German traffic signs classifier report

Mahdi Helioui – 3AMIndS- 04/01/2019

1. Introduction:  
   The main goal of this project is to create a deep neural network to classify a German traffic sign database. In order to reach this goal, the following steps will be presented in this report:
   1. Load the data set (see below for links to the project data set): Done.
   2. Explore, summarize and visualize the data set: Done.
   3. Design, train and test a model architecture: Done.
   4. Use the model to make predictions on new images: Not done.
   5. Analyze the softmax probabilities of the new images: Not done.
   6. Summarize the results with a written report: Done.
2. Load the data set:  
   To load the data set into the project and to be more easy to work with, I preferred to store the important files (train.p, valid.p and test.p) on my Google Drive. Then, I execute a script which allows me to link my jupyter notebook repository to my Google Drive account. It's easy to use and it needs only to copy and paste two verification codes. After that, the ".p" files can be opened with the "pickle" package.
3. Data Set Summary & Exploration:  
   The data set is constituted by 43 classes of different German traffic signs. They are labeled from 0 to 42. The data set consists in three sets:
   1. The train set with 34799 samples.
   2. The validation set with 4410 samples.
   3. The test set with 12630 samples.

Each sample from this sets has a (32x32x3) shape. The samples are colored and the three colors are red, green and blue (RGB). We could see some of the signs in the figure (Figure 1)



Figure 1: Some images from the German traffic signs data set.

We can see also the classes distribution’s histogram within the different data sets (Figure2, Figure3 and Figure4). And we remark that most of the classes labeled between 18 and 43 are underrepresented. We can try to generate new data same as this classes to balance the classes’ distribution but I will not explore this in this project.



Figure 2: Classes distribution’s histogram within the train set.

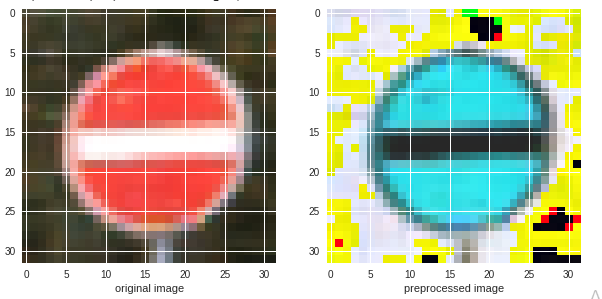


Figure 3: Classes distribution’s histogram within the valid set.



Figure 4: Classes distribution’s histogram within the test set.

1. Design and Test a Model Architecture:  
   The first step is to preprocess the data. There are multiple preprocessing functions we can apply. In this project I just applied normalization and brightness as preprocessing. The goal of the normalization is to have data which will be treated by the same way from the model. And the goal of the brightness is the data augmentation by changing randomly the brightness of the images. We can see an example of “before and after” preprocessing the data in the figure (Figure 5).  
     
     
     
   Figure 5: The difference between the original image and the preprocessed image.



The second step is the model design which is resumed in the figure (Figure 6). For all the layers I applied L\_2 regularization with a parameter equal to 0.001 for both the coefficients and the bias. The activation function for all the layers is “Relu”.  
  
The third step is the training. I used an Adam optimizer with a 0.0009 learning rate and the cross entropy as a loss function. I trained the model for 10 epochs with 256 as a batch size. The model reached an accuracy of 99.6% on the training set and an accuracy of 95% on the test set. We can see the evolution of the training accuracy and validation accuracy in the figure (Figure 7).

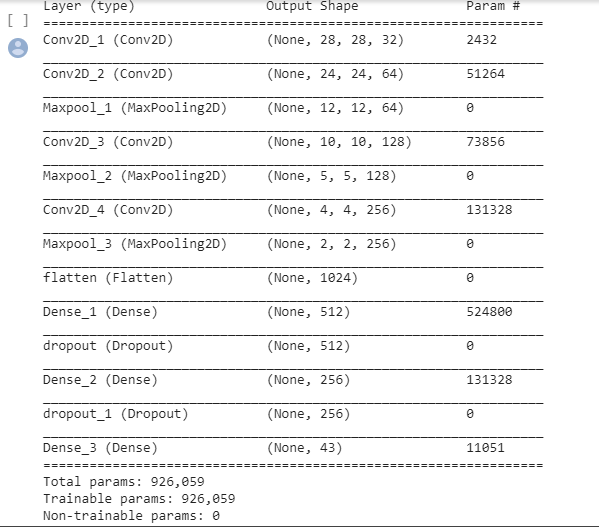


Figure 6: The summary of my model.

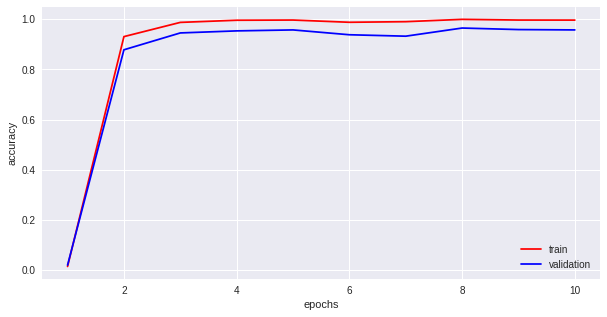


Figure 7: Evolution of training accuracy and validation accuracy.

1. Test a Model on New Image:  
   I didn’t do this part due to time issues.