

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

**CERTIFICATE**

Certified that the Mini Project entitled “**Road Sign Classification using CNN**” is carried out by **Nikhil Dwivedi (1NH17EC729), Ritwik Shome (1NH17EC738), Sahib Arora (1NH17EC742) and Prathamesh Kadam (1NH17EC420)** bonafide students of **New Horizon College of Engineering**, **Bengaluru** in partial fulfilment for the award of **Bachelor of Engineering** in **Electronics & Communication** of the **Visvesvaraya Technological University** during the year **2019-20**.

It is certified that all corrections/suggestions indicated for internal assessment has been incorporated in the report deposited to the department library. The Mini Project report has been approved as it satisfies the academic requirements in respect of the Mini Project work prescribed for the said degree.

Signature of HOD Signature of Internal Guide

Dr Sanjeev Sharma Mrs. Ishani Mishra

Professor and HOD Assistant Professor

Dept. of ECE Dept. of ECE

NHCE, Bangalore NHCE, Bangalore

EXTERNAL EXAMINER

Name of Examiner Signature with Date

1.

2.

**ACKNOWLEDGEMENT**

We would like to thank the management Dr Mohan Maghnani, Chairman of New Horizon Educational Institutes for providing us the necessary infrastructure and creating a good environment.

We would like to express our sincere gratitude to our principal, Dr Manjunatha for giving us this opportunity to undertake this project.

We also wish to express our gratitude to Mr. Sanjeev Kumar, HOD of Electronics & Communication Engineering Department, for his kind hearted support.

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our guide Mrs. Ishani Mishra and project coordinator Mr. Ashok for their valuable guidance, encouragement and help throughout this project. The useful suggestion for this Mini Project and cooperative behavior is sincerely acknowledged.

We also wish to express our indebtedness to our parents as well as our family members whose blessings and support always helped us to face the challenges ahead. At the end we would like to express our sincere thanks to all our friends who helped us directly or indirectly during this project work.

Rigorous hard work has been put in this project to ensure that it proves to be the best. We hope that this project will prove to be a breeding ground for the next generation of students and will guide them in every possible way.

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| 1 | **Introduction** | 4 |
| 2 | **Literature Review** | 5 |
| 3 | **Existing System and Problem Statement** | 6 |
| 4 | **Hardware and Software Requirements** | 7 |
| 5  5.1  5.2 | **Proposed System**  Block Diagram  Flow Chart | 8  8  9 |
| 5.3  5.4 | Algorithm  Architecture & Working | 10  11 |
| 6  7  8 | **Result**  **Advantages and Applications**  **Conclusion** | 15  23  25 |
| 9  10  11 | **Future Scope**  **References**  **Appendix** | 26  27  28 |

**CHAPTER 1**

**INTRODUCTION**

This project road sign classification is done by using convolutional neural networks. In this project the computer recognizes the image that has been given to it. The ability to recognize an image by the computer was the main goal by this project. As we know that the computer does not understand images therefore these images are taken in the form of pixels or matrices. After this these images are fed into the neural network. The neural network is broken into layers. The first layer in the neural network is termed as input layer where the images from the dataset are taken. The second layers or the forth coming layers or the layers between are termed as hidden layers. The last layer is the output layer. The output layer decides whether the input image entered from the dataset in correctly recognized by the machine or not. The main work of this algorithm convolutional neural networks mainly functions in the hidden layers. The process on how the images are recognized by the machine is done by convolutional neural networks. The images entered are recognized part by part by all the hidden layers. Convolution of this images are done that is the images are passed through filters which helps the computer learn the image step by step. Another step is that pooling is used. Pooling is same as convolution but no hyperparameters are there unlike convolution. As there are so many classes or output labels are therefore SoftMax function is used. SoftMax function basically predicts the suitable label for the input image.

**CHAPTER 2**

**LITERATURE REVIEW**

The following literature resources were used to complete this project

* [ZF Net](http://arxiv.org/pdf/1311.2901v3.pdf)(2013)
* [AlexNet](https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf) (2012)
* [VGG Net](http://arxiv.org/pdf/1409.1556v6.pdf)(2014)
* [GoogLeNet](http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf)(2015)
* [Microsoft ResNet](https://arxiv.org/pdf/1512.03385v1.pdf) (2015)
* [Generative Adversarial Networks](https://arxiv.org/pdf/1406.2661v1.pdf) (2014)
* Chen T, Lu S, “Accurate and Efficient Traffic Sign Detection Using Discriminative Adaboost and Support Vector Regression,” IEEE Transactions on Vehicular Technology, 2016, 65(6):4006-4015.
* Xing M, Chunyang M, Yan W, et al, “Traffic sign detection and recognition using color standardization and Zernike moments,” Chinese Control and Decision Conference, 2016.
* R. Qian, B. Zhang, Y. Yue and Z. Wang, “Robust Chinese traffic sign detection and recognition with deep convolutional neural network,” Proc. IEEE International Conference on Natural Computation (ICNC), Aug. 2015, pp. 791-796, doi: 10.1109/ICNC.2015.7378092.
* P. Sermanet and Y. LeCun, “Traffic sign recognition with multi-scale convolutional networks,” in Proc. Int. Joint Conf. on Neural Networks (IJCNN’11) (San Jose, 2011).
* M. Haloi, “Traffic sign classification using deep inception based convolutional networks” (2015). arXiv:1511.02992, 2015.
* D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, “A committee of neural networks for traffic sign classification,” in Proc. Int. Joint Conf. on Neural Networks (IJCNN) (San Jose, 2011), pp. 1918–1921.
* J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “The German traffic sign recognition benchmark: a multi-class classification competition,” in Proc. Int. Joint Conf. on Neural Networks (San Jose, 2011).

**CHAPTER 3**

**EXISTING SYSTEM AND PROBLEM STATEMENT**

**3.1 EXISTING SYSTEM**

Earlier Road Sign Classification system using the same architecture was done in MATLAB with 85% accuracy.

**3.2 PROBLEM STATEMENT**

To build a Road Sign Classification System using Convolutional Neural Networks (CNN) on Python 3.7.6 with 95% accuracy.

**CHAPTER 4**

**HARDWARE AND SOFTWARE SPECIFICATION**

**4.1 HARDWARE SPECIFICATION**

* Processor: X86 Compatible processor with 1.7 GHz Clock speed or above
* RAM: 512 MB or more
* Hard disk: 250 GB or more

**4.2 SOFTWARE SPECIFICATION**

* Python 3.7
* Spyder 4.0 or any other compatible Python IDE
* Windows XP or above Operating System

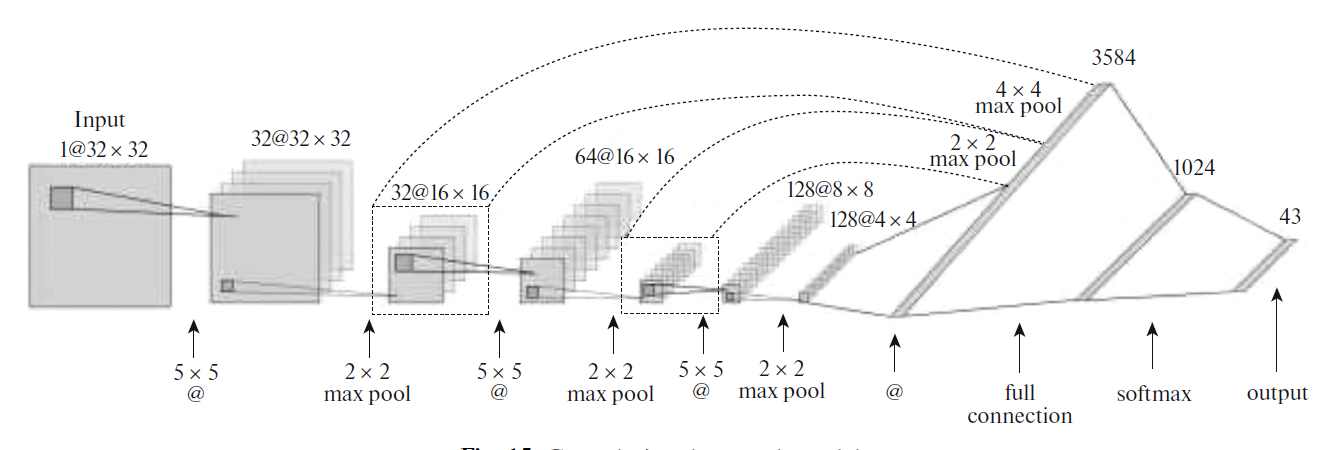
**4.3 PYTHON LIBRARIES**

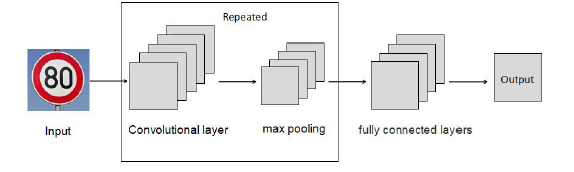
* + Keras
  + Tensorflow
  + OpenCV
  + Matplotlib
  + Scikit-learn
  + Pandas
  + Python Image Classification
  + Tkinter
  + Numpy

**CHAPTER 5**

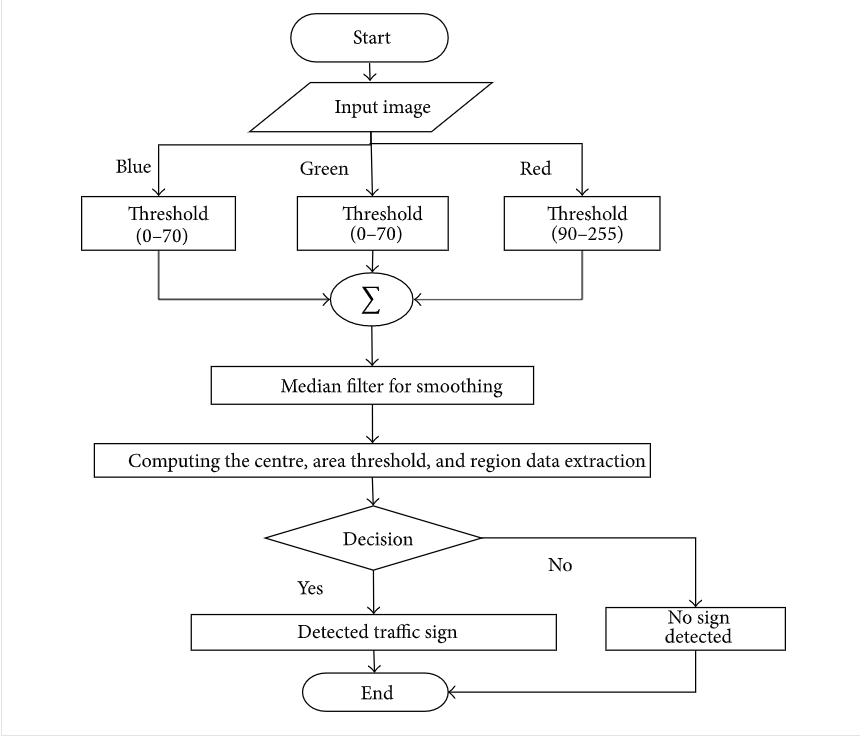
**PROPOSED SYSTEM**

**5.1 BLOCK DIAGRAM**





**5.2 FLOW CHART**



**5.3 ALGORITHM**

Step 1**:** Capture the image

Step 2: Break down the image to get 3 layers (Red, Green, Blue)

Step 3: Select a suitable size(5x5,3x3) and number (32,64) of function blocks.

Step 4: Select a suitable stride (number of pixels the function block moves after each iteration value)

Step 5: Run the function block through the image to identify small pixels.

Step 6: Repeat step 5 for all function blocks till all the pixels in the image are covered

Step 7: Increase the size of function block and check whether the function blocks are forming and particular pattern

Step 8: Repeat step 8 until the whole picture is covered

Step 9: Make the decision based on result of step 8

**5.5 ARCHITECTURE AND WORKING**

There are 43 categories in the data set and thus our model can classify the road signs into the following 43 categories.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Speed limit (20km/h) | 23 | Bumpy road |
| 2 | Speed limit (30km/h) | 24 | Slippery road |
| 3 | Speed limit (50km/h) | 25 | Road narrows on the right |
| 4 | Speed limit (60km/h) | 26 | Road work |
| 5 | Speed limit (70km/h) | 27 | Traffic signals |
| 6 | Speed limit (80km/h) | 28 | Pedestrians |
| 7 | End of speed limit (80km/h) | 29 | Children crossing |
| 8 | Speed limit (100km/h) | 30 | Bicycles crossing |
| 9 | Speed limit (120km/h) | 31 | Beware of ice/snow |
| 10 | No passing | 32 | Wild animals crossing |
| 11 | No passing veh over 3.5 tons | 33 | End speed + passing limits |
| 12 | Right-of-way at intersection | 34 | Turn right ahead |
| 13 | Priority road | 35 | Turn left ahead |
| 14 | Yield | 36 | Ahead only |
| 15 | Stop | 37 | Go straight or right |
| 16 | No vehicles | 38 | Go straight or left |
| 17 | Veh > 3.5 tons prohibited | 39 | Keep right |
| 18 | No entry | 40 | Keep left |
| 19 | General caution | 41 | Roundabout mandatory |
| 20 | Dangerous curve left | 42 | End of no passing |
| 21 | Dangerous curve right | 43 | End no passing veh > 3.5 tons |
| 22 | Double curve |  |  |

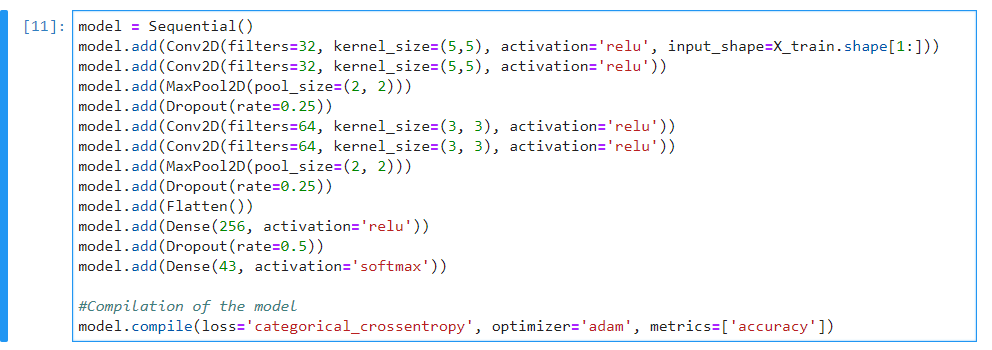
The architecture of our model is

* 2 Conv2D layer (filter=32, kernel\_size=(5,5), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* 2 Conv2D layer (filter=64, kernel\_size=(3,3), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* Flatten layer to squeeze the layers into 1 dimension
* Dense Fully connected layer (256 nodes, activation=”relu”)
* Dropout layer (rate=0.5)
* Dense layer (43 nodes, activation=”SoftMax”)

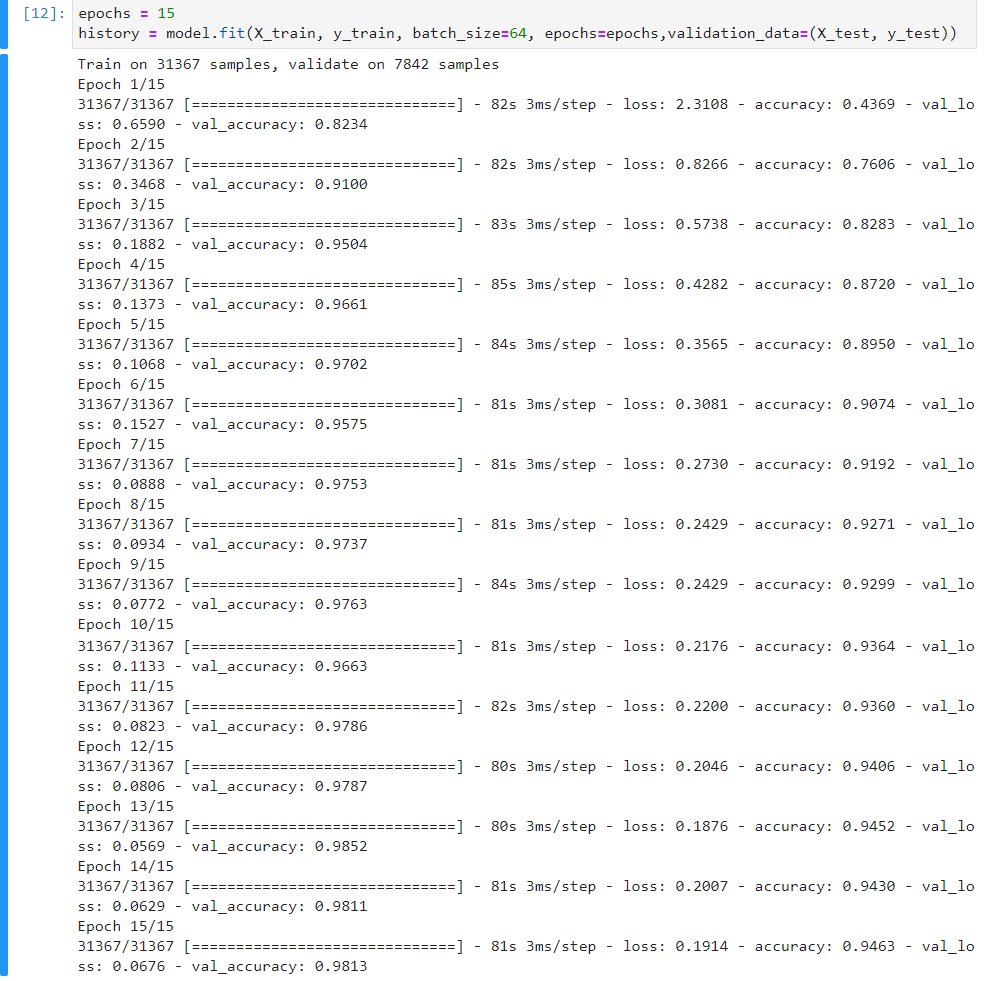
The code first takes the data as images and stores it in form of matrices. Then splits the dataset into 2 parts, testing and training set, we have chosen 80% of the data set to train the model and 20% to test our model.

The given image is broken down to 3 layers to separate the RGB values based on the threshold. After the image has been broken down the program learns the pattern of small filters from the training set, this is done using Keras library, in this project we have used 32 5x5 filters and 64 3x3 filters for each image. These filters once decided by the program are run through the new image one by one. These filters are of different layers, the filters in the first layer checks if a pixel is present or not in a particular place or not. The next layer is run on the first layer and not on the picture itself and checks if the presence or absence of pixels combine to form a particular pattern or not(check if they form a curve or edge etc.).The next layer checks if the patterns combine together to form another pattern, several layers of convolutional neural networks are run before the combination of layer can result in the formation of an object. The model here can only recognize patterns it has learnt from the training data set. For identifying patterns, each 0f the 64 and 32 filters gives a numeric value if a pixel is present in a particular cell or not(close to 0 if its absent and close to 1 if present), All the filters will give different numeric values as which are quiet close to each other as patterns in road signs do not differ vastly, therefore the model takes a mean of all the filters and if the mean is greater than 0.5 it decides that the filter is present else absent. Once decided if a particular pattern is present or not. This is repeated by different layers until an object is formed. If the object formed is recognized by the model the output with accuracy is shown.

We compile the model with Adam optimizer which performs well and loss is “categorical\_crossentropy” because we have multiple classes to categorize.

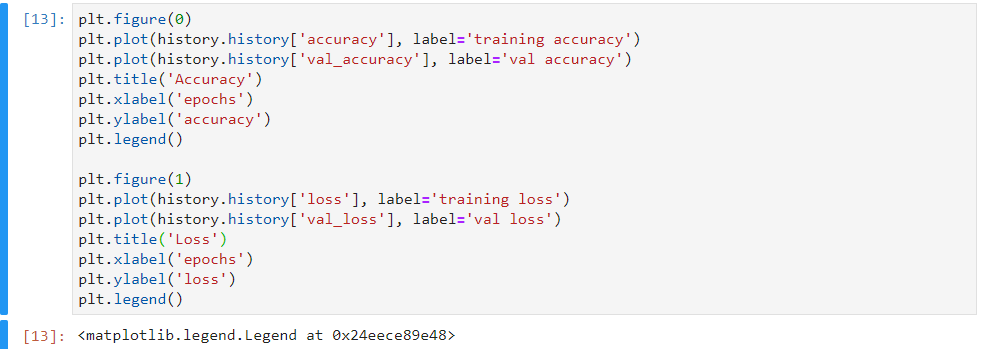


Once the model architecture has been built we then proceed to train the model using the function model.fit(). We tried the model with different batch sizes but we got the stable accuracy at batch size 64. The accuracy was stable after 15 forward and backward runs.



We achieved 95% accuracy on the training dataset.

We then plotted the accuracy and loss curve using matplotlib library.



Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30×30 pixels and make a NumPy array containing all image data. From the sklearn.metrics, we imported the accuracy\_score and observed how our model predicted the actual labels. We achieved a 95% accuracy in this model.



The model traffic\_classifier.h5 was then saved using Keras model.save() function.

**CHAPTER 6**

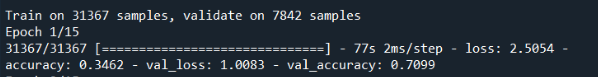
**RESULT**

**6.1 TRAINING AND VALIDATING THE MODEL**

We train the model using model.fit(), with batch size 64 the model was able to achieve stable accuracy. The total epochs given to train the model was 15 and the accuracy of the model increased with increase in the number of epochs.

1. Epoch=1

One forward pass and one backward pass of all the training examples



1. Epochs=2

Two forward passes and two backward passes of all the training examples



1. Epochs=5

Five forward passes and five backward passes of all the training examples



1. Epochs=10

Ten forward passes and ten backward passes of all the training examples



1. Epochs=15

Fifteen forward passes and fifteen backward passes of all the training examples



1. Epochs=20

Twenty forward passes and twenty backward passes of all the training examples



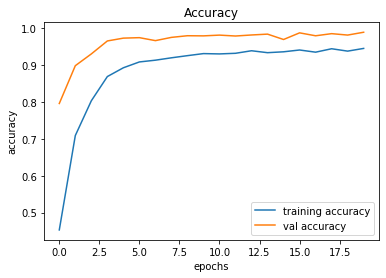
The overall accuracy obtained on the training dataset after running 20 passes is 94%

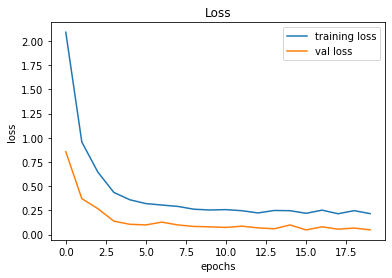
**6.2 TRAINING AND VALIDATING THE MODEL**

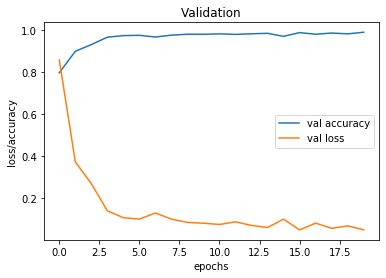
From the sklearn.metrics, we imported the accuracy\_score and observed how our model predicted the actual labels. We achieved a 95% accuracy in this model.

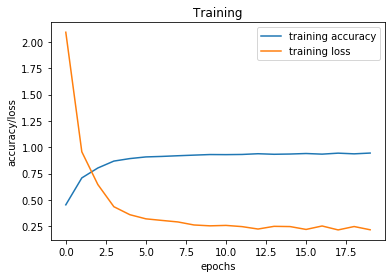


* 1. **ACCURACY AND LOSS CURVE**









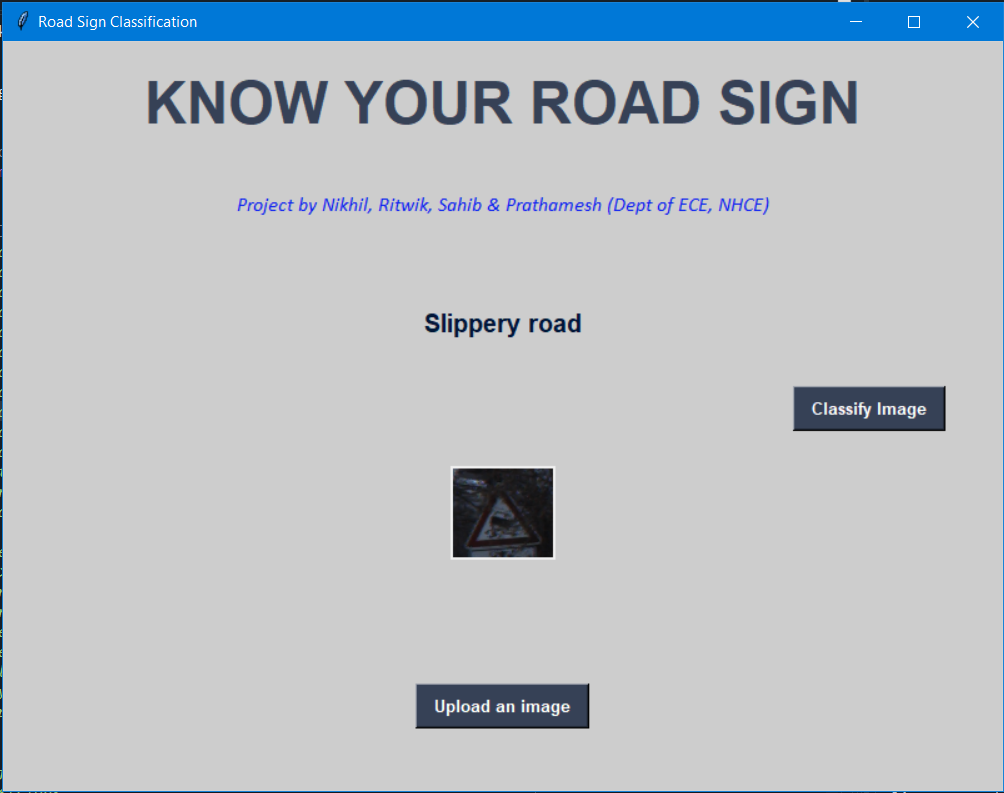
**6.3 OUTPUT**

1. Graphical User Interface Window



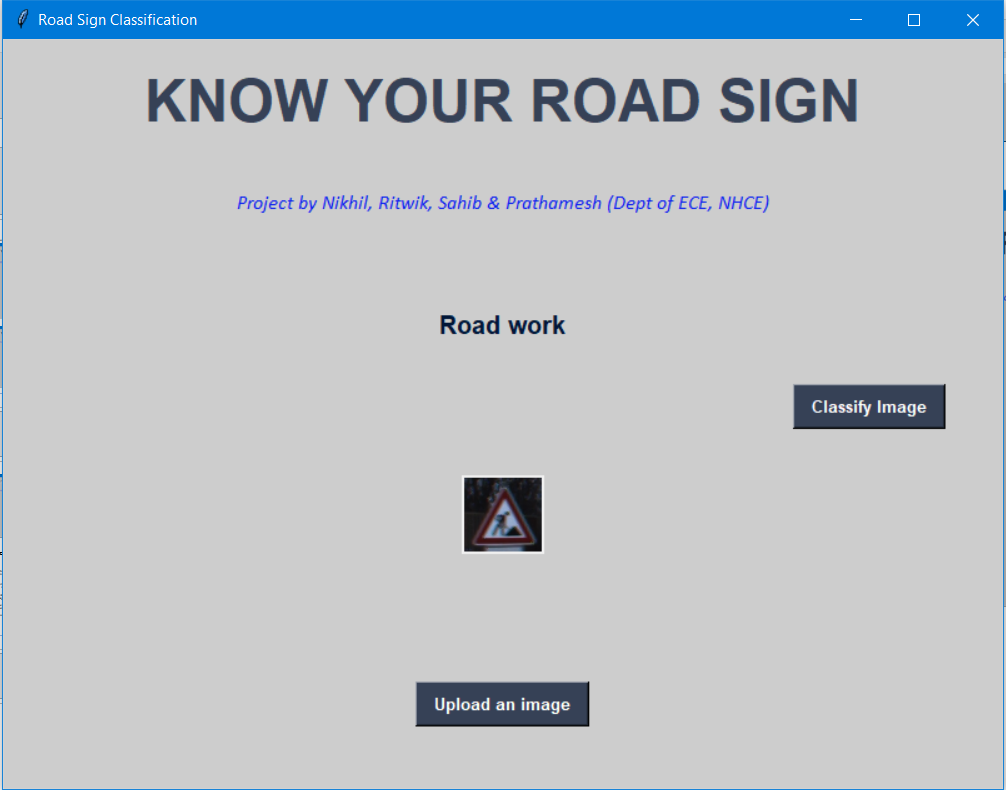
1. Image from Training set





1. Image from Testing set







1. Random image taken from Google



**CHAPTER 7**

**ADVANTAGES AND APPLICATION**

**7.1 ADVANTAGES**

* The model we developed using the Convolutional Neural Network achieved accuracy up to 95% which means that our model is highly reliable compared to many other Road Classification model.
* Although the model takes more time to train (due to system specifications), it can classify the images within 1-2 seconds, which is very good for a CNN model.
* The Graphical User Interface built for using the project is very basic and minimalist and can be operated by anyone, even with people having very basic computer knowledge.
* Once the model is trained it can be used in any system to test and implement which means the system requirement for running the project is very low.
* The project has been made open source and can be used and modified by anyone.

**7.2 APPLICATIONS**

* The Road Sign Classification project is very useful for the autonomous vehicles which are going to be very common in the near future. It is very important for the autonomous vehicles to capture and identify in real time and this entire process should be very quick for proper functioning of the autonomous vehicles.
* Many big companies like Tesla, Uber, Google-Waymo, Mercedes-Benz, Toyota, Ford, Audi etc. are working on autonomous vehicles and self-driving cars in fact Tesla has already implemented the self-driving mode in some of its models.
* This image classification model can be used for many other applications if the proper dataset is available like Road Signal Recognition, Hand Sign recognition etc.

**CHAPTER 8**

**CONCLUSION**

We built a road sign recognition system which recognizes and classifies the road sign from an input image into the 43 categories.

The image processing techniques used in this software include a pre-processing stage, regions of

interest detection, potential traffic sign detection, according to the traffic sign shape patterns, and

finally, the recognition and classification of these potential traffic signs according to a database of

traffic sign patterns. The performance of this application depends on the quality of the input image,

in relation to its size, contrast and the way the signs appear in the image. With this consideration,

the percentages of recognized signs for this application are high.

We were able to achieve 94% accuracy on the training set and 95% on the validation set which is pretty good for a simple CNN based model. We visualized how our accuracy and loss changes with time and also visualized the accuracy and loss on the training and validation data.

**CHAPTER 9**

**FUTURE SCOPE**

* The model we developed achieves up to 95% accuracy which is highly reliable but still it’s not completely reliable. In order to achieve complete reliability, we need the accuracy to be more than 99%. So, this model can be further developed using different architecture on different datasets for even better accuracy.
* Our model only works for the static inputs i.e. it works only for the images. We can further make this model dynamic so that it can work on live data where it will automatically capture the road signs using dashcam and classify it immediately. This will be very much helpful in the Self Driving vehicles.
* The time taken for the model to train once (1 epoch) takes more than 60 seconds and thus running 15 epochs took around 20 minutes. We can further develop the model and improve its time.
* The training and validation losses has decreased significantly as the no of epochs has increased. The losses can further be reduced.
* The GUI developed for this project is quite basic and it can be made much more Interactive and user friendly.
* The model right now only accept image within the dimensions of shape (1, 30, 30, 3). Thus, it can be further developed to accept images of larger dimensions.
* Our model classifies the road sign only into 43 categories as per the dataset as of now. There are still many more categories of road signs which it can’t classify. So, if the datasets with much more classes are made available this model can be altered to classify even more different types of road signs.
* We can also increase the no of blobs for each category.
* We also need to focus our work in the previous steps, especially the segmentation, since this step is crucial for the correct operation of the whole system. We also have to deal with other kind of problems like partial occlusions, shadows, bad illuminations, etc.

**CHAPTER 10**

**REFERENCES**

[1] <https://www.kaggle.com/fab711/gtsrb-traffic-signs-classification-with-cnn>

[2] <https://analyticsindiamag.com/8-uses-cases-of-image-recognition-that-we-see-in-our-daily-lives/>

[3] <https://docs.python.org/3.7/>

[4] <https://scholar.google.co.in/scholar?q=traffic+sign+recognition&hl=en&as_sdt=0&as_vis=1>

[5] <https://en.wikipedia.org/wiki/Traffic-sign_recognition>

[6] Wang C Y, “Research and application of traffic sign detection and recognition based on deep learning,” International Conference on Robots & Intelligent System (ICRIS), 2018.

[7] Chen T, Lu S, “Accurate and Efficient Traffic Sign Detection Using Discriminative Adaboost and Support Vector Regression,” IEEE Transactions on Vehicular Technology, 2016, 65(6):4006-4015.

[8] Xing M, Chunyang M, Yan W, et al, “Traffic sign detection and recognition using color standardization and Zernike moments,” Chinese Control and Decision Conference, 2016.

[9] <http://benchmark.ini.rub.de/?section=gtsrb&subsection=news>

**CHAPTER 11**

**APPENDIX**

**11.1 SOURCE CODE**

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import cv2
5. import tensorflow as tf
6. from PIL import Image
7. import os
8. from sklearn.model\_selection import train\_test\_split
9. from keras.utils import to\_categorical
10. from keras.models import Sequential, load\_model
11. from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
12. data = []
13. labels = []
14. classes = 43
15. cur\_path = os.**getcwd**()
16. #Retrieving the images and their labels
17. for i in **range**(classes):
18. path = os.path.**join**(cur\_path,'train',**str**(i))
19. images = os.**listdir**(path)
20. for a in images:
21. try:
22. image = Image.**open**(path + '\\'+ a)
23. image = image.**resize**((30,30))
24. image = np.**array**(image)
25. #sim = Image.fromarray(image)
26. data.**append**(image)
27. labels.**append**(i)
28. except:
29. **print**("Error loading image")
30. #Converting lists into numpy arrays
31. data = np.**array**(data)
32. labels = np.**array**(labels)
33. **print**(data.shape, labels.shape)
34. #Splitting training and testing dataset
35. X\_train, X\_test, y\_train, y\_test = **train\_test\_split**(data, labels, test\_size=0.2, random\_state=42)
36. **print**(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)
37. #Converting the labels into one hot encoding
38. y\_train = **to\_categorical**(y\_train, 43)
39. y\_test = **to\_categorical**(y\_test, 43)
40. #Building the model
41. model = **Sequential**()
42. model.**add**(**Conv2D**(filters=32, kernel\_size=(5,5), activation='relu', input\_shape=X\_train.shape[1:]))
43. model.**add**(**Conv2D**(filters=32, kernel\_size=(5,5), activation='relu'))
44. model.**add**(**MaxPool2D**(pool\_size=(2, 2)))
45. model.**add**(**Dropout**(rate=0.25))
46. model.**add**(**Conv2D**(filters=64, kernel\_size=(3, 3), activation='relu'))
47. model.**add**(**Conv2D**(filters=64, kernel\_size=(3, 3), activation='relu'))
48. model.**add**(**MaxPool2D**(pool\_size=(2, 2)))
49. model.**add**(**Dropout**(rate=0.25))
50. model.**add**(**Flatten**())
51. model.**add**(**Dense**(256, activation='relu'))
52. model.**add**(**Dropout**(rate=0.5))
53. model.**add**(**Dense**(43, activation='softmax'))
54. #Compilation of the model
55. model.**compile**(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])
56. epochs = 15
57. history = model.**fit**(X\_train, y\_train, batch\_size=32, epochs=epochs, validation\_data=(X\_test, y\_test))
58. model.**save**("my\_model.h5")
59. #plotting graphs for accuracy
60. plt.**figure**(0)
61. plt.**plot**(history.history['accuracy'], label='training accuracy')
62. plt.**plot**(history.history['val\_accuracy'], label='val accuracy')
63. plt.**title**('Accuracy')
64. plt.**xlabel**('epochs')
65. plt.**ylabel**('accuracy')
66. plt.**legend**()
67. plt.**show**()
68. plt.**figure**(1)
69. plt.**plot**(history.history['loss'], label='training loss')
70. plt.**plot**(history.history['val\_loss'], label='val loss')
71. plt.**title**('Loss')
72. plt.**xlabel**('epochs')
73. plt.**ylabel**('loss')
74. plt.**legend**()
75. plt.**show**()
76. plt.**figure**(2)
77. plt.**plot**(history.history['accuracy'], label='training accuracy')
78. plt.**plot**(history.history['loss'], label='training loss')
79. plt.**title**('Training')
80. plt.**xlabel**('epochs')
81. plt.**ylabel**('accuracy/loss')
82. plt.**legend**()
83. plt.**show**()
84. plt.**figure**(3)
85. plt.**plot**(history.history['val\_accuracy'], label=’val\_accuracy’)
86. plt.**plot**(history.history['val\_loss'], label='val loss')
87. plt.**title**('Validation')
88. plt.**xlabel**('epochs')
89. plt.**ylabel**('loss/accuracy')
90. plt.**legend**()
91. plt.**show**()
92. #testing accuracy on test dataset
93. from sklearn.metrics import accuracy\_score
94. y\_test = pd.**read\_csv**('Test.csv')
95. labels = y\_test["ClassId"].values
96. imgs = y\_test["Path"].values
97. data=[]
98. for img in imgs:
99. image = Image.**open**(img)
100. image = image.**resize**((30,30))
101. data.**append**(np.**array**(image))
102. X\_test=np.**array**(data)
103. pred = model.**predict\_classes**(X\_test)
104. #Accuracy with the test data
105. from sklearn.metrics import accuracy\_score
106. **print**(**accuracy\_score**(labels, pred))
107. model.**save**(‘traffic\_classifier.h5’)

**11.2 GUI CODE**

1. import tkinter as tk
2. from tkinter import filedialog
3. from tkinter import \*
4. from PIL import ImageTk, Image
5. import numpy
6. #load the trained model to classify sign
7. from keras.models import load\_model
8. model = **load\_model**('traffic\_classifier.h5')
9. #dictionary to label all traffic signs class.
10. classes = { 1:'Speed limit (20km/h)',
11. 2:'Speed limit (30km/h)',
12. 3:'Speed limit (50km/h)',
13. 4:'Speed limit (60km/h)',
14. 5:'Speed limit (70km/h)',
15. 6:'Speed limit (80km/h)',
16. 7:'End of speed limit (80km/h)',
17. 8:'Speed limit (100km/h)',
18. 9:'Speed limit (120km/h)',
19. 10:'No passing',
20. 11:'No passing veh over 3.5 tons',
21. 12:'Right-of-way at intersection',
22. 13:'Priority road',
23. 14:'Yield',
24. 15:'Stop',
25. 16:'No vehicles',
26. 17:'Veh > 3.5 tons prohibited',
27. 18:'No entry',
28. 19:'General caution',
29. 20:'Dangerous curve left',
30. 21:'Dangerous curve right',
31. 22:'Double curve',
32. 23:'Bumpy road',
33. 24:'Slippery road',
34. 25:'Road narrows on the right',
35. 26:'Road work',
36. 27:'Traffic signals',
37. 28:'Pedestrians',
38. 29:'Children crossing',
39. 30:'Bicycles crossing',
40. 31:'Beware of ice/snow',
41. 32:'Wild animals crossing',
42. 33:'End speed + passing limits',
43. 34:'Turn right ahead',
44. 35:'Turn left ahead',
45. 36:'Ahead only',
46. 37:'Go straight or right',
47. 38:'Go straight or left',
48. 39:'Keep right',
49. 40:'Keep left',
50. 41:'Roundabout mandatory',
51. 42:'End of no passing',
52. 43:'End no passing veh > 3.5 tons' }
53. #initialise GUI
54. top=tk.**Tk**()
55. top.**geometry**('800x600')
56. top.**title**('Traffic sign classification')
57. top.**configure**(background='#CDCDCD')
58. label=**Label**(top,background='#CDCDCD', font=('arial',15,'bold'))
59. sign\_image = **Label**(top)
60. def **classify**(file\_path):
61. global label\_packed
62. image = Image.**open**(file\_path)
63. image = image.**resize**((30,30))
64. image = numpy.**expand\_dims**(image, axis=0)
65. image = numpy.**array**(image)
66. pred = model.**predict\_classes**([image])[0]
67. sign = classes[pred+1]
68. **print**(sign)
69. label.**configure**(foreground='#011638', text=sign)
70. def **show\_classify\_button**(file\_path):
71. classify\_b=**Button**(top,text="Classify Image",command=lambda: **classify**(file\_path),padx=10,pady=5)
72. classify\_b.**configure**(background='#364156', foreground='white',font=('arial',10,'bold'))
73. classify\_b.**place**(relx=0.79,rely=0.46)
74. def **upload\_image**():
75. try:
76. file\_path=filedialog.**askopenfilename**()
77. uploaded=Image.**open**(file\_path)
78. uploaded.**thumbnail**(((top.**winfo\_width**()/2.25),(top.**winfo\_height**()/2.25)))
79. im=ImageTk.**PhotoImage**(uploaded)
80. sign\_image.**configure**(image=im)
81. sign\_image.image=im
82. label.**configure**(text='')
83. **show\_classify\_button**(file\_path)
84. except:
85. pass
86. upload=**Button**(top,text="Upload an image",command=upload\_image,padx=10,pady=5)
87. upload.**configure**(background='#364156', foreground='white',font=('arial',10,'bold'))
88. upload.**pack**(side=BOTTOM,pady=50)
89. sign\_image.**pack**(side=BOTTOM,expand=True)
90. label.**pack**(side=BOTTOM,expand=True)
91. heading = **Label**(top, text="Know Your Traffic Sign",pady=20, font=('arial',20,'bold'))
92. heading.**configure**(background='#CDCDCD',foreground='#364156')
93. heading.**pack**()
94. sub\_heading = **Label**(top, text="Project by Nikhil, Ritwik, Sahib & Prathamesh (Dept of ECE, NHCE)",pady=20, font=('arial',20,'bold'))
95. sub\_heading.**configure**(background='#CDCDCD',foreground='#0A1EF2')
96. sub\_heading.**pack**()
97. top.**mainloop**()