

Traffic Sign Classifier

Anton Pavlov
ap@antonpavlov.ca

October 2017

Introduction

This report describes an architecture and particularities of realization of a neural network for German traffic sign classification. The neural network was developed within Udacity's Self-Driving Car Nanodegree. This report should be read with Jupyter notebooks located at <https://github.com/antonpavlov/traffic-sign-recognition>

Dataset augmentation

Along with neural network architecture described further, the training dataset augmentation was essential to network's accuracy. Pierre Sermanet and Yann LeCun, in their work¹, list the following possible techniques among others:

- Random position perturbation
- Random scale
- Image rotation $-15/+15$ deg.
- Brightness
- Contrast
- Blur
- Some other affine transformations

In this project, an image rotation, a random brightness and a blur were used to extend the training dataset. Application of these techniques permitted to improve accuracy from 0.85 to 0.94 on Test dataset, and 0.96 on Validation dataset. On this very important point, the neural network satisfies the project requirements of 0.93 on Validation dataset.

Neural network architecture

The neural network used in the classification task was derived from Yann LeCun's MNIST network with a changes at the following points:

1. First and second convolutional layers use *tanh* activation function instead of *relu*, and bias are initialized as 0.1, not zeros.
2. Flatten layer flattens only a second convolutional layer.

¹ Pierre Sermanet and Yann LeCun: **Traffic Sign Recognition with Multi-Scale Convolutional Networks**, Proceedings of International Joint Conference on Neural Networks (IJCNN'11), 2011

- Fully connected layers also use *tanh* and 0.1 for biases. They also are separated by two dropout layers.

Neural network architecture is depicted in a Figure 1.

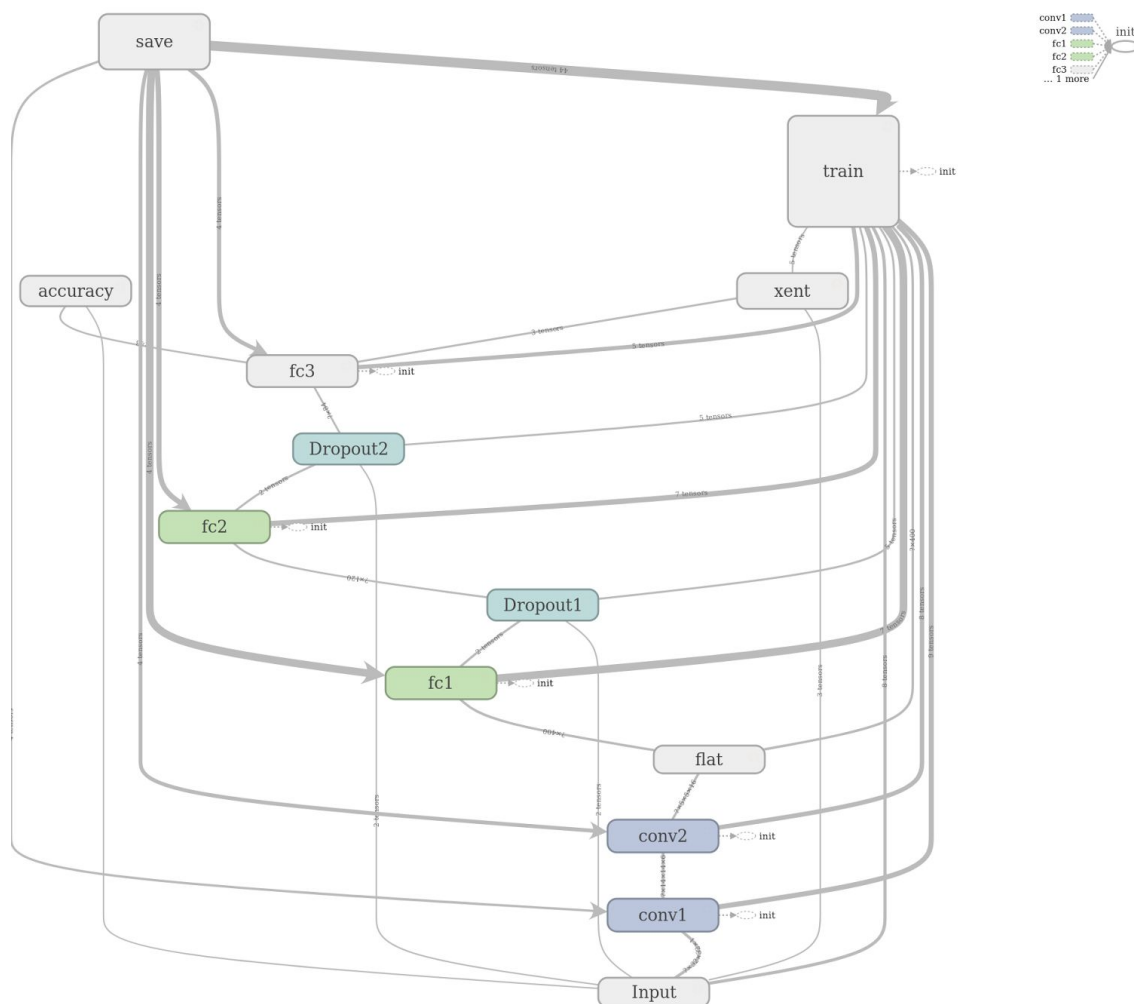


Figure 1. Neural network architecture **convnet**.

As it might be observed on Figure 1, the layers are interconnected on sequential way. Train box has connections to every unit of the network.

An optimization algorithm Adam was used during training procedure. This algorithm is known to be universal for a variety of problems where neural networks are applied. At this very beginning, it is very hard to choose the right one from a number of implemented optimizers in the Tensorflow package: https://www.tensorflow.org/api_guides/python/train#Optimizers. So, the versatility is definitely an important characteristic to be considered at decision point. More about optimizers: <http://runder.io/optimizing-gradient-descent/index.html>

A variety of hyperparameters were implemented in the convnet, for 100 epochs:

- Batch size is 128.

- Learning rate is 0.0005.
- Weights and bias particular initialization.
- Dropout rate switching between 0.5/1.0.

Unfortunately, there are only generalized instructions regarding hyperparameters setup. So, all of them were set empirically, on trial-and-error basis.

For better understanding of neural network architecture and its performance on hyperparameters variation, a Tensorboard was used. The Table 1 shows weights, biases, and activation function histograms updated during training procedure. Moreover, the Table 2 gives another perspective (distributions) to the same weights, biases and activations. All figures showed in the Tables 1 and 2 confirm a convergence of all parameters without anomalies.

Table 1. Histograms of weights, biases, and activation functions.

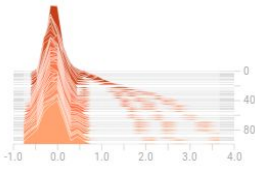
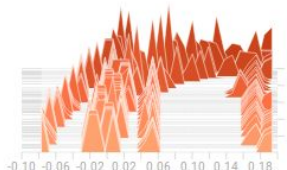
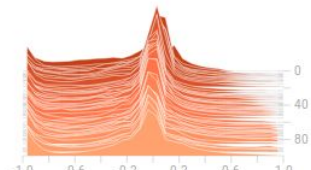
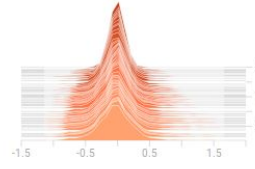
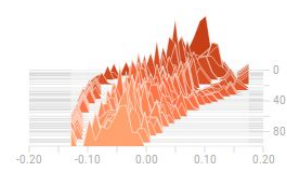
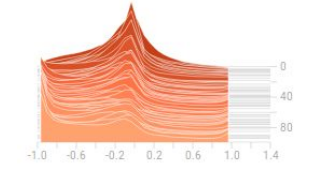
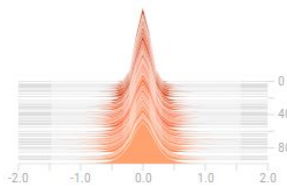
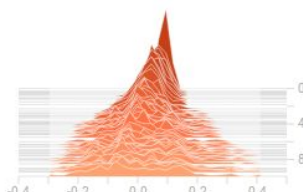
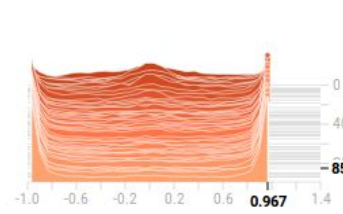
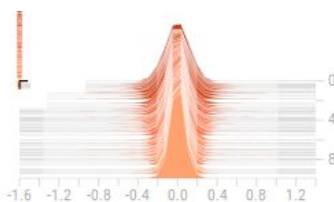
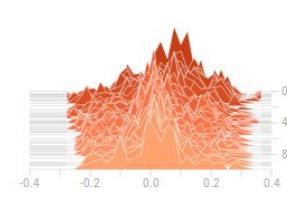
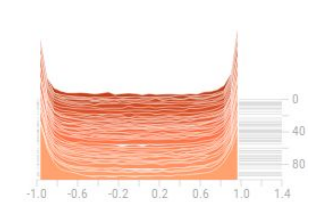
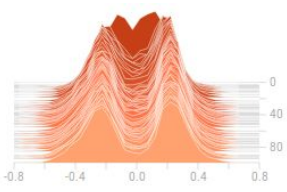
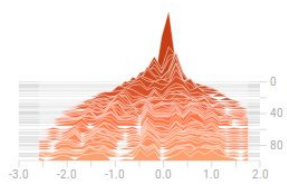
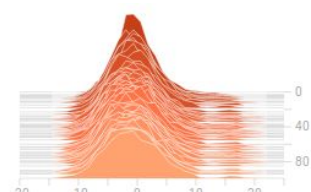
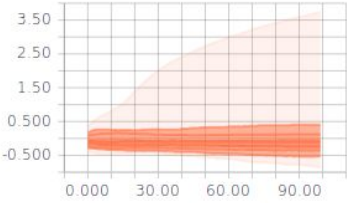
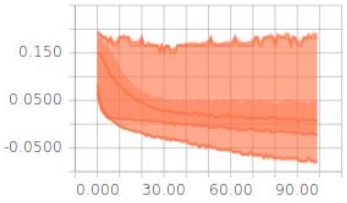
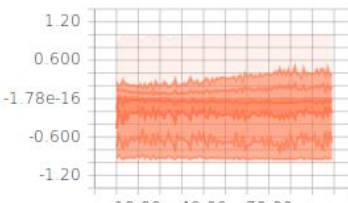
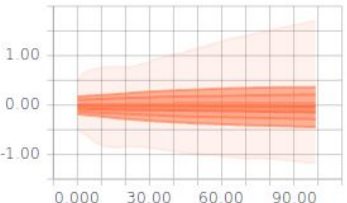
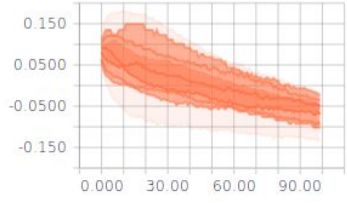
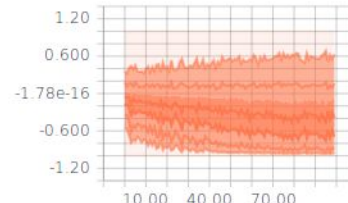
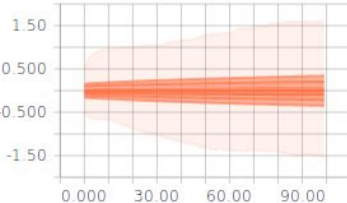
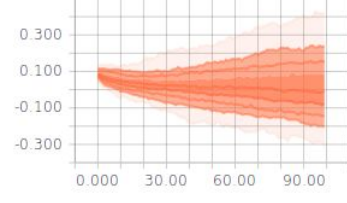
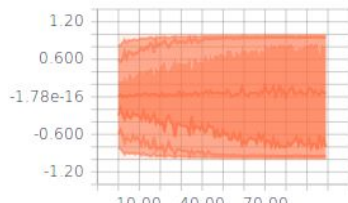
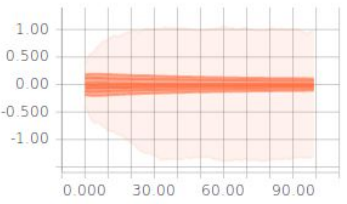
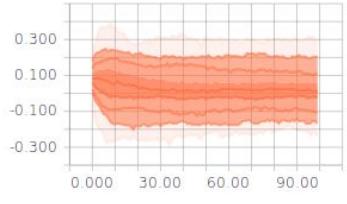
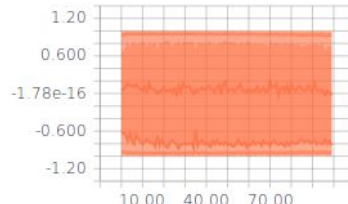
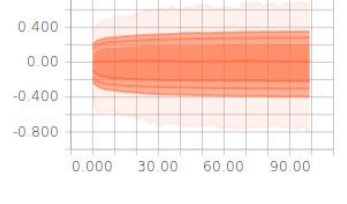
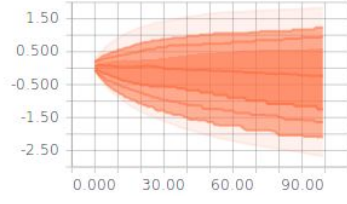
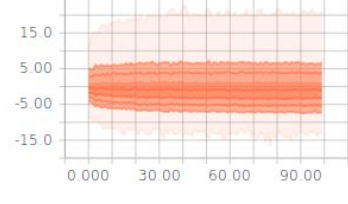
Weights	Bias	Activations
1 convolutional layer		
		
2 convolutional layer		
		
1 fully connected		
		
2 fully connected		
		
3 fully connected		
		

Table 2. Distributions of weights, biases, and activation functions.

Weights	Bias	Activations
1 convolutional layer		
		
2 convolutional layer		
		
1 fully connected		
		
2 fully connected		
		
3 fully connected		
		

Conclusion

The neural network architecture used in this project showed its robustness with high accuracy rates on different datasets. The classification task is not trivial due to the nature of sign images. Sometimes they are very similar to each other. Signs were extracted from video that, in general, has lower image quality comparing to a photograph. The image size 32X32, despite it is related to training and processing, is very small. Obstruction of some parts of a sign is a big problem for classifier as well. Nevertheless, the convolutional neural network were able to classify correctly the most of the traffic sign images.

This project is definitely the right way to enter into neural networks competitions organized by Kaggle and others. In fact, the sign classification was one of them.