Writeup / README :

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

The code for this is contained in the Step 1 of this Notebook

2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

The code for this is contained in the Step 2 of this notebook

Design and Test a Model Architecture

1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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.

In the step 4 of this notebook preprocessing of the images is done.

Here the images are normalized. This increases the accuracy by 2 percent and the algorithm reaches its peak accuracy faster than doing without normalization.

I tried preprocessing the images to gray scale. But it did not seem to increase the accuracy at all. So I ignored Greyscaling the image.

Further the training images were shuffled in order to prevent any preordering within them.

2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, identify where in your code, and provide example images of the additional data)

Since the number of Images for some of the class IDs are less, I decided to generate new images by rotating them slightly. The code for this can be found under step 3. The dict plot after generating the images show the increase in the number of images in the classes where there were less images to train.

After generating new images the number of training images were 40637.

Since the images I used came in three groups as test, train, valid I did not do any further split the traing set.

If I had to split I would have used test_train_split from sklearn.model_selection Library

3. Describe, and identify where in your code, 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.

The code for this section can be found in Step 4 of this notebook. I modified the LeNet architecture so that it can observe more feature data. The strucure of my network is as follows, The input dimention is 32x32x3 (RGB Image)

Layer 1 - Convolution:

Input shape = (32,32,3)
Strides = 1
Padding = SAME
Window size for convolution stride = (4, 4, 3, 32)
Activation = Relu (Rectifier)
Dropout = 0.9
Max pool with stride (2)

Layer 2 - Convolution:

Input shape = (16, 16, 32)
Strides = 1
Padding = SAME
Window size for convolution stride = (4, 4, 32, 64)

Activation = Relu (Rectifier)
Dropout = 0.8
Max pool with stride (2)

Layer 3 - Convolution:

Input shape = (8, 8, 64)
Strides = 1
Padding = SAME
Window size for convolution stride = (4, 4, 64, 128)
Activation = Relu (Rectifier)
Dropout = 0.7
Max pool with stride (2)

Layer 4 - Flat and fully connected:

Input shape = (4, 4, 128)
Activation = Relu (Rectifier)
Dropout = 0.5
Number of weights in output = 1024

Layer 5 - Flat and fully connected:

Input shape = (1024)
Activation = Relu (Rectifier)
Dropout = 0.5
Number of weights in output = 256

Layer 6 - Flat and fully connected:

Input shape = (256)
Activation = Relu (Rectifier)
Number of weights in output = 43 = Number of traffic sign classes

4. Describe how, and identify where in your code, 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.

The code for this section is found in Step 5 of this notebook. The type of Optimizer used is Adam Optimiser. This is because of the following reason.

"The tf.train.AdamOptimizer uses Kingma and Ba's Adam algorithm to control the learning rate. Adam offers several advantages over the simple tf.train.GradientDescentOptimizer. Foremost is that it uses moving averages of the parameters (momentum); Bengio discusses the reasons for why this is beneficial in Section 3.1.1 of this paper. Simply put, this enables Adam to use a larger effective step size, and the algorithm will converge to this step size without fine tuning.

The main down side of the algorithm is that Adam requires more computation to be performed for each parameter in each training step (to maintain the moving averages and variance, and calculate the scaled gradient); and more state to be retained for each parameter (approximately tripling the size of the model to store the average and variance for each parameter). A simple tf.train.GradientDescentOptimizer could equally be used in your MLP, but would require more hyperparameter tuning before it would converge as quickly." - Source http://stats.stackexchange.com/questions/184448/difference-between-gradientdescentoptimizer-and-adamoptimizer-tensorflow

Batch Size = 128
number of Epochs = 1000
Learning rate = 0.0005
mu = 0
sigma = 0.1

5. Describe the approach taken for finding a solution. 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 code for calculating the accuracy of the model is located in Step 5 of the Ipython notebook.

training set accuracy = 99%
validation set accuracy = 96%
test set accuracy = 86%

I used an already known implementation of conv network (LeNet) which is originally used for classifying letters and numbers in US postal system and extended its layers by adding an additional convolution layer and a fully connected layer so that it can store more features which can be used for identifying the traffic signs.

1.What was the first architecture that was tried and why was it chosen?

The first Architecture that I chose for a LeNet convolution network. I chose this because it is already proven to detect numbers from texts and currectly classify them. I got an accuracy about 91% with this model.

2. What were some problems with the initial architecture?

I could get the validation accuracy upto 91 % but not more than that. So I decided to add more feature detectors to the network.

3. How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to over fitting or under fitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

The Architechture was adjusted by adding a third convolution layer. The depth was in increased to 32 in the first convolution layer. In the second convolution layer, I changed the depth to 64. The third has a depth of 128. This layer is then flattened to 2048.

Initially with the LeNet architecture I found out that the Training accuracy reached 99.5% within the first 20 epocs. I considered it to be overfitting and hence I added more dropout Layers to improve the generalization of the model and it worked. It improved the validation accuracy of my model to more that 95%.

I tried with preprocessing the images to Grey scale but it made the accuracy worser and hence I removed this preprocessing step.

Normalization of the Matrix values increased the accuracy by more than 2%.

The final plot of the Validation and training can be found under the section Step 6 of the Ipython notebook.

4. Which parameters were tuned? How were they adjusted and why?

The leaning rate was tuned many times. I initially tried with 0.1 but the accuracy did not seem to go above 90 %. I then tried 0.0001 but it made the learning process very slow and I did not get much more accuracy with this rate too. Finally I tried 0.0005 and it seemed to give the best possible accuary.

Similary I also tried with the dropout percentage and ended up having 0.9, 0.8, 0.7 for the conv layers and 0.5 for the fully connected layers.

I also played around with the value of sigma and ended with with 0.1

5. What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

According to this paper, https://arxiv.org/pdf/1506.02158v6.pdf adding more dropout in the fully connected layers is better than having more dropouts in the convolution layers. This is the reson why is chose the keep probability to be 0.9, 0.8, 0.7 for conv layers and 0.5 for the fully connected layer.

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 code for this can be found in step 7 of this notebook. Five traffic signs from Germany have been chosen and they are used for testing the training model.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

The prediction and accuracy can be found in Step 7 and Step 8 of this notebook. The trained model seems to perform well in real life and I get constantly >80% accuracy with the new images.

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 and identify where in your code softmax probabilities were outputted. 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 code can be found in step 9