Adrian Chow

Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Data Set Summary & Exploration

The following data set is a set of 43 different classifications of German Traffic Signs.

```
Number of training examples = 35839
Number of validation examples = 8960
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of unique classes = 43
Ratio between valid/train = 0.2
```

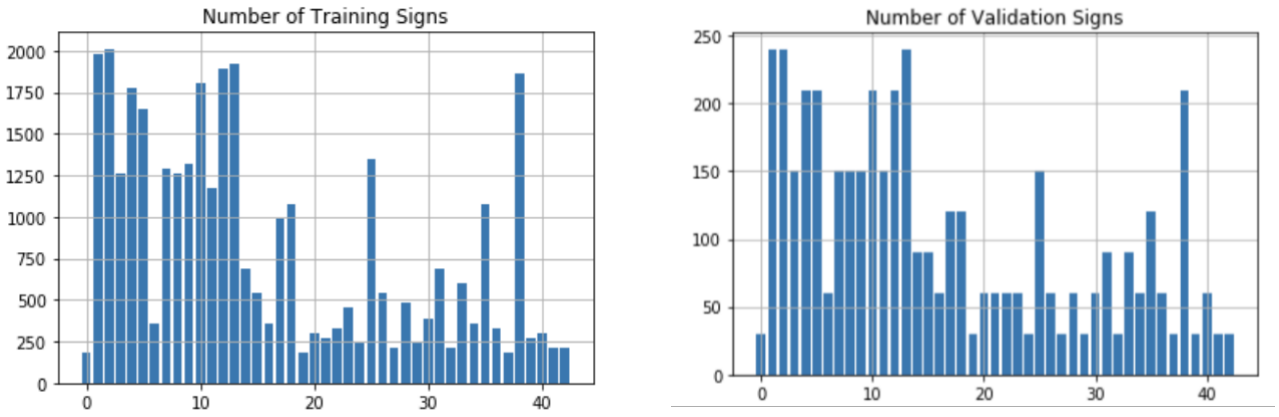


Figure 1: Distribution of Training and Validation Data.

Adrian Chow

Data Set Summary & Exploration

Preprocessing Steps:

1. Convert grayscale
2. Normalize
3. Data Augmentation

I first converted every image to grayscale to reduce the number of color channels from 3 to 1. This essentially would reduce the cost of training the neural network due to three times less parameters. The following is an image before and after converting to grayscale.

Figure 2: Images after gray-scaling.

Next the images were normalized by dividing the pixel intensities by 255 to standardize the intensities from 0 to 1. The following will allow the model to converge faster. Lastly, new data was generated using the data generator from Keras library. The additional images were distorted, rotated and dislocated to diversify the dataset. Here are some of the new images which were distributed to the data set.

Figure 3: Augmented images using existing dataset.

Adrian Chow

Model Architecture:

LAYER	DESCRIPTION (INPUT)
INPUT	32x32x1 RGB Image
CONVOLUTION	28x28x60
RELU	28x28x60 -
CONVOLUTION	24x24x60
RELU	24x24x60
MAX POOLING	12x12x60
CONVOLUTION	12x12x60
RELU	10x10x60
CONVOLUTION	10x10x30
MAX POOLING	10x10x30
RELU	8x8x30
FLATTEN	480
DROPOUT	480
FULLY CONNECTED	480
RELU	500
DROPOUT	500
FULLY CONNECTED	500
RELU	43
TOTAL PARAMETERS	378,023

```
#Parameters
EPOCHS = 27
BATCH_SIZE = 128
rate = 0.00095
mu = 0
sigma = 0.1
keep_prob = 0.5 (Dropout Rate)

optimizer = tf.train.AdamOptimizer(learning_rate = rate)
```

To achieve an accuracy of 0.93, a couples of a measurements were taken:

1. Hyperparameter Tuning
2. Data Augmentation
3. Train-Test-Split of Training Data and Test Data

Adrian Chow

First the epochs were set to a higher value than what was observed in the lab due to the proportion of extra data. For me 27 epochs sufficed to train a well-model, before overtraining would become an issue. A slower rate was used due to the amount of classifications that were handled. In addition, dropout of 0.5 probability forced the weights and biases to become tuned. As mentioned before more data was generated using augmentation images, which forced the network to train on low-quality and diverse images in the effort of improving the accuracy. Lastly, the training and validation datasets were balanced to a 1:0.2 ratio to make sure the training would be balanced with validation step.

Here are the results:

```
INFO:tensorflow:Restoring parameters from ./lenet
Validation Accuracy = 0.958
INFO:tensorflow:Restoring parameters from ./lenet
Test Accuracy :  0.949326999095
```

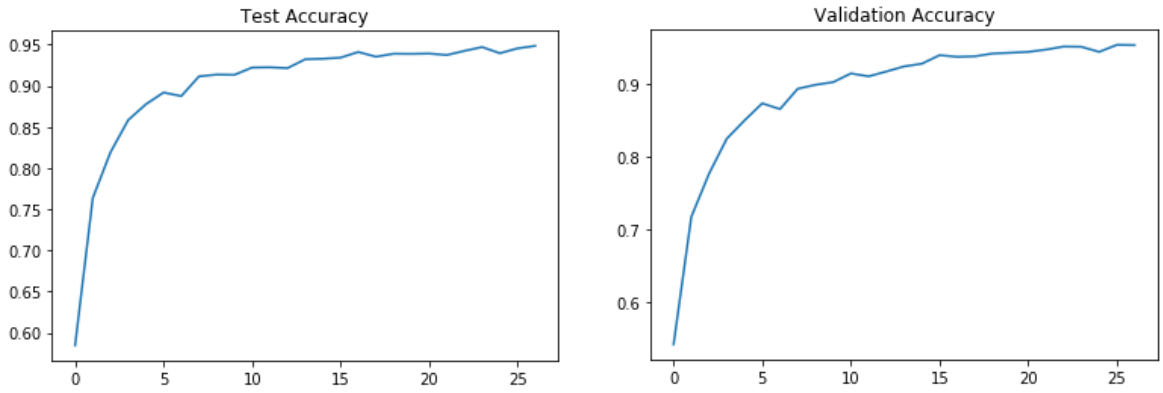


Figure 4: Progression of test and validation accuracy.

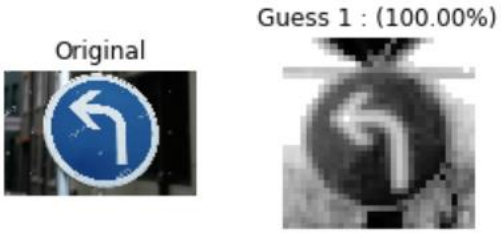
Adrian Chow

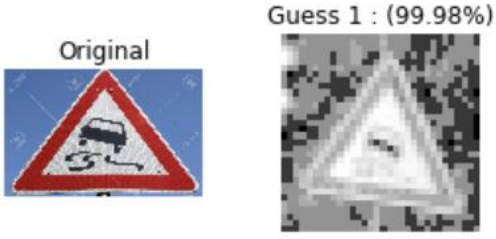
Testing New Images

I chose the following 5 images for testing the model.

Here are the results of the prediction:

Image	Prediction
Yield	Yield
Turn left ahead	Turn left ahead
Bicycles crossing	Bicycles crossing
Slippery road	Slippery road
Speed limit (30km/h)	Speed limit (30km/h)





The model guessed 5 out of 5 correct, scoring 100% accuracy. One way to improve the model Would be further diversifying the data set and Training on a more inspired network model.