Udacity: Self Driving Car Engineer Nanodegree

Project 3 – Traffic Sign Classifier

Submitted by Ayeda Sayeed

Dataset Summary & Exploration

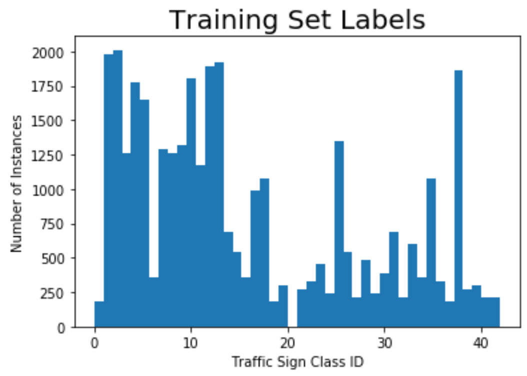
1. Summary

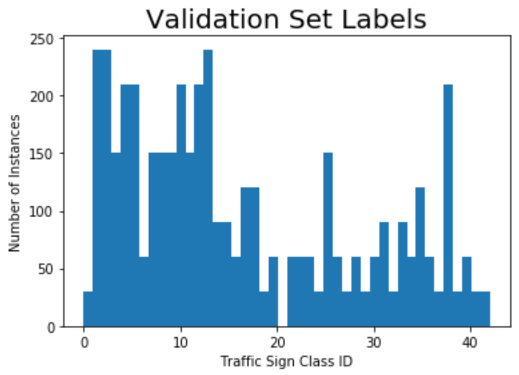
I used a combination of Python and Numpy methods to summarize the dataset info as follows:

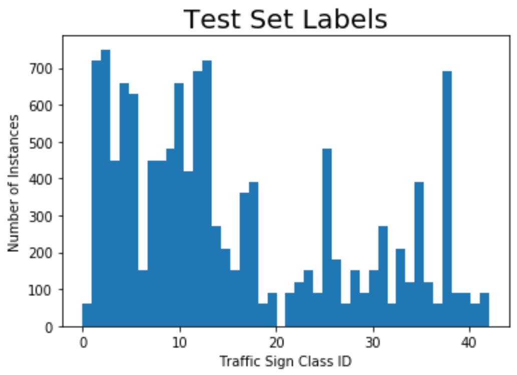
```
Number of training examples = 34799
Number of validation examples = 4410
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 44
```

2. Visualization

I used a histogram for each of the training, validation, and test data to show each set's distribution across the different types or classes of traffic signs:







Class 2, the "Speed Limit (50km/h)" sign, is the most common traffic sign in all three datasets, while class 21, "double curve", does not make an appearance in any of the datasets. Thus, the former may be the easiest to identify (and perhaps dominate other identifications), while the latter is likely to be unrecognizable to the model as the model will not be trained for it. Since the distributions of all three datasets are quite similar, their final classification accuracies should be very similar as well. If they were wildly different from the training set, we could expect the validation and test sets to be much less accurate.

Design and Test a Model Architecture

1. Preprocessing

I decided against converting the colour images to grayscale, as I would lose a lot of information this way. I also decided against normalizing from -1 to +1, as the (pixel – 128)/128 method would do, since all three channels are already within the same range of 0-255. Instead, I normalized the pixels using the standard deviation, after subtracting the mean value so as to center the data around zero and achieve zero-mean.

To display an example image, I normalized the image from 0-1.

An example original image, and pre-processed image, are shown below.

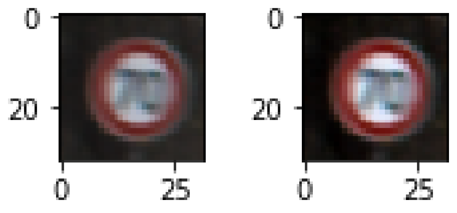


Figure 0.1: Image from Training Images, Type/Class 4; before (left) and after (right) preprocessing

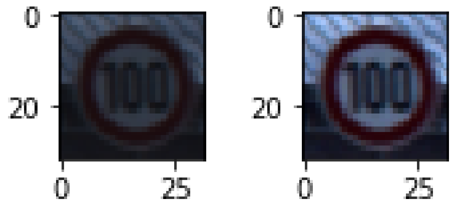


Figure 0.2: Image 24299 from Training Images, Class 7; before (left) and after (rght)

As we can see, the preprocessing doesn't necessarily make the image clearer or remove a significant amount of noise, but it does increase the contrast between the text and the sign background, which is helpful to identifying the text.

2. Final model architecture

The table below describes the architecture of the final model saved after training was completed.

Layer #	Layer Name	Description
1	Convolution	Input = 32x32x3, Filter = 5x5x3, Output = 28x28x6, # of Filter = 6, stride = 1,1,1,1
2	ReLu	
3	Max Pooling	Input = 28x28x6, Output = 14x14x6, kernel = 2x2
4	Convolution	Input = 14x14x6, Filter = 5x5x6, Output = 10x10x16, stride = 1,1,1,1
5	ReLu	
6	Max Pooling	Input = 10x10x6, kernel = 2x2, Output = 5x5x16
7	Fully Connected	Input = 400, Output = 120
8	ReLu	
9	Fully Connected	Input = 120, Output = 84
10	Dropout	Keep probability of 75%
11	ReLu	
12	Fully Connected	Input = 84, Output = 44

Epochs: 75

Batch Size: 128

Learning Rate: 0.001

3. Training

In training my model, I used the Adam optimizer to minimise the loss at each iteration. I maintained a small batch size to ensure enough training sets per epoch, and I didn't let my learning rate go below 1e-4 to ensure the training didn't get stuck or the calculations of gradients didn't become too heavy.

4. Approach for finding solution

The final model accuracy results are as follows:

Training	100%
Validation	94.4%
Test	93.8%

I decided to use a LeNet architecture because it is robust and I am familiar with implementing the architecture from the course labs. The first architecture had 10 epochs, a batch size of 128, and a learning rate of 0.001, with valid padding on all of the filters. The table below describes the details:

Layer #	Layer Name	Description
1	Convolution	Input = 32x32x3, Filter = 5x5x3, Output = 28x28x6, # of Filter = 6, stride = 1,1,1,1
2	ReLu	

3	Max Pooling	Input = 28x28x6, Output = 14x14x6, kernel = 2x2
4	Convolution	Input = 14x14x6, Filter = 5x5x6, Output = 10x10x16, stride = 1,1,1,1
5	ReLu	
6	Max Pooling	Input = 10x10x6, kernel = 2x2, Output = 5x5x16
7	Fully Connected	Input = 400, Output = 120
8	ReLu	
9	Fully Connected	Input = 120, Output = 84
10	ReLu	
11	Fully Connected	Input = 84, Output = 44

On the first attempt, I found that the test accuracy was 95%, but the final validation accuracy was only 89%, meaning that the model was overfitting. To combat this, I decreased the learning rate to 0.0001, which increased the validation accuracy to 92.4%, and test accuracy to 99%. I needed a more reliable increase in accuracy, so I increased the number of epochs to 50. With this architecture, the validation accuracy rose quickly to 90% and then was oscillating around 93% until around the last 5 epochs, where it maintained above 93% accuracy and a cost of 0, finally ending at 94% validation accuracy and 100% training accuracy. While this validation accuracy is satisfactory, I am not partial to the overfitting demonstrated by the 100% training accuracy. So, I added a dropout layer, with a keep probability of 75%, between the final two fully connected layers to prevent this overfitting (making sure the validation mode had 100% keep probability to negate the dropout layer), and increased the epochs to 75. This architecture gave validation accuracies more consistently above 93%, even as high as 95.6%, with a final epoch accuracy of 94.4%. Therefore, adding the dropout layer achieved a more reliable architecture than only increased epochs.

Test a Model on New Images

1. Five German traffic signs

I found 5 images of German traffic signs that belong to the 44 classes that this model is trained for. I cropped the images to contain only the sign, and then resized them to 32x32 pixels. This work can be found in the Jupyter notebook “Playground.ipynb”. The images are shown below:

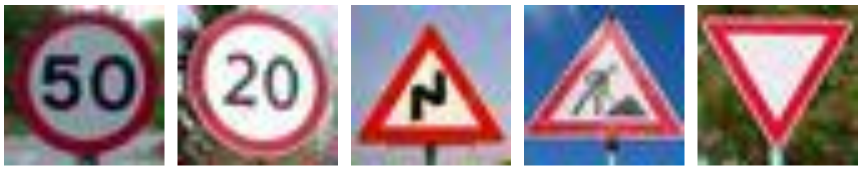


Image 1, the 50km/h speed limit sign, is the most common sign in the training set. Thus, I anticipate that this image will be easily identified. It is a little dark, with lower contrast between the numbers and the sign background, so this might make classification slightly more difficult.

Image 2 is on a slight angle, which might contribute to a lower probability of classification.

Image 3, the double curve sign, was not in the training set at all, and so I believe it will likely not classified correctly, perhaps classified as either Class 19 “dangerous curve to the left” or Class 20 “dangerous curve to the right”.

Image 4 has some glare at the top, and the image boundaries touch the bottom corners of the sign, so this might cause some loss of information.

Image 5 is quite a good candidate for classification.

All images have noise and colours in the background, unlike the training sets whose backgrounds were mostly black. This may introduce some disruptions in the classification process.

A note on preprocessing: When resizing the images to 32x32 pixels, I used the Lanczos downsampling method, and so subtracting the mean and dividing by the standard deviation makes no difference on the image because they are already preprocessed enough.

2. Models’ Predictions

Below is a chart showing a comparison between the actual sign classes and the classes predicted by the model. As we can see, something has gone wrong with the classification since every sign has been given the same ID, which is not correct for any of them.

Image #	Actual Class ID	Actual Class Name	Predicted Class ID	Predicted Class Name
1	2	*Speed limit (50km/h)*	42	*End of no passing by vehicles over 3.5 metric tons*
2	0	*Speed limit (20km/h)*	42	*End of no passing by vehicles over 3.5 metric tons*
3	21	*Double Curve*	42	*End of no passing by vehicles over 3.5 metric tons*
4	25	*Road work*	42	*End of no passing by vehicles over 3.5 metric tons*
5	13	*Yield*	42	*End of no passing by vehicles over 3.5 metric tons*

3. Softmax probabilities

With further analysis, we look at the top 5 softmax probabilities for each sign. The entirety of the probability has been given to ID 42 for every sign, with almost-zero numbers left for the other 4. The probabilities that are exactly 0 are simply assigned the first consecutive class IDs with no meaning.

This is not the behaviour that I expected. I anticipated that Image 3 (double curve) would have similar top 2 probabilities, identifying as either Class 19 or 20, since the double curve class was not trained for. I anticipated at least an above 50% probability for Images 1, 2 and 5. The second probability for Image 5 identifies a No Entry sign, which is quite similar to the yield sign, especially when the triangle of the yield sign is warped in low-resolution.

Still, these results indicate that something has gone wrong with these new images. Perhaps they are zoomed in too closely to the sign, and so the boundaries of the sign are not where the model expects them. I understand this this is likely not a satisfactory submission, so I would appreciate feedback how to solve this issue.

Image #	**#1 Probability**	**#2 Probability**	**#3 Probability**	**#4 Probability**	**#5 Probability**
1	1.00	3.29678959e-19	9.00067100e-24	1.31562471e-29	9.09337133e-31
	End of no passing by vehicles over 3.5 metric tons	*Speed limit (70km/h)*	*Turn left ahead*	*Keep left*	*Dangerous curve to the left*
2	1.00	1.28940667e-35	0	0	0
	End of no passing by vehicles over 3.5 metric tons	*Turn left ahead*	*0,Speed limit (20km/h)*	*1,Speed limit (30km/h)*	*2,Speed limit (50km/h)*
3	1.00	6.18722599e-30	0	0	0
	End of no passing by vehicles over 3.5 metric tons	*No vehicles*	*0,Speed limit (20km/h)*	*1,Speed limit (30km/h)*	*2,Speed limit (50km/h)*
4	1.00	6.68577209e-24	3.88849138e-29	3.26974912e-34	4.55986377e-36
	End of no passing by vehicles over 3.5 metric tons	*Speed limit (70km/h)*	*No vehicles*	*Go straight or left*	*Speed limit (120km/h)*
5	1.00	4.85620431e-38	0	0	0
	End of no passing by vehicles over 3.5 metric tons	*No entry*	*0,Speed limit (20km/h)*	*1,Speed limit (30km/h)*	*2,Speed limit (50km/h)*