Project: Traffic Sign Classifier

Patrick Cleeve

Objective

The objective of the project is to develop a traffic sign classifier. The classifier was developed using a convolutional neural network in tensorflow. The training, validation and testing data is available to download here. The data sets contains German traffic signs from forty-three different sign types.

The model was trained on a single AWS g2.2xlarge with the udacity-carnd AMI.

The project code is available here:
https://github.com/patrickcleeve/CarND-Term1-TrafficSignClassifer-P2

Dataset Exploration

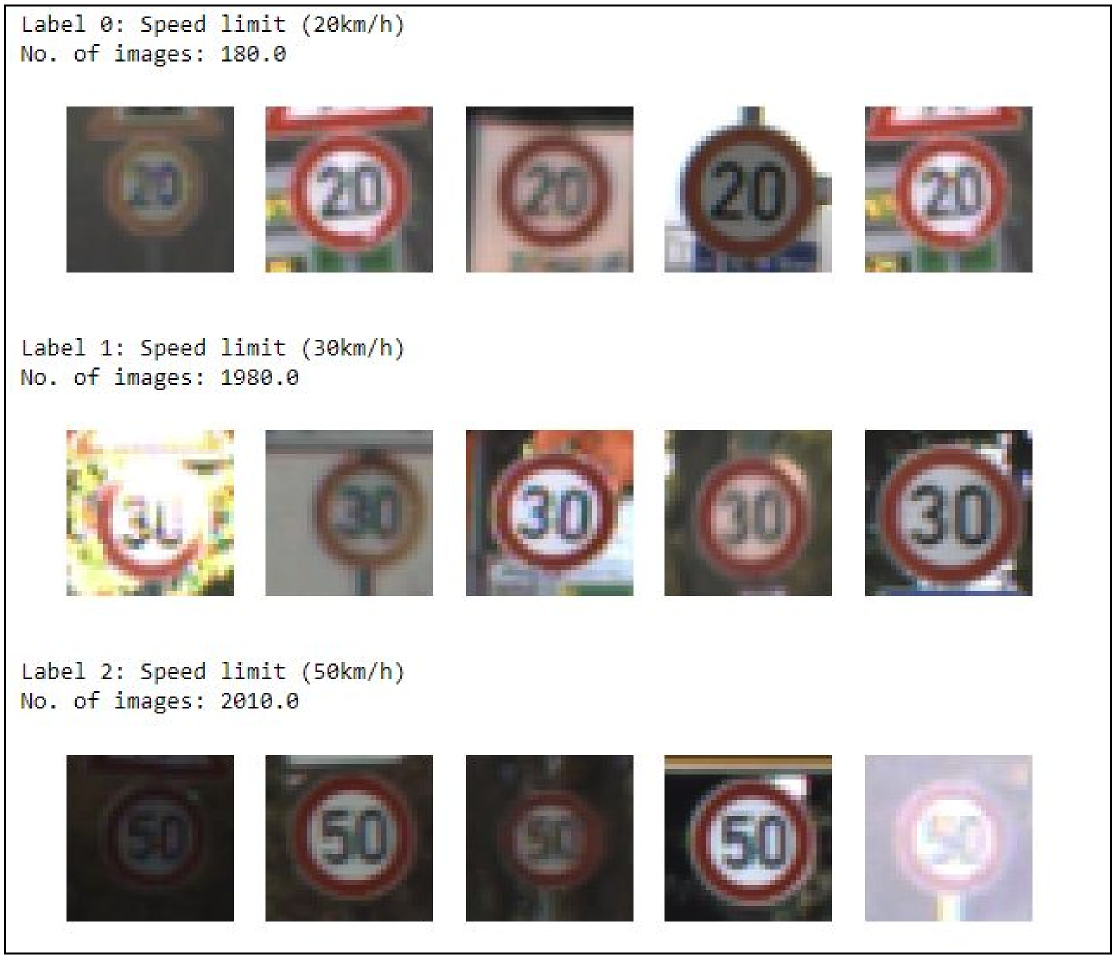
Dataset Summary

The following is a summary of the datasets used for training, validation and testing:

- Number of training examples = 34799
- Number of validation examples = 4410
- Number of testing examples = 12630
- Image data shape = (32, 32, 3)
- Number of classes = 43

Exploratory Visualisation

The following shows a random sample of images of the training set for each of the traffic sign classes (note: not all classes shown here):

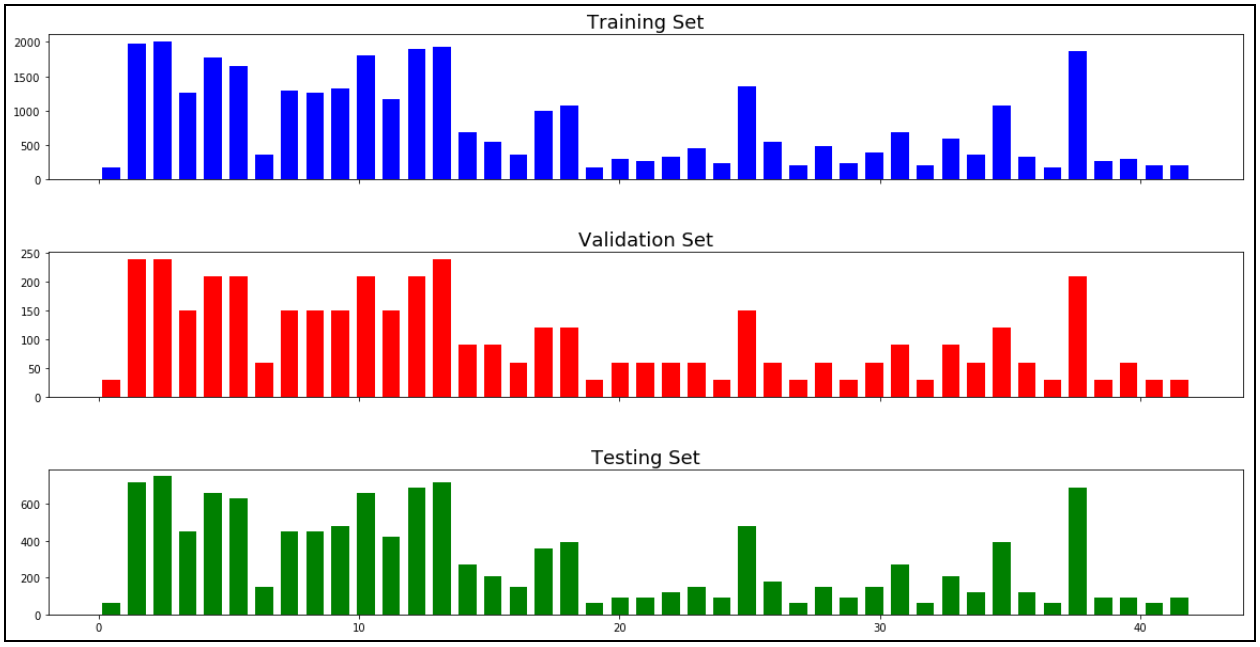


```
Label 0: Speed limit (20km/h)
No. of images: 180.0
```

```
Label 1: Speed limit (30km/h)
No. of images: 1980.0
```

```
Label 2: Speed limit (50km/h)
No. of images: 2010.0
```

The following shows the distribution of image classes within the training, validation and testing datasets:



From the distributions we can see the following characteristics:

- The distribution of classes is highly uneven. Some classes have a large amount of examples, whilst others have very few.
- The distribution of classes is consistent between the training, validation and testing sets.

Design and Test a Model Architecture

Preprocessing

- The image datasets were already sized a 32x32x3 (RGB), therefore no resizing of dataset was required.
- The image datasets were normalised using the following equation (pixel - 128) / 128 so the image data has mean zero and equal variance. This was done to improve training convergence.

Other preprocessing steps that could be added to improve model performance include:

- Grayscale conversion: The dataset can be converted to grayscale to reduce the three RGB colour channels to one. If the colour of the signs is unimportant to the classification task this can improve the as the neural net does not have to learn the colour structure.
- Image augmentation: The dataset can be augmented by rotating, flipping, scaling or transforming images to provide extra training images to the classifier without having to collect additional images. This would expose the classifier to signs at different angles during training to hopefully improve classification of real world images.

Model Architecture, Training and Solution

The base model used for the classifier is the LeNet-5 implementation used in the Convolutional Neural Network section of the class. This architecture contains:

- Two convolutional layers with ReLU activation and max pooling.
- Three fully connected layers with ReLU activation

Dropout has been added to each of the layers. During training, a proportion of the activation between are set to zero. This prevents the network from relying on any particular activation, as it may be dropped. Therefore the network must learn additional, redundant representations in order to achieve high accuracy during training. However, as it is forced to learn the redundant representations the network will take longer to train, but will be more robust.

Additional architecture modifications that may improve model performance include:

- Adding additional layers. When adding additional layers, we may provide additional classifying power to the network, but have to be more concerned about overfitting the training data.
- Training for longer. We could let the network train for longer to see if it improves accuracy.In order to prevent overfitting, we can use a technique such as early termination when the model performance begins to decline.
- Regularisation: We can add a regularisation technique such as L2 Regularisation to penalise large weights and reduce overfitting.

Solution Approach

The following iterative changes were made to the model through the development process.

Iteration	**Image Pre-processing**	**Model Architecture**	**Hyperparameters**	**Validation Accuracy**

1		LeNet	EPOCHS = 10
			BATCH_SIZE = 128
				0.863

2	Normalisation	LeNet	EPOCHS = 10
			BATCH_SIZE =128
				0.911

3	Normalisation	LeNet
		Dropout (fc1)
			EPOCHS = 10
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.899

4	Normalisation	LeNet
		Dropout (fc1, fc2)
			EPOCHS = 10
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.939

5	Normalisation	LeNet
		Dropout (fc1, fc2)
			EPOCHS = 15
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.946

6	Normalisation	LeNet
		Dropout (fc1, fc2, conv2)
			EPOCHS = 15
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.921

7	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 15
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.902

8	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 30
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.941

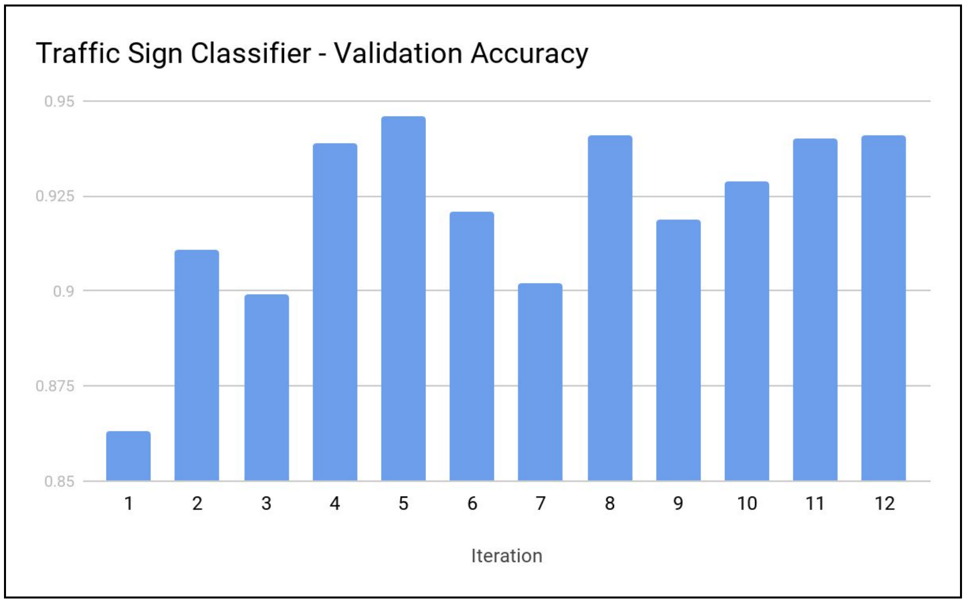
9	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 50
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.919

10	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 30
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.929

11	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 40
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.940

12	Normalisation	LeNet
		Dropout (fc1, fc2, conv2, conv1)
			EPOCHS = 45
			BATCH_SIZE = 128
			KEEP_PROB = 0.5
				0.941

The following chart shows validation accuracy throughout the development process.



- The initial architecture chosen (LeNet-5) performed very well as a baseline, achieving 86.3% validation accuracy, with no adjustments.
- Image normalisation pre-processing dramatically increased the validation accuracy, as the model became easier to train.
- As dropout was added each layer, the model performance degraded. This is because the model required more training as it was required to learn the redundant representations discussed previously.
- As the training epochs were increased, in conjunction with dropout validation accuracy gradually increased.
- However, if the epochs were too high (50) the model performance began to degrade.
- Once the model achieved over 93% consistently no further experimentation was conducted.

Additional parameters that were not iterative tuned include:

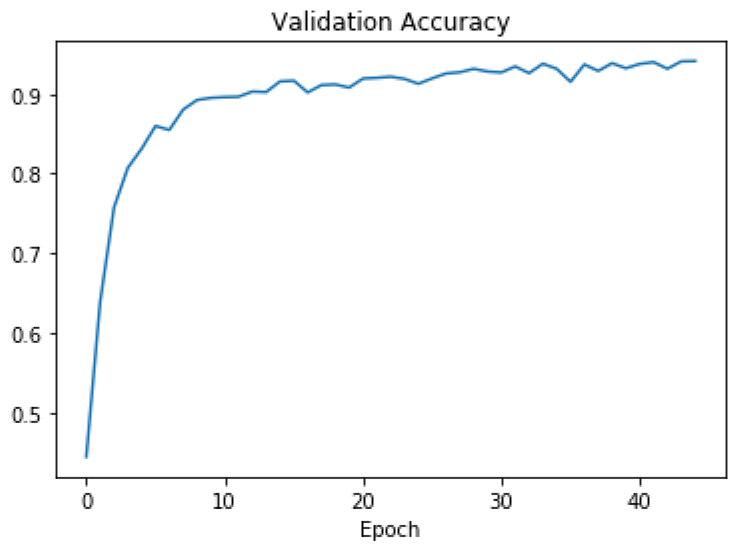
- Learning rate: The learning rate was set at 0.001 and not adjusted. This represents the size of the “step” taken during backpropagation. This could be reduced to test if the model was able to train faster. However, too large of a rate will cause the solution to diverge, and perform poorly.
- Keep probability: This represents the probability that an activation is kept during dropout. Currently it is set at 50% for all dropout layers. This parameter could be tuned for each layer individually to adjust training.
- Batch size: This represents the size of the batch used during training. Larger batches may provide faster training.
- Optimiser: The default Adam optimiser was used, but this is another architectural choice that can be adjusted.

Final Model and Performance

Layer	**Description**
Input	32x32x3 RGB Image
Convolution 5x5 RELU Dropout	1x1 strides, Valid padding, outputs 28x28x6
Max Pooling	2x2 strides, Valid padding, outputs 14x14x6
Convolution 5x5 RELU Dropout	1x1 strides, Valid padding, outputs 10x10x16
Max Pooling	2x2 strides, Valid padding, outputs 5x5x16
Flatten	Outputs 400x1
Fully Connected RELU Dropout	Outputs 120x1
Fully Connected RELU Dropout	Outputs 84x1
Fully Connected Logits	Outputs 43x1

The final model results were:

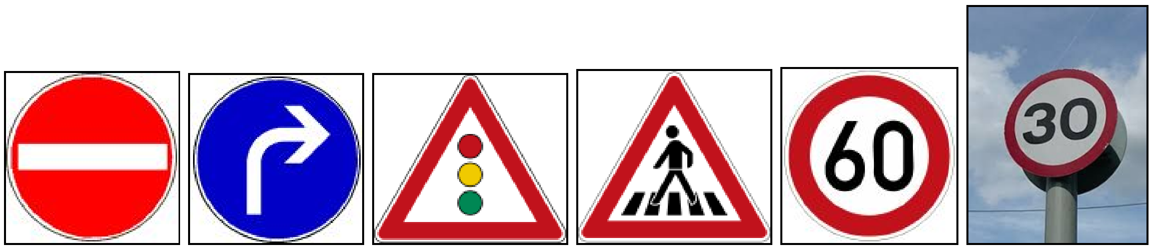
- Validation Accuracy = 0.941
- Test Accuracy = 0.930



Test a Model on New Images

Acquiring New Images

Five images of German traffic signs were downloaded from the internet.



The images were resized to 32x32x3 and preprocessed. Two of the images had transparency channels (Images 3 and 4), these channels were removed with the cv2.cvtColor(image, cv2.COLOR_RGBA2RGB) function.

Performance on New Images

The performance of the classifier was measured on the six new images.

Image	**Prediction**
No entry	No entry
Turn right ahead	Turn right ahead
Traffic signs	Traffic signs
Pedestrians	Pedestrians
Speed limit (60km/h)	Speed limit (60km/h)
Speed limit (30km/h)	Vehicles over 3.5 metric tons prohibited

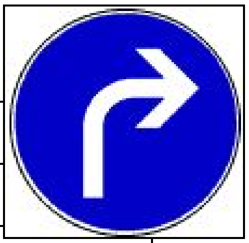
The prediction accuracy of the classifier on the new images is 83.33%.

Model Certainty - Softmax Probabilities

Image 1: No entry

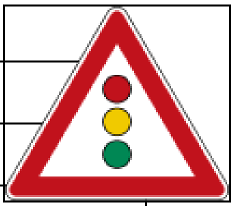
Prediction	Probability
No entry	99.960%
Stop	0.040%
Speed limit (20km/h)	0.041%
Speed limit (30km/h)	0.001%
Bicycles crossing	0.001%

Image 2: Turn right ahead



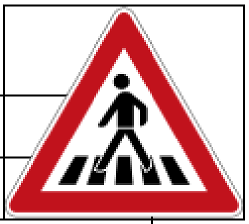
Prediction	Probability
Turn right ahead	99.893%
Keep left	0.066%
Roundabout mandatory	0.022%
Ahead only	0.018%
Turn left ahead	0.000%

Image 3: Traffic signs



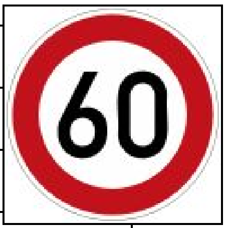
Prediction	Probability
Traffic signals	99.787%
General cauton	0.160%
Priority road	0.047%
Road work	0.003%
No entry	0.002%

Image 4: Pedestrians



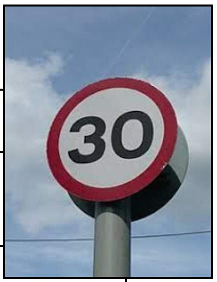
Prediction	Probability
Pedestrians	33.6238%
General caution	21.948%
Right-of-way at the next intersection	13.276%
Road narrows on the right	10.739%
Dangerous curve to the right	5.823%

Image 5: Speed limit (60km/h)



Prediction	Probability
Speed limit (60km/h)	95.355%
Speed limit (80km/h)	4.503%
Speed limit (50km/h)	0.116%
Speed limit (30km/h)	0.046%
End of speed limit (80km/h)	0.000%

Image 6: Speed limit (30km/h)



Prediction	Probability
Vehicles over 3.5 metric tons prohibited	100.000%
Speed limit (20km/h)	0.000%
Speed limit (30km/h)	0.000%
Speed limit (50km/h)	0.000%
Speed limit (60km/h)	0.000%

The model gets five of the six (83.333%) predictions correct, lower than the 93.2% testing accuracy. The following are reflections regarding the images and performance:

- The model is relatively certain about most of the correctly classified images (>90.0%). Lower certainty, such as image 4 may occur due to a lack of training examples (Only 210 pedestrian images in the training set.
- For the image that the model classifies incorrectly (image 6), it has 100.0% certainty regarding its prediction.
- The image that is misclassified is the most “real world” image, the other images are closer to icons than real world images. This may suggest that the model has much worse real world performance than assumed when testing.
- The correctly classified images are perfect representations of the signs. These represent the optimal case that the classifier could expect to receive. In reality, real world images would contain different lighting conditions, viewing angles and other different discrepancies. More real world images, such as Image 6 should be collected to test the performance of the model.

Reflection

Dataset:
The highly uneven distributions of the dataset may be providing a false sense of accuracy. Because we are only measuring total validation accuracy, the classifier may be performing poorly on a number of classes with low representation in the datasets, without a dramatic decrease in overall validation accuracy.

To evaluate this, we should evaluate the accuracy for each class during validation. These would provide extra information to help understand where the classifier is performing poorly, allowing more directed iteration during development.

Neural Network:
The black box nature of neural networks became very apparent during development and testing. When the model performed poorly, it was very difficult to understand why. It was hard to predict how much changes would affect the model. The only way to evaluate the model was to experiment, retrain and measure the change in results.

Ultimately, I developed a better understanding of how convolutional neural networks work, and how architecture changes affect the requirements and performance of the model (e.g. dropout improving model robustness, but requiring extra training). It was also important to understand the computer vision and image aspects, to assess how the pre-processing techniques would affect the model performance.

Improvements

Future improvements and experimentation regarding the pre-processing, model architecture and performance testing include:

- Convert images to grayscale.
- Augment data set with image rotations, blur and/or cropping.
- Add extra layers to neural network, or other techniques discussed
- Measure individual class accuracy during training to assess poorly trained classes.
- Collect and test more ‘real world’ images.
- Complete Step 4: Visualise the Neural Network’s State with Test Images