Self Driving Car Nano Degree

Traffic Sign Recognition Project

Bastian Dittmar

May 10, 2017

1 Introduction

In this project, deep neural networks and convolutional neural networks are used to classify traffic signs. Specifically, a model is trained to classify 43 traffic signs from the German Traffic Sign Dataset. First the data set is explored. Then the data is reprocessed and prepared as input to the model. After the neural network has been modeled, the training and the validation set are used to train the networks parameters. After a satisfyingly accuracy has been reached the performance is checked against the test set. At last, additional traffic sign images from the web are downloaded and classified by the neural network.

2 Dataset Exploration

The Dataset contains three sets of images, for training, validation and testing along with three labels for classification.

2.1 Dataset Summary

The training set contains 43 different traffic signs that have to be learned. The training set consists of 34799 samples, the validation set of 4410 and the test set has 12530 sample images. Only a subimage of the actual traffic sign is used. While all images are resized to 32x32 pixels, the average size of the subimages before resizing is about 39x39 pixels.

2.2 Exploratory Visualization

For a better understanding one sample of each traffic sign label is shown in the notebook as well as a chart that illustrates the distribution of samples per label (Figure 1). One apparent issue with the data set seems to be its non-uniform distribution. Some labels have very few labels which can lead to overfitting.

3 Design and Test a Model Architecture

3.1 Preprocessing

The preprocessing of the data is quite simple. While I explored feeding grayscale images to my network, I eventually went back to color images, thus that specific preprocessing step is gone. The only thing left is normalizing the image data from an interval of [0, 255] to [-1, +1], which speeds

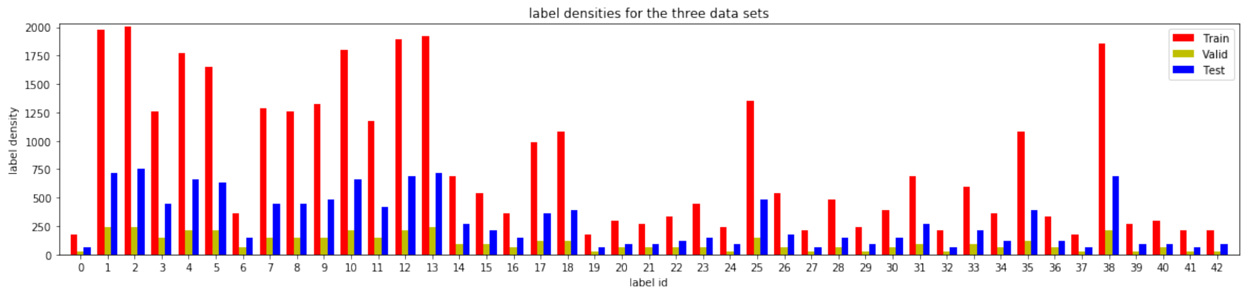


Figure 1: Distribution of samples per label

Layer	Type	Input	Outut	Kernel	Stride	Padding	Activation
1	Convolution	(32, 32, 3)	(28, 28, 48)	5x5	(1, 1, 1, 1)	VALID	relu
2	Max Pool	(28, 28, 48)	(14, 14, 48)	3x3	(1, 2, 2, 1)	SAME	
3	Convolution	(14, 14, 48)	(12, 12, 64)	3x3	(1, 1, 1, 1)	VALID	relu
4	Convolution	(12, 12, 64)	(12, 12, 64)	3x3	(1, 1, 1, 1)	SAME	relu
5	Max Pool	(12, 12, 48)	(6, 6, 64)	3x3	(1, 2, 2, 1)	SAME	
6	Flatten	(6, 6, 64)	2304				
7	Fully Connected	2304	1024				relu
8	Dropout p=0.5	1024	1024				
9	Fully Connected	1024	512				relu
10	Dropout p=0.5	512	512				
11	Fully Connected	512	43				

Table 1: Network layers for traffic sign recognition

up the convergence of the training. Besides that the training set is sorted ascending by label-id which makes it easier to draw a certain label by choice later on.

3.2 Model Architecture

In the final draft of my notebook I used a network layout similar to le-net (LeCun et al. 1998) and alex-net (Krizhevsky, Sutskever, and Hinton 2012). The layers are illustrated in table 1 and start with a series of convolutions where two of them are intermitted by a max pooling. The result of the last convolution gets flattened and then passed on to a series of fully connected layers which are intermitted by dropout layers. As activation functions I used relu. Input into the network is a 32x32 rgb color image. Output is a vector of size 43 (number of signs).

3.3 Model Training

To train the model I minimize the cross entropy of the softmax classifier to the logits while the labels are being encoded. The softmax function converts the logits to probabilities, one for each of my labels. The cross entropy is our distance function that measures the distance between the softmax probabilities and our one-hot encoded labels. The one-hot encoding make the classification mutual exclusive, thus only one label can be assigned to a single input. After each epoch the model is evaluated on the validation set and the weighs are modified using the AdamOptimizer. The learning rate starts at 0.001 and is divided by two every 4 epochs with a total of 20 epochs.

Since the training set is not unoformely distributed I did not just feed randomy ordered batches to the network during an epoch. Instead I used 3 randomly choses samples of each label per batch.

That yields a batch size of 129. The samples are then augmented to lessen the effect of overfitting. Common augmentation techniques like horizontal flipping did not make sense in this application as some of the traffic signs are not invariant to such operations. Instead I used fancy PCA as described by Krizhevsky, Sutskever, and Hinton (2012), which alters the intensities of RGB channels in the training images. This operation simulates variations of color and intensity of the traffic sign's illumination. Examples of such augmentation iss depicted in figure 2

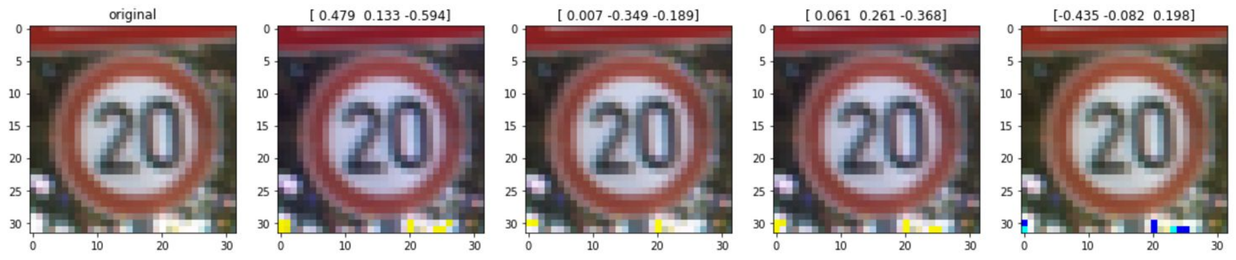


Figure 2: Distribution of samples per label

3.4 Solution Approach

My finals model results were 0.972 on the validation set and 0.960 on the test set. To solve this project I used an iterative approach. I started with the le-net architecture (LeCun et al. 1998) from the LeNet-Lab, added another convolutional layer and added more depth. That alone pushed my validation results to 0.93. Then I added a single dropout before the last fully connected layer which resulted on an accuracy of 0.95. I tested for rgb and grayscale input but the differences were neglectable when dropout was present. I played around with adding more layers and increasing the depth but yield no improvement. In the last step I added fancy PCA for data augmentation and climbed about 2% in accuracy to 0.96 to 0.98.

I also played around with different dynamic learning rates. I tried subtracting a constant amount every n iterations and divisions by different factors. A division of the learning rate by 2 every 4 rounds seemed to yield the best results.

If I had more time for development and training I would increase the network to 5 convolutional and 3 fully connected layers similar to Krizhevsky, Sutskever, and Hinton (2012) who claim a reduction of the number of layers results in quite worse performance of the model. Also I would add other data augmentations like edge cropping and small rotations.

4 Test a Model on New Images

As an additional test I downloaded 5 traffic sign images from the web and classified them using my trained model.

4.1 Aquiring New Images

The pictures I chose are depicted in figure 3. They were resized to 32x32 to be conform with the expected input of the model.

4.2 Performance on New Images

The model guessed 5 out of 5 correctly and thus achieved an accuracy of 1.0 in its final state.

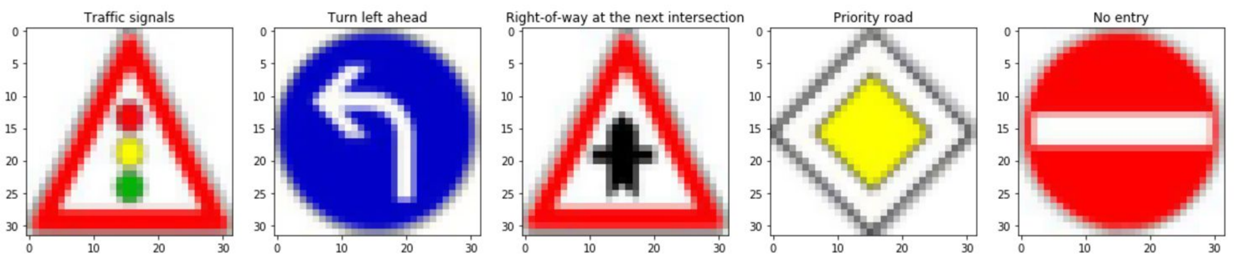
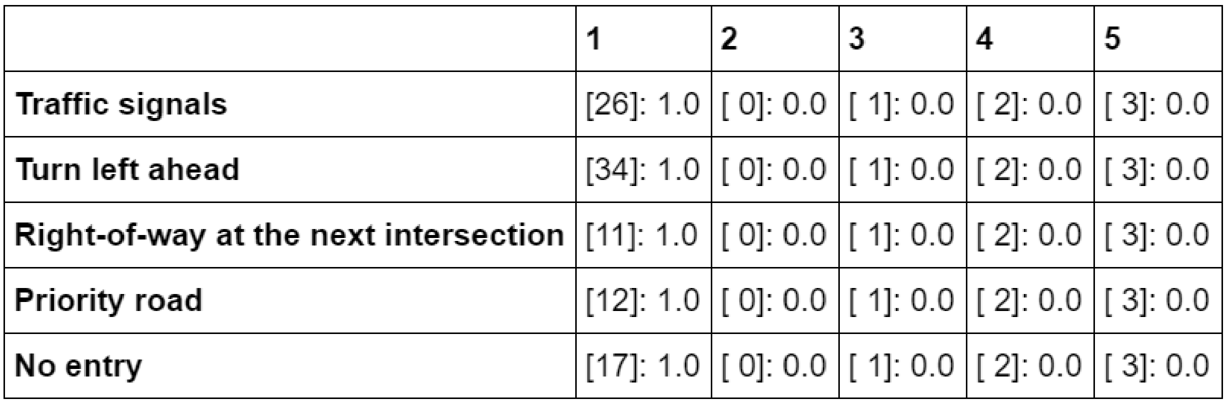


Figure 3: Five random traffic signs from the web

4.3 Model Certainty

The models certainty about its guesses is depicted in figure 2. The models certainty is 1.0 for all five traffic signs with a precision of 8. I trained the final model 3 times with similar parameters and the resulting certainty on this test set was never lower than 0.98.



	1	2	3	4	5
Traffic signals	[26]: 1.0	[0]: 0.0	[1]: 0.0	[2]: 0.0	[3]: 0.0
Turn left ahead	[34]: 1.0	[0]: 0.0	[1]: 0.0	[2]: 0.0	[3]: 0.0
Right-of-way at the next intersection	[11]: 1.0	[0]: 0.0	[1]: 0.0	[2]: 0.0	[3]: 0.0
Priority road	[12]: 1.0	[0]: 0.0	[1]: 0.0	[2]: 0.0	[3]: 0.0
No entry	[17]: 1.0	[0]: 0.0	[1]: 0.0	[2]: 0.0	[3]: 0.0

Table 2: Top five softmax probabilities of the five traffi signs

References

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, et al. Curran Associates, Inc., pp. 1097–1105.

LeCun, Y., L. Bottou, Y. Bengio, et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.