December 4, 2017 – P2 Write-up

### Data Set Summary & Exploration

#### **1.** *Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.*

The training test size is 34799; The test set size is 12630. Some of the test and training test size was also remade later on to increase and balance out the validation test set.

The entire sign image data set has the size of 32 by 32 pixels with an image depth of 3 (1 for each color in the RGB).

It should also be noted that there are 43 unique classes for each of the data sets

#### **2.** *Include an exploratory visualization of the dataset.*

In the 3rd cell an exploratory visualization was included firstly to demonstrate a random set of four (4) images from the data set. In the 4th cell each of the data sets was plotted via the self-defined "plot_graph" function in order to get a better idea how the sets are distributed – and potentially even out the distribution a little bit to help with training and validation.

### Design and Test a Model Architecture

#### **1**. *Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.*

It should be noted that some of the starting points for my project was in reference to the technical document by Pierre Sermanet – with a few steps left out for further study at own time with own neural networks ☺

1.) In cells 7 and 8 I applied the first step of pre-processing: **Grayscaling.**

Greyscaling improves the learning process for relatively simple to identify objects while reducing the computational cost (image depth reduction from 3 to 1). While I didn't get a chance to experiment with grayscaling ON or OFF every time I made a change or addition, during the early phases of just having the LeNet convolutional network I noticed a slight performance increase for my validation data – hence I kept it. I tried once more after normalizing and obtained the same results. Once I have a bit more time on my hands I will experiment further to obtain a better idea of when exactly grayscaling helps and in what conditions it doesn't!

In cell 9 I outputted the data once again in a random format as before, just to make sure that the gray scaling process went smoothly.

2.) In cell 10 a **Randomization** technique was performed.

Randomization typically not only reduced the likelihood of overfitting data – but can also improve performance.

In cell 11 the randomized image can be seen on the left while the original is on the right. The image on the left appears to have a softer color; however it is also a product of tilting, and then concatenating the data to increase the data set.

3.) In cell 13 a **Normalization** technique was performed.

As was mentioned in the lesson, creating a data set with zero mean and equal variance reduces the likely hood of error propagation – helping the network learn more effectively.

#### 2. *Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.*

Following this, all of the data – now processed was fed into my neural network on cell 16. It is a slightly altered LeNet with a few points of drop-out and some lesson inspiration:

Layer	**Description**
Input	32x32x1
Conv 5x5 +RELU	2x2 stride, valid padding, with an output of 28x28x6
Max pooling	2x2 stride, with an output of 14x14x6
Conv 5x5 +RELU	2x2 stride, valid padding, with an output of 10x10x16
Max pooling	2x2 stride, with an output of 5x5x16
Conv 1x1 +RELU	2x2 (5) stride, valid padding with an output of 1x1x412
Fully Connected + RELU +dropout	412 -> output of 122
Fully Connected + RELU +dropout	122 -> output of 84
Fully Connected	84 -> output of 43 (# of unique classes for each of the data sets)

#### 3. *Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.*

A more detailed explanation about my training procedure can be found in the next section, as my training technique was the same as my technique for improving validation set accuracy. The epoch number, learning rate number and batch size were all changed (in coarse gradients) every time I implemented a new technique with regards to pre-processing or in the neural network pipeline itself. For the final iterations of learning rate I tweaked it in a fine manner, but also in a way that didn't run too many training sessions.

Final Settings used were -> Epoch: 15, Batch Size: 156, learning rate: 0.001

#### 4. *Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.*

The process was rather sequential in terms of pre-processing – taking inspiration from technical documents, the lesson as well as good practices techniques and then modifying the learning rate, batch size and number of epochs with every change to get the optimal –but not overpowered performance (hence my validation accuracy of exactly 93.0%! ☺). With regards to the optimizer used – I used the Adam Optimizer as I read very good things about it. As first I was playing with exponential decay for the learning rate, however once I dove a little bit under the hood of how the Adam Optimizer works I learned that it takes care of this, along with momentum – built it! And it does it rather efficiently, hence I stuck with it.

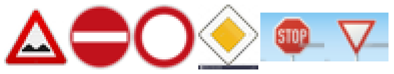
The randomization technique was applied to improve performance and reduce the chance of overfitting. Similarly other techniques were performed, some in different orders until I got the best performance for the criteria that I could – then once again adjusting the learning rate, epochs, batch size, ect.

In terms of the convolutional network – conv networks are among the best when it comes to image processing techniques that involve classification, hence I took the LeNet and unchanged it for the most part, other than playing with or without max pooling or applying drop-out in strategic locations. Once I have a bit more time I will craft a network different than the LeNet, perhaps even surpassing its performance.

### Test a Model on New Images

#### 1. *Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.*

The following are the traffic signs I found from the web – strategically choosing the most obvious looking signs for the network in order to test it on those first. I think what might be the most confusing for the network is the watermark associated with some of the images like the stop sign.. as the training data did not contain the watermark! Additionally, while I tried to choose images that were as close as possible, it might not be the most obvious for the network as some of the images were a little bit cut off (like the no vehicle sign which ended up performing the worst).



#### 2. *Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).*

To my surprise, my accuracy was not as high as I have expected, however it is not a bad start (I expected at least 93%, but only got **83%**). I think if I had time the next step would be to find images without watermarks. Additionally, perhaps going back to my code and increasing the amount of feature maps, reducing the batch size and learning rate a little bit along with increasing the number of epochs would allow my network to overfit the training set a little bit less, and be better equipped for new images *(as taken from my submission review).*

Overall, the predictions were 83% accuracy, with the following distribution of Accuracy for the images:

Image 1: Accuracy 1.000

Image 2: Accuracy 1.000

Image 3: Accuracy 0.667

Image 4: Accuracy 0.750

Image 5: Accuracy 0.800

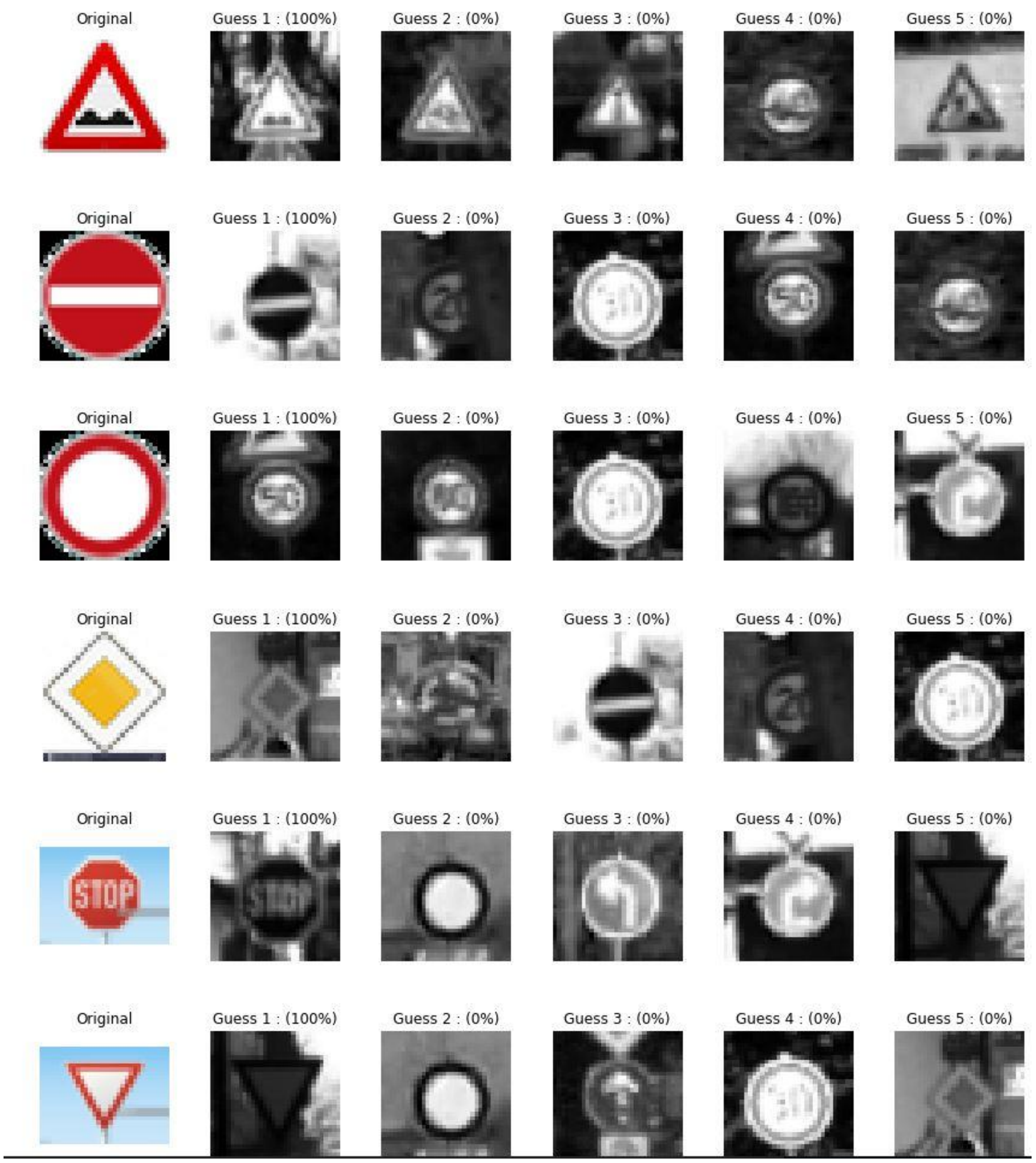
Image 6: Accuracy 0.833

While my test set accuracy was 0.938 (Train Accuracy 0.999, Validation Accuracy 0.930).

#### 3. *Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)*

The model was able to correctly guess all of my 6 traffic signs with an accuracy of 100% (6 out of 6). This guess accuracy of 100% is relatively close to the accuracy of my model on this test set (German traffic signs from web).

A plot of my outputs can be provided on the next page which neatly summarizes the answers to the question:



Final Notes: Overall, Throughout this project I learned that deep learning truly lives up to its name in more ways than one – noticing patterns and forming abstractions while being able to apply them at a later time to ever changing situations is remarkably similar to deep learning in humans.