparing the data for use in the machine learning algorithms. The images may need to be resized, cropped, and normalized to ensure consistency and reduce noise.

Feature extraction: This involves extracting relevant features from the images that can be used to train the machine learning models. These features can include colour, shape, and texture.

Model training: This involves selecting an appropriate machine learning algorithm and training it on the extracted features. Common algorithms used in traffic sign classification include convolutional neural networks (CNNs) and support vector machines (SVMs).

Model evaluation: This involves testing the trained model on a separate dataset to evaluate its performance. Metrics such as accuracy, precision, and recall can be used to assess the performance of the model.

Deployment: Once a satisfactory model has been developed, it can be deployed in real-world applications such as autonomous vehicles or traffic control systems.

METHODOLOGY

Data collection: Collect a dataset of traffic sign images from various sources, including cameras and online datasets.

Data pre-processing: Pre-process the data by resizing, cropping, and normalizing the images to ensure consistency and reduce noise.

Feature extraction: Extract relevant features from the images that can be used to train the machine learing model. These features can include color, shape, and texture.

Model selection: Select an appropriate machine learning algorithm and architecture for the task, Convolutional neural networks (CNNs) are commonly used for image classification tasks.

Model training: Train the selected machine learning model on the extracted features. Divide the dataset into training, validation, and testing sets to evaluate the performance of the model.

Model evaluation: Evaluate the performance of the trained model on the testing set using metrics such as accuracy, precision, and recall.
Hyperparameter tuning: Optimize the hyperparameters of the model to improve its performance.

Deployment: Deploy the trained model in real-world applications, such as autonomous vehicles or traffic control systems.

It's important to note that this is a high-level methodology and the specific steps may vary depending on the particular project and dataset. However, these steps provide a general framework for developing a traffic sign classification system using machine learning

CODE AND TESTING

```
pip install tensorflow
pip install tensorflow keras
pip install tensorflow sklearn
pip install tensorflow matplotlib
pip install tensorflow pandas
pip install tensorflow pil
```

Dataset exploration
Around 43 subfolders(ranging from 0 to 42) are available in our ‘train’ folder, and each subfolder represents a different class. We have an OS module that helps in the iteration of all the images with their respective classes and labels. To open the contents of ideas into an array, we are using the PIL library.

```
import numpy as np
import pandas as pd
 import matplotlib.pyplot as plt
 import tensorflow as tf
from PIL import Image
 import os
from sklearn.model_selection
import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
data = []
labels = []
classes = 43
cur_path = os.getcwd()
for i in range(classes):
  path = os. path.join(cur_path,'train', str(i))
  images = os.listdir(path)
  for a in images:
     try:
     image = Image.open(path + '\'+ a)
     image = image.resize((30,30))
     image = np.array(image)
     data.append(image)
     labels.append(i)
   except:
print("Error loading image")
data = np.array(data)
labels = np.array(labels)
```

In the end, we have to store every image with its corresponding labels into lists. A NumPy array is needed to feed the data to the model, so we convert this list into an array. Now, the shape of our data is (39209, 30, 30, 3), where 39209 represents the number of images, 30*30 represents the image sizes into pixels, and the last 3 represents the RGB value(availability of coloured data).

```
print(data.shape, labels.shape)
#Splitting training and testing dataset
X_t1, X_t2, y_t1, y_t2 = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_t1.shape, X_t2.shape, y_t1.shape, y_t2.shape)
#Converting the labels into one hot encoding
y_t1 = to_categorical(y_t1, 43)
y_t2 = to_categorical(y_t2, 43)

#Building the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

eps = 15
anc = model.fit(X_t1, y_t1, batch_size=32, epochs=eps, validation_data=(X_t2, y_t2))

#testing accuracy on test dataset
from sklearn.metrics import accuracy_score
y_test = pd.read_csv('Test.csv')
labels = y_test["ClassId"].values
imgs = y_test["Path"].values
data=[]
for img in imgs:
   image = Image.open(img)
   image = image.resize((30,30))
```

```
data.append(np.array(image))
X_test=np.array(data)
pred = model.predict_classes(X_test)
#Accuracy with the test data
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))
```

Output

```
0.9532066508313539
```

```
model.save('traffic_classifier.h5')
```

SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection
import train_test_split
from keras.utils
import to_categorical
from keras.models
import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten,
Dropout
data = []
labels = []
classes = 43
cur_path = os.getcwd()
for i in range(classes):
   path = os.path.join(cur_path,'train',str(i))
   images = os.listdir(path)
for a in images:
try:
           image = Image.open(path + '\'+ a)
           image = image.resize((30,30))
           image = np.array(image)
           data.append(image)
           labels.append(i)
except:
           print("Error loading image")
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
X_t1,X_t2,y_t1,y_t2=train_test_split(data,labels,test_
size=0.2, random_state=42)
print(X_t1.shape, X_t2.shape, y_t1.shape, y_t2.shape)

y_t1 = to_categorical(y_t1, 43)
y_t2 = to_categorical(y_t2, 43)
model = Sequential()
```

```
model.add(Conv2D(filters=32,kernel_size=(5,5),
activation='relu', input_ shape=X_ train. shape[1:]))

model.add(Conv2D(filters=32,kernel_size=(5,5),
activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64,kernel_size=(3,3),
activation='relu'))
model.add(Conv2D(filters=64,kernel_size=(3,3),
activation='relu'))


model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
#Compilation of the model
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
eps = 15
anc= model.fit(X_t1,y_t1,batch_size=32,epochs=eps, validation_
data=(X_t2, y_t2))
model.save("my_model.h5")
plt.figure(0)
plt.plot(anc.anc['accuracy'], label='training accuracy')
plt. plot(anc.anc['val_ accuracy'], label='val accuracy')
plt. title('Accuracy')
plt. xlabel('epochs')
plt. ylabel('accuracy')
plt. legend()
plt. show()
plt. figure(1)

plt. plot(history. history['loss'], label='training loss')
plt. plot(history. history['val_ loss'], label='val loss')
plt. title('Loss')
plt. xlabel('epochs')
plt. ylabel('loss')
plt. legend()
plt. show()
```

```
from sk learn. metrics
import accuracy _score
y_ test = pd. Read _csv('Test.csv')
labels = y_ test["Class  Id"].values
imgs = y_ test["Path"].values
data=[]
for img in imgs:
   image = Image. open(img)
   image = image. resize((30,30))
   data. append(np. array(image))
X_ test= np. array(data)

pred = model.predict_classes(X_test)
from sklearn.metrics
import accuracy_score

print(accuracy_score(labels, pred))
model.save(‘traffic_classifier.h5’)
```

REFERENCES

[1] Hernández, E.; Sanchez-Anguix, V.; Julian, V.; Palanca, J.; Duque, N. Rainfall prediction: A deep learningapproach. In International Conference on Hybrid Artificial Intelligence Systems; Springer: Cham, Switzerland,2016; pp. 151–162.

[2] Goswami, B.N. The challenge of weather prediction. Resonance **1996**, 1, 8–17.

[3] Nayak, D.R.; Mahapatra, A.; Mishra, P. A survey on rainfall prediction using artificial neural network. Int. J.Comput. Appl. **2013**, 72, 16.

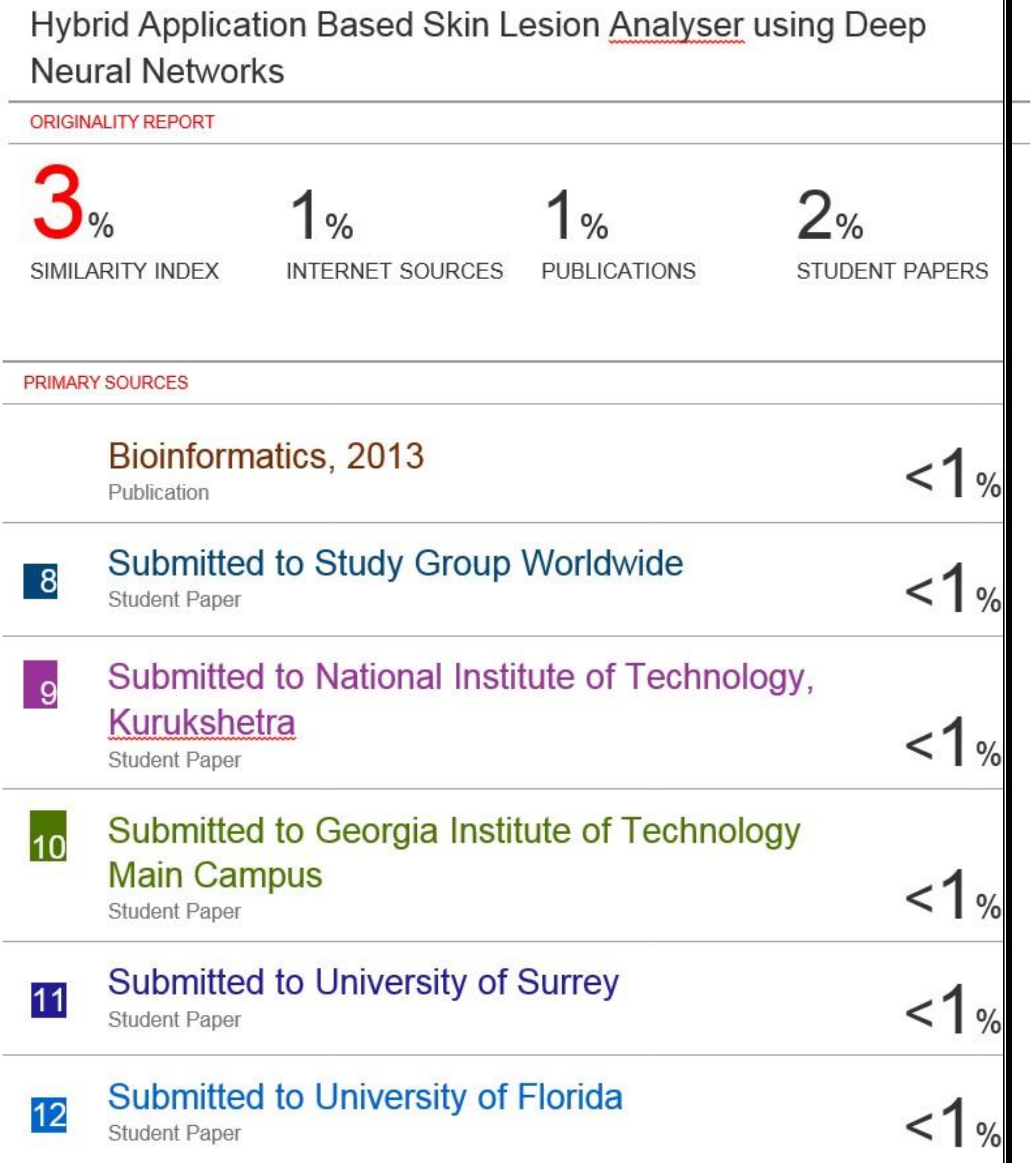
[4] Kashiwao, T.; Nakayama, K.; Ando, S.; Ikeda, K.; Lee, M.; Bahadori, A. A neural networkbased localrainfall prediction system using meteorological data on the internet: A case study using data from the Japanmeteorological agency. Appl. Soft Comput. **2017**, 56, 317–330.

[5] Mislan, H.; Hardwinarto, S.; Sumaryono, M.A. Rainfall monthly prediction based on artificial neural network:A case study in Tenggarong Station, East Kalimantan, Indonesia. Procedia Comput. Sci. **2015**, 59, 142–151.

[6] Muka, Z.; Maraj, E.; Kuka, S. Rainfall prediction using fuzzy logic. Int. J. Innov. Sci. Eng. Technol. **2017**, 4, 1–5.

APPENDIX C

PLAGIARISM REPORT



Hybrid Application Based Skin Lesion Analyser using Deep Neural Networks

ORIGINALITY REPORT

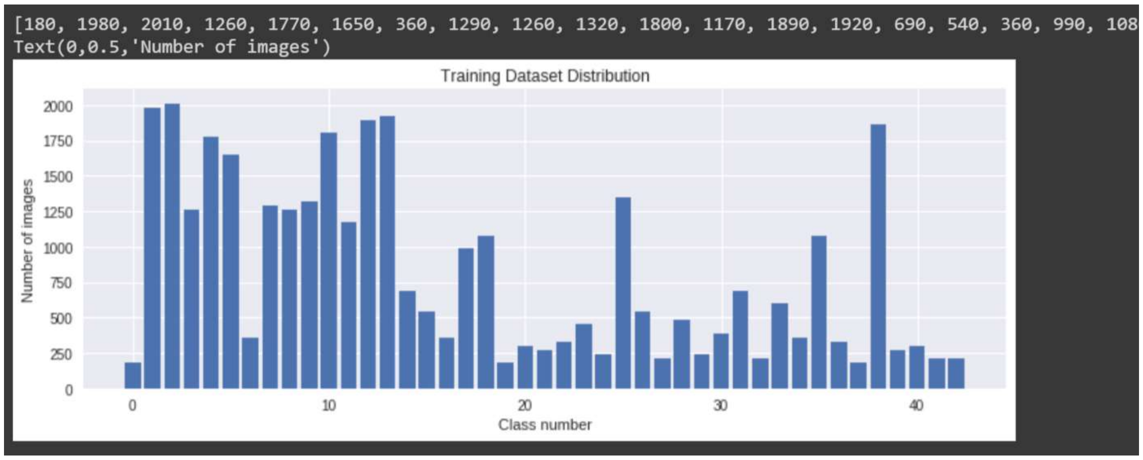
3%	1%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

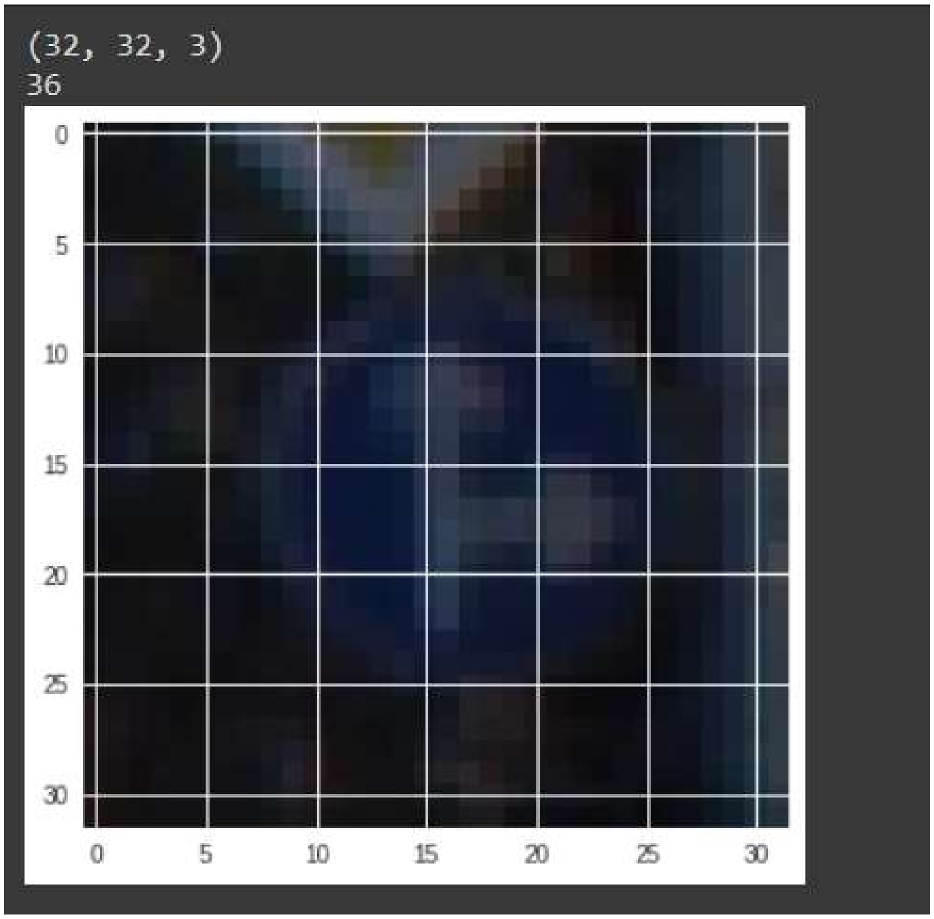
	Bioinformatics, 2013 Publication	<1%
8	Submitted to Study Group Worldwide Student Paper	<1%
9	Submitted to National Institute of Technology, Kurukshetra Student Paper	<1%
10	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1%
11	Submitted to University of Surrey Student Paper	<1%
12	Submitted to University of Florida Student Paper	<1%

RESULTS AND DISCUSSIONS

num_of_samples



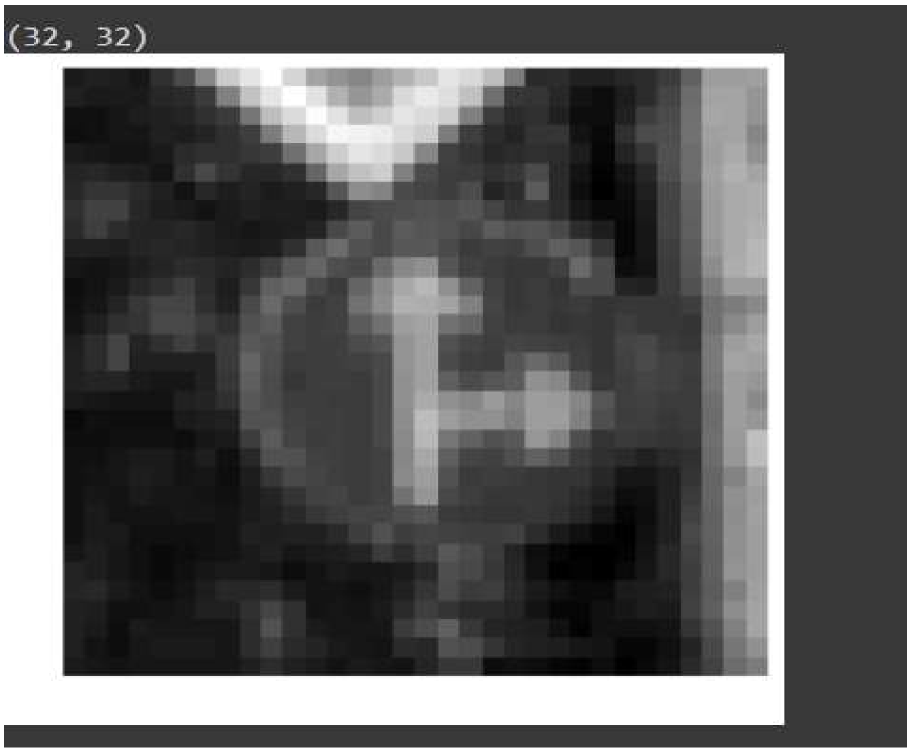
```
[180, 1980, 2010, 1260, 1770, 1650, 360, 1290, 1260, 1320, 1800, 1170, 1890, 1920, 690, 540, 360, 990, 108
Text(0,0.5,'Number of images')
```



```
(32, 32, 3)
36
```

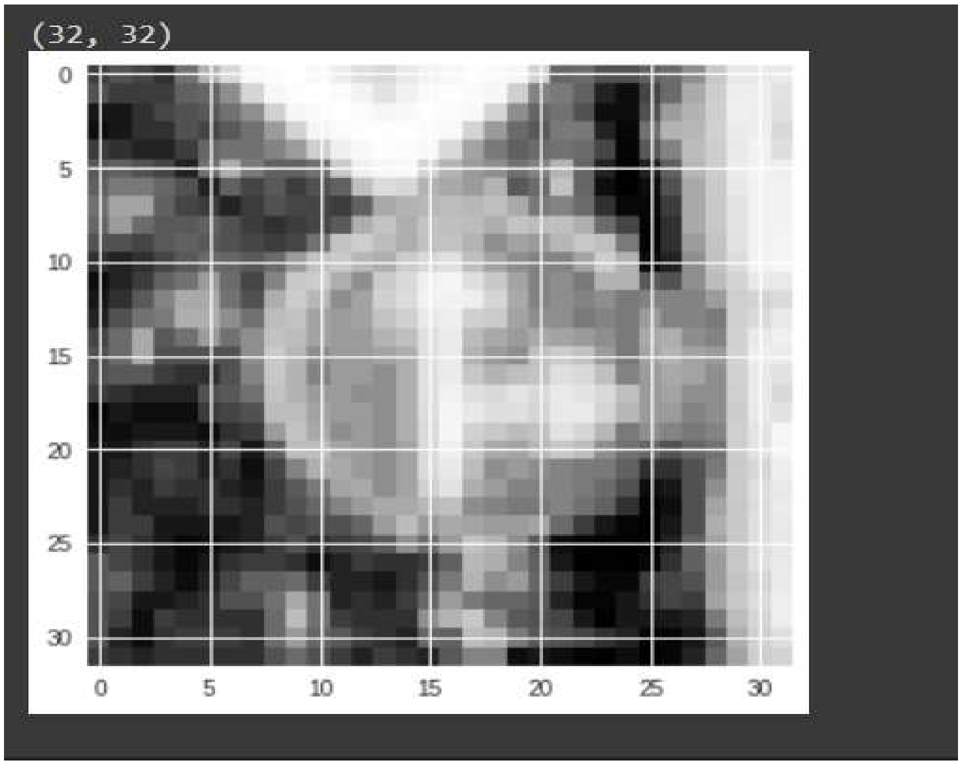
RESULTS AND DISCUSSIONS

GREY SCALE



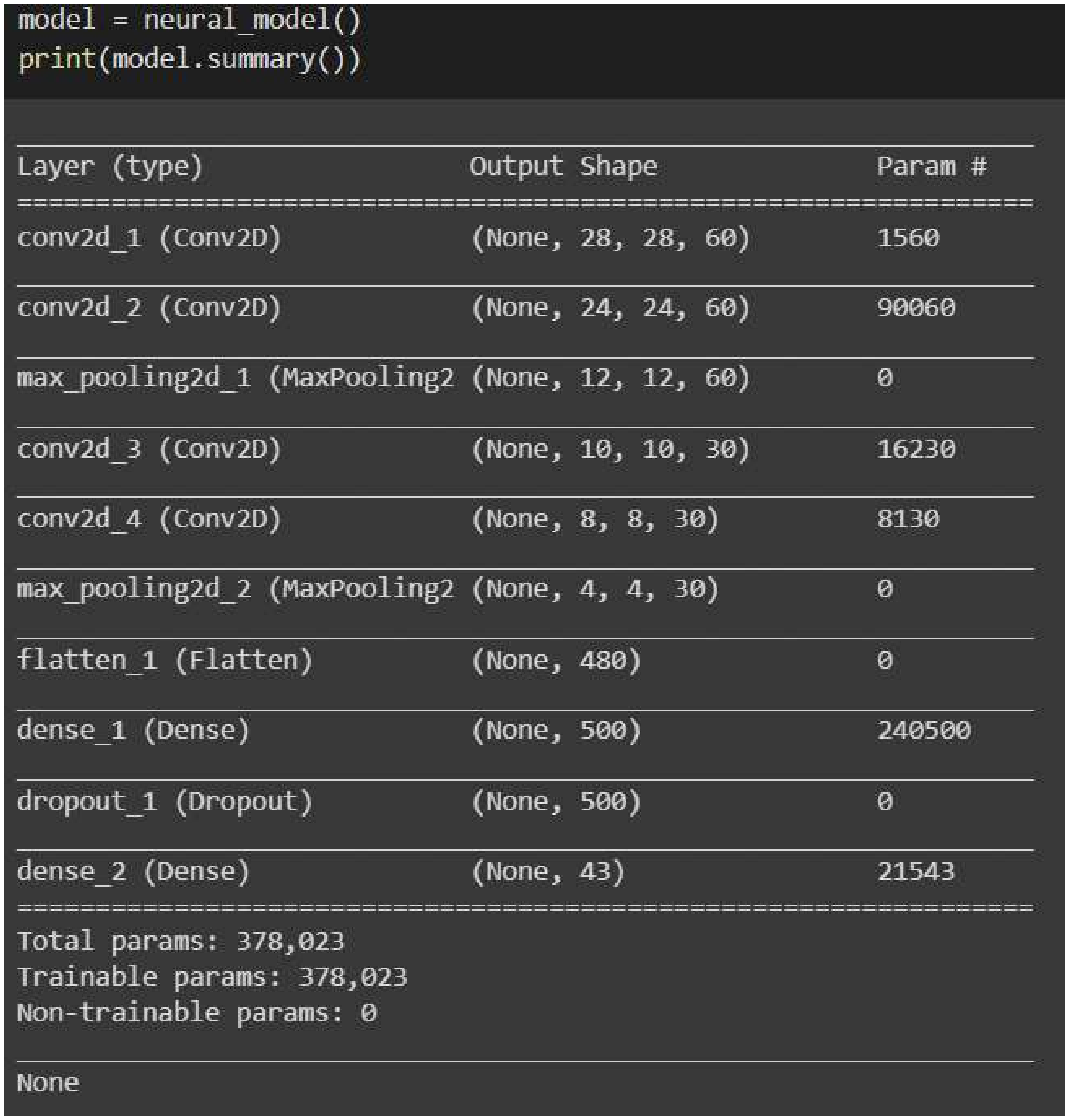
```
(32, 32)
```

NORMALIZE



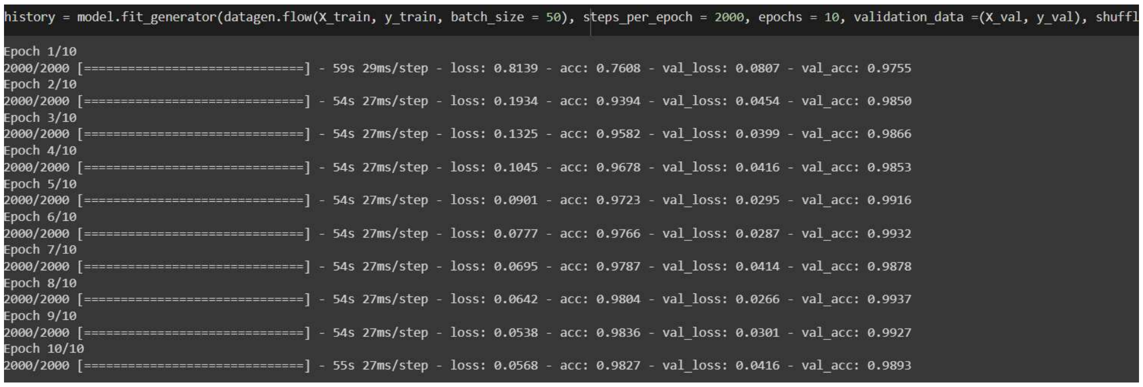
```
(32, 32)
```

RESULTS AND DISCUSSIONS



```
model = neural_model()
print(model.summary())
```

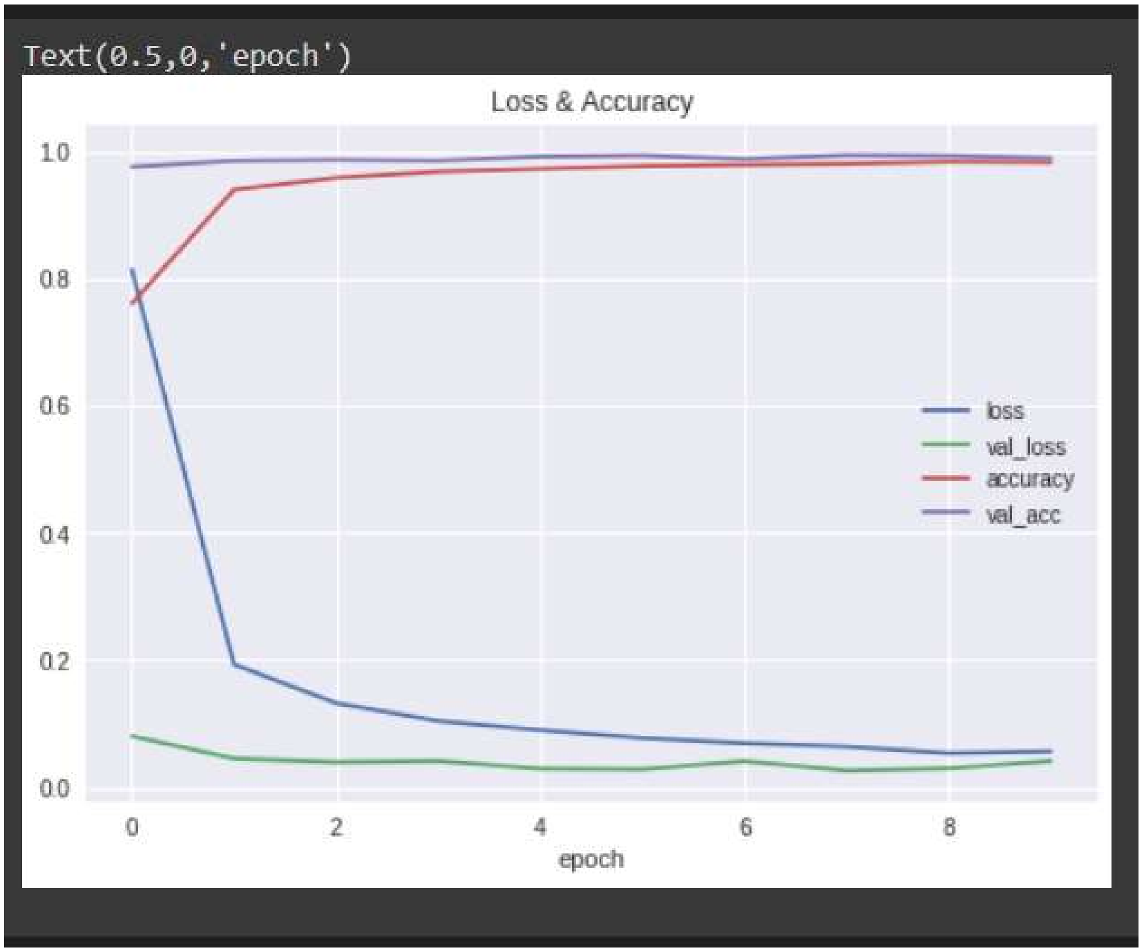
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 60)        1560      
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 60)        90060     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 60)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 30)        16230     
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 30)          8130      
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 30)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 500)               240500    
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 43)                21543     
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
_________________________________________________________________
None
```



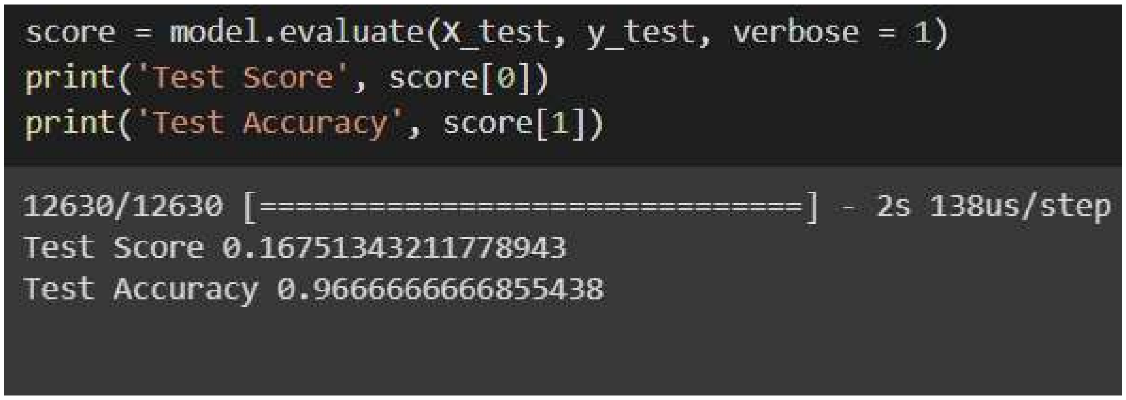
```
history = model.fit_generator(datagen.flow(X_train, y_train, batch_size = 50), steps_per_epoch = 2000, epochs = 10, validation_data =(X_val, y_val), shuffl

Epoch 1/10
2000/2000 [==============================] - 59s 29ms/step - loss: 0.8139 - acc: 0.7608 - val_loss: 0.0807 - val_acc: 0.9755
Epoch 2/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.1934 - acc: 0.9394 - val_loss: 0.0454 - val_acc: 0.9850
Epoch 3/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.1325 - acc: 0.9582 - val_loss: 0.0399 - val_acc: 0.9866
Epoch 4/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.1045 - acc: 0.9678 - val_loss: 0.0416 - val_acc: 0.9853
Epoch 5/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.0901 - acc: 0.9723 - val_loss: 0.0295 - val_acc: 0.9916
Epoch 6/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.0777 - acc: 0.9766 - val_loss: 0.0287 - val_acc: 0.9932
Epoch 7/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.0695 - acc: 0.9787 - val_loss: 0.0414 - val_acc: 0.9878
Epoch 8/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.0642 - acc: 0.9804 - val_loss: 0.0266 - val_acc: 0.9937
Epoch 9/10
2000/2000 [==============================] - 54s 27ms/step - loss: 0.0538 - acc: 0.9836 - val_loss: 0.0301 - val_acc: 0.9927
Epoch 10/10
2000/2000 [==============================] - 55s 27ms/step - loss: 0.0568 - acc: 0.9827 - val_loss: 0.0416 - val_acc: 0.9893
```

RESULTS AND DISCUSSIONS



```
Text(0.5,0,'epoch')
```



```
score = model.evaluate(X_test, y_test, verbose = 1)
print('Test Score', score[0])
print('Test Accuracy', score[1])
```

```
12630/12630 [==============================] - 2s 138us/step
Test Score 0.16751343211778943
Test Accuracy 0.9666666666855438
```

DISCUSSIONS

1)Dataset: The first discussion would be about the dataset. You will need a large dataset of labeled images of traffic signs to train your machine learning model. The dataset should be representative of the real-world distribution of traffic signs. You may discuss where to find such a dataset, how to curate it, and how to split it into training, validation, and test sets.

2)Model Selection: Next, you will need to decide on a suitable machine learning model for this task. There are several models that can be used for image classification, such as convolutional neural networks (CNNs), support vector machines (SVMs), and decision trees. You may discuss the pros and cons of each model and decide on the one that is best suited for your project.

3)Preprocessing: Before training your model, you may need to preprocess the images. This may include resizing, normalization, and data augmentation to increase the size of the dataset. You may discuss the various techniques for image preprocessing and which ones are most appropriate for this task.

4)Training and Evaluation: Once you have selected a model and preprocessed your dataset, you will need to train your model and evaluate its performance. You may discuss the various techniques for training and fine-tuning your model, such as transfer learning, and the metrics used for evaluation, such as accuracy, precision, and recall.

5)Deployment: Finally, you will need to deploy your model to a real-world environment. You may discuss the various options for deploying your model, such as deploying it on a web server or on a mobile device. You may also discuss how to monitor the performance of your model in a real-world setting and how to retrain it if necessary.

Overall, these are some of the key discussions that may arise when working on a traffic sign classification project using machine learning.

REFERENCES

1)Sermanet, R, & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In Proceedings of the International Joint Conference on Neural Networks (pp. 2809-2813). IEEE.

2)Ciresan, D., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2012). Deep, big, simple neural nets for handwritten digit recognition. Neural computation, 24(8), 1907-1931.

3)Zhu, X., Zhang, W., & Wang, M. (2016). A deep learning approach towards traffic-sign detection and recognition. In Proceedings of the International Joint Conference on Artificial Intelligence (pp. 3001-3007). AAAI Press.

4)Stenborg, E., Forsberg, D., & Olofsson, B. (2017). A comparison of different machine learning approaches to traffic sign classification. In Proceedings of the International Conference on Image Analysis and Recognition (pp. 501-509). Springer.

5)GTSRB dataset: https://benchmark.ini.rub.de/gtsrb_dataset.html

6)German Traffic Sign Recognition Benchmark (GTSRB) website: http://benchmark.ini.rub.de/?section=gtsrb&subsection=news

7)Udacity Self-Driving Car Engineer Nanodegree: https://www.udacity.com/course/self-driving-car-engineer-nanodegr ee--n d013