Self-driving cars program - project 3: Traffic Signs Classifier

Francesco Boi

Contents

1 Content of the project

Here is the content of the project:

- *writeup.pdf* (this file): report of the project;
- *writeup.tex*: tex source code for;
- *writeUpImgs*: folder containing the images shown in this document;
- *Traffic_Sign_Classifier.ipynb*: ipython notebook with the code for the classifier;
- *webSigns*: folder containing traffic sign images downloaded from the Web to test the NN on;

- *OtherStudentsImgs*: folder containing traffic sign images used by other students' projects to test the NN on.

Here is the link to the github project.

2 Project goals

The steps of this project are the following:

- Download and load the dataset.
- Explore, summarise and visualize the data.
- Design, train and test a neural network.
- Use the model to predict new images.
- Analyse the softmax probabilities of the new images.
- Show the activation of the hidden convolutional layers.

3 Dataset summary and exploration

The dataset consists of a training, validation and test set with the following characteristics:

- The size of the training set is 34799.
- The size of the validation set is 4410.
- The size of the test set is 12630.
- The shape of a traffic sign image is $(32, 32, 3)$.
- The number of unique classes/labels in the data set is 43.

The set sizes and images shape have been calculated using the `shape` attribute of the numpy arrays. The number of classes have been calculated in two ways: looking at the maximum value of `y_train` and adding 1 and confirmed by using the `numpy.unique` function.

```
n_classes = max(y_train)+1
unique, counts = np.unique(y_train, return_counts=True)
assert n_classes == len(counts)
```

3.1 Visualisation

Figure 1 shows some images of the dataset whereas Figure 2 shoes the histograms of the classes on the training, validation and test set. Note two things: some images are of really bad quality; note also that the classes are not equally represented: class 0, 41 and 42 have really few smaples whereas class 1 and 2 are overrepresented, 10 times more than class 0 for example.

(a) Speed limit of 20 Km/h sign

(b) Stop sign

Figure 1: Example of images from the dataset.

4 Design and test a model architecture

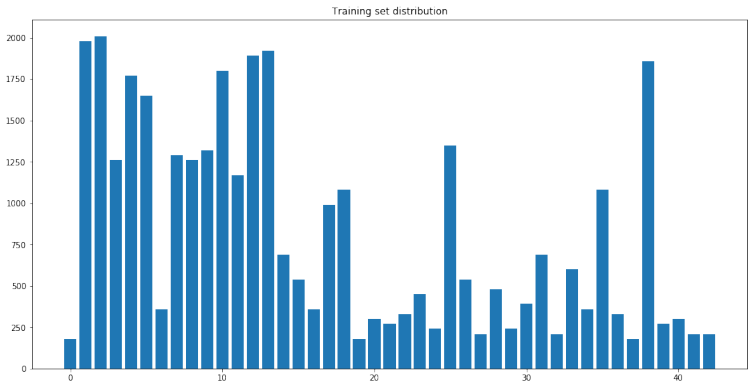
4.1 Preprocessing

The preprocessing consists of two steps: converting the RGB images to grayscale and normalising the value between -1 and 1 (almost 1 actually, since $(255 - 128)/128$ is not 1 exactly). Normalisation should improve and speed up the training process. Gray scale conversion should also speed up the training process since there are less parameters to train. Surprisingly, after running some trainings, it seems that the training process benefits from gray scale conversion. However, colours are very important in traffic sign recognition and a deeper network might take advantage of it.

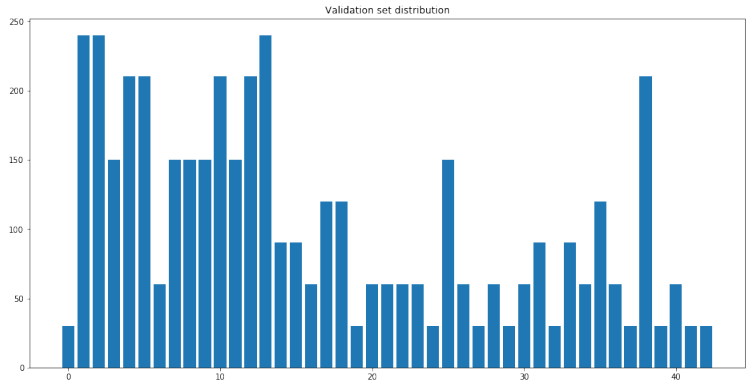
Some gray scale images are shown in Figure 3

4.2 Model

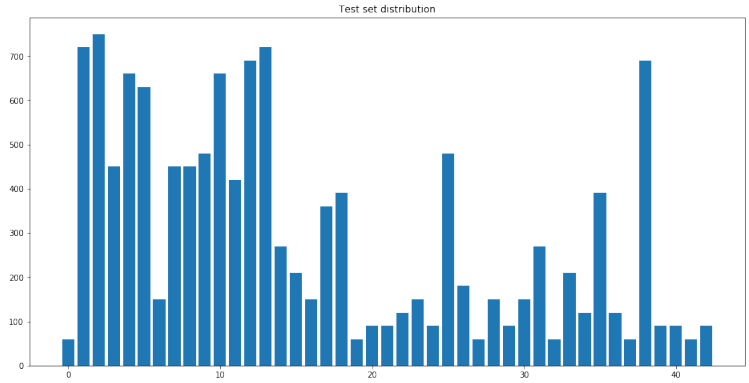
The model architecture is the following:



(a) Histogram of the classes in the training set



(b) Histogram of the classes in the validation set



(c) Histogram of the classes in the test set

Figure 2: Example of images from the dataset.

(a) Speed limit of 20 Km/h sign in gray scale

(b) Stop sign in gray scale

Figure 3: Example of images from the dataset.

Neural network layers					
Layer	Layer shape	Input shape	Output shape	Stride	Pad
Input	(32, 32, 1)	/	/	/	/
Conv	(5,5,6)	(32,32,1)	(28,28,6)	(1,1)	valid
Relu	/	(28,28,6)	(28,28,6)	/	/
Max Pool	(2,2)	(28, 28, 6)	(14, 14, 6)	(2,2)	valid
Conv	(5,5,16)	(14, 14, 6)	(10,10,16)	(1,1)	valid
Relu	/	(10,10,16)	(10,10,16)	/	/
Max Pool	(2,2)	(10,10,16)	(5, 5, 16)	(2,2)	valid
Flatten	/	(5,5,16)	(400)	/	/
Dropout	/	(400)	(400)	/	/
Full	(120)	(400)	(120)	/	/
Relu	/	(120)	(120)	/	/
Dropout	/	(120)	(120)	/	/
Full	(84)	(120)	(84)	/	/
Relu	/	(120)	(120)	/	/
Dropout	/	(84)	(84)	/	/
Full	(10)	(84)	(10)	/	/

The softmax activation is called outside the `LeNet()` function as in the lecture.

To find a good set of hyper parameters, different models have been trained using a batch size of 128 or 256, a probabililty for the drop out layers of $0.5, 0.6, 0.7$ and 5 rates between 0.0001 and 0.001. Among these, the best result was with batch size 128, rate 0.00055 and probaility 0.7.

The number of epochs is 45 but there is little to no improvement after 40. Such a high number was chosen since a relatively small rate was used, hence the model kept improving. The Adam optimizer has been used. With this configuration I got a validation accuracy of 0.946 and a test accuracy of 0.929 (section *Test accuracy* and *Validation accuracy* in the notebook). On the

training set the final accuracy is of 0.994 so the model is slightly overfitting (section *Training accuracy* in the notebook).

Adding the dropout layer significantly improved training and validation accuracy since it prevents overfitting.

The LeNet architecture was used. This works quite well thanks to the convolutioal layers that are suitable for image applications. A convolutional layer uses less parameters than a fully connected layer and it allows to treat each region of the image equally since the parameters are the same.

5 Test a model on new images

Figure 4 shows some images found on the Web.

4f might be difficult to classify because differently from the other training images, the sign has a yellow background instead of a white one, although the grayscale conversion might mitigate this. The other images look quite clean. 4e is not actually a picture but a digital representation of the sign.

5.1 Accuracy on web images

The accuracy in this case is very poor, of 0.286, infinitely worse than the test set.

Figure 5 shows the classification results. Only two classes are classified correctly: 4e as expected and 4c. The wild crossing sign is confused with the slippery road sign, which is unerstandable: they look very similar even to the human eye for such low resolution images. The general caution sign is not classified correctly, probably due to the yellow colour, but at least the NN recognised a triangular shape. The other stop sign (class 14) and the 20 km/h speed limit (class 0) are not classified correctly. Very likely, this is because these classes are underrepresented in the dataset. Especially for the 20 km/h sign, the class 30 km/h (class 1) is very similar but has roughly ten times more instances (see Figure 2), so the NN has learnt it has more chances of doing a better job by outputting class 1, since class 0 is very unlikely. In fact, this is exactly what it is doing: the 20 km/h speed limit is classified as th 30 km/speed limit.

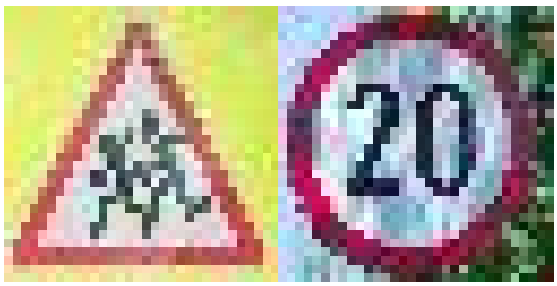
Further analysis should include a confusion matrix. Getting more instances of the underrepresented classes might be beneficial.

6 Test a model on new images

Figure 4 shows some images found on the Web.

6.1 Accuracy on other students' images

In the section *Test on other students' images*, the NN has been tested on images used by other students for the same project. In this case the accuracy is 0.643,



(a) Children crossing (b) Speed limit 20 Km/h

(c) Slippery road sign (d) Stop sign

(e) Stop sign clean (f) Warning sign

(g) Wild animals sign

Figure 4: Web images to test the neural network on.

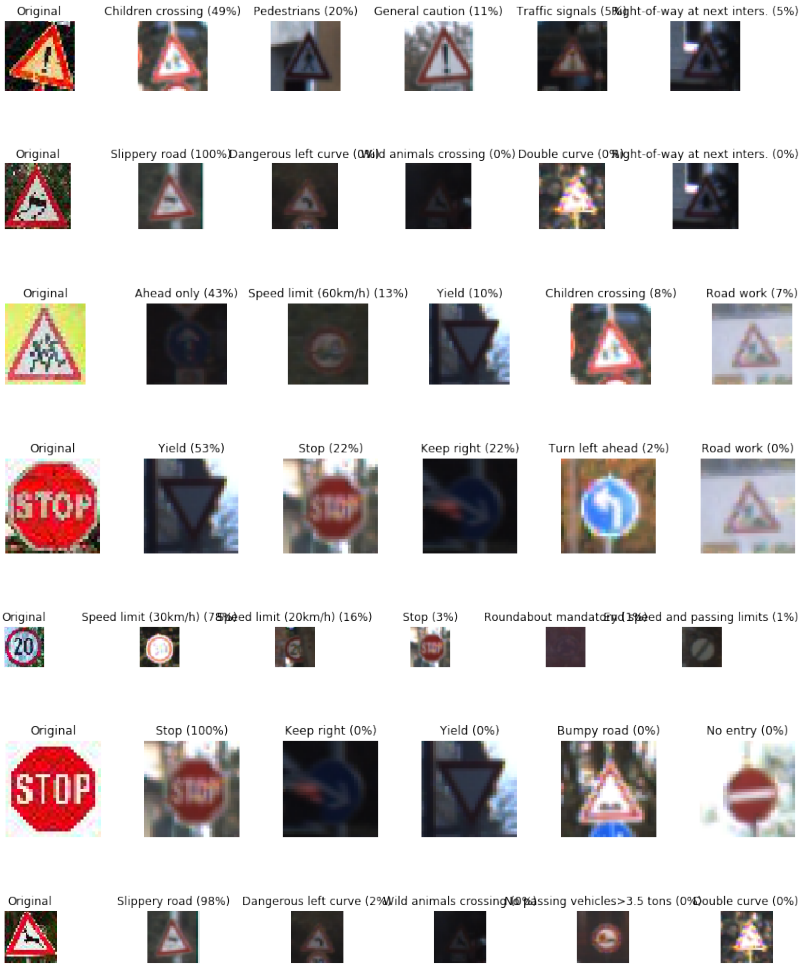
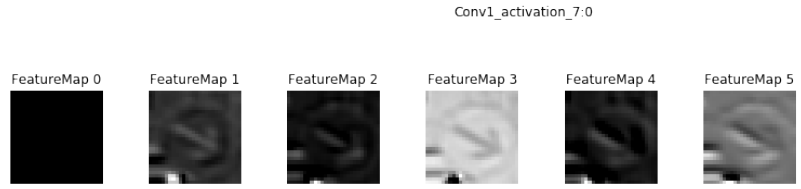
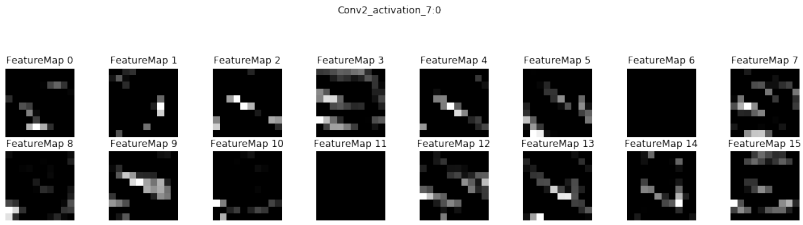


Figure 5: Predictions on the new images.



(a) First conv layer



(b) Second conv layer

Figure 6: Activations of the convolutional hidden layers.

which is signifiantly better than the previous one. Probably I have chosen some bad images to test the NN.

7 Visualisation of the hidden layers

Figure 6 shows the activations of the convolutional layer. Note that some feature maps are not activated (at least for this input). The first layer seems to learn a general shape whereas the second layer seems to highlight some details in the image. The result of the first layer looks similar to different derivative filters: the image is still recognisable. The second layer seems to look for the diagonal arrow.