Writeup – Traffic Sign Classifier

Author: Jelena Kocić

Date: 15th April 2017.

1. Goals

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

2. Data Set Summary & Exploration

Here the basic summary of the data set is presented. Training, validating and testing data are stored on my PC, location: /Users/Jelena/P2/. For all computations and training model I am using personal PC with OS Windows 10 and NVIDIA Geoforce 940M GPU. Previously, I have installed CUDA and all other dependencies in order to achieve usage of GPU for training data.

Load the data:

```
training_file = '/Users/Jelena/P2/train.p'
validation_file = '/Users/Jelena/P2/valid.p'
testing_file = '/Users/Jelena/P2/test.p'
```

After that I open these files and extract data: X_train, y_train, X_valid, y_valid, X_test, y_test.

I calculated summary statistics of the traffic signs data set:

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

* The size of training set is 34799.

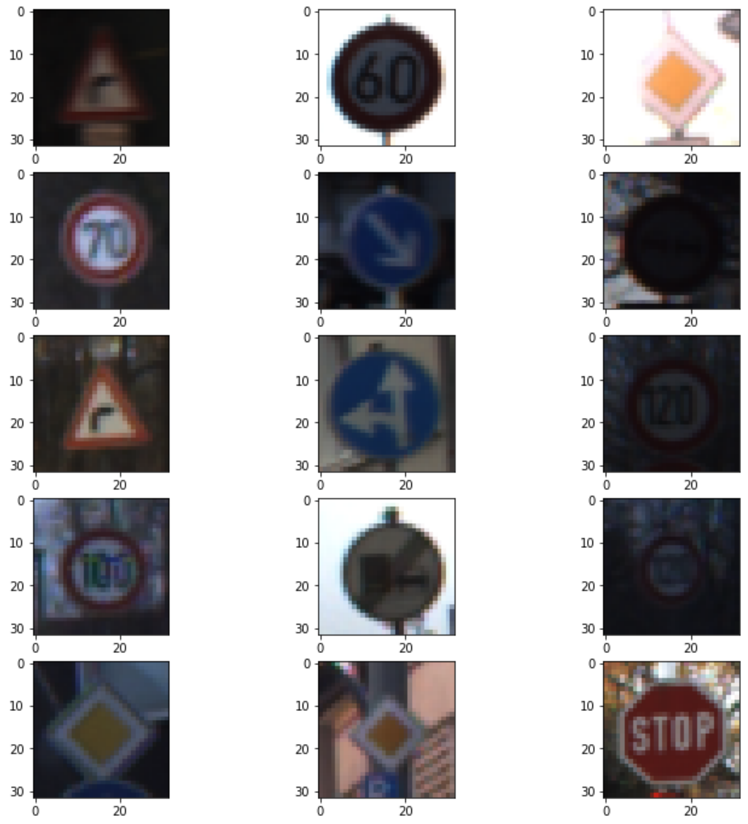
* The size of the validation set is 4410.

* The size of test set is 12630.

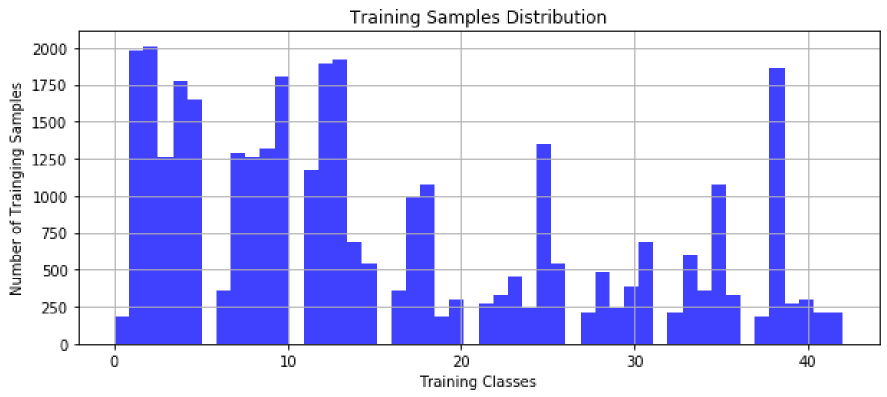
* The shape of a traffic sign image is 32x32x3.

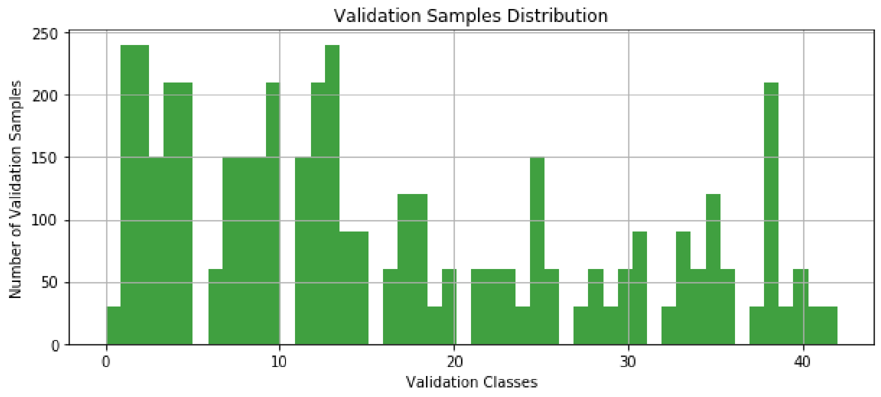
* The number of unique classes/labels in the data set is 43.

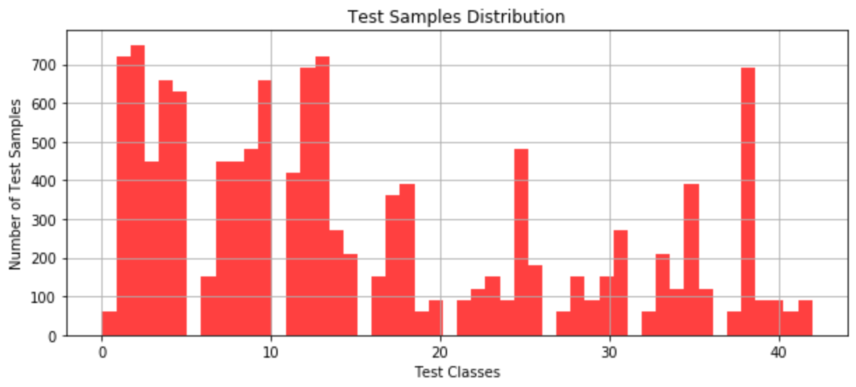
I included an exploratory visualization of the dataset using MATLAB's subplots, and imshow commands. I chose to show 12 pictures:



Also, I made histograms for all three groups of data: train, validation and test:





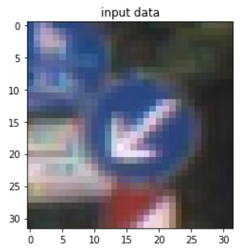


3. Design and Test a Model Architecture

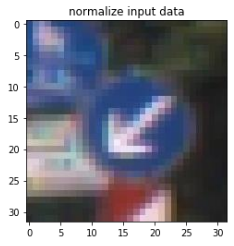
3.1. Preprocessing image data

For the preprocessing image data, I started first with normalization where image have mean zero and equal variance: X_train_scaled = (X_train - 128)/(128). But I explore other options as well. First, I done image reshape. Than MinMaxScaler for scaling each feature to a given range.

Results when we are using only reshape + MinMaxScaler:

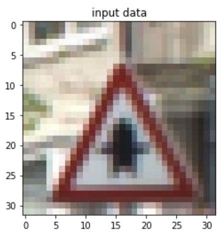


Original image:

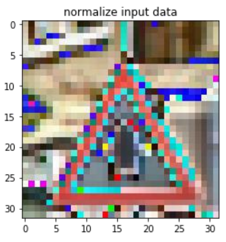


Processed image:

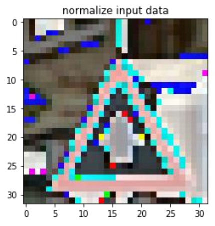
Results when we are using: step 1: (X_train - 128)/(128) and step 2: reshape + MinMaxScaler:



Original image:



Processed image 1:

Processed image 2: 

I trained model several times, and the better results was when I chose the first option.

3.2. Model Architecture

My final model consisted of the following layers:

- Layer 1: Convolutional. Input = 32x32x3 RGB image. Output = 28x28x6.
- Activation.
- Pooling. Input = 28x28x6. Output = 14x14x6.
- Layer 2: Convolutional. Output = 10x10x16.
- Activation.
- Pooling. Input = 10x10x16. Output = 5x5x16.
- Flatten. Input = 5x5x16. Output = 400.
- Layer 3: Fully Connected. Input = 400. Output = 120.
- Activation.
- Layer 4: Fully Connected. Input = 120. Output = 84.
- Activation.
- Layer 5: Fully Connected. Input = 84. Output = 43.

3.3. Trained model and approach

After many trials of the model I chose: EPOCHS = 15, BATCH_SIZE = 128. The ideal case for epochs seems to be 30, or even better 100. But since I am working on local GPU, the training model vas slow, so I choose less number of epochs to train all data. Batch size of 128 sounds as reasonable solution.

For hyperparameters, I chose: mu = 0, sigma = 0.1 (arguments used for tf.truncated_normal, randomly defines variables for the weights and biases for each layer).

Learning rate = 0.0035. I experimented with 0.0001, 0.0005, 0.001, 0.01. But somehow the very small learning rate didn’t reached desired accuracy, and the faster one (0.01, 0.001) is too fast, and again didn’t achieved desired accuracy. So, 0.003 seemed like the right choice.

Accuracy that I managed to achieve is 0.947. I believe that better preprocessing data will lead to better accuracy, but that will remain for another project. Also, taking more epochs (e.g. 100) will give better performance, but also the training time would be much longer.

Architecture which is chosen is LeNet. I just adopted LeNet code from the lessons to the Traffic Sign Classifier.

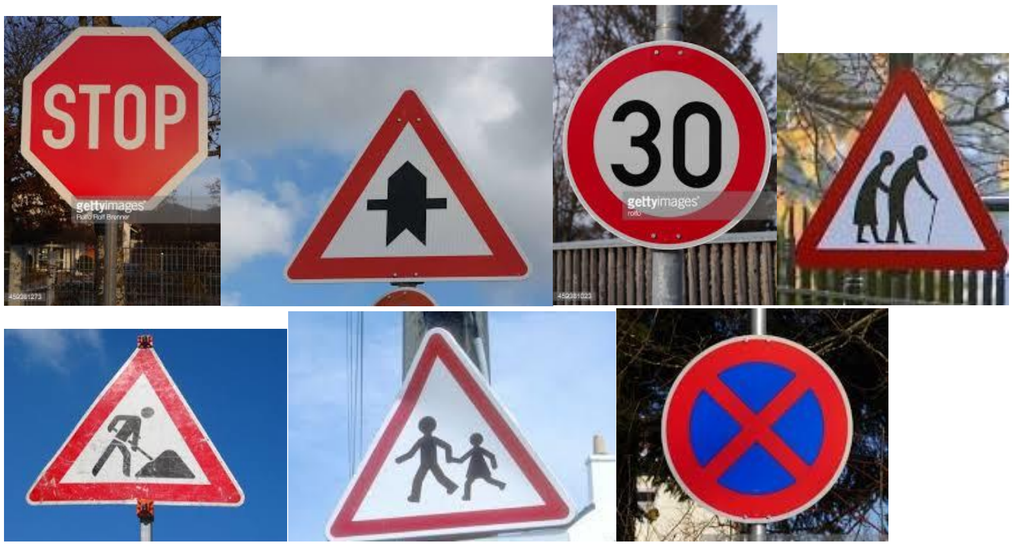
Final model have validation set accuracy of 0.947.

4. Test a Model on New Images

4.1. Acquiring New Images

I picked seven new images that I found on the web. I saved images to the folder: /Users/Jelena/P2/test_signs/

Input:

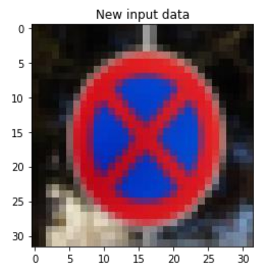


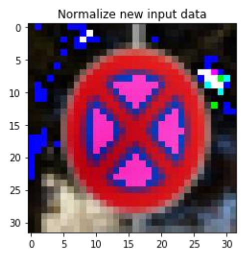
Images are with different lighting conditions, some of them have background as trees or clouds, and some of them have reflection from the sun, which may be additional difficulty to the model to classify these images. Two of those pictures, 4th - “Elderly people” and the 7th - “No stopping'” are not even in the training dataset, and I added labels for them randomly.

Output:

4.2. Performance on New Images

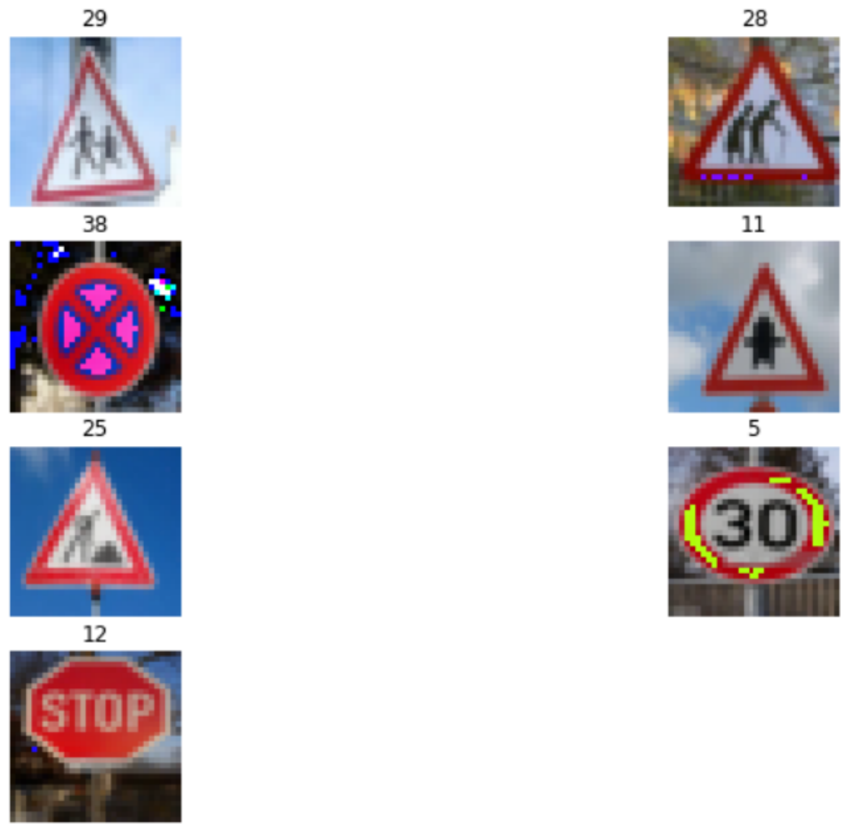
I used the same way to pre-process the new images, as the training and validation set was pre-processed, see the images below.





I run predictions on new data set and here are the results:

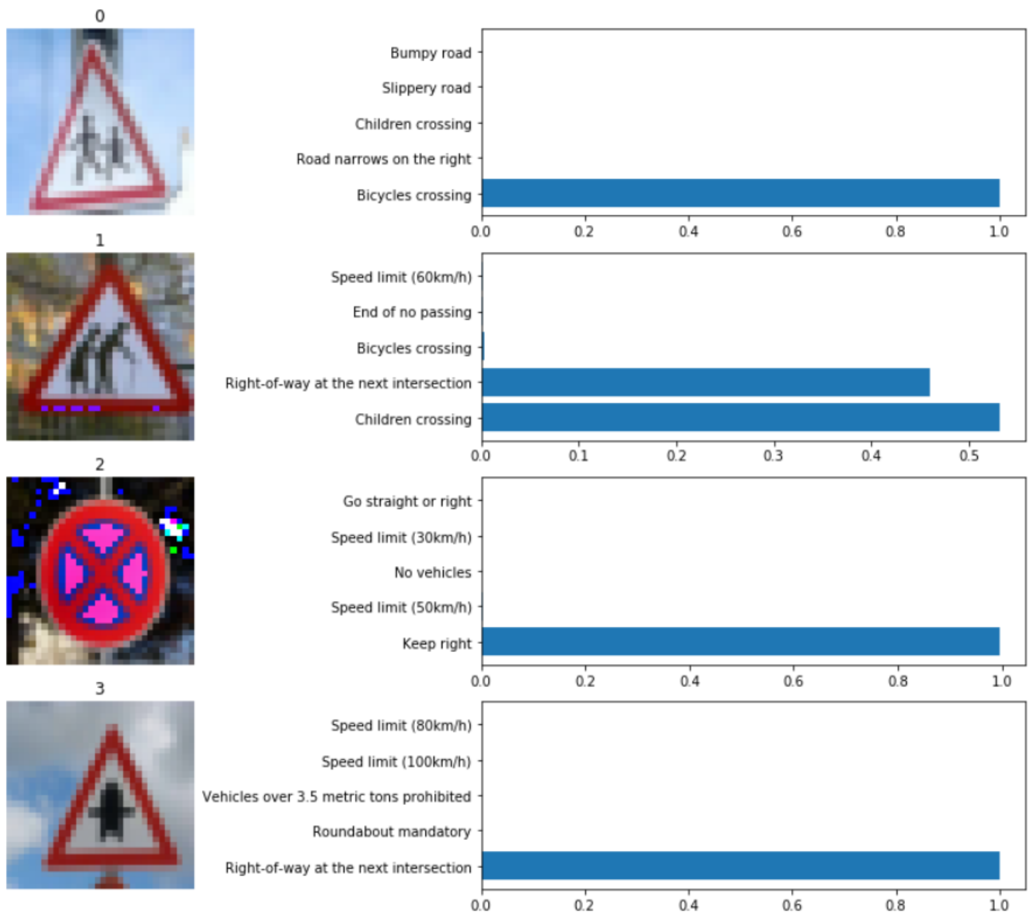
```
Predictions:
[29 28 38 11 25  5 12]
```

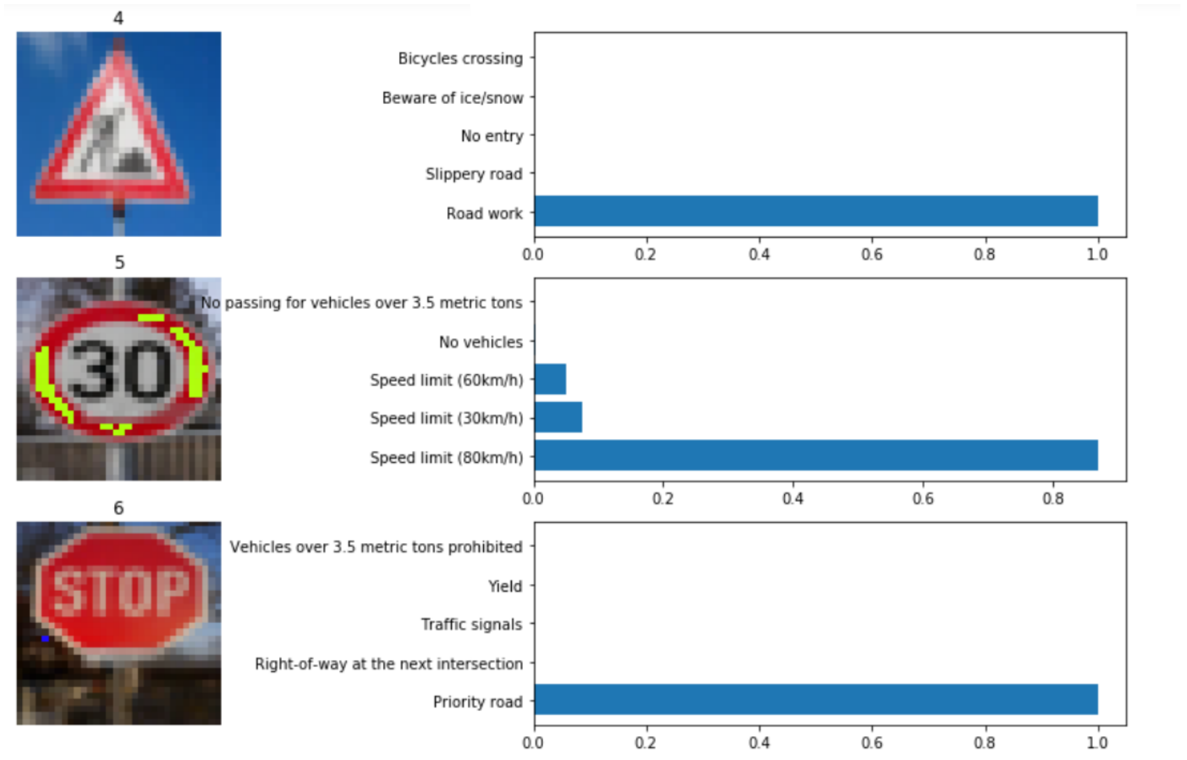


Accuracy of new image set is only about 15%, while we previously saw that model accuracy on test set is 92.8%. This means that one of the seven images is classified correctly. However, for e.g. sign “Elderly people” is not in the training set, so the model recognized it as “Children crossing”, label 28, which in deed is similar image. Also, the image “No stopping” is no existing in training set and this is also the reason of low performance on classifying of new images.

4.3. Model Certainty - Softmax Probabilities

For the end of this project I print out the top five softmax probabilities:





First sign "Children crossing" is incorrectly predicted. Although there is label "Children crossing" on third place for this softmax, it is recognized as "bicycles crossing". It can be partially because low resolution of the particular picture. For the second picture "Elderly people", we do not have this picture in training set. Model recognized it very well as similar pictures "Children crossing" and "right-on-way on the next intersection". "No stopping" is incorrectly predicted, but again this sign is not in input base. "Priority road" is not correctly predicted. Fifth image, "Road work" have accurate prediction! "Speed limit (30km/h)" is recognized as: Speed limit 80, 30 and 60km/h, which is acceptable. At the end "Stop" is incorrectly predicted. In this case background and lighting conditions was aggravating factor.

In total, we can say that model is uncertain, but there is a room for improvement.

5. Summary

Learning about neural networks, tensor flow, deep neural networks and convolutions was quite a challenge for me. I am aware that in this project I "just" apply the knowledge that I gained through lessons. For me it is great success, even though accuracy of validation set is not so great, 0.947. I can say that I really understand the concept of deep learning. I can say that I do understand how LeNet works. And I can say that I succeeded in adopting this network to traffic sign classifier, all by myself, only learning from Udacity lessons and reading additional documents which I found online. It was fun to learn about predictions, to measure (un)certainty by softmax, and so on. I hope that in next project I will achieve even better results. I hope that my knowledge will grow. So far, I can say that I am very satisfied how much I gain from these lessons, how much I learned.