Writeup of Thierry Monteil project

Traffic Sign Recognition

1) Build a Traffic Sign Recognition Project

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

a) Summary of data set

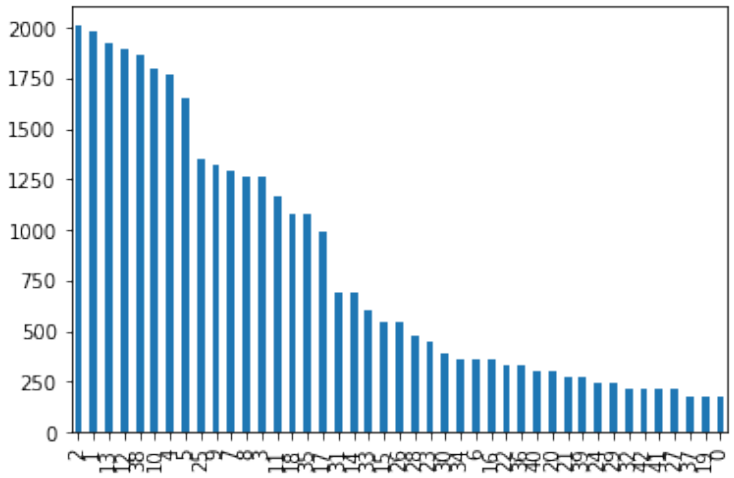
After opening the different sets, the calculation of the size and characteristics is done using len function and panda library to build a serie with labels and then keep only one entry per type of label.

I used the pandas library to calculate summary statistics of the traffic signs data set:

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

b) visualization of the dataset.

It is a bar chart showing how the data the distribution of labels. This is showing that some sign occurrence is very different. It varies from about 2000 to a few occurences.



3) Design and Test a Model Architecture

a) Pre-processing

As a first step, I have decided to convert the images to grayscale because the RGB color is not an important parameter in this classification and it helps to reduce the size of the models.
Then, I normalized the image data for stability reasons of the algorithm. I shuffle the image to see the impact of the order of images.

The size of the image is coherent with the size use in LeNet neuronal network so I don't need padding. I have done not more pre-processing because I would like first to test other mete-parameters of the models. And in fact, I get correct results to validate this project, even if for real use I will need to improve it.

b) Model description

As suggested, I use the LeNet models as base. The only modification has been to add the dropout mechanism at different levels if the neuronal network.

- After Prepocessing: 32x32x1 image shape
- Convolution layer 1. The output shape should be 28x28x6.
- Activation 1. Relu activation
- Pooling layer 1. The output shape should be 14x14x6.
- Convolution layer 2. The output shape should be 10x10x16.
- Activation 2. Relu activation
- Pooling layer 2. The output shape should be 5x5x16.
- Flatten layer. Flatten the output shape of the final pooling layer such that it's 1D instead of 3D, output 400
- Fully connected layer 1. This should have 120 outputs.
- Activation 3. Relu activation
- Dropout with a keep probability of 0.6
- Fully connected layer 2. This should have 84 outputs.
- Activation 4. Relu activation
- Dropout with a keep probability of 0.6
- Fully connected layer 3. This should have 43 outputs.

c) Training

The LeNet models has been chosen for is capabilities of classification of images. The Adam optimizer is used to find the parameters of the models. To train the model, I used an epoch of 50 and a batch size of 500. Those values have been found through an iterative process to get a validation and test accuracy over 0.93.

As suggested in the course over different strategies: experiment different network architecture, change dimension of the network, add regularization, avoid overfitting, tune the hyperparameters, improve data pre-processing or augment the training data, I have chosen between different strategy the simplest ones. First, I change the epoch and batch to see how it can improve accuracy. I have first started with the parameters given in the example in the lesson and then I have increased them by step. Then I have added dropout after activation in the last 2 layers of the network. I have tuned the" keep probability" and decrease it to make the network more robust until I get a correct value of accuracy (from 0.75 to 0.6).

My final model results were:

- validation set accuracy of 0,953
- test set accuracy of 0,938

This rate change between two different training session due to the shuffle of images.

4) Test a Model on New Images

I spend a long time on this part just to integrate images due to difficulties to manipulate images in python and make them at the correct characteristics for the model. First, I thought that I have done error in the model, Finally, I pick images on a project in github and find that it is only a problem of format.

Here are five German traffic signs that I found on the web:

On this set the accuracy is 0.20, so only one image has been really recognized.
The probabilities of top 3 softmax logits are:

```
TopKV2(values=array([[9.9230403e-01, 6.8645533e-03, 7.7699055e-04],
       [9.7705078e-01, 2.1359736e-02, 6.9532683e-04],
       [7.3659205e-01, 2.4977009e-01, 6.5564099e-03],
       [9.7676575e-01, 2.0086534e-02, 3.0461350e-03],
       [1.0000000e+00, 1.0799141e-11, 9.9727258e-12]],dtype=float32),
indices=array([[32,  3, 12],
       [ 1, 14, 35],
       [22, 25, 36],
       [ 3,  6, 32],
       [13, 35,  1]], dtype=int32))
```

I try several runs after different training process and get a maximum results of 0,60 so 3 of 5 fives signs have been recognized.

5) **Conclusion and Improvement**

The process can be optimized at several levels:

- Pre-processing: I just have centered the values representing the image with (x-128)/128 supposing that the mean is 128. It can be more precise with the real calculation of statistic models and specific correction. I haven't try other more sophisticated method to increase diversity of the set.
- Training: add more sophisticated layer suggested in the course

I am still needed to better understand the impact of different layers and how they can be tuned. After this course, I am just the level to apply consigns and all runtime errors generated in tensorflow are difficult to manage and understand.