Project Report

on

TRAFFIC SIGN CLASSIFICATION

Submitted to

NMAM INSTITUTE OF TECHNOLOGY, NITTE

(An Autonomous Institution Under VTU, Belagavi)

In partial fulfilment of the requirements for the award of the

Degree of Bachelor of Engineering

In

Computer Science and Engineering

By

Ankith Bhandary	**4NM17CS020**
Dhruv Shetty	**4NM17CS057**
KISHAN	**4NM17CS090**
MANUKASHYAP U V	**4NM17CS101**

Under the guidance of

Mr. RAMESHA SHETTIGAR

Asst. Professor

Dept. of CSE, NMAMIT, NITTE

CERTIFICATE

Certified that the project work entitled Traffic Sing Classification is a bonafide work carried out by Ankith Bhandary (4NM17CS020), Dhruv Shetty (4NM17CS057), Kishan (4NM17CS090), Manukashyap U V (4NM17CS101) in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project prescribed for the said Degree.

Name and Signature of the guide Signature of HOD Signature of Pricipal

External Viva

Name of the examiners	**Signature with date**
1._____	**1.**_____
2._____	**2.**_____

ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without the mention of all the people, without whom this endeavour would have been a difficult one to achieve. Their constant blessings, encouragement, guidance and suggestions have been a constant source of inspiration.

First and foremost, our gratitude to our project guide, **Mr. Ramesha Shettigar** for his constant guidance throughout the course of this project Phase - 2 and for the valuable suggestions.

We also take this opportunity to express a deep sense of gratitude to the project coordinators for their valuable guidance and support.

We acknowledge the support and valuable inputs given by, **Dr. Uday Kumar Reddy** the Head of the Department, Computer Science and Engineering, NMAMIT, Nitte.

Our sincere thanks to our beloved principal, **Dr. Niranjan N Chiplunkar** for permitting us to carry out this project at our college and providing us with all the needed facilities.

Finally, thanks to staff members of the Department of Computer Science and Engineering and our friends for their honest opinions and suggestions throughout the course of our project.

Ankith Bhandary (4NM17CS020)
Dhruv Shetty (4NM17CS057)
Kishan (4NM17CS090)
Manukashyap U V (4NM17CS101)

ABSTRACT

With the advent of electric vehicles and drive by wire, the idea of adding autonomy to motor vehicles became mainstream. Even though traffic signs have been around for ages now, people tend to ignore most of them, of which the speed limit is the most prominent. If we can create a machine learning model that will detect and classify the traffic signs, the ECU can take over, under the scenario where the driver ignores the signs.

With this project we aim to create a machine learning model that is trained over a bench mark set of traffic signs and be able to detect most of the traffic signs with most accuracy. We also aim to create a camera interface to this model that can recognise the traffic signs in front of it in close to real time.

Table of Contents

CHAPTER - 1

INTRODUCTION

1.1 Computer Vision

Definition: “Computer vision is an interdisciplinary scientific field that deals with how computers can gain a high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do.”

Computer vision enables the computer to partially interact with the real world and gain valuable data over time that will enable next level of automation in the day-to-day tasks leading to a safer and comfortable world for humans while also decreasing the global energy footprint.

1.2 Convolution Neural Network

Definition: “In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analysing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics.”

Neural networks try to mimic the behaviour of the human brain to learn from real-world data and to solve real-world problems in the process. Convolution neural networks are one of the more efficient ways of using machine learning on images since they are capable of differentiating and identifying the different characters in each image, thereby accurately predicting the image class.

1.3 Computer Vision in Automobiles

Lately, the use of computer vision in the field has skyrocketed. Computer vision provides unprecedented security and comfort for the automobile and its occupants. Some of these services are collision detention, traffic sign detection, driver awareness detection, side collision detection, traffic awareness etc. But these features require a huge amount of Research and Development (R&D) budget and also highly capable processing units to be installed on the vehicles. Thus, these features are only available in luxury vehicles whose cost would be the entire lifetime earnings of a common man a century ago.

1.4 Traffic Sign Classification

Traffic sign classification is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs etc. Being able to automatically recognize the traffic sign enables us to build smarter and safer automobiles.

CHAPTER - 2

LITERATURE SURVEY

2.1 Paper - 1

Paper Name: Towards Reliable Traffic Sign Recognition

-Benjamin H¨oferlin
Intelligent Systems Group Universit¨ at Stuttgart, Germany Benjamin.
höferlin@vis.uni-stuttgart.de
-Klaus Zimmermann
European Technology Centre (EuTEC) Sony Deutschland GmbH Stuttgart,
Germany klaus.zimmermann@sony.de

Abstract:

- System architecture for the reliable detection of circular traffic signs.
- 80s first research into computer-aided traffic sign detection.
- How old research is using colour segmentation, colour thresholding, Bayesian classification of colour.
- Hough transform and its derivatives.

Proposed Method:

- Detection
  - Twofold detection stage
    - Shape-based detection
    - Content-based detection
- Refinement
  - Contracting Curve Density Algorithm
- Classification
  - Two Multi-Layered Perceptron

Conclusion:

- The 30-minute-long test yielded 96.4% correct detections.

Future Work:

- Detection of non-circular signs
- Better methods for classification [1].

2.2 Paper - 2

Paper Name: Detection and Recognition of Indian Traffic Signs

- Pritika Priya, Dhara Modha, Mansi Agrawal Department Of Information Technology, Bharati Vidyapeeth College Of Engineering For Women, Pune-43

Abstract:

- An automatic system which would detect, recognize and interpret the meaning of the traffic signs for the driver.
- Use of several image processing techniques to enhance the efficiency and speed of the system.
- Disburden the drivers and reduce road accidents for better and safe driving, hence implementing the concept of the intelligent vehicle.

Proposed Method:

- Detection
  - Image blurring algorithm
  - Colour filtering
  - Blob detection
- Classification
  - Based on shapes: circle, rectangle and triangle.
- Recognition
  - Use of pattern matching algorithms to compare extracted ROI with standard templates.
  - If a pattern is found, sound notification is given to the driver otherwise the image is discarded.

Conclusion:

- Describes the system that is strictly used to differentiate Indian Traffic Signs that is subdivided into three classes according to the shapes.

Future Work:

- Improve the robustness of the system.
- Make the system work for highly tilted signs [2].

2.3 Paper - 3

Paper Name: **Deep Learning for Large-Scale Traffic-Sign Detection and Recognition**

-Domen Tabernik and Danijel Skoˇcaj Faculty Computer and Information Science, University of Ljubljana Veˇcna pot 113,1000 Ljubljana {domen.tabernik,danijel.skocaj}@fri.uni-lj.si

Abstract:

- Use of Mask R-CNN for Traffic sign detection and recognition.
- Using deep learning method for detection of traffic signs with large intra-category appearance variation.
- This approach is used for the detection of 200 traffic-sign categories.

Proposed Method:

- Detection and Recognition using Mask R-CNN
  - Online hard-example mining (OEHM)
  - Distribution of selected training sample
  - Sample weighting
  - Adjusting region pass-through during detection
- Data augmentation technique
  - This technique is used to generate several instances of traffic-signs and hence provide diverse data for the deep learning model.

Conclusion:

- Describes the system that is strictly used to differentiate Indian Traffic Signs that is subdivided in three classes according to the shapes.

Future Work:

- Average precision of 97.5% achieved in correct detections with an error rate of just 2%-3%.
- Improving the system to achieve ideal performance [3].

2. 4 Paper - 4

Paper Name: **Traffic Sign Classification Using Deep Inception Based Convolutional Networks**

-Mrinal Haloi, IIT Guwahati

mrinal.haloi11@gmail.com

Abstract:

- Use of spatial transformer layers and a modified version of inception module specifically designed for capturing local and global features together.
- Classify precisely intraclass samples even under deformations.
- This approach addresses the concern of exploding parameters and augmentations.

Proposed Method:

- Transformation invariant
  - Localization network
  - Grid generator
  - Sampling unit
- Proposed Pipeline
  - A modified version of GoogLeNet Inception module is used for the classification task.
- GTSRB data set is used for training and testing.

Conclusion:

- Achieves the state-of-the-art performance of 99.81% on GTSRB dataset.

Future Work:

- Improving the system to achieve ideal performance [4].

2. 5 Paper - 5

Paper Name: **Indian Traffic Sign Detection and Classification Using Neural Network**

-Arun Nandewal, CSE Department, arunnandewal@gmail.com

-Abhishek Tripathi, IT Department, abhishek.tripathi2421@gmail.com

-Satyam Chandra, EEE Department, satyam9871@gmail.com

NITK Surathkal

Abstract:

- This paper presents an automatic Indian Road Traffic Sign Detection and Classification system based on Multiple Neural Networks.
- Validated on a standard data set of Indian Traffic Signs.
- The proposed methodology works with real-time images invariant to rotation, illumination and partially distorted and occluded images.

Proposed Method:

- The proposed system has 4 stages:
  - Image procurement and pre-processing
  - Colour segmentation
  - Blob Detection using Binarization and Otsu Thresholding.
  - Classification using Multiple Neural Networks to decide the type of sign.

Conclusion:

- When the NN is trained over a standard database, the recognition of ROI has high accuracy.

Future Work:

- Real-time implementation requires a more robust system which has reduced proceeding time [5].

CHAPTER - 3

PROBLEM DEFINITION

3. 1 Problem Statement

Creation of an automatic, fast and light system that is capable of real-time detection, classification and interpretation of the traffic signs.

3. 2 Definition

Traffic sign classification is one of the most important uses of computer vision in automobiles. It has several aspects that include differentiating the road and surroundings, identifying the sign poles, closing in on the traffic sign and classifying it. We plan to tackle the issue of classifying the traffic signs once we have the sign in the view of the camera since it is the basic block of a traffic sign classification system.

Our goal is to create a machine learning model that is capable of classifying most of the traffic signals and also enable real-time interpretation of those signals. We aim to create an end product that can see the traffic signal through the camera and can interpret it onto the display while also being light on its resources.[8]

We plan to use OpenCV for the image input part of the system and the CNN model for the classification and interpretation of the traffic signs.[6]

CHAPTER - 4

SYSTEM REQUIREMENTS SPECIFICATION

4. 1 Model Creation

- Any one of the following
  - Google Collab Notebook with GPU acceleration enabled.
  - Kaggle Notebook with GPU acceleration and Network enabled.
  - PC or Laptop with
    - Intel i5 or Ryzen 5 1600 or higher.
    - Nvidia GTX or RTX or Radeon series graphics card.
    - 1GB of storage space.
    - 8GB or more RAM
    - Anaconda or PyCharm IDE installed

4. 2 Model Execution

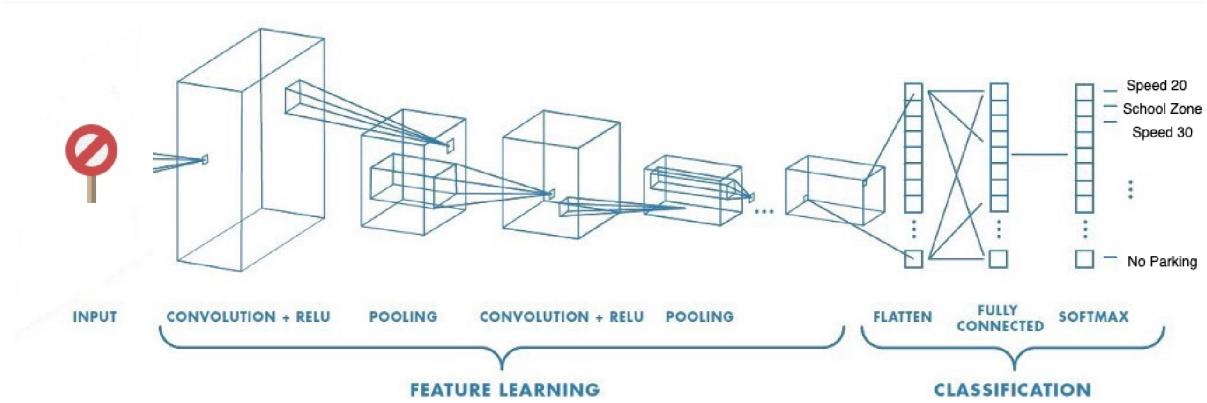
- Average PC or Laptop with Webcam.

4. 3 Python Libraries

- Python libraries such as NumPy, Matplotlib, TensorFlow, Keras, Cv2, Sklearn etc are used.

CHAPTER - 5

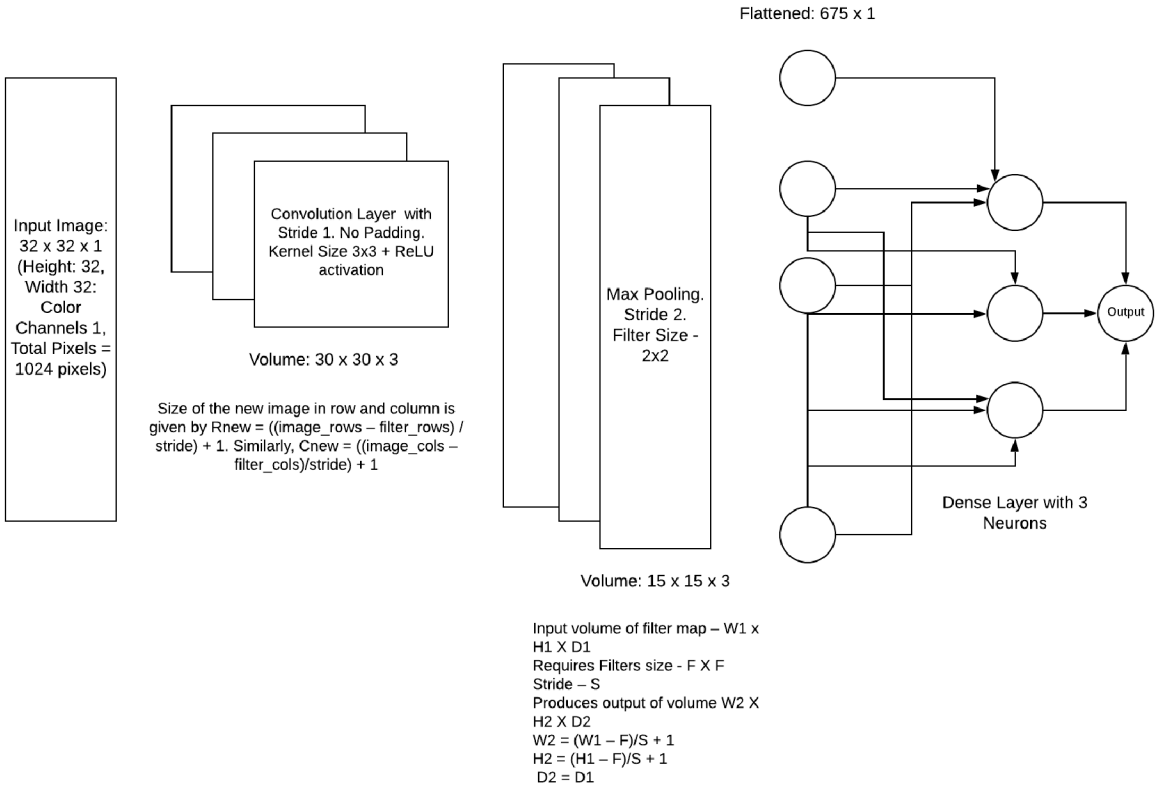
SYSTEM DESIGN



CNN is a machine learning algorithm that can take an input image, assign importance to various aspects or objects in the image, and be able to differentiate one from another.[7]

[6]CNN works by extracting the features from the images. Any CNN consists of the following:

1. The input layer which is a grayscale image.
2. The output layer is a binary or multi-class label.
3. Hidden layers consist of convolution layers, ReLU (Rectified Linear Unit) layers, the pooling layer, and a fully connected Neural Network.



CHAPTER - 6

IMPLEMENTATION

6. 1 Importing the required libraries

```
import NumPy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import cv2
from sklearn.model_selection import train_test_split
import pickle
import os
import pandas as pd
import random
from keras.preprocessing.image import ImageDataGenerator
```

6. 2 Setting up the parameters

Different parameters such as the path for the dataset, batch size, epochs, steps per epoch etc are set up.

```
data_path = "../input/tsc-data-set/myData/myData"  # folder with all the class folders
label_file = "../input/tsc-data-set/labels.csv"  # file with all names of classes
batch_size = 50  # how many to process together
steps_per_epoch = 400
epochs = 50
image_dimensions = (32, 32, 3)
test_ratio = 0.2    # if 1000 images split will 200 for testing
validation_ratio = 0.2  # if 1000 images 20% of remaining 800 will be 160 for validation
```

6. 3 Importing the images

The images are imported from the dataset and loaded into the kernel for further processing.

```
count = 0
images = []
classNo = []
myList = os.listdir(data_path)
print("Total Classes Detected:", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
for x in range(0, len(myList)):
    myPicList = os.listdir(data_path+"/"+str(count))
    for y in myPicList:
        curImg = cv2.imread(data_path+"/"+str(count)+"/"+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

Output

```
Total Classes Detected: 43
Importing Classes.....
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42
```

6. 4 Splitting the data

The dataset is split into test, train and validation sets.

```
X_train, X_test, y_train, y_test = train_test_split(
    images, classNo, test_size=test_ratio)
X_train, X_validation, y_train, y_validation = train_test_split(
    X_train, y_train, test_size=validation_ratio)
```

6.5 Dataset Validation

The dataset and the label files are compared to check if there are any mismatches in the dataset.

```
print("Data Shapes")
print("Train", end="")
print(X_train.shape, y_train.shape)
print("Validation", end="")
print(X_validation.shape, y_validation.shape)
print("Test", end="")
print(X_test.shape, y_test.shape)
assert(X_train.shape[0] == y_train.shape[0]
    ), "The number of images in not equal to the number of lables in training set"
assert(X_validation.shape[0] == y_validation.shape[0]
    ), "The number of images in not equal to the number of lables in validation
set"
assert(X_test.shape[0] == y_test.shape[0]
    ), "The number of images in not equal to the number of lables in test set"
assert(X_train.shape[1:] == (image_dimensions)
    ), " The dimesions of the Training images are wrong "
assert(X_validation.shape[1:] == (image_dimensions)
    ), " The dimesionas of the Validation images are wrong "
assert(X_test.shape[1:] == (image_dimensions)
    ), " The dimesionas of the Test images are wrong"
```

Output

```
Data Shapes
Train(22271, 32, 32, 3) (22271,)
Validation(5568, 32, 32, 3) (5568,)
Test(6960, 32, 32, 3) (6960,)
```

6.6 Reading the label file and displaying sample images

Each label is read from the CSV file and corresponding images in the dataset is displayed for visual validation of the dataset.

```
# READ CSV FILE
data = pd.read_csv(label_file)
print("data shape ", data.shape, type(data))
```

```
# DISPLAY SOME SAMPLES IMAGES  OF ALL THE CLASSES
num_of_samples = []
cols = 5
num_classes = noOfClasses
fig, axs = plt.subplots(nrows=num_classes, ncols=cols, figsize=(5, 100))
fig.tight_layout()
for i in range(cols):
    for j, row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(
            0, len(x_selected) - 1), :, :], cmap=plt.get_cmap("gray"))
        axs[j][i].axis("off")
        if i == 2:
            axs[j][i].set_title(str(j) + "-"+row["Name"])
            num_of_samples.append(len(x_selected))
```

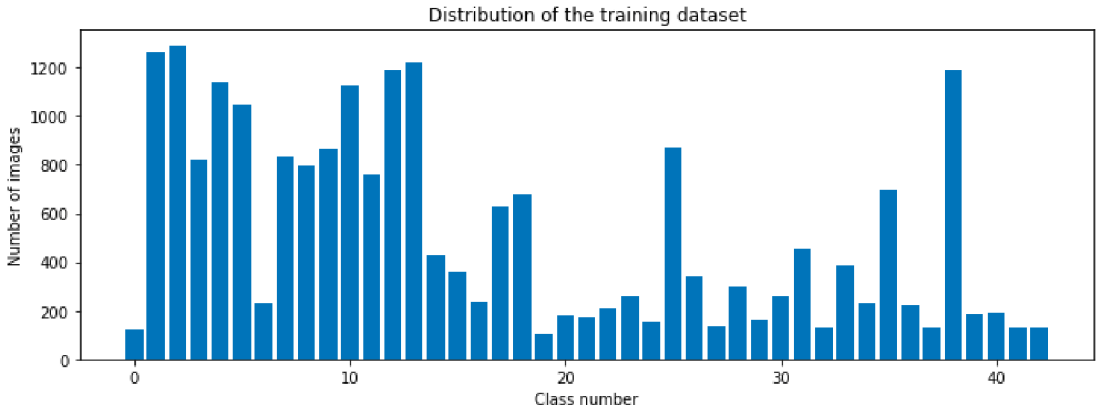
Output

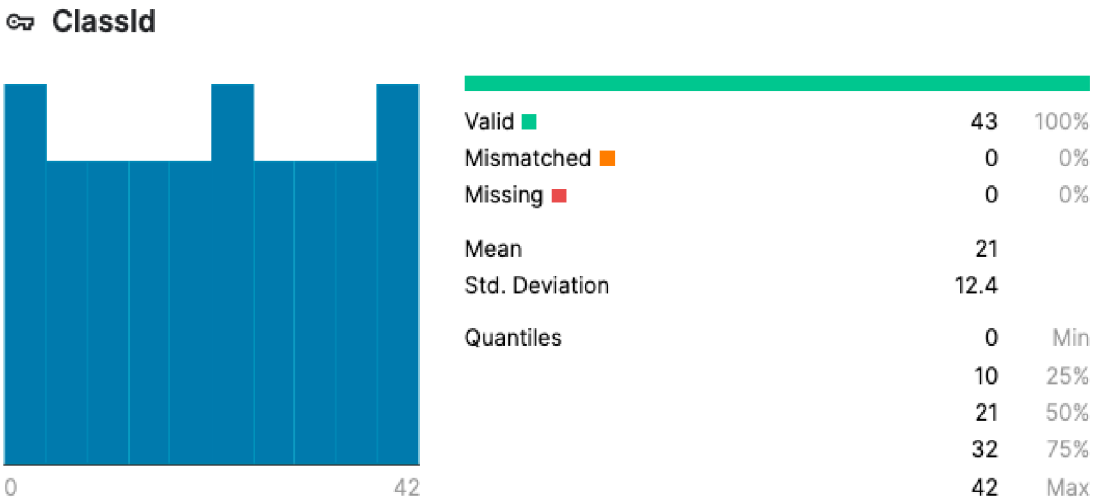
6.7 Display information about the data set

Some information about the distribution of images of different classes in the dataset is displayed.

```
# DISPLAY A BAR CHART SHOWING NO OF SAMPLES FOR EACH CATEGORY
print(num_of_samples)
plt.figure(figsize=(12, 4))
```

```
plt.bar(range(0, num_classes), num_of_samples)
plt.title("Distribution of the training dataset")
plt.xlabel("Class number")
plt.ylabel("Number of images")
plt.show()
```





6.8 Pre-processing the images

All of the images in the dataset is pre-processed to make all of them have the same dimensions and all of them are converted to grey scale images for easier processing.

```
# PREPROCESSING THE IMAGES
def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img

def equalize(img):
```

```
    img = cv2.equalizeHist(img)
    return img

def preprocessing(img):
    img = grayscale(img)     # CONVERT TO GRAYSCALE
    img = equalize(img)      # STANDARDIZE THE LIGHTING IN AN IMAGE
    img = img/255            # TO NORMALIZE VALUES BETWEEN 0 AND 1 INSTEAD OF
0 TO 255
    return img
# TO IRETATE AND PREPROCESS ALL IMAGES
X_train = np.array(list(map(preprocessing, X_train)))
X_validation = np.array(list(map(preprocessing, X_validation)))
X_test = np.array(list(map(preprocessing, X_test)))
# ADD A DEPTH OF 1
X_train = X_train.reshape(
    X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)
X_validation = X_validation.reshape(
    X_validation.shape[0], X_validation.shape[1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)
```

6.9 Image Augmentation

```
# AUGMENTATAION OF IMAGES: TO MAKEIT MORE GENERIC
dataGen = ImageDataGenerator(width_shift_range=0.1,   # 0.1 = 10%     IF MORE
THAN 1 E.G 10 THEN IT REFFERS TO NO. OF  PIXELS EG 10 PIXELS
                             height_shift_range=0.1,
                             zoom_range=0.2,  # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
                             shear_range=0.1,  # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10)  # DEGREES
dataGen.fit(X_train)
# REQUESTING DATA GENRATOR TO GENERATE IMAGES  BATCH SIZE = NO. OF
IMAGES CREAED EACH TIME ITS CALLED
batches = dataGen.flow(X_train, y_train, batch_size=20)
X_batch, y_batch = next(batches)
# TO SHOW AGMENTED IMAGE SAMPLES
fig, axs = plt.subplots(1, 15, figsize=(20, 5))
fig.tight_layout()

for i in range(15):
    axs[i].imshow(X_batch[i].reshape(image_dimensions[0], image_dimensions[1]))
    axs[i].axis('off')
```

```
plt.show()


y_train = to_categorical(y_train, noOfClasses)
y_validation = to_categorical(y_validation, noOfClasses)
y_test = to_categorical(y_test, noOfClasses)
```

Output

6.10 Creating the Convolution Neural Network

After experimenting with different model parameters the most ideal CNN model is setup.

```
# CONVOLUTION NEURAL NETWORK MODEL
def myModel():
    no_Of_Filters = 60
    # THIS IS THE KERNEL THAT MOVE AROUND THE IMAGE TO GET THE
FEATURES.
    size_of_Filter = (5, 5)
    # THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN USING 32 32
IMAGE
    size_of_Filter2 = (3, 3)
    # SCALE DOWN ALL FEATURE MAP TO GERNALIZE MORE, TO REDUCE
OVERFITTING
    size_of_pool = (2, 2)
    no_Of_Nodes = 500   # NO. OF NODES IN HIDDEN LAYERS
    model = Sequential()
    # ADDING MORE CONVOLUTION LAYERS = LESS FEATURES BUT CAN CAUSE
ACCURACY TO INCREASE
    model.add((Conv2D(no_Of_Filters, size_of_Filter, input_shape=(
        image_dimensions[0], image_dimensions[1], 1), activation='relu')))
    model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu')))
    # DOES NOT EFFECT THE DEPTH/NO OF FILTERS
    model.add(MaxPooling2D(pool_size=size_of_pool))

    model.add((Conv2D(no_Of_Filters//2, size_of_Filter2, activation='relu')))
    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5))
```

```
    model.add(Flatten())
    model.add(Dense(no_Of_Nodes, activation='relu'))
    # INPUTS NODES TO DROP WITH EACH UPDATE 1 ALL 0 NONE
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses, activation='softmax'))  # OUTPUT LAYER
    # COMPILE MODEL
    model.compile(Adam(lr=0.001), loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

6.11 Training the model

```
# TRAIN
model = myModel()
print(model.summary())
history = model.fit_generator(dataGen.flow(X_train, y_train,
batch_size=batch_size),
                              steps_per_epoch=steps_per_epoch, epochs=epochs,
validation_data=(X_validation, y_validation), shuffle=1)
```

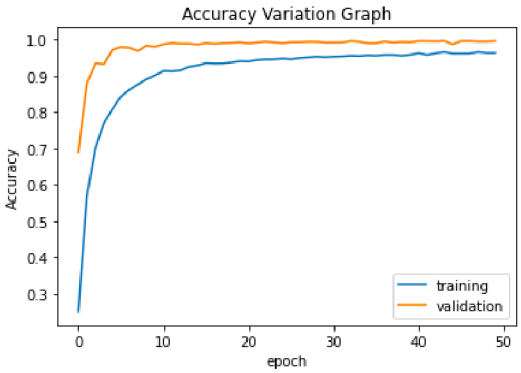
6.12 Plotting the Loss and Variation Graph

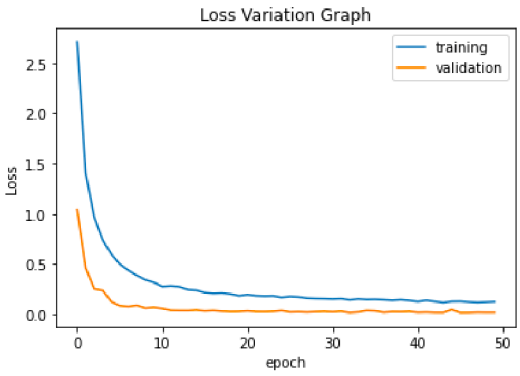
The graph of how the accuracy and loss varies with each of the epoch is plotted.

```
# PLOT
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('Loss Variation Graph')
plt.xlabel('epoch')
plt.ylabel('Loss')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'])
plt.title('Accuracy Variation Graph')
plt.xlabel('epoch')
plt.ylabel('Accuracy')
```

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

Output





Test Score: 0.012194344773888588
Test Accuracy: 0.995976984500885

6.13 Saving the model

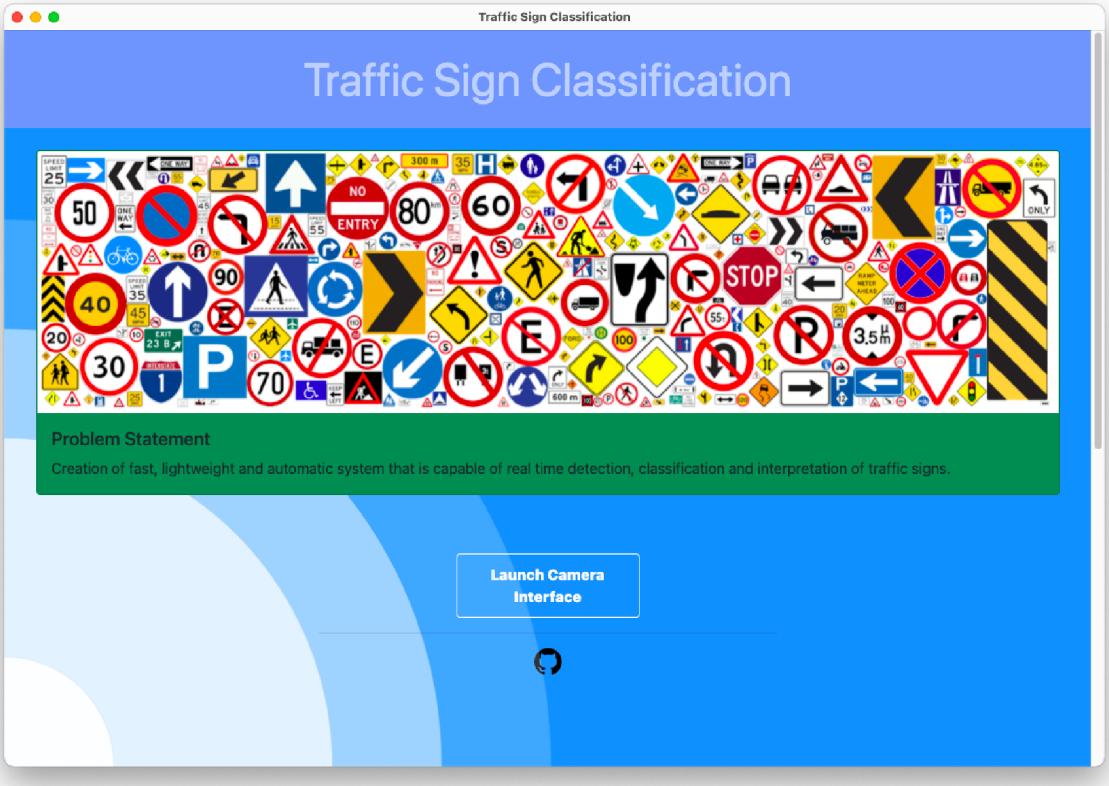
The model is saved to the disk in the form of JSON and HDF5 for future operations.

```
# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
```

6.14 Creating the GUI for the practical application of the model

We are using eel python library to write the GUI code for the application using HTML, CSS and JavaScript.

We use the bootstrap framework to create the container, div and cards and custom css for animation rendering. The GUI also points to the open sourced code for this particular application of the model stored in github.



6.15 Creating the OpenCv interface for the practical application of the model

We also created a OpenCv webcam interface for the model to identify the traffic signs. Our practical demonstration also has a logging system that will log the identified signs along with the probability and identified time for review and safety purposes.

The contents of the log file:

No: 11 Time: 20:48:04
[14] Stop 97.35%

No: 12 Time: 20:48:05
[12] Priority road 95.93%

No: 13 Time: 20:48:05
[14] Stop 92.57%

No: 14 Time: 20:48:05
[14] Stop 99.61%

No: 15 Time: 20:48:05
[12] Priority road 93.11%

CHAPTER - 7

RESULTS AND CONCLUSION

7.1 Results

1. The data set is cleaned and made error free.
2. The distribution of images amound different classes is found satisdactory.
3. The CNN model was created and trained to achieve theoretical accuracy as per objective.
4. The model was serialized succssfully.
5. The practical application of the model works well under ideal conditions.
6. All the objectives are satisfied within the stipulated time.

7.2 Conclusion

The idea of applying convolution neural networks to solve the problem of traffic sign classification proved feasible and the results were satisfactory. With time this technique can be more refined and can be used in real life and also a replacement for some of the old traffic sign classification systems.

CHAPTER - 8

FUTURE WORKS

1. The data set can be furter enlarged by adding more signs and sign classes.
2. The model can be further refined to predict more real life conditions.
3. The practical implementation and logging system can be refined further.
4. The model can be used for differet implentation other than the one provided here.

CHAPTER - 9

REFERENCES

[1]. Towards Reliable Traffic Sign Recognition
Benjamin Hoferlin - Intelligent Systems Group Universitat Stuttgart Stuttgart, Germany benjamin.hoeferlin@vis.uni-stuttgart.de
Klaus Zimmermann - European Technology Center (ESTEC) Sony Deutschland GmbH Stuttgart, Germany klaus.zimmermann@sony.de

[2]. Detection And Recognition Of Indian Traffic Signs
Pritika Priya, Dhara Modha, Mansi Agrawal Department Of Information Technology, Bharati Vidyapeeth College Of Engineering For Women, Pune-43

[3]. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition
Domen Tabernik and Danijel Skocaj Faculty of Computer and Information Science, University of Ljubljana Vecna pot 113, 1000 Ljubljana {domen. tabernik,danijel.skocaj}@fri.uni-lj.si

[4]. Traffic Sign Classification Using Deep Inception Based Convolutional Networks Mrinal Haloi IIT Guwahati1, mrinal.haloi11@gmail.com

[5]. Indian Traffic Sign Detection and Classification Using Neural Networks
Arun Nandewal - CSE Department, arunnandewal@gmail.com
Abhishek Tripathi - IT Department, abhishek.tripathi2421@gmail.com
Satyam Chandra - EEE Department, satyam9871@gmail.com
NITK Surathkal

[6]. Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.

[7]. Stanford University (2018). Introduction to Convolutional Neural Networks.

[8]. Visin, V.D., Francesco (2016). English: Animation of a variation of the convolution operation. Blue maps are inputs, and cyan maps are outputs.
[online] Wikimedia Commons.

[9]. Bonner, A. (2019). The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks.