AI Course

Team Project Final Report

For students (instructor review required)

ⓒ2019 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of this document.

This document is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this document other than the curriculum of Samsung innovation Campus, you must receive written consent from copyright holder.

|  |
| --- |
| < Traffic Sign Recognition System Using CNN> |

< 12/10/22 >

Dark Web Crusaders

<Mirza Toheed Arsal>

<Shamaem Saqib>

Content

1. Introduction

1.1. Background Information

1.2. Motivation and Objective

1.3. Members and Role Assignments

1.4. Schedule and Milestones

2. Project Execution

2.1. IoT Service Model

2.2. Data Processing

2.3. Service Implementation

2.4. System Diagram

3. Results

3.1. Data Acquisition (Sensor, Actuator, Controller)

3.2. Network & Communication

3.3. Hardware Implementation

3.4. Data Visualization

3.5. Testing and Improvements

4. Projected Impact

4.1. Accomplishments and Benefits

4.2. Future Improvements

5. Team Member Review and Comment

6. Instructor Review and Comment

1. Introduction

* 1. Background Information

Traffic Signs play a significant role in managing the traffic on the roads, avoiding any kind of

road accidents and ensuring the safety of people by conveying different rules that need to be

followed on the road. The annual globe roach crash statistics say that over 3280 people die

every day in a road accident. These numbers would be much higher in case if there were no

traffic signs. There are several types of traffic signs on the roads like speed limits, not entry,

traffic signals, turn to left or right, children crossing, parking, no parking etc. In this era of

artificial intelligence, humans are becoming more dependent on technology. With the increasing

demand for the intelligence of vehicles in automobile industry and increased production of self

driving cars by many multinational companies’ like Google, Tesla, Uber, Ford, Toyota Mercedes-Benz and many others are working on driverless cars with help of enhanced technology. You have might heard about self-driving cars, where the vehicle itself behaves like a driver and does not need any human guidance and assistance on the roads. So the companies have also think about the safety aspects as there All these companies are trying to make more accurate autonomous or driverless vehicles. it is extremely necessary to detect and recognize the traffic signs automatically through enhanced computer technology to reduce the accidents on roads, make the flow of

traffic smooth and ensure the safety of the other people on the road. Hence, it is very important to design an automatic real time driver assistance system to detect and recognize traffic signs in

automobiles, most specifically in driverless automobiles.

1.2 Motivation and Objective

Traffic signs such as speed limit, turn left, parking etc. play a significant role in managing the

road traffic, avoiding any accidents and ensuring safety by conveying different rules to be

followed on the roads. In this era of artificial intelligence, humans are becoming more dependent on technology. Many Multinational automobile companies like Google, Tesla, Uber.

Ford, Toyota, Mercedes-Benz and many other are working on driverless cars with the help of

enhanced technology. With the increasing demand for the intelligence of vehicles in

automobile industry, and increased production of self-driving cars, it is extremely necessary to

detect and recognize the traffic signs automatically through computer technology. For this,

neural networks can be applied for analyzing visuals of the traffic signs for cognitive decision

making by automatic vehicles. Neural networks are the computing systems which act as a means of performing machine learning. In the system, a convolutional neural network (CNN)

based machine learning model is built and trained for the recognition of traffic signs in the driverless cars.

Traffic Sign recognition is very important and could potentially be used for driver assistance to reduce the accidents in driverless cars. Traffic sign detection and recognition is an important feature for driver assistance, contributing to safety of drivers, pedestrians and vehicle. For this purpose, we build a CNN model (convolution neural network).

1.3 Members and Role Assignments

|  |  |
| --- | --- |
| **Member Name** | **Role Assignments** |
| Mirza Toheed Arsal | Building , training, tuning and testing the model |
| Shamaem Saqib | Data And Preprocessing And User Interface |

* 1. Schedule and Milestones

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Time taken (days)** | **Done By** |
| Exploring Generalized Traffic Sign Dataset | 1 day | Shamaem Saqib |
| Data Acquisition and EDA | 2 day | Shamaem Saqib |
| Pre Processing | 2 day | Shamaem Saqib |
| Building the model | 1 day | Mirza Toheed Arsal |
| Training and its evaluation | 1 day | Mirza Toheed Arsal |
| Testing and its evaluation | 1 day | Mirza Toheed Arsal |
| User Interface | 2 day | Shamaem Saqib |
| Final Testing | 1 day | Mirza Toheed Arsal |

2. Project Execution

2.1 Data Acquisition

There are many generalized dataset of Traffic Signs are available such as GTSRB, GTSDB, BTSCB, BTSDB. But the most common dataset is the GTSRB (German Traffic Sign Recognition Benchmark) dataset because it contains large number of traffic signs images (50,000 images and 43 classes) of different variety, background, and color variation which help to model to perform accurately. The dataset is further spilt into training, testing and validation dataset. The training dataset is used to train the model. The validation dataset is used to evaluate the model. The test dataset is used to check whether the model make correct predictions or not and it is only used once the model is trained. Further, histogram is plotted to show number of images in each class, for training, testing and validation datasets.

2.2 Training Methodology

The steps which will be followed during implementation method are as follow:

1. Exploring Generalized Traffic Sign Dataset
2. Building a CNN Image recognition model
3. Training the model along with validations
4. Testing the trained model via test dataset

The approach followed for building this computer vision model involves importing the data

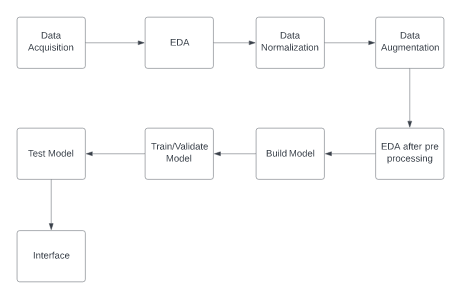
Set and loading it, performing exploratory data analysis (EDA) on dataset and visualizing

The dataset variables and their characteristics, Pre-processing data so that model is finely

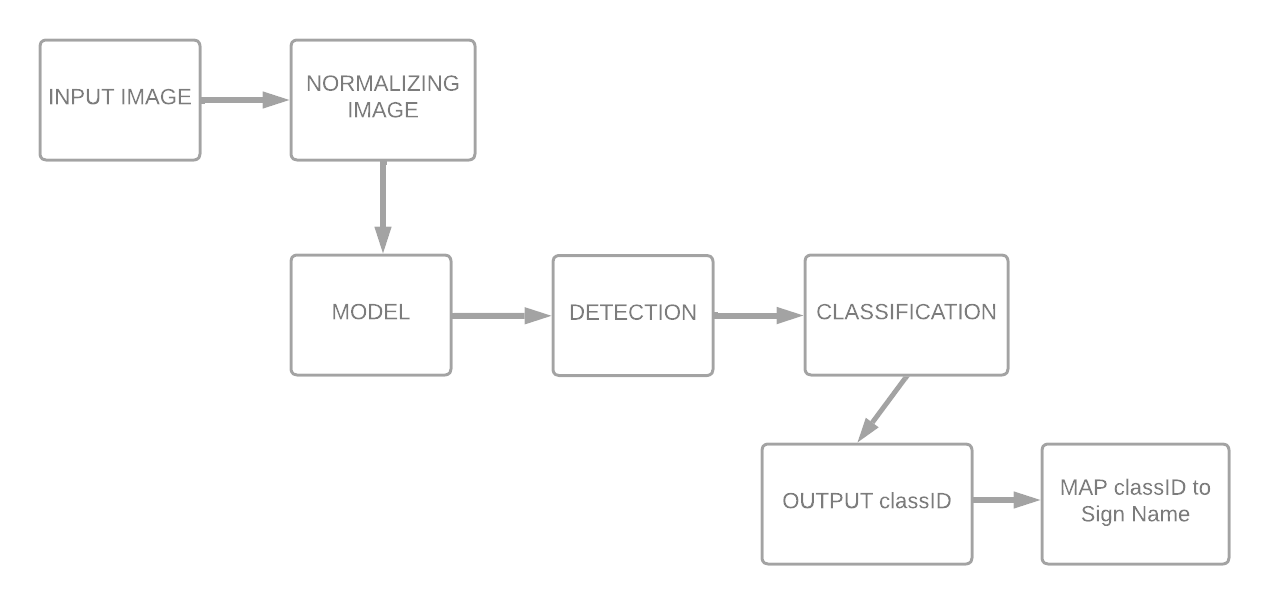
Tuned, signing the model using CNN and training and testing the designed model with

Testing samples.

2.3 Workflow



2.4 System Diagram



3. Results

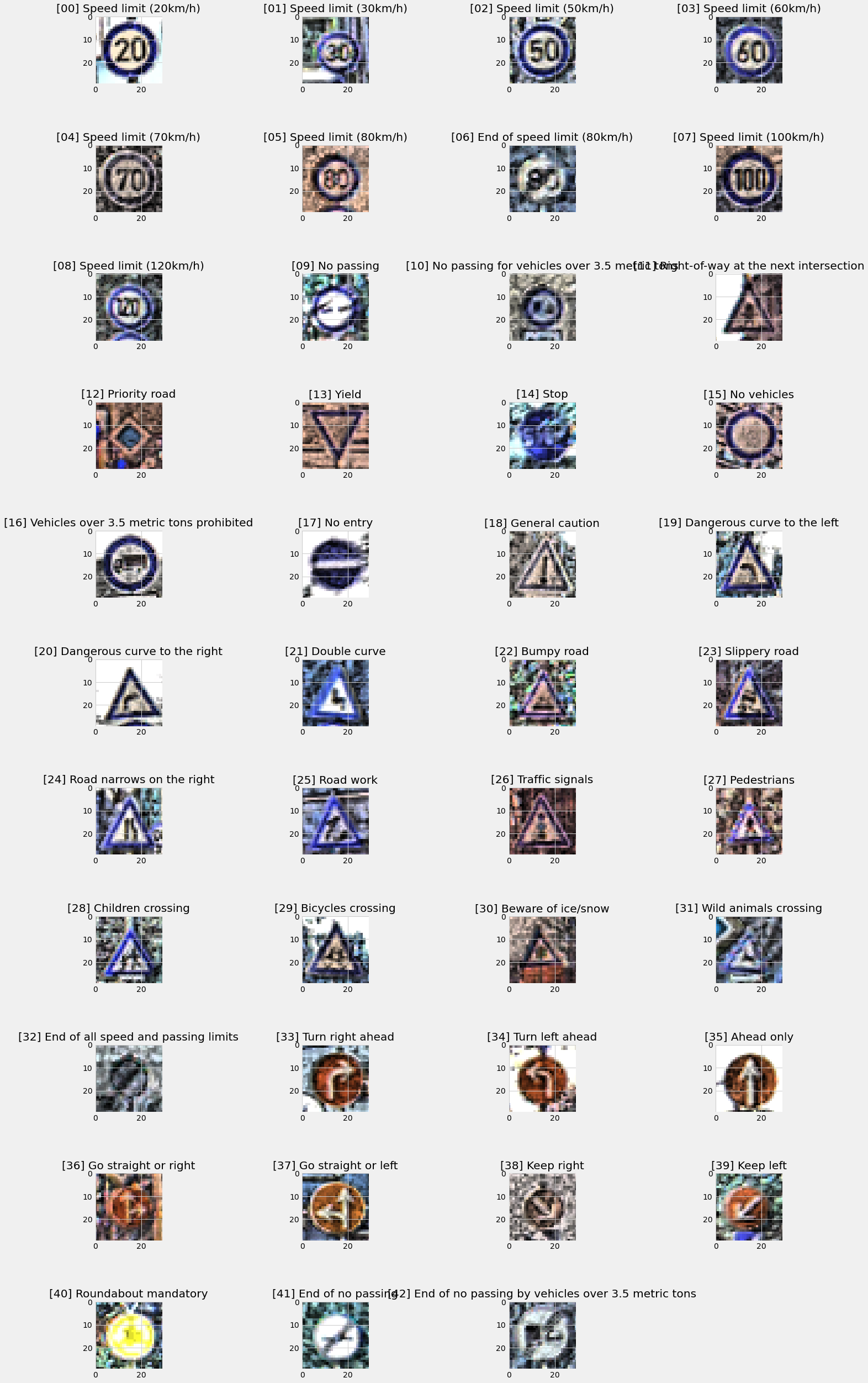
3.1. Data Preprocessing

Given the issues identified below in EDA section, I decided to explore the following preprocessing operations (in addition to data normalization):

3.1.1. Contrast enhancement / Data Normalization

I used a Scikit histogram equalization function, which not only normalizes the images but also enhances local contrast details in regions that are darker or lighter than most of the. You can see from the image sample below this also inherently increases the brightness of the image.

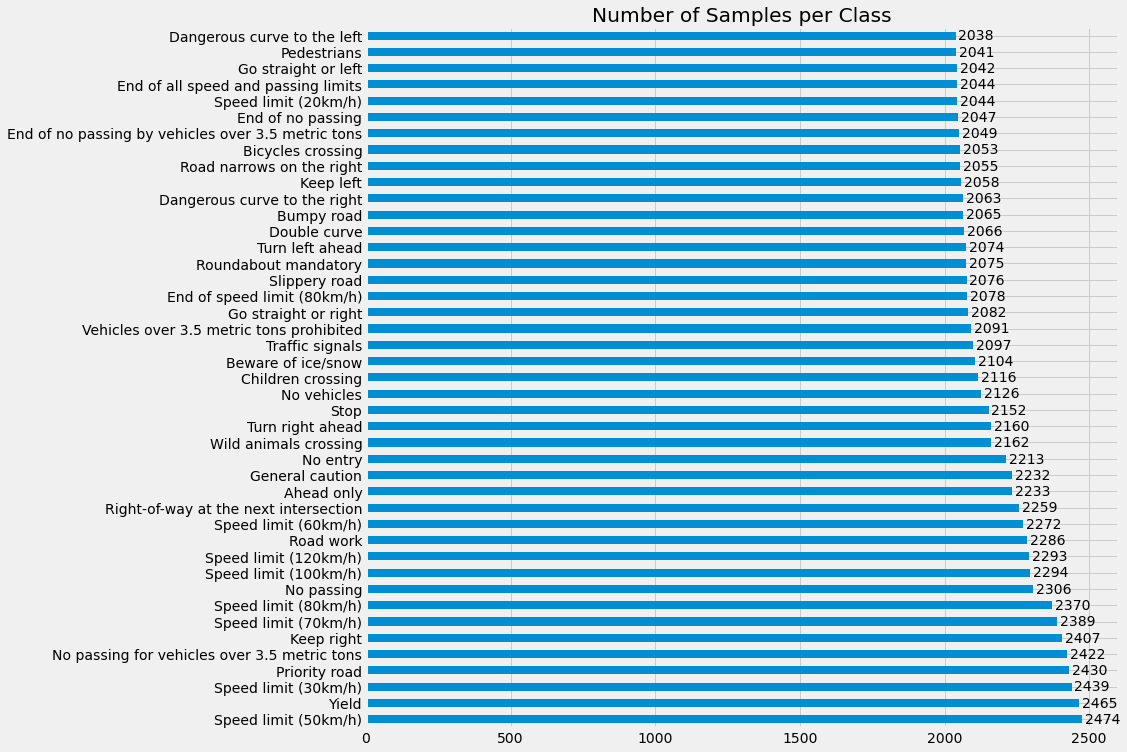




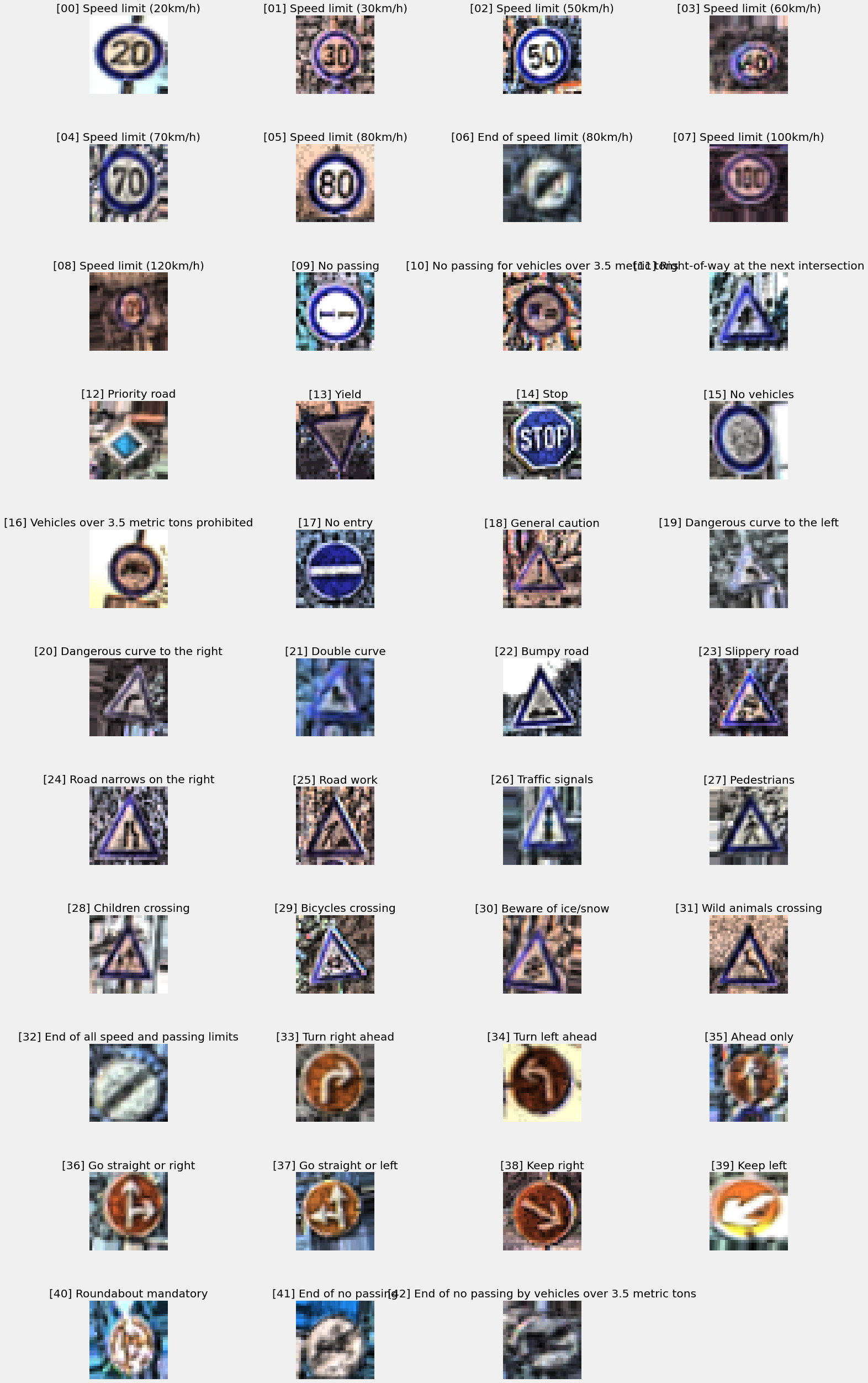
3.1.2. Data Augmentation

Did the following to augment dataset

* **Create an equal distribution of images** i.e., the same number of images per class, so that the model has a sufficient number of training examples in each class. I generated a set of 2k images per class for the final model.



* **Apply affine transformations.** Used to generate images with various sets of perturbations. Specifically: rotation, shift, shearing, and zoom. But, I decided not to apply horizontal/vertical flipping as this didn’t seem applicable to real-life use cases.

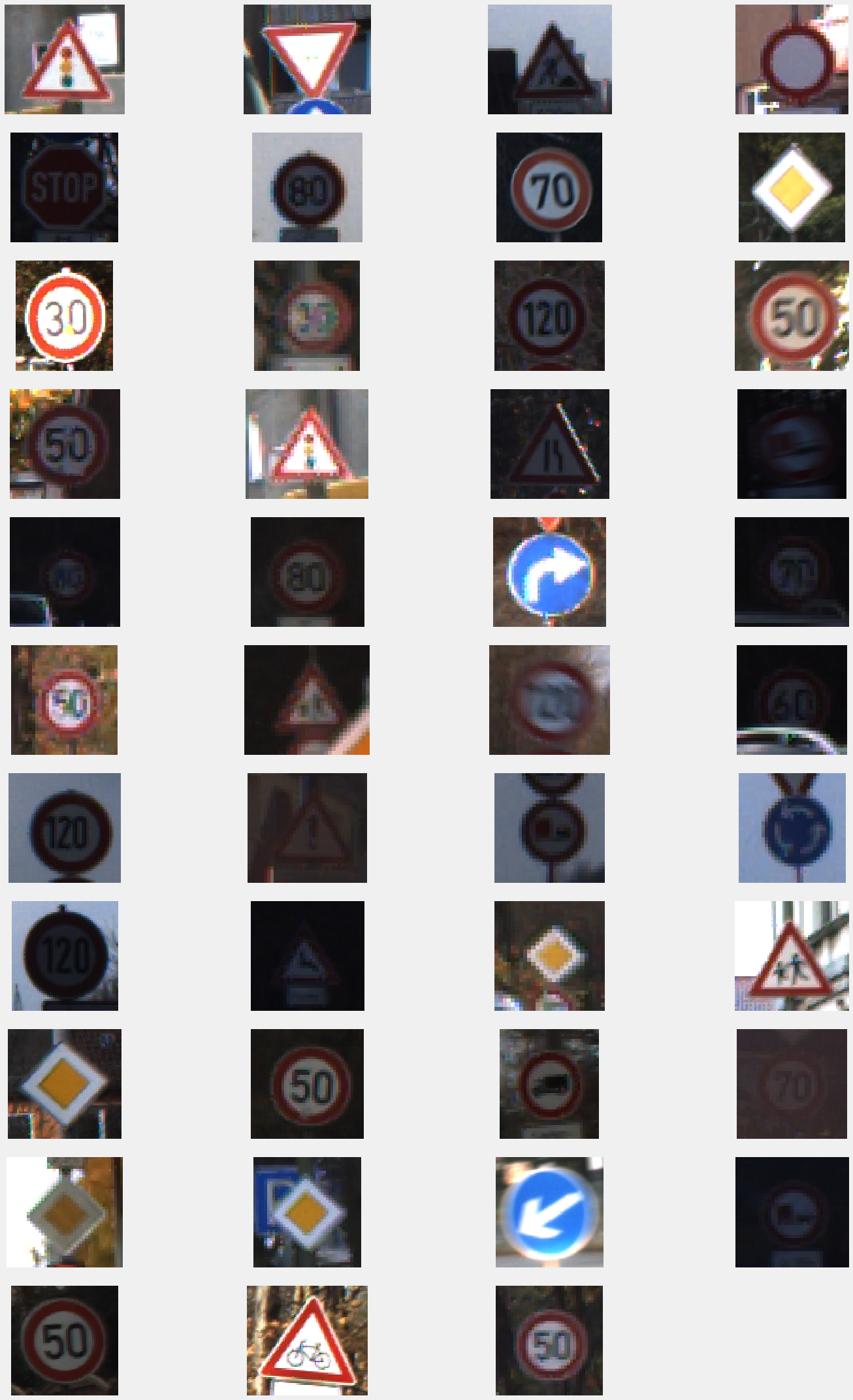


* 1. Exploratory Data Analysis (EDA)
     1. Folder Structure of Dataset
* The data-set contains Train and Test Folder. Train folder has 43 different folders named from 0 to 42. Each folder contains images from the perspective class.
* There are 39,209 train images and 12,630 test images. All images are 3- channel RGB-image
* There are also Train.csv and Test.csv files which contains information about the train and test images.
* All the images are distributed in 43 classes.
  + 1. Data Size and Shape before pre-processing
* Number of training examples: 31367
* Number of validation examples: 7842
* Number of testing examples: 12630
* Image data shape: (30, 30, 3)
* Number of labels/classes: 43
  + 1. Data Size and Shape after pre-processing
* Number of training examples = 86000
* Number of validation examples = 7842
* Number of test examples = 12630
* Image data shape = (30, 30, 3)
* Number of labels/classes = 43
  + 1. Data Visualization before pre-processing

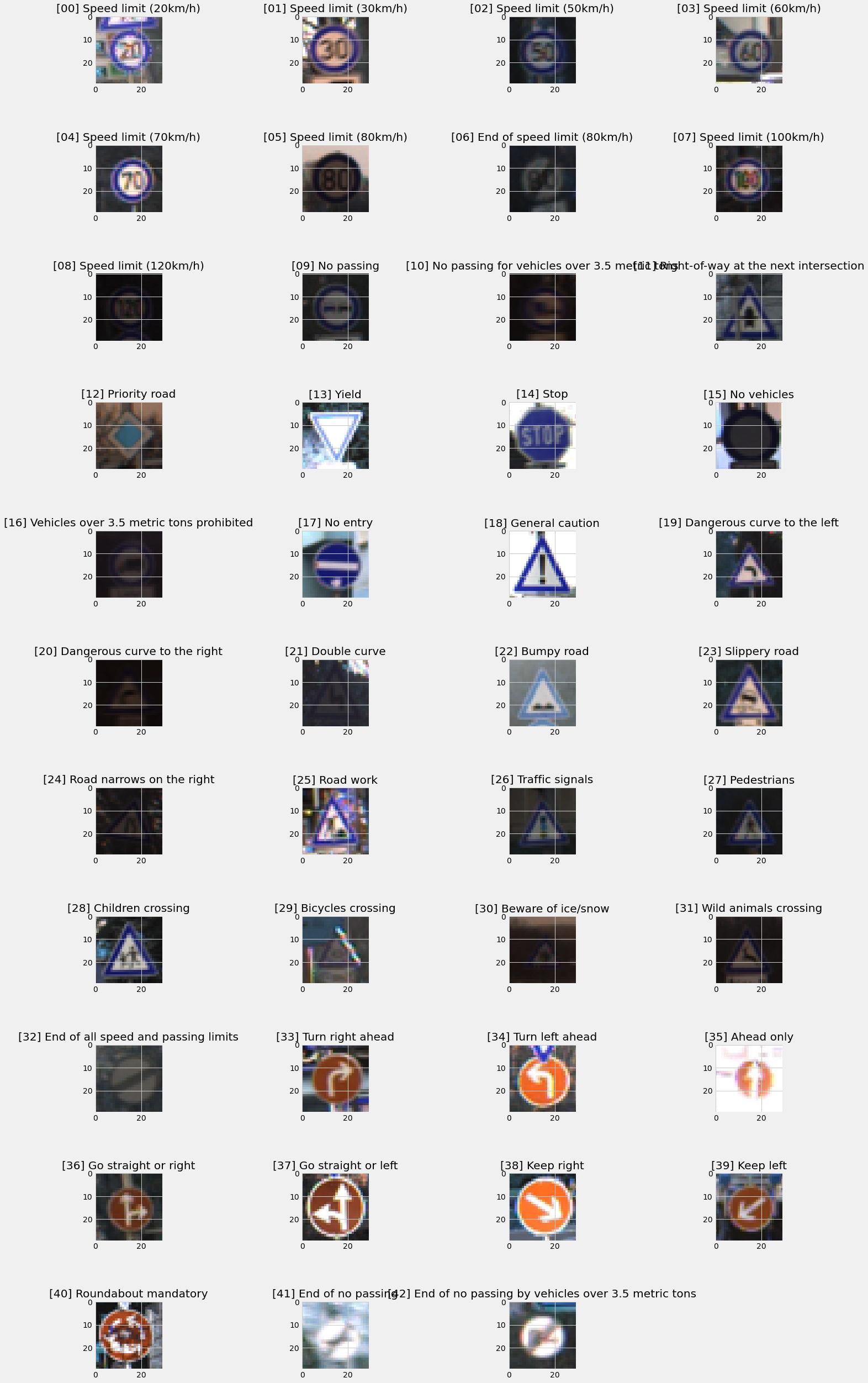
Before designing the model and start training it, it is really important to visualize the data in various ways to gain some intuition for what the model will “see.” This not only informs the model structure and parameters, but it also helps determine what types of preprocessing operations should be applied to the data (if any).

I Visualized the data through the following way:

* **Inspect a sample of images:** Inspected a sample of 25 images to take a peek of what the data looks like.



* **Image & Label Sample:** Inspected an image from each class to know whether do the labels make sense? Do they accurately correspond with images in the data set?



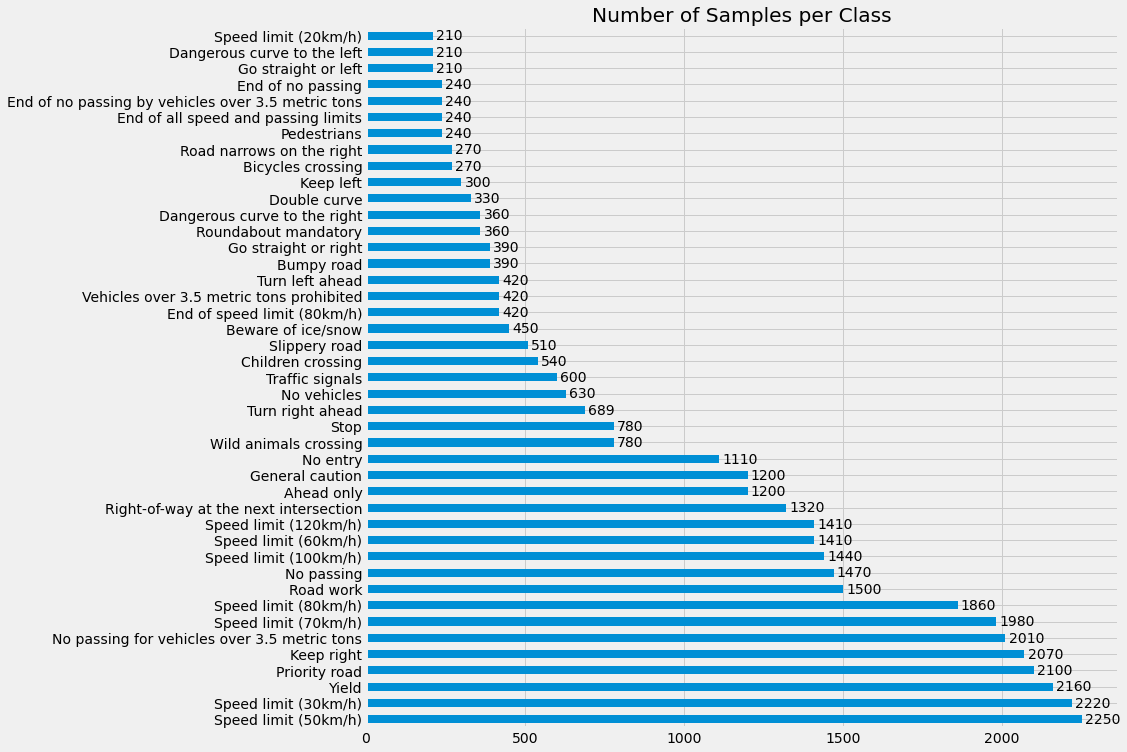
However, immediately you can notice a few things that can be adjusted during preprocessing:

* The issue of **images being dark and of low contrast** makes many signs hard to recognize.
* There is **little variation in the sign shape and viewing angle**. Most of the pictures are

taken with a straight on view of the sign, which is good for the core data set. However,

in real life, signs are viewed from different angles.

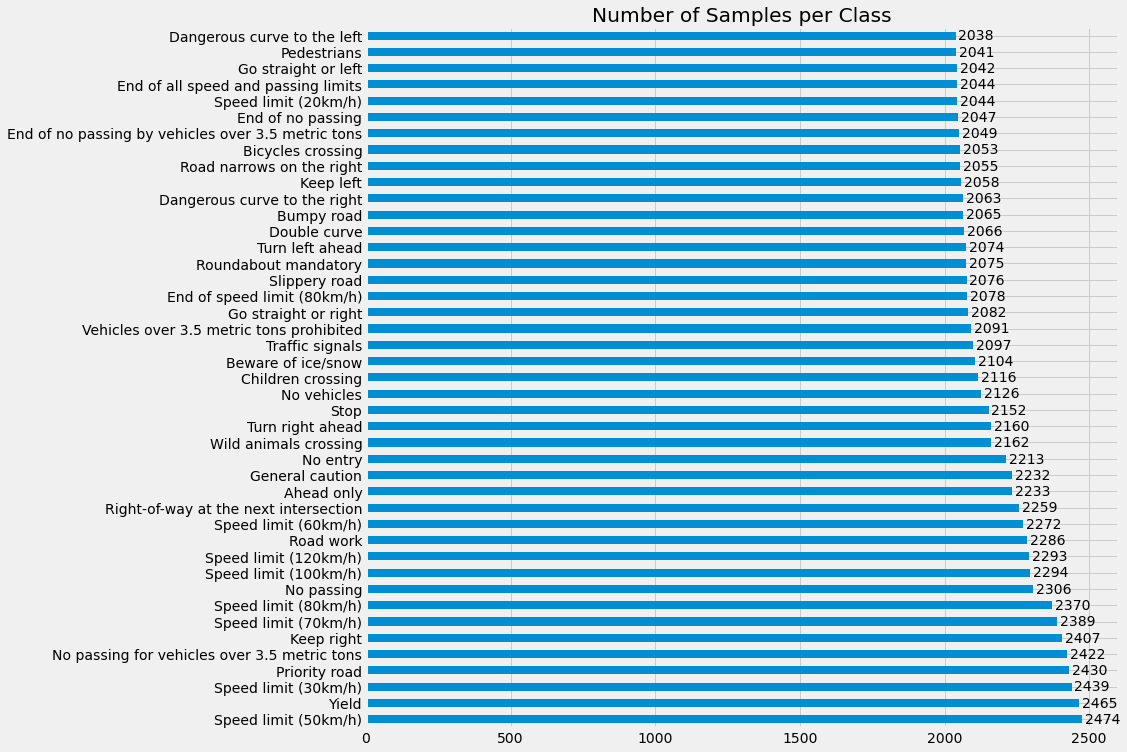
* **Class/Label Distribution:** Create a Bar chart showing the distribution of classes/labels to know whether how balanced is the dataset? Are there certain classes that dominate? Are there others that are under-represented?



As you can see in the above given picture, the distribution is not uniform. The largest classes have 10x the number of traffic sign images than the smallest classes. This is expected given that in real-life there are certain signs which appear more frequently than others. However, when training the model, a more uniform distribution is desirable so that each class has the same number of training examples and the model, therefore, has an equal number of opportunities

to learn each sign.

So, after pre-processing mainly data augmentation, class/image distribution was as follows:



3.3 Modeling

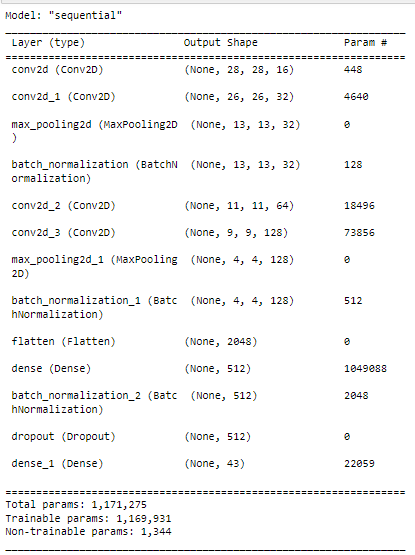
**3.3.1. Building the model**

For building the Traffic Sign Recognition System, we will build a CNN model. As the CNN is the best for classification problems. The architecture of our model is:

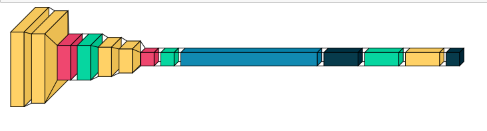
* Conv2D layer (filter=16, kernel\_size= (3,3), activation=” relu”)
* Conv2D layer (filter=32, kernel\_size= (3,3), activation=” relu”)
* MaxPool2D layer (pool\_size= (2,2))
* BatchNormalization(axis=-1)
* Conv2D layer (filter=64, kernel\_size= (3,3), activation=” relu”)
* Conv2D layer (filter=128, kernel\_size= (3,3), activation=” relu”)
* MaxPool2D layer (pool\_size= (2,2))
* BatchNormalization(axis=-1)
* Flatten layer to squeeze the layers into 1 dimension
* Dense Fully connected layer (512 nodes, activation=” relu”)
* Dropout layer (rate-0.5)
* BatchNormalization(axis=-1)
* Dense layer (43 nodes, activation=” softmax”)

43 nodes as we have 43 classes and for dense layer the activation function is softmax as is a multi-class classification problem. We compile the model with Adam optimizer which performs well and loss is “categorical\_crossentropy” because we have multiple classes to categorize.

The summary of the model is a follow:



Layered View of Model:

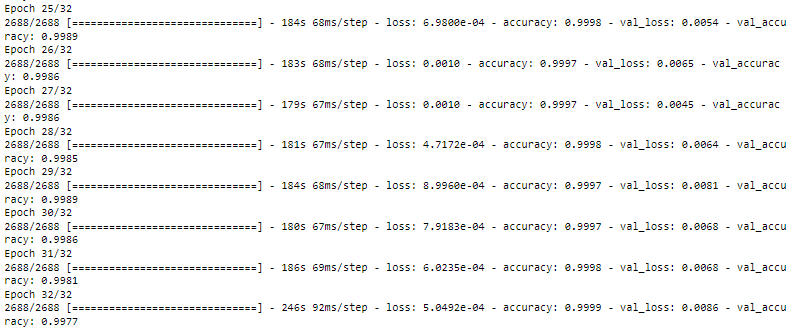


**3.3.2. Train and validate the model:**

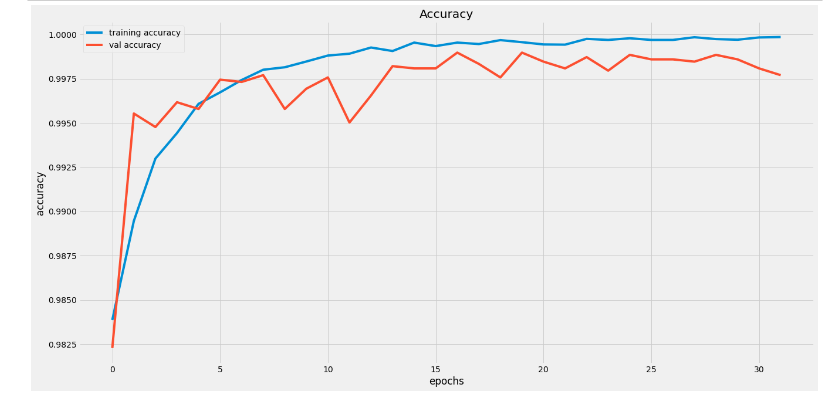
After building the model architecture we then train the model using model. fit (). I tried batch

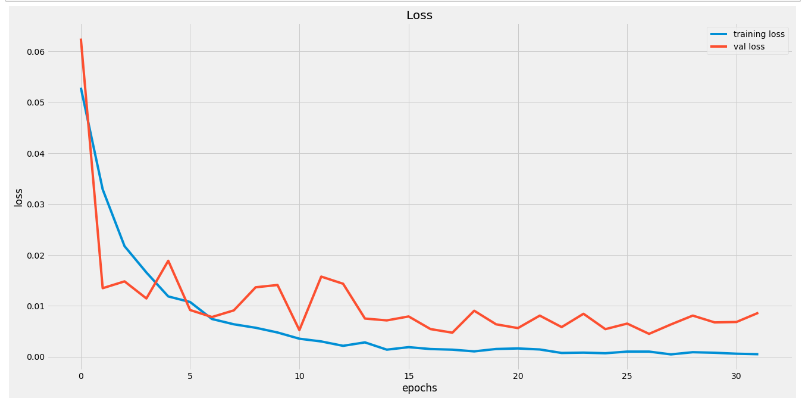
size 32 and 64. Our model performed better with 32 batch size, And after 32 epochs, the

accuracy was stable.



Our model got 99 % accuracy on the validation dataset. With matplotlib, we plot the graph for accuracy and the loss.





**Final Model Results:**

* training accuracy: 100%
* validation accuracy: 99%
* test accuracy: 98.34%

3.4 User Interface (Interface).

We are going to build a graphical user interface for out traffic signs classifier with Tkinter.

Tinkter is a GUI toolkit in the standard python library. We will make a new file in the project

Folder and save is as gui.py and we can run the code by typing python gui.py in the command

line. In this file, we have first loaded the trained model using Keras. And then we build a GUI

for uploading the image and a button is used to classify which calls the classify () function. The

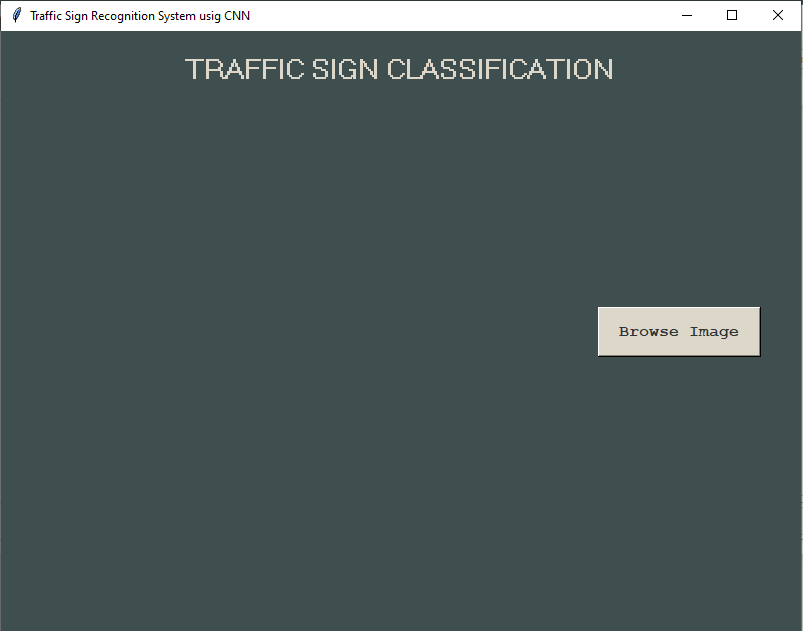
classify () function is useful for converting the image into dimension of shape (30, 30, 3). This

is because to predict the traffic sign we have to provide the same dimension we have used

when building the model, then we predict the class, the model. predict(image) returns

us a number between (0- 42) which represents the class it belongs to. We use the dictionary to

get the information about the class. The interface is shown below in the image. It has a browse image button to upload an image from your system.



After uploading the image from you image, the classify sign button will be displayed.



After clicking the classify sign button, the system will tell you that what sign is this.



Here are some more outputs.





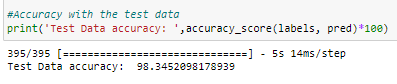


3.5. Testing and Improvements.

**Test our model with test data:**

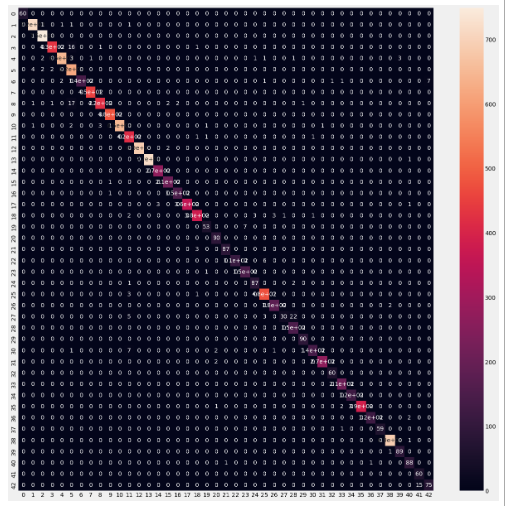
Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30 X30 pixels and make a numpy array containing all image data. From the sklearn. metrices, we imported the accuracy\_score and observed how our model predicted the actual labels. We achieved a 98% accuracy in this

Model.



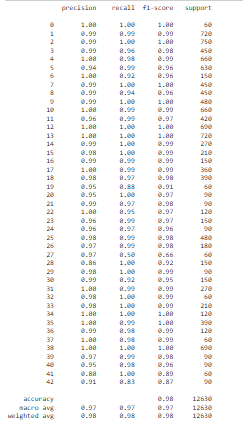
**Confusion Matrix:**

We have also generated the confusion matrix, and display it seaborn. Heatmap. The generated confusion matrix is as follow:



**Classification Report:**

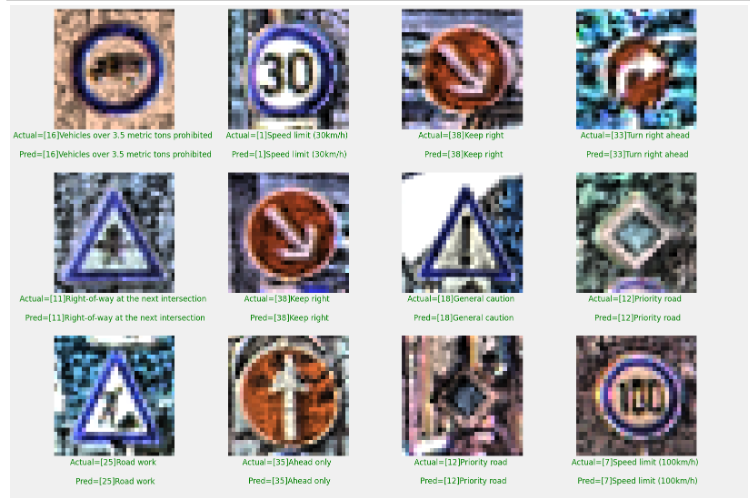
Listed below are the precision, recall and F1 scores for the original set of test images.

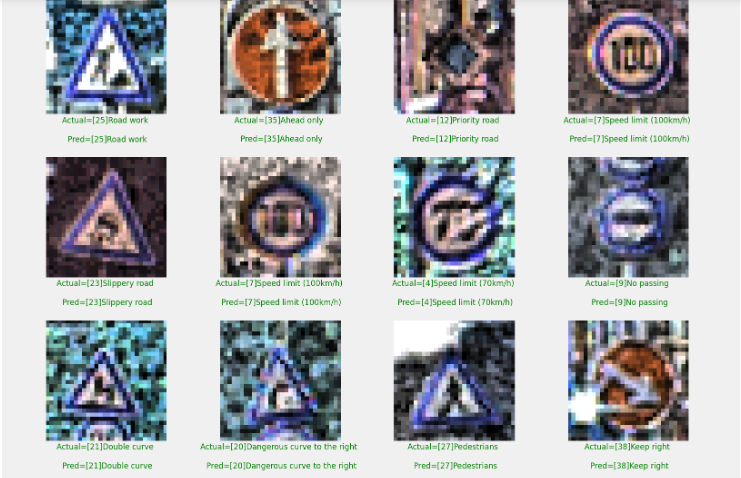


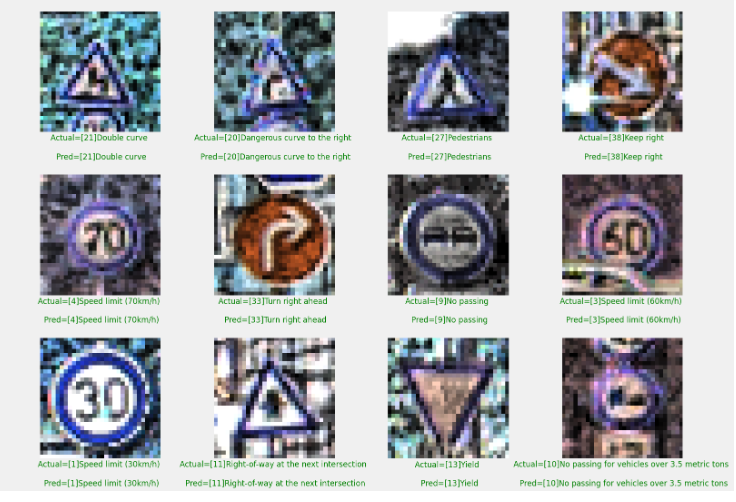
**Predictions On Test Data:**

We have also done predictions on the test data and compare it with actual data. It gives us

100 percent accuracy. For this we have selected 28 images, and he predict all of them correct.







4. Projected Impact

4.1. Accomplishments and Benefits

4.2 Future Improvements

5. Team Member Review and Comment

|  |
| --- |
| <ATTACH A TEAM PICTURE HERE> |

|  |  |
| --- | --- |
| NAME | REVIEW and COMMENT |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

6. Instructor Review and Comment

|  |  |  |
| --- | --- | --- |
| CATEGORY | SCORE | REVIEW and COMMENT |
| IDEA | \_\_/20 |  |
| CODING | \_\_/20 |  |
| PROJECT MANAGEMENT | \_\_/30 |  |
| PRESENTATION & REPORT | \_\_/30 |  |
| TOTAL | \_\_/100 |  |