



ABSTRACT

In this report we discuss about using a CNN to identify 43 different traffic signs

REPORT PRESENTED BY

Rutuparn Pawar
Shreyas Kulkarni
Nagarjuna Vatti
Aditya Sangle

INDEX

Sr.No.	Content	Page No.
1	Introduction.....	2
2	Problem definition and algorithm.....	2
2.1	Problem at hand.....	2
2.2	Algorithm in use.....	2
3	Implementation.....	3
3.1	Methodology.....	3
3.2	Results.....	7
3.3	Discussion.....	8
4	Deployment in real-time system.....	9
5	Conclusion.....	10
6	References.....	10

1. INTRODUCTION

Traffic signs are an integral part of our road infrastructure. They provide critical information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behaviour to make sure they adhere with whatever road regulation currently enforced. Without such useful signs, we would most likely be faced with more accidents, as drivers would not be given critical feedback on how fast they could safely go, or informed about road works, sharp turn, or school crossings ahead. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs. Naturally, autonomous vehicles must also abide by road legislation and therefore recognize and understand traffic signs.

2. PROBLEM DEFINITION AND ALGORITHM

2.1 PROBLEM AT HAND

Traditionally, standard computer vision methods were employed to detect and classify traffic signs, but these required considerable and time-consuming manual work to handcraft important features in images. Instead, by applying deep learning to this problem, we create a model that reliably classifies traffic signs, learning to identify the most appropriate features for this problem by itself. In this report we show how we can create a deep learning architecture that can identify traffic signs with close to 98% accuracy on the test set.

2.2 ALGORITHM IN USE

We have utilized a convolutional neural network (CNN) also referred to as ConvNet for classifying images containing traffic signs

A CNN is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.

3. IMPLEMENTATION

3.1 METHODOLOGY

A. Project Setup

- This project has been implemented in Python 3 and Tensorflow
- In addition to Tensorflow the following modules have been used

numpy	pandas	matplotlib
opencv2 (cv2)	PIL	os
sklearn	tkinter	pickle
serial	pyttsx3	speech_recognition

B. About the dataset

- Number of entries in train dataset = 39209
- Number of entries in test dataset = 12630
- Columns in the train and test dataset

Width	Width of image
Height	Height of image
Roi.X1	Upper left X coordinate of sign on image
Roi.Y1	Upper left Y coordinate of sign on image
Roi.X2	Lower right X coordinate of sign on image
Roi.Y2	Lower right Y coordinate of sign on image
ClassId	Class of provided image
Path	Path to provided image

The dataset is available at <https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

C. Images and Distribution

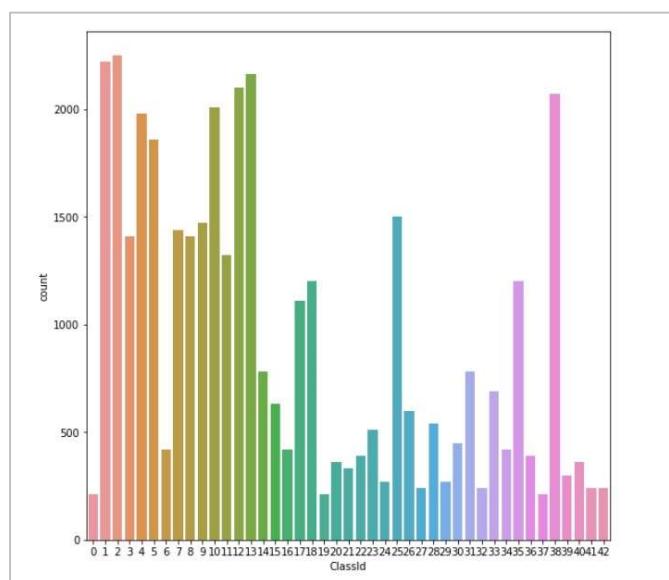
There are a total of 51,839 RGB images of size 30x30 pixels. Paths to the images have been stored in the data set

You can see below a sample of the images from the dataset, with labels displayed above the row of corresponding images. Some of them are quite dark so we will look to improve contrast a bit.

Vehicle > 3.5 tons prohibited 	Speed limit (30km/h) 	Keep right 	Turn right ahead 	Right-of-way at intersection 
--	---	---	---	---

Sample of Training Set Images with Labels Above

There is also a significant imbalance across classes in the training set, as shown in the histogram below. Most classes have less than 500 images, while a few have over 2000. This means that our model could be biased towards over-represented classes, especially when it is unsure in its predictions. However we can mitigate this issue using data augmentation.



Distribution of images in training set – not quite balanced!

The following images containing the following traffic signs are classified by the classifier

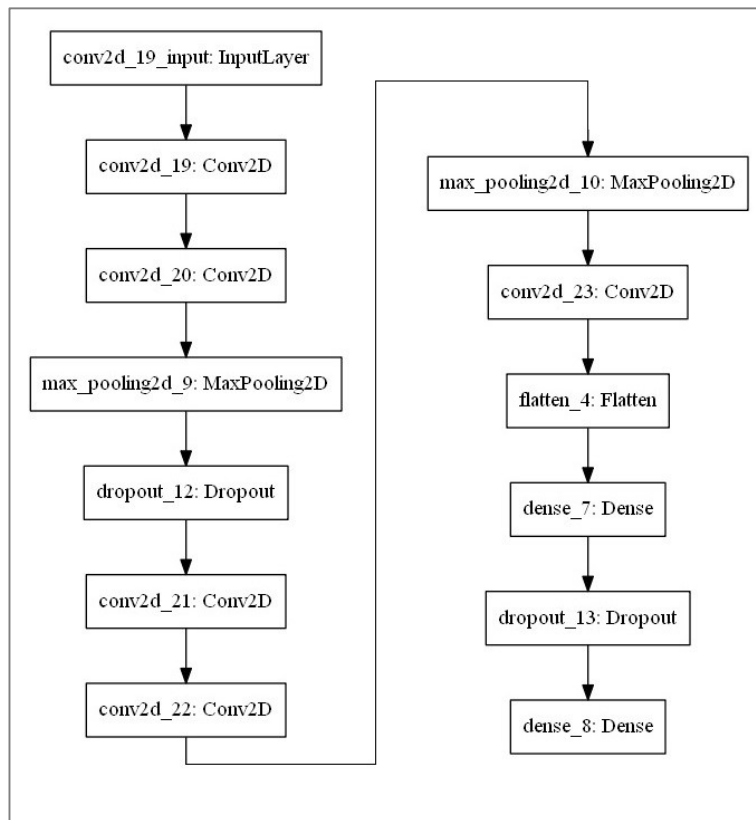
1	Speed limit (20km/h)
2	Speed limit (30km/h)
3	Speed limit (50km/h)
4	Speed limit (60km/h)
5	Speed limit (70km/h)
6	Speed limit (80km/h)
7	End of speed limit (80km/h)
8	Speed limit (100km/h)
9	Speed limit (120km/h)
10	No passing
11	No passing vehicle over 3.5 tons

12	Right-of-way at intersection
13	Priority road
14	Yield
15	Stop
16	No vehicles
17	Vehicle > 3.5 tons prohibited
18	No entry
19	General caution
20	Dangerous curve left
21	Dangerous curve right
22	Double curve
23	Bumpy road
24	Slippery road
25	Road narrows on the right
26	Road work
27	Traffic signals
28	Pedestrians
29	Children crossing
30	Bicycles crossing
31	Beware of ice/snow
32	Wild animals crossing
33	End speed + passing limits
34	Turn right ahead
35	Turn left ahead
36	Ahead only
37	Go straight or right
38	Go straight or left
39	Keep right
40	Keep left
41	Roundabout mandatory
42	End of no passing
43	End no passing vehicle > 3.5 tons'

D. Model Architecture

Our initial model has the following layers:

- 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")



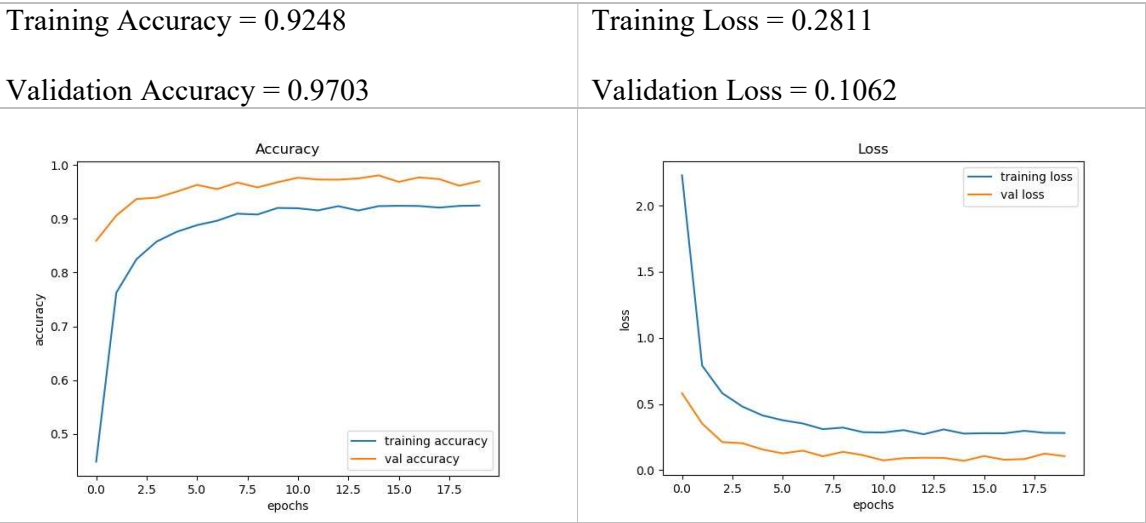
Model Architecture

Model: "sequential_5"		
Layer (type)	Output Shape	Param #
conv2d_19 (Conv2D)	(None, 28, 28, 32)	2432
conv2d_20 (Conv2D)	(None, 24, 24, 32)	25632
max_pooling2d_9 (MaxPooling2D)	(None, 12, 12, 32)	0
dropout_12 (Dropout)	(None, 12, 12, 32)	0
conv2d_21 (Conv2D)	(None, 10, 10, 64)	18496
conv2d_22 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_10 (MaxPooling2D)	(None, 4, 4, 64)	0
conv2d_23 (Conv2D)	(None, 2, 2, 128)	73856
flatten_4 (Flatten)	(None, 512)	0
dense_7 (Dense)	(None, 256)	131328
dropout_13 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 43)	11051
Total params: 299,723		
Trainable params: 299,723		
Non-trainable params: 0		

Model Summary

3.2 RESULTS

Initial model

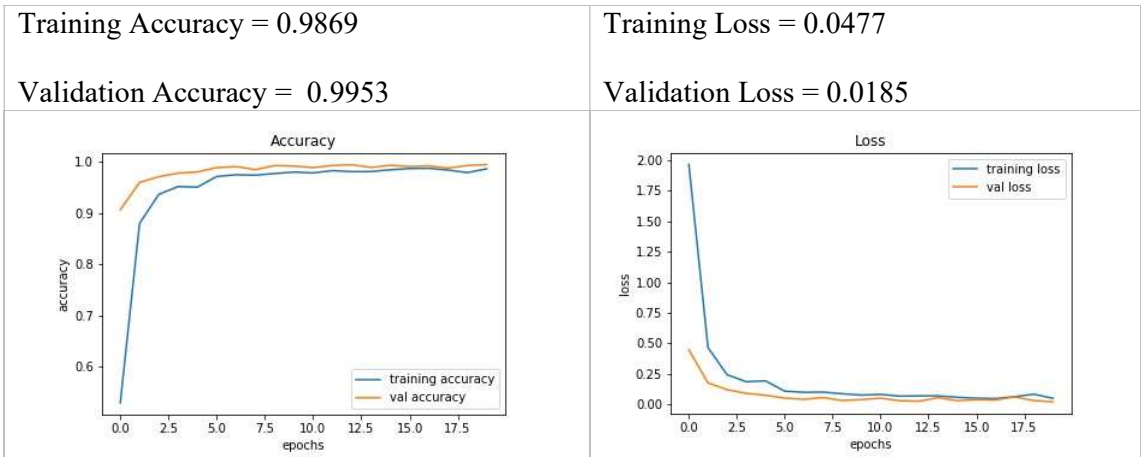


Final model having the highest accuracy we could achieve

Number of convolutional layers: 5

Optimizer is Adam and Epochs = 15.

This is the best output which we could achieve.



3.3 DISCUSSION

- We were able to develop a CNN model having 98.25% training accuracy and 99.45% test accuracy.
- Effect on accuracy when model is slightly modified using intuition

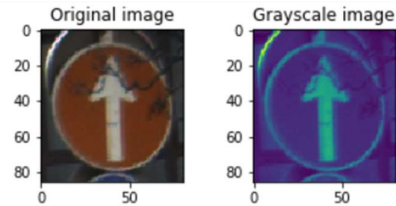
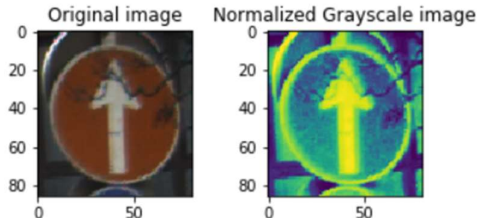
<i>No. of Convolutional Layers</i>	<i>Optimizer</i>	<i>Epochs</i>	<i>Training Loss</i>	<i>Training Accuracy</i>	<i>Validation Loss</i>	<i>Validation Accuracy</i>
4	Adam	15	0.0939	0.9940	.1212	0.9590
5	Adam	15	0.0689	0.9825	0.0196	0.9945
4	Adam	20	0.281	0.9248	0.1062	0.9703
5	Adagrad	15	0.6569	0.8284	0.2884	0.9454
5	SGD	15	0.1953	0.9515	0.0559	0.9885

4. DEPLOYMENT IN REAL-TIME SYSTEM

We worked on developing a real-time implementation of the model. Here is a brief on what we did.

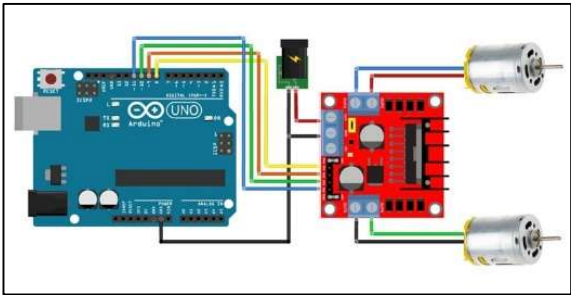
STEP 1: Image pre-processing

We initially apply two pre-processing steps to our images:

<p>A. Grayscale</p> <p>We convert our 3 channel image to a single grayscale image.</p>	 <p>Original image Grayscale image</p> <p>Random original image and its Grayscale image</p>
<p>B. Image Normalization</p> <p>We centre the distribution of the image dataset by using histogram equalization. This helps our model treating images uniformly. The resulting images look as follows:</p>	 <p>Original image Normalized Grayscale image</p> <p>Normalised grayscale images</p>

STEP 2: Classification of image by model

The normalized grayscale image obtained in the previous step is passed to the model to get its prediction. Some predictions are utilized to control DC motors connected to the hardware model which is interfaced with the processor using serial communication. A diagrammatic representation of the hardware model can be seen below



Schematic of hardware model

HARDWARE IMPLEMENTATION- ACTUAL IMAGES



5. CONCLUSION

In this project, we have successfully classified the traffic signs classifier with more than 95% accuracy which is quite good from a simple CNN model. We also visualized how our accuracy and loss change with each epoch.

Our model reached close to close to 98% accuracy on the test set, achieved about 99% on the validation set.

We thoroughly enjoyed this project and gained practical experience using Tensorflow and investigating artificial neural network architectures. Moreover, we delved into some seminal papers in this field, which reinforced my understanding and more importantly refined my intuition about deep learning.

6. REFERENCES

<https://data-flair.training/blogs/python-project-traffic-signs-recognition/>

<https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aadc2ab>