German Traffic Sign Classifier

Brendan Kam

The purpose of this project was to create a traffic sign classifier by building a convolutional neural network (CNN). The traffic sign classifier had to be able to accurately predict at least 93% of the traffic signs from the validation set. In this report, the steps taken to build the convolutional neural network are introduced. The final performance of the traffic sign classifier is also discussed.

In this project, the traffic signs provided were German traffic signs. The data provided consists of a training set, a test set and validation set. The traffic signs in the data sets come from 43 different types of traffic signs. A summary of the data sets can be found in table 1.

Table 1: Summary of German Traffic Sign Data Sets

Number of training images	34799
Number of testing images	12630
Number of validation images	4410
Image Data Shape	32 x 32 x 3
Number of traffic sign classes	43

Figure 1 below shows 5 random traffic signs obtained from the training data. As seen in figure 1, some traffic signs are brighter than other traffic signs. Exploring the different images in the training images, it was found that some images are also more visible and identifiable than other images.

Figure 1: Random Traffic Signs from Traning Data

Figure 2 shows the number of time each traffic sign appears in the training data set. As shown in figure 2, there is an uneven distribution of the types of traffic signs in the training data set. Some traffic signs have more than 1500 samples while others have less than 250 samples.

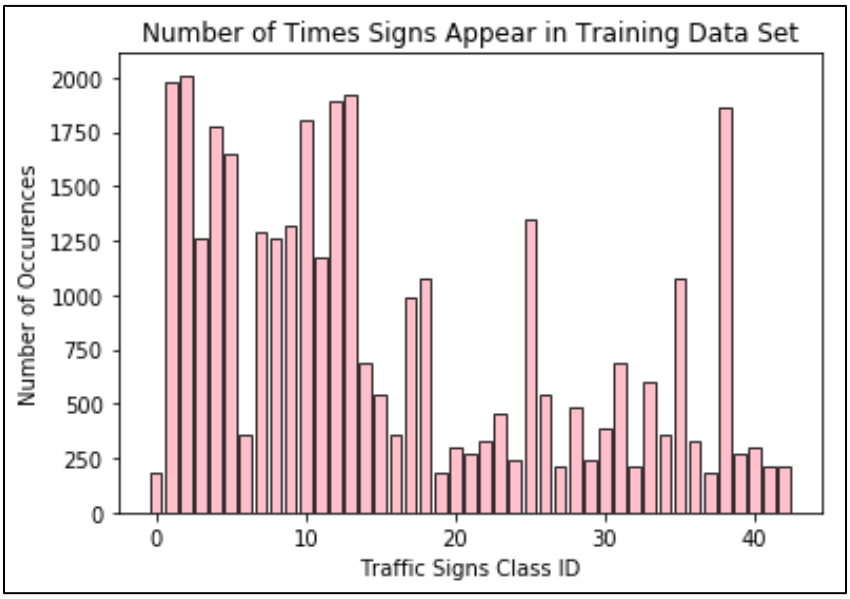

Figure 2: Number of examples of each traffic sign in the training data set

For this project, there were 4 steps taken to preprocess the training data. The first was to convert the images to gray scale. Changing the images to grayscale allows the convolutional neural network to identify edges of the traffic signs more easily. The next step was to normalize the training data. Normalizing the training data narrows the search space leading to more accurate models. Then, the training data histogram was equalized using the adaptive equalize histogram function from the Scikit library. As seen in figure 1, some images are darker than the others, by equalizing the data histogram, the images become brighter which leads to an increase in prediction accuracy from the traffic sign classifier. The final step in preprocessing the data was to shuffle the training, test and validation data. This allows for more variability in the final results while reducing the chances of the model overfitting.

As seen in figure 2, some traffic signs had a lot more samples than other traffic signs. This could lead to the models being better at predicting a certain sign over the others. It could also lead to the model overfitting for certain signs. To overcome this problem, the training data was augmented by signs that originally has less than 750 samples. First the traffic signs with less than 750 samples

were identified. Then each image is augmented by randomly rotating it between 20 degrees clockwise and anti-clockwise about the center of the image. Figure 3 shows 5 random augmented traffic signs that have been rotated.

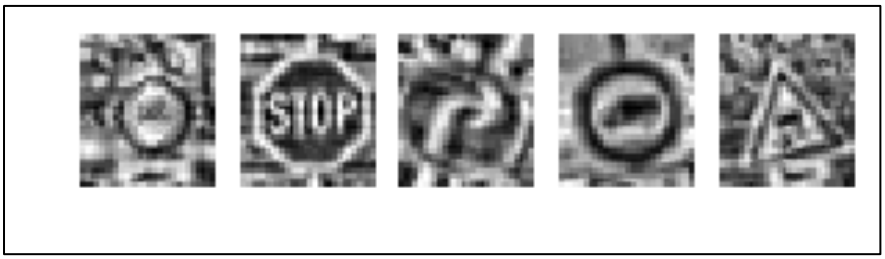

Figure 3: Augmented Images after preprocessing

Figure 4 is a bar chart which shows the increase in number of traffic sign samples in the training data. Only traffic signs with less than 750 samples were augmented. As shown in figure 4, the traffic signs with less than 750 samples were augmented 3 times so that all traffic signs will have at least 750 samples.

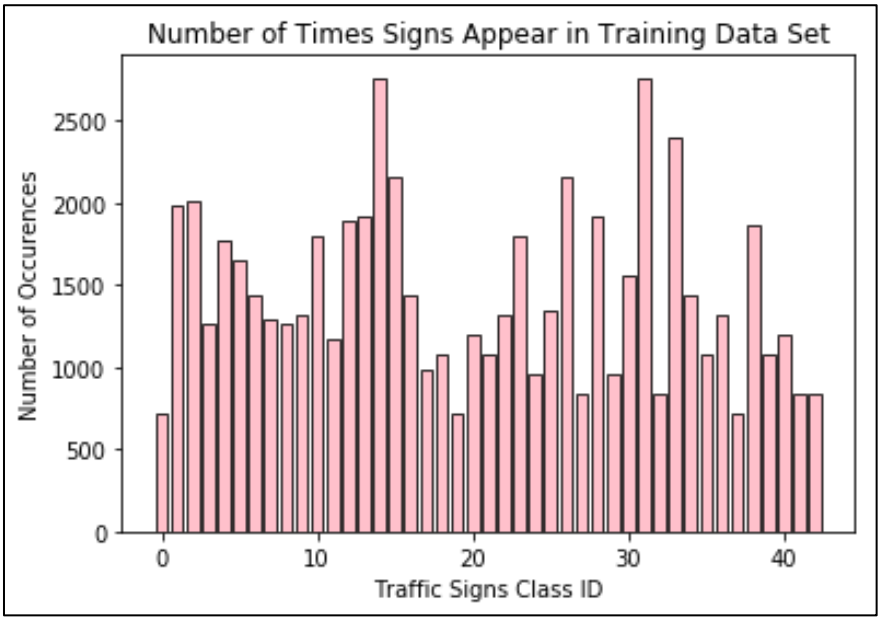

Figure 4: Bar chart showing increase in augmented data

The final model used is a slight variation of the LeNet architecture. It consists of 5 main layers, 2 convolutional neural networks and 3 fully connected networks. A summary of the final model can be found in Table 2. The selection of the architecture is discussed further in a later section.

Table 2 : Description of Final Architecture

Layer	Description
Input	32x32x1 grayscale image
Convolution 1	1x1 stride, Valid Padding Output: 28x28x6
RELU	
Dropout	Keep Probability:
Max Pooling	2x2 stride Output: 14x14x6
Convolution 2	2x2 stride Valid Padding Output: 10x10x16
RELU	
Dropout	Keep Probability:
Max Pooling	2x2 stride Output: 5x5x6
Fully Connected 3	Output: 120
RELU	
Fully Connected 4	Output: 84
RELU	
Fully Connected 5	Output: 43

While training, a batch size of 128 was chosen. Generally, a larger batch size would lead to a more accurate model, however due to hardware memory limitations, a batch size of 128 was chosen as a compromise. The Adam Optimizer was chosen as the optimizer for this classifier. The Adam Optimizer is an algorithm that uses a decaying learn rate to achieve better accuracy. The training was done in 30 epochs and a learn rate of 0.00085. Through an reiterative process, it was found that beyond 30 epochs, there was no significant increase in prediction accuracy and that a learn

rate of 0.00085 would lead to the maximum prediction accuracy. For the drop out layer, a keep probability of 0.6 was chosen as it was found to provide the best accuracy.

Table 3 shows the final prediction accuracy of the traffic sign classifier for each data set.

Table 3 Final Prediction Accuracy of Data Sets

Data Set	Prediction Accuracy
Training	0.999
Test	0.945
Validation	0.959

The first iteration of the traffic sign classifier was the original LeNet architecture without any modifications. The code was based on the Udacity's Self Driving Car Nano Degree LeNet lab. The only data preprocessing done was to shuffle the data. The training parameters were the same as those found in the LeNet lab. In the first iteration, a maximum accuracy of 86% was achieved.

To improve the accuracy, over several reiterations, the optimal learning rate and number of epochs were determined. Next, three extra steps were added to the data preprocessing. First the images were converted to grayscale, then there were normalized. Finally, they were brighten using adaptive histogram equalization function. The training data was also augmented by randomly adding rotated traffic signs. Having completed those steps, a maximum validation accuracy of 92.5% was achieved by the traffic sign classifier.

It was found that the traffic sign classifier was overfitting the training data. Therefore, two dropout layers were added to prevent overfitting. Through several reiterations, the optimal keep probability of 0.6 was found. A final accuracy of 95.9% was achieved after adding the drop out.

To further test the performance of the classifier, 8 traffic signs were chosen from the internet. Figure 5 below shows the signs that were chosen.

Figure 5: Traffic Signs Taken from the internet

The first two signs were the No Entry sign and the Stop sign, which are very common signs on the road. Two direction signs were chosen to see if the classifier could differentiate between directions. Two speed limit signs were chosen to see if the classifier would be able to identify different numbers. Two priority signs were chosen, one sign which had been spray painted on. The purpose of the spray-painted priority sign was to determine if the classifier could identify still identify a sign correctly if it had been vandalized in real life.

Table 4 shows the prediction of the web images and the top 5 probabilities of each traffic sign. The final prediction accuracy was 100%. However, looking at the top 5 probabilities, for certain signs, the probabilities were relatively close to one another. This might mean the classifier is unsure about its prediction.

Table 4: Results of Predicting Web Images

Traffic Sign from Web	Predicted Sign	Top 5 probabilities	
No Entry	No Entry	No Entry	0.842
		Stop	0.090
		No vehicles	0.022
		No passing	0.020
		Priority Road	0.015

Stop	Stop	Stop	0.999
		Go straight or left	0.000
		Speed limit (20km/h)	0.000
		Speed limit (30km/h)	0.000
		Speed limit (120km/h)	0.000
Turn Right Ahead	Turn Right Ahead	Turn right ahead	0.747
		Ahead only	0.136
		Speed limit (30km/h)	0.051
		Vehicles over 3.5 tons prohibited	0.0216
		Speed limit (20km/h)	0.0181
Go Straight or Right	Go straight or right	Go straight or right	0.481
		Roundabout mandatory	0.312
		Speed limit (70km/h)	0.045
		Stop	0.039
		Speed limit (20km/h)	0.021
Speed Limit: 120km/h	Speed Limit (120km/h)	Speed limit (120km/h)	0.400
		Speed limit (100km/h)	0.200
		Speed limit (70km/h)	0.166
		Speed limit (60km/h)	0.108
		Speed limit (80km/h)	0.059
Speed Limit: 20km/h	Speed limit (20km/h)	Speed limit (20km/h)	0.973
		Speed limit (70km/h)	0.026
		Speed limit (30km/h)	0.002
		Speed limit (80km/h)	0.000
		Speed limit (120km/h)	0.000
Priority Road	Priority Road	Priority Road	0.999
		No passing	0.000
		Vehicles over 3.5 tons prohibited	0.000
		Round about mandatory	0.000

		Speed limit (50km/h)	0.000
Priority Road (Sprayed)	Priority Road	Priority Road	0.999
		Speed limit (50km/h)	0.000
		Stop	0.000
		No passing	0.000
		Round about mandatory	0.000

Overall, the traffic sign classifier could predict the traffic signs obtained from the web accurately. It could differentiate direction signs, and between numbers. The classifier was also able to predict the correct sign despite having one sign spray painted on. Since traffic signs in real life my be vandalized or worn down by weather, the traffic sign shows promise in identifying them correctly.

The classifier managed to surpass the 93% accuracy as required by the project. However, it could be further improved. One way is to add more training data by augmenting it with other transformations besides a rotation. This would give the model more scenarios to learn from. Besides that, more convolutions or depth can be added to LeNet architecture to improve the accuracy. In this project, the only two dropout layers were added to the LeNet architecture. Finally, it would be beneficial to be able to know which signs LeNet tends to misclassify or over predict. This would help in fine tuning the parameters for the training set.

References

The codes used in this project are based on the Deep Learning and LeNet labs from the Udacity Self Driving Car Nanodegree Term 1. Please check Udacity's GitHub repo for the original codes used. This project writeup was based on the write up sample provided for this project. The traffic signs used to predict the internet traffic signs were obtained from various websites. Various ideas for preprocessing, augmenting the data and improving the models were obtained from the course forum discussion board.