**Mini Project Report on**



**TRAFFIC SIGN CLASSIFICATION**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Manas Pant 2018916**

***Under the Mentorship of***

**Mr. Ashwini Kumar**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

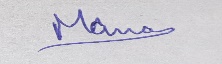
**Dehradun, Uttarakhand**

**July 2023**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Traffic Sign Classification”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Mr. Ashwini Kumar, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Manas Pant 2018916 

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter I | Introduction | 1 |
| Chapter II | Literature Survey | 4 |
| Chapter III | Methodology | 6 |
| Chapter IV | Result and Discussion | 9 |
| Chapter V | Conclusion and Future Work | 11 |
|  | References | 12 |

**Chapter I**

**Introduction**

* 1. **Introduction**

The invention of the wheel is said to be a historic moment in the evolution of humanity. The domino effect that started from the invention of wheels led us to the modern-day drivable motor cars. The next step in the same direction is making these cars self-drivable. But there exists a trivial obstacle to achieving this goal.

* 1. **The Problem**

The first thing that’s important while driving a car is the need to understand and follow the traffic rules to ensure the safety of self and of others on the road. That’s why the first step in teaching someone to drive is to make them learn the rules and the traffic signs.

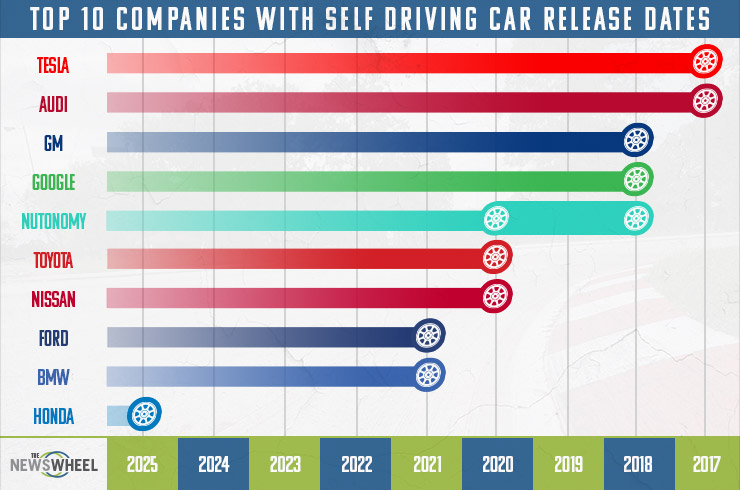
The same is true for any vehicle as well. To teach a vehicle to drive on its own, it is essential for it to learn, understand and implement the traffic rules and the traffic signs. Without this trivial detail, this next step in the evolution of cars will just be a dream and nothing more than that.

Even with automated vehicles being out on the market, there is always a scope of improvement, and the classification can always be better.

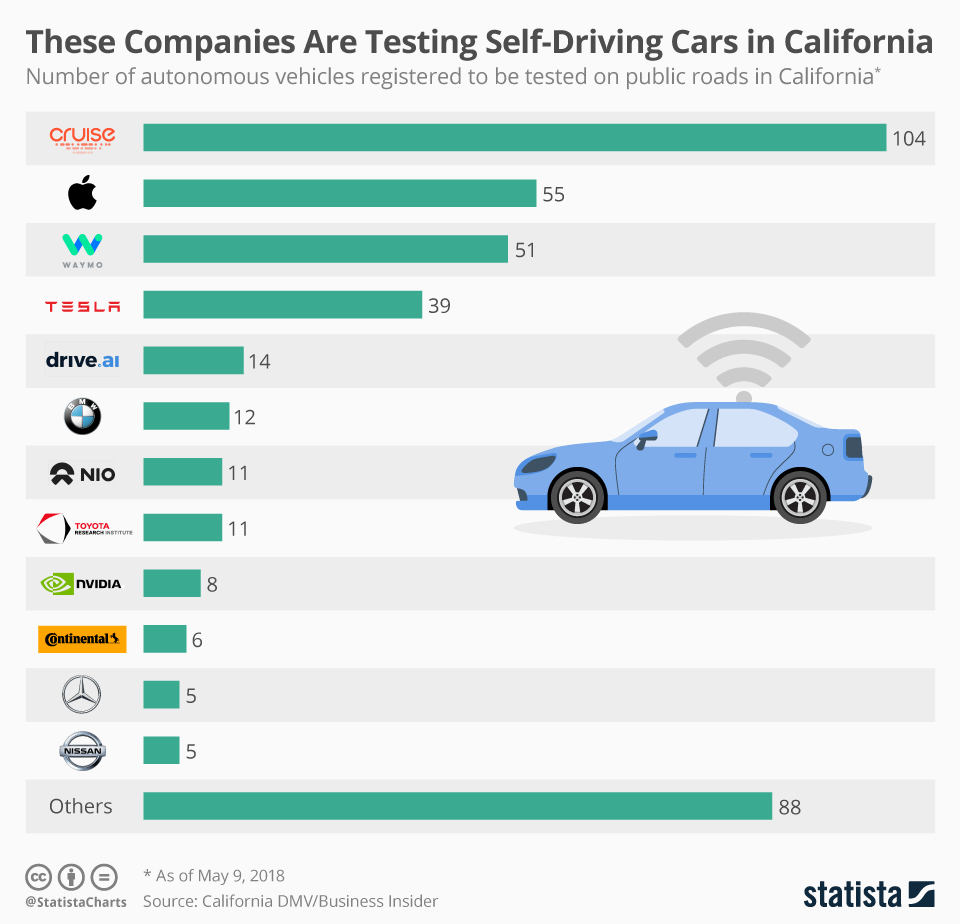
* + 1. **The Extent**

The zeal of people who are in invested in achieving the automation of vehicles is so high that there is quiet a race amongst various companies out there to put up the best product and to provide the best service.

All this can be easily understood by the following data :-



**Fig. 1.1 A graph representing the race between companies to achieve self-driving cars**



**Fig. 1.2 Ongoing tests for self-driving cars by different companies**

* 1. **The Need of a Solution**

In order to reach level 5 Automation, a vehicle needs to clearly understand and classify amongst different traffic signs, and it is the most trivial but the most essential detail for the same.

This is where the need for this project arises.

**Chapter II**

**Literature Survey**

**2.1 First Survey**

They created a quick CNN implementation using graphics processing units (GPUs) to quickly test the categorization performance of various architectures. Their implementation of CNN is flexible and fully online (i.e., weight updates after each image), in contrast to earlier GPU-based CNN implementations that were hardcoded to satisfy GPU hardware limits. The most recent GPUs are not being fully utilized by other flexible implementations. Large CNNs may now be trained in days as opposed to months, enabling research into the effects of different structural factors through extensive parameter exploration and error analysis across numerous experiments.[1]

They provided the findings from the 43-class single-image classification benchmark for the recognition of German traffic signs. They first provided a quick overview of their CNN model, followed by a description of the development of the training set and data preprocessing. They concluded by presenting experimental findings that demonstrate how improving recognition performance by combining a CNN trained on raw pixels with an MLP trained on common feature descriptors.[1]

They describe the method that defeated human recognition rates of 98.98% to win the preliminary round of the German traffic sign benchmark. They trained the nets further to achieve an even higher identification rate of 99.15%.[1]

**2.2 Second Survey**

They classify 43 different types of traffic signs using K-d trees and Random Forests. On images from the German Traffic Sign Benchmark data set, these two tree classifiers’ performances are assessed using various size HOG descriptors and Distance Transforms. The outcomes demonstrate that the HOG descriptor's tighter spatial binding produces superior outcomes. With HOG descriptors, the K-d tree produces a classification result of 92.9%, while with distance transforms, it achieves a result of 67%. The findings are improved to 97.2% and 81.8%, respectively, by the Random Forests.[2]

Because the variables were chosen at random, the Random Forests are less susceptible to changes in the backdrop than the K-d trees. This is demonstrated by comparing photos with and without a 10% border around the traffic sign when computing HOG descriptors. When the border is removed, the K-d tree's performance improves by up to 20%, whereas the Random Forests' performance is unaffected by changes in the background. K-d trees have the benefit of being quicker to build, query, and update than Random Forests. Since they are made up of a single tree rather than an ensemble, they also use less memory.[2]

**Chapter III**

**Methodology**

**3.1 Introduction**

The language used for implementing the project is Python due to its wide range of available libraries for training machine learning models and tools for the same. The project is made possible by training deep neural network model using CNN and Keras with the help of a dataset namely GTSRB (German Traffic Sign Recognition Benchmark) downloaded from Kaggle which contains over 50,000 images of different traffic signs which are then further classified into 43 classes.

This trained model is then used to classify the input image containing a traffic sign into its type. The model so made shows 95% accuracy in classifying the traffic sign.

**3.2 The Necessities**

There are various things that I required for the completion of this project are :-

* Visual Studio Code – It is a code editor by Microsoft. I chose it due to its easy to use and helpful interface and the wide variety of extensions it provides us.[3]
* Python – It is a high level and easy to understand programming language. I chose it due to the wide variety of first-party and third-party libraries it provides the support for.[4]
* Dataset - A dataset is must to train any model such that it classifies future inputs. The one I used is GTSRB - German Traffic Sign Recognition Benchmark. It is a public domain dataset with over 50,000 images segregated into 43 classes.[5]
* CNN - A Convolutional Neural Network (CNN) is a specialized type of deep learning neural network designed to process and analyze visual data, such as images and videos.
* Keras - It is an open-source high-level neural networks API written in Python. It is designed to provide a user-friendly, modular, and intuitive interface for building and experimenting with deep learning models.

**3.3 Exploring the dataset**

With the help of the OS module, I iterated over all 43 (0-42 folders) classes and appended images and their respective labels in the data and labels list.

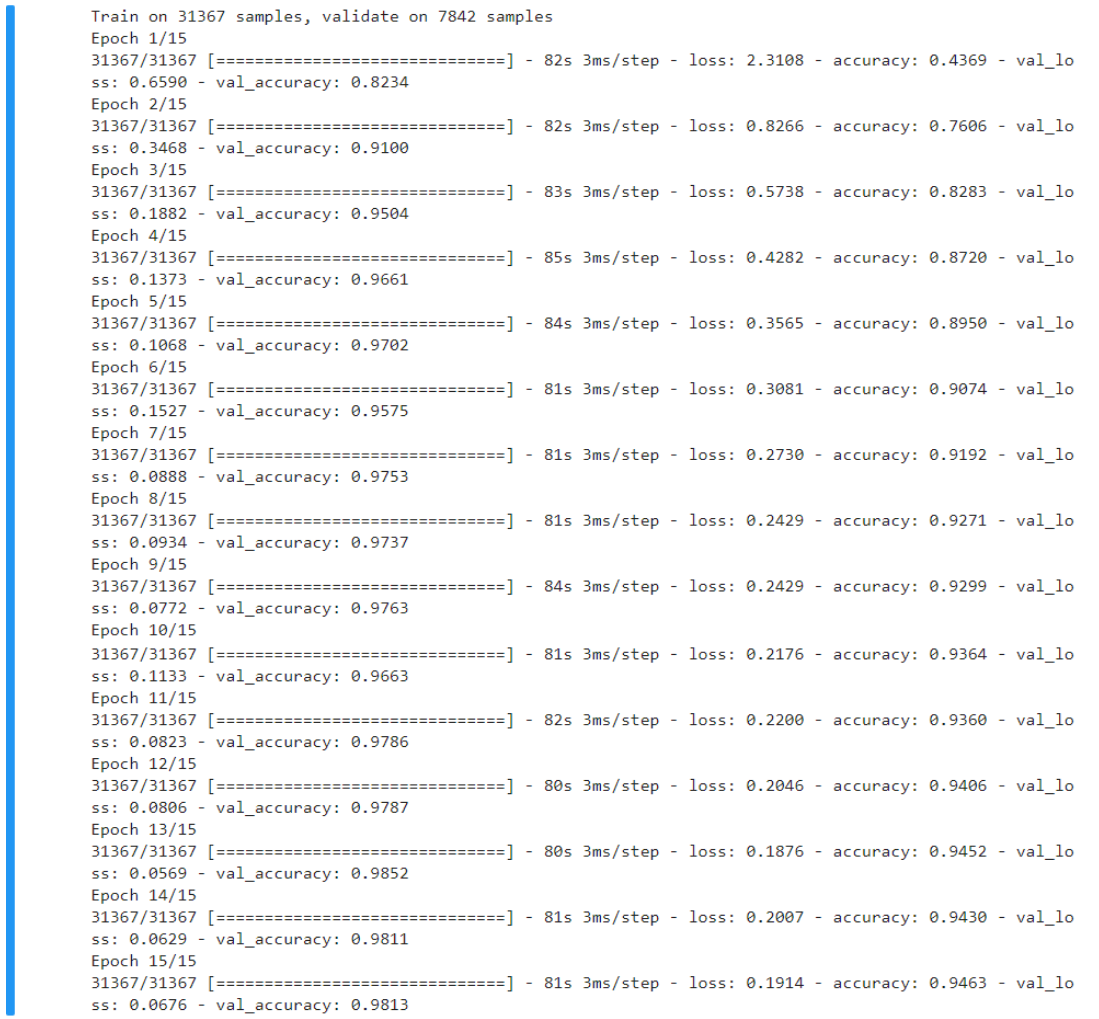
The PIL library was used to open image content into an array.

**3.4 Building a CNN model**

To classify the images into their respective categories, I built a CNN model (Convolutional Neural Network) as it is best for image classification purposes.

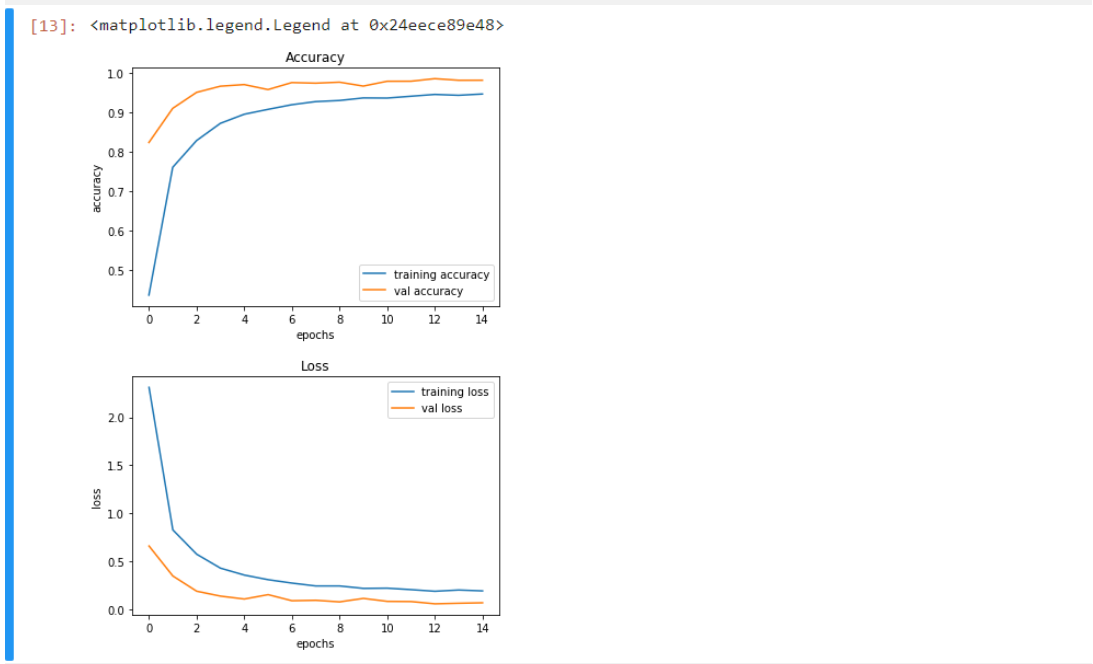
**3.5 Training and validating the model**

After building the model architecture, I then trained the model using model.fit().



**Fig. 3.1 Training the model**

The model got 95% accuracy on the training dataset. Now, with matplotlib, I plotted the graph for accuracy and loss.



**Fig. 3.2 Accuracy and Loss graphs**

**3.5 Testing our model with test dataset**

I extracted the image path and labels using pandas. Then to predict the model, I had to resize the images to 30×30 pixels and make a numpy array containing all image data. From the sklearn.metrics, I imported the accuracy\_score and observed how the model predicted the actual labels. The model achieved 95% accuracy.

Finally, the model was saved using the model.save() function.

**3.5 Building a GUI**

I built a graphical user interface (GUI) for the traffic signs classifier with Tkinter. Tkinter is a GUI toolkit in the standard python library.

A screenshot of a computer

Description automatically generated

**Fig. 3.3 The GUI**

**Chapter IV**

**Result and Discussion**

**4.1 Result**

The resulting classifier model can now be used to classify any input image of a traffic sign into its respective type.

The working of the model is as follows :-

A screenshot of a computer screen

Description automatically generated

**Fig. 4.1 First output demonstration**

A screenshot of a computer screen

Description automatically generated

**Fig. 4.2 Second output demonstration**

A screenshot of a computer screen

Description automatically generated

**Fig. 4.3 Third output demonstration**

**4.2 Discussion**

This final model has shown an accuracy of 95% and has also visualized how our accuracy and loss changes with time, which is quite good for a simple CNN model.

**Chapter V**

**Conclusion and Future Work**

**5.1 Conclusion**

This project provides us with a deep learning model that is able to classify different traffic signs into their respective classes.

The model has shown an accuracy of 95% which is very good for a basic CNN model.

**5.2 Future Work**

The few sectors where future work is possible are :-

* Model Interpretability: Explore methods for interpreting the model's decisions, such as class activation maps or gradient-based visualization. Understanding which parts of the image the model focuses on can help identify potential areas for improvement.
* Real-World Testing: Evaluate the model's performance on real-world traffic sign images or videos collected in different conditions, such as varying lighting, weather, and camera angles. Fine-tuning the model on domain-specific data can help adapt it to real-world scenarios.
* Deployment: Consider deploying the trained model on embedded devices or in real-time applications to enable real-time traffic sign recognition for autonomous vehicles or smart traffic systems.

**References**

[1] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, "A committee of neural networks for traffic

sign classification," The 2011 International Joint Conference on Neural Networks, San Jose,

CA, USA, 2011, pp. 1918-1921, doi: 10.1109/IJCNN.2011.6033458.

[2] F. Zaklouta, B. Stanciulescu and O. Hamdoun, "Traffic sign classification using K-d trees and

Random Forests," The 2011 International Joint Conference on Neural Networks, San Jose, CA,

USA, 2011, pp. 2151-2155, doi: 10.1109/IJCNN.2011.6033494.

[3] Visual Studio Code website was used to download the setup of VS Code.

<https://code.visualstudio.com/download>

[4] Python website was used to download Python 3.10.9 setup.

<https://www.python.org/downloads/release/python-3109/>

[5] Kaggle website was used to download the GTSRB dataset.

<https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>