**TRAFFIC SIGN RECOGNITION**

**A MINI PROJECT REPORT**

***Submitted by***

**Prajwal**

**(4NM20MC062)**

**Prajwal B S**

**(4NM20MC063)**

***of***

**MASTER OF COMPUTER APPLICATIONS**

**NMAM Institute of Technology, Nitte**

**(An Autonomous Institution Affiliated to VTU Belgaum)**



(An Autonomous Institution affiliated to VTU, Belagavi)

Nitte Mahalinga Adyanthaya Memorial Institute of Technology, Nitte

Department of MCA

JANUARY 2022

**NMAM Institute of Technology, Nitte-574110**

**Department of MCA**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Traffic SignRecognition”**is the bonafide

workof“**Prajwal(USN:4NM20MC062),Prajwal B S(USN:4NM20MC063)”**

who carried out the project work under my supervision.

SIGNATURE SIGNATURE

**Dr. Surendra ShettyProject Guide**

**Professor & HeadMrs. Sarita Shetty**

**Department of MCA**Assistant Professor

**NMAM Institute of Technology, Department of MCA**

**NitteNMAM Institute of Technology,**

**Nitte**

**Abstract**

Traffic sign recognition feature is widely employed in industry today by researchers working in artificial intelligence and machine learning fields targeted to create an autonomous driving assistance system. This will help the driver’s human eye and manoeuvring capabilities which can’t be presumed to be consistent for high accuracy for all weather condition challenges. As the limitation of the human eye perception on a road sign is upsetting the road safety.

The traffic signs are images captured on the road by static people or moving automated cars and vehicles. This classification system is extremely useful in automated vehicles that use sensors and cameras to an4alyze their surrounding environment and take decisions.

The dataset used is the benchmarked “German Traffic Sign Recognition” dataset.

It has traffic sign images for training and testing purposes. A very crucial part of

the project is pre-processing the dataset to maximize accuracy. We use techniques

like normalization and data augmentation to make the image more contrasted and

clearer and to make the system more robust. The proposed approach consists of

building a model using convolutional neural networks by extracting traffic signs

from an image using colour information. We have used convolutional neural

networks (CNN) to classify the traffic signs and we used colour-based

segmentation to extract/crop signs from images.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**1. INTRODUCTION 1-2**

* 1. **Problem Definition**
  2. **Objectives**
  3. **Challenges**
  4. **Literature Survey**

**2. Software Requirement Specification 3**

**2.1 Functional Requirements**

**2.2 Non-functional Requirements**

**2.3 Hardware and Software Requirements**

**3. System Design 4-6**

**3.1 High Level Design**

**3.2 Detailed Design**

**4. System Implementation 7-13**

**4.1 Module Description**

**4.2 Tools and Language**

**4.3 Screen Shots**

**5. Testing 14**

**6. Conclusion and Future enhancements 15-16**

**Reference**

**1. INTRODUCTION**

Traffic sign classification is an artificial intelligence project to identify or divide the traffic signs into their respective classes. Traffic sign classification involves collecting and analysis of traffic signs images in the early stages. The traffic signs are images captured on the road by static people or moving automated cars and vehicles. This classification system is extremely useful in automated vehicles that use sensors and cameras to an4alyze their surrounding environment and take decisions. It is very important to classify the seen traffic signs along the road into their specific classes like “60 kmph speed limit” sign or a “school zone” sign.

The dataset used is the benchmarked “German Traffic Sign Recognition” dataset. It has traffic sign images for training and testing purposes. A very crucial part of the project is pre-processing the dataset to maximize accuracy. We use techniques like normalization and data augmentation to make the image more contrasted and clearer and to make the system more robust. Then we use Convolutional Neural networks to build the model. CNN is used in the system because it is the best fit for analysing image datasets.

* 1. **Problem Definition:**

Traffic Sign Recognition project is to build a Convolutional Neural Network (CNN) which is used to classify traffic signs. We should train the model so it can decode traffic signs from natural images using the [German Traffic Sign Dataset](http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset). This data should be firstly pre-processed in order to maximize the model performance. After choosing model architecture, fine tuning and training, the model will be tested on new images of traffic signs found on the web. Because we deal with images classification, a Convolutional Neural Network is chosen as a type of DNN, which is a common choice for this type of problems. The code is written in Python with use of TensorFlow library. It’s great for making quick, high-level changes in our model architecture. In addition, TensorFlow supports computation on GPU which can really speed up required calculations.

* 1. **Objectives**

The main objective of our project is to design and construct a computer-based system which can automatically detect the traffic signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using colour information. We have used convolutional neural networks (CNN) to classify the traffic signs and we used colour-based segmentation to extract/crop signs from images.

* 1. **Challenges**

The problem we are trying to solve has some advantages such as traffic signs being unique thereby resulting in object variations being small and traffic signs are clearly visible to the driver/system. The other side of the coin is that we have to contend with lighting and weather conditions.

* 1. **Literature Survey**
* A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape and Motion Information.
* Road Sign Detection and Recognition Based on Support Vector Machines.
* An Incremental Framework for Video-based Traffic Sign Detection, Tracking and Recognition.
* Project pipeline consists of 7 steps, which are quite common in classification problems:
* Loading the data
* Dataset exploration and visualization
* Data preprocessing
* Data augmentation
* Designing, training and testing a CNN model
* Using the model on new images
* Analyzing SoftMax probabilities

**2. SOFTWARE REQUIREMENT SPECIFICATION**

**2.1. Functional Requirements:**

* User should be able to give new Image.
* The System must classify the image with accuracy.

**2.2. Non-functional Requirements:**

Maintainability: The software must support for the future updates like to get more traffic signs.

**2.3. Hardware and Software Requirements:**

**2.3.1. Hardware Requirements**

* PC with webcam or other external camera
* Windows 10 or higher version operating system is required.
* Minimum 4GB RAM is necessary.
* Minimum 256GB Storage

**2.3.2.Software Requirements**

* Anaconda 5.0.1
* TensorFlow 0.12.1
* Python (version>3.0)
* OpenCV
* NumPy

**3. SYSTEM DESIGN**

**3.1. High Level Design**

German traffic sign data set

Test

Train

Pre-Processing

Process

Pre-Processing

Process

Train a CNN Model

Learned CNN Model

Classification

Prediction

**3.2. Detailed Design**

Various steps have been followed to build this project. Each step has its own significance. There is total 5 steps that we have followed. They are, Image Collection, Pre-processing, Labelling, Data Training, Testing.

**3.2.1. Image Collection:**

Image collection or data collection is the first step in building the project. The data can be anything. It can be text, image, audio etc. In this project Image is used as our dataset. A huge number of images needs to be collected to get the results accurately.

**3.2.2. Pre-processing:**

Image pre-processing is the process in which some algorithms and operations are performed on the images to enhance the quality of the images. This is the one of the most essential processes. Because the accuracy of the results depends on the how well the images are pre-processed. Pre-processing also helps in the further operations like training and testing.

In this step, we will apply several pre-processing steps to the input images to achieve the best possible results.

* **Shuffling**: In general, we shuffle the training data to increase randomness and variety in training dataset, in order for the model to be more stable. We will use sklearn to shuffle our data.
* **Gray scaling**: Grayscale images instead of colour improves the ConvNet's accuracy. We will use OpenCV to convert the training images into grey scale.
* **Local Histogram Equalization**: This technique simply spreads out the most frequent intensity values in an image, resulting in enhancing images with low contrast. Applying this technique will be very helpful in our case since the dataset in hand has real world images, and many of them has low contrast. We will use skimage to apply local histogram equalization to the training images.
* **Normalization**: Normalization is a process that changes the range of pixel intensity values. Usually, the image data should be normalized so that the data has mean zero and equal variance.

**3.2.3. Labelling:**

Data labelling is a critical step in developing a high-performance ML model. Labelling is part of the pre-processing stage when developing a [machine learning](https://www.ibm.com/cloud/learn/machine-learning#toc-what-is-ma-qhM6PX35) (ML) model. It requires the identification of raw data (i.e., images, text files, videos), and then the addition of one or more labels to that data to specify its context for the models, allowing the machine learning model to make accurate predictions.

**3.2.4. Data Training:**

Data training is the process where the data is inserted into our Machine Learning Model. Training data is the part of data we use to train our model. This is the data that your model actually sees (both input and output) and learns from.

In this project the images undergo various algorithms and the result is stored as excel file. In the excel file the data stored is the labelled data.

**4. SYSTEM IMPLEMENTATION**

**4.1. MODULE DECRIPTION**

**4.1.1. German traffic sign data set:**

The dataset we’ll be using to train our own custom traffic sign classifier is the [German Traffic Sign Recognition Benchmark (GTSRB)](https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign).The GTSRB dataset consists of **43 traffic sign classes**and**nearly 50,000 images.**

We notice how the traffic signs have been pre-cropped for us, implying that the dataset annotators/creators have manually labelled the signs in the images and extracted the traffic sign Region of Interest (ROI) for us, thereby simplifying the project.

**4.1.2. Pre-processing:**

Image pre-processing is the process in which some algorithms and operations are performed on the images to enhance the quality of the images. This is the one of the most essential processes. Because the accuracy of the results depends on the how well the images are pre-processed. Pre-processing also helps in the further operations like training and testing. In this step, we will apply several pre-processing steps to the input images to achieve the best possible results.

As we have taken a prominent data set present in this domain, the images were of good quality. The main pre-processing that we had to do was resizing the images to a lower size for the ease of computation. We did not change the colour of the images as we wanted the images to retain maximum of its properties for the model to learn. Once the images have been resized to the same size, we have converted the images to a NumPy array an appended that array to a list. Similarly, outputs are also converted to integers and appended with another list, depicting the output labels of each image that is being trained.

**4.1.3. Train CNN Model:**

In implementing the model training pipeline, we will typically go through tuning hyperparameters and using training data for training our Deep Learning Algorithm into a model and repeat this process until we are satisfied with the model's performance. Specifically, TensorFlow was used for implementing model training pipeline, and so hyperparameters and variables were set, a one hot encoded feature engineering function was used and a SoftMax cross entropy scorer metric was used in this training pipeline to train the CNN algorithm into a model.

The **learning rate was set to 0.001** since it is a good default value. The learning rate tells TensorFlow how quickly to update the weights. The **epochs were set to 10** to tell TensorFlow to run the training data through the network 10 times to train the model. As a result, training the model did not take too long and the model's validation accuracy was near 100%. The **batch size was set to 400** to tell TensorFlow to run 400 training images at a time. A smaller batch size was chosen to consider processors with memory limitations. The downside of using 400 batches is that the model training would take longer, yet by having the epochs set to 10, then the batch size would not be a problem.

**4.1.4. Learn CNN Model:**

After the successive train of CNN model, we have to test the CNN model with the respective outputs.

**4.1.5. 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 traffic signs enables us to build “smarter cars”.

Self-driving cars need traffic sign recognition in order to properly parse and understand the roadway. Similarly, “driver alert” systems inside cars need to understand the roadway around them to help aid and protect drivers.

Traffic sign recognition is just one of the problems that computer vision and deep learning can solve.

**4.1.6. Predict:**

When the model was trained and evaluated for the first time, it's SoftMax cross entropy scorer metric result for **validation accuracy = 89.5**. However, to meet the requirements for this project, the validation accuracy needed to be **93%** or higher. After trial and error with tuning the hyperparameters for the model, it was determined that changing the **batch size to 64** increased the validation accuracy. Likewise, the **validation accuracy increased**to **95.1%** by adding **dropout regularization layers** at the end of the first two fully connected layers. Finally, the validation accuracy increased to **98.4%** once the training data was augmented with 6 rotations from -15 to +15 degrees followed by normalization applied to that data.

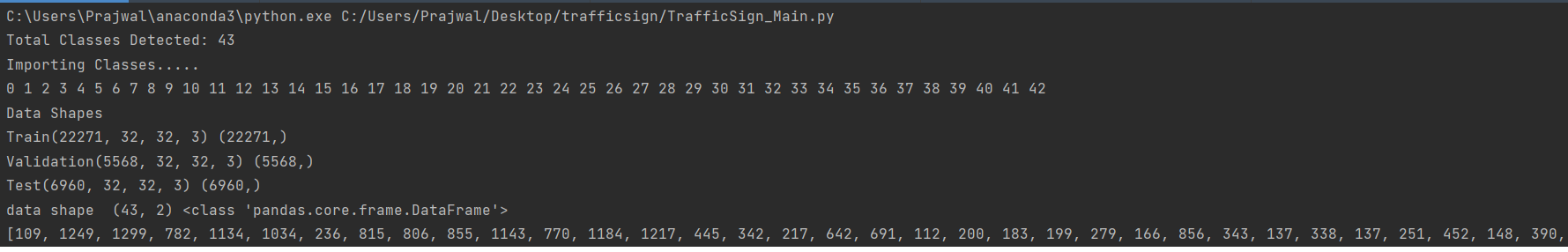
* 1. **Tools and languages used**

To create the model Kaggle’s dataset is used and PyCharm IDE is used to build the recognize the traffic sign.

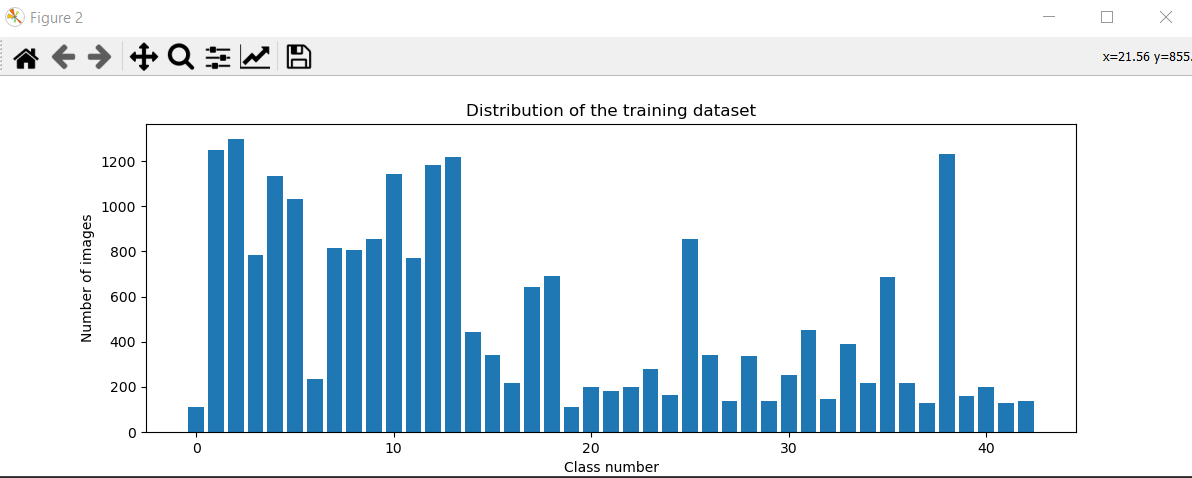
To develop this project, Python has been used. Python has been used because it is very flexible language and has a lot of library support, which eases the work of the programmer.

**4.3. SCREENSHOTS:**

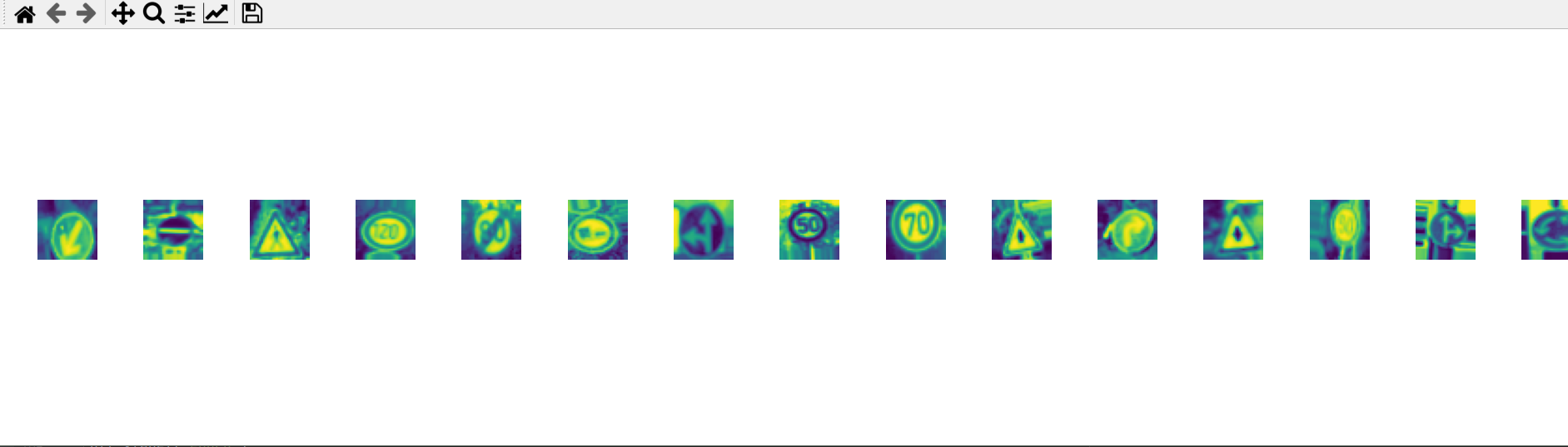
**Importing Classes:**

****

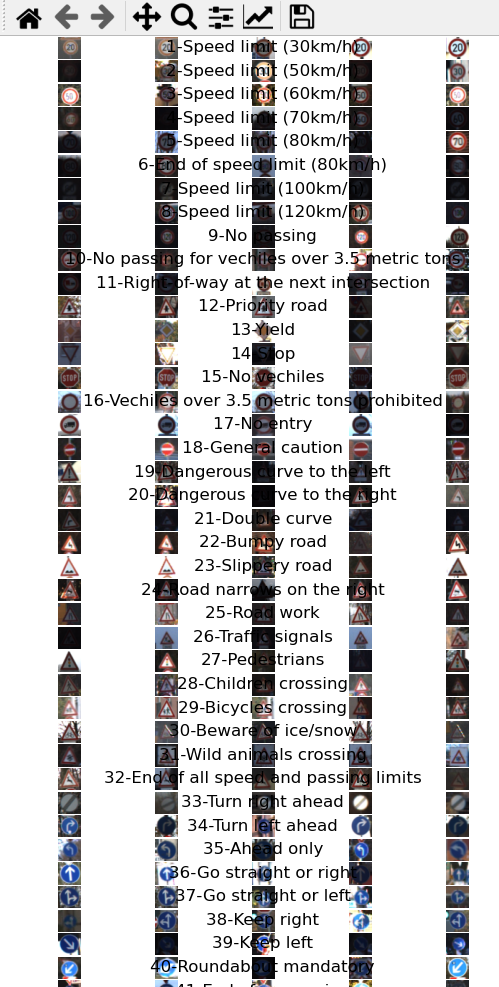
**Distribution of Training Dataset:**

****

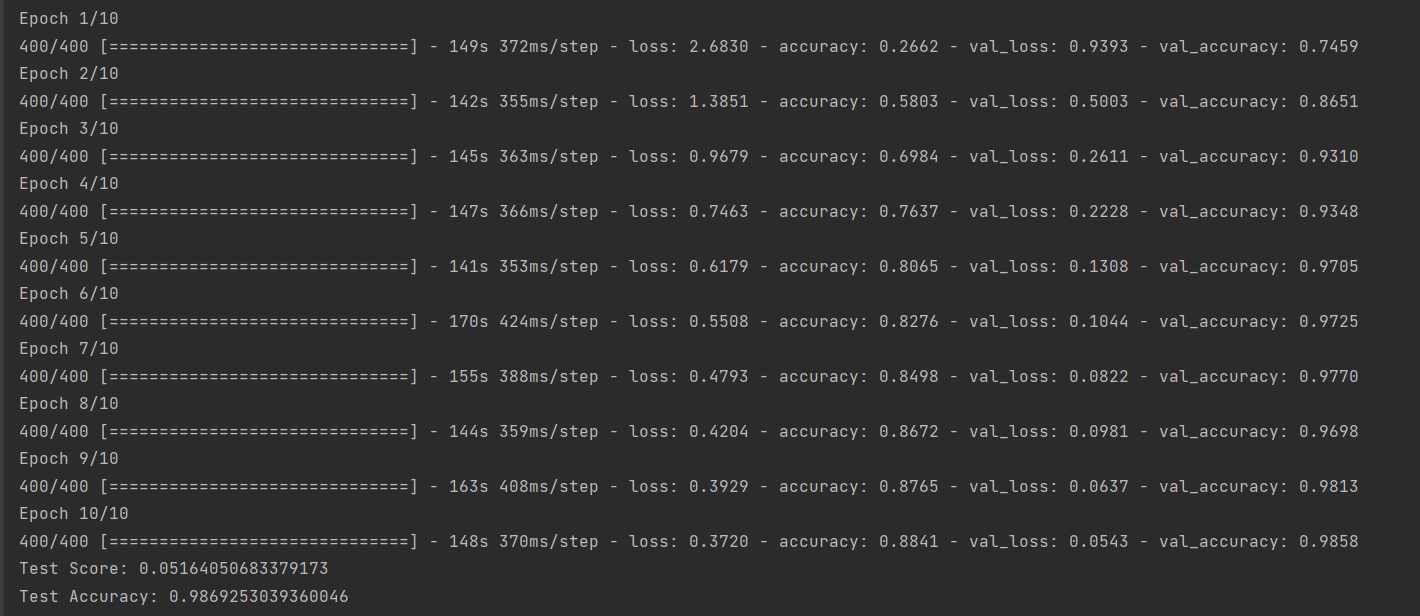
**Image After Pre-processing Process:**

****

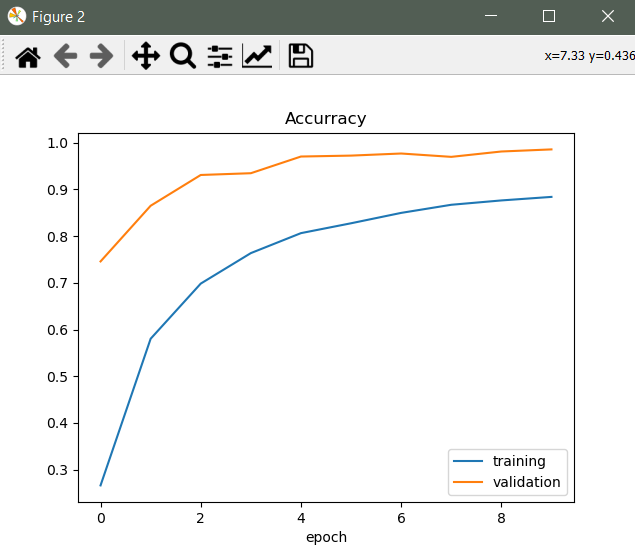
**GTSRB Image Classes:**

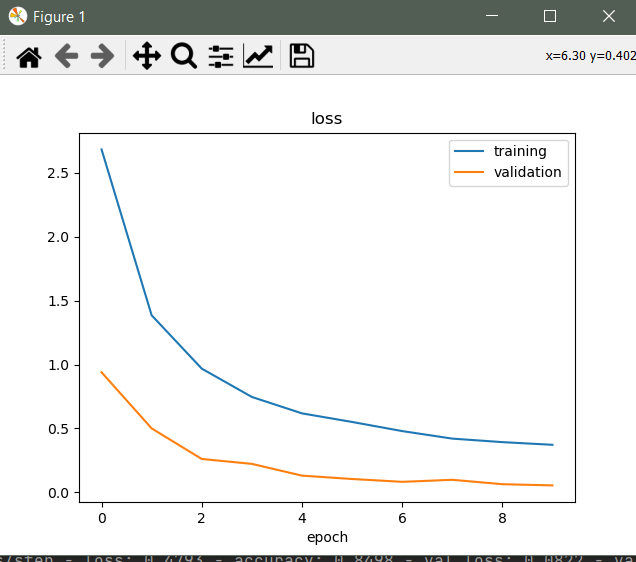
****

**Training and validatethe CNN model:**

****

**Accuracy Graph:**

****

**Loss Graph:**

1. **TESTING**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **Traffic Sign Description** | **Image** | **Result**  **(Success/Fail)** |
| 1 | Stop |  | Success |
| 2 | Speed 80 km/h |  | Success |
| 3 | Turn Right |  | Success |
| 4 | Turn Left |  | Success |
| 5 | Bumpy Road |  | Success |
| 6 | Road Work |  | Success |

**6. CONCLUSION AND FUTURE ENHANCEMENTS**

The pre-processing on the dataset, Feature extraction, Result calculation is done successfully. In the future some new algorithms and models can be incorporated into this system to process even more large amount data and get more accurate results.

This project focused on building a traffic sign recognition application using exploratory data analysis (NumPy, pandas, matplotlib), computer vision (OpenCV) and deep learning (TensorFlow). During this project, I learned to interpret a deep neural network architecture from a research paper and implement it with TensorFlow. I learned to visualize all sorts data to bring meaningful insight to users and increase the model's prediction accuracy using data preparation techniques, feature engineering techniques, and tuning the model's hyperparameters. I would highly recommend this project for anyone who wants to gain hands-on experience with TensorFlow, data preparation, feature engineering, deep learning and more.

**REFERENCES**

[1]. <https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

[2]. <https://www.youtube.com/hashtag/keras>

[3]. <https://en.wikipedia.org/wiki/TensorFlow>

[4]. <https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html>

[5]. <https://www.w3schools.com/python/matplotlib_pyplot.asp>