**Traffic Sign Recognition**

**Writeup**

**You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.**

**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

**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.**

I used the python/numpy to calculate summary statistics of the traffic signs data set and for the classes/ label I have written function called counter\_classes where I used the following line **for i in range(43):** at start of the project I used the value of 50 at place of 43, once I found the no of images after 42 is zero and moreover I already know the no of classes and later to make it easy and fast I hardcoded value.

n\_train = len(X\_train)

n\_validation = len(X\_valid)

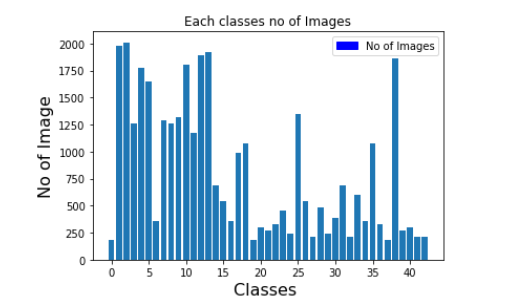
n\_test = len(X\_test)

X\_train[0].shape

* The size of training set is 34799
* The size of the validation set is 4410
* The size of 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

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

Here is an exploratory visualization of the data set. It is a bar chart showing how the data ...



**CHART A**

**Design and Test a Model Architecture**

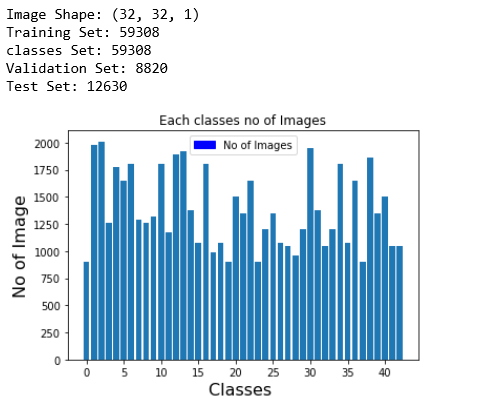
**1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As 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, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)**

As a first step I decided to get more data to equalize all classes, because the gradients of dominating classes will have a strong pull towards it, as we can see from chart A above there are some classes of images which are almost equal or greater than 2000 and some classes less than 300.

I have used cropping, padding, zoom in, zoom out to equalize classes.

As a next step, I decided to convert the images to grayscale because the for the classifier it will be easier to learn (CNN Lecture Color topic)

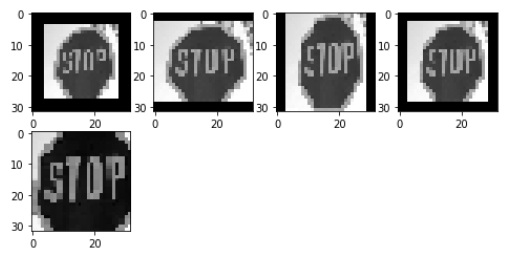
As a last step, I normalized the image data because the values involved in loss function should never be too big or too small and keep zero mean and equal variance (tensorflow lecture Normalized inputs and Initial weights topic)



**CHART B**



The Output shown below is after getting mode data and applying grayscale and normilization.

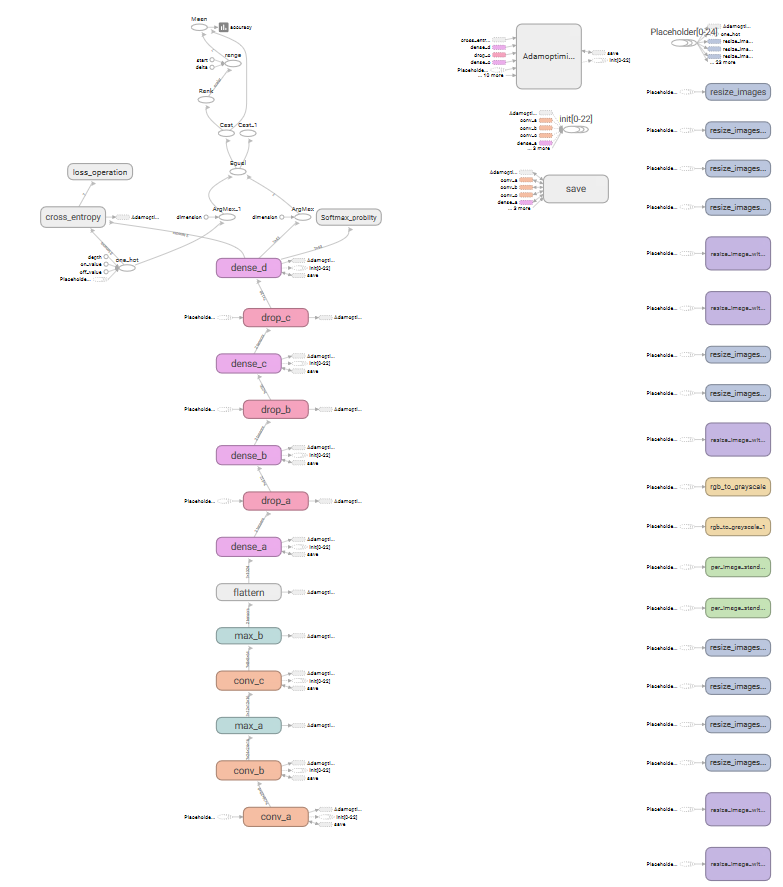


**Preprocessing images**

**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.**

My final model consisted of the following layers:

|  |  |  |  |
| --- | --- | --- | --- |
| **Layers** | **Input Shape** | **Output Shape** | **Description** |
| convolution 5x5 | 32x32x1 | 28x28x8 | Valid, 1x1 stride |
| RELU |  |  |  |
| convolution 5x5 | 28x28x8 | 24x24x16 | Valid, 1x1 stride |
| RELU |  |  |  |
| Max pooling | 24x24x16 | 12x12x16 | Valid, 2x2 stride |
| convolution 5x5 | 12x12x16 | 8x8x32 | Valid, 1x1 stride |
| RELU |  |  |  |
| Max pooling | 8x8x32 | 4x4x32 | Valid, 2x2 stride |
| Flattern | 4x4x32 | 1024 |  |
| Dense | 1024 | 512 |  |
| RELU |  |  |  |
| Dropout |  |  | 0.4 for Traning/ 1.0 for Validation |
| Dense | 512 | 256 |  |
| RELU |  |  |  |
| Dropout |  |  | 0.4 for Traning/ 1.0 for Validation |
| Dense | 256 | 128 |  |
| RELU |  |  |  |
| Dropout |  |  | 0.4 for Traning/ 1.0 for Validation |
| Dense | 128 | 43 |  |
| Softmax |  |  | Used softmax\_cross\_entropy |



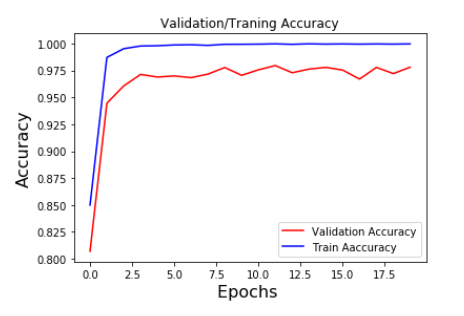
**3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.**

To train the model, I used a learning rate of 0.001, batch size 128, epochs 20, and Adam Optimizer dropout of 0.4 for traning and 1.0 for validation.

**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.**

I started with (LNet as base to start) two convolution layer, single maxpooling, and two dense layer, as I saw accuracy on traning and validation (calculated in cell [15] and output will be observed in cell [16] while traning) for the first time observed the overfitting and nearly 80% accuracy. At this point I added the dropout layers, as still results were not promising, I decided to make layer more depth (Trail error method going through lectures again and again depending on problem facing adjusting padding, depth learning rate, mean mu etc. to come up with final design). The most important part was dropout for my overfitting. Getting more data for unbalanced classes was helpful getting this accuracy

* training set accuracy of 1.0
* validation set accuracy of 97.8
* test set accuracy of 97.2



**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.**

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



I was expecting all the six images at least of 0.85 accuracy.

**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).**

Here are the results of the prediction:

| **Image** | **Prediction** |
| --- | --- |
| Stop Sign | Stop sign |
| No Entry | No Entry |
| Yield | Yield |
| 70 km/h | 70 km/h |
| Slippery Road | Slippery Road |

Priority Priority

The model was able to correctly guess 6 of the 6 traffic signs, which gives an accuracy of 100%.

**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.**

The code for making predictions on my final model is located in the 15 cell (softmax\_problity and top\_k functions) and 19th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a No Entry (probability of 1.0), and the image does contain a No entry the top five soft max probabilities were.

| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | No entry |
| 9.47919336e-16 | Go straight or left |
| 8.95115302e-18 | Priority Road |
| 2.87246471e-18 | No Passing |
| 1.72627445e-18 | Slippery Road |

For the second image the model is relatively sure that this is a Priority Road (probability of 1.0), and the image does contain Priority Road the top five soft max probabilities were

| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | Priority Road | |
| 2.32945466e-23 | Roundabout mandatory | |
| 7.20869861e-30 | 50km/hr | |
| 4.77663314e-30 | 100km/hr | |
| 3.61934485e-30 | No Vehicle | |

For the Third image the model is relatively sure that this is a No Entry (probability of 1.0), and the image does contain Slippery Road the top five soft max probabilities were

| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | Slippery Road | |
| 6.53336166e-17 | Right-of-way at the next intersection | |
| 2.60535274e-17 | Wild animals crossing | |
| 6.48634918e-18 | No passing for vehicles over 3.5 metric tons | |
| 1.99711220e-18 | No passing | |

For the Fourth image the model is relatively sure that this is a 70 km/hr (probability of 1.0), and the image does contain a 70 km/hr the top five soft max probabilities were

| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | 70 km/hr | |
| 3.55372043e-09 | 80 km/hr | |
| 2.41422748e-09 | 20km/hr | |
| 1.35169120e-09 | 30km/hr | |
| 1.08458395e-10 | 120 km/hr | |

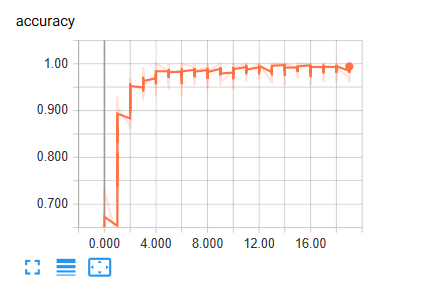
For the Fifth image the model is relatively sure that this is a Stop (probability of 1.0), and the image does contain Stop the top five soft max probabilities were

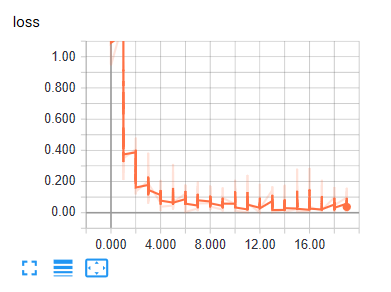
| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | Stop | |
| 4.63841727e-17 | No Vehicles | |
| 1.56711567e-17 | 60 km/hr | |
| 2.24779611e-18 | 80 km/hr | |
| 1.57740096e-18 | 50 Km/hr | |

For the sixth image the model is relatively sure that this is a Yield (probability of 1.0), and the image does contain Yield the top five soft max probabilities were

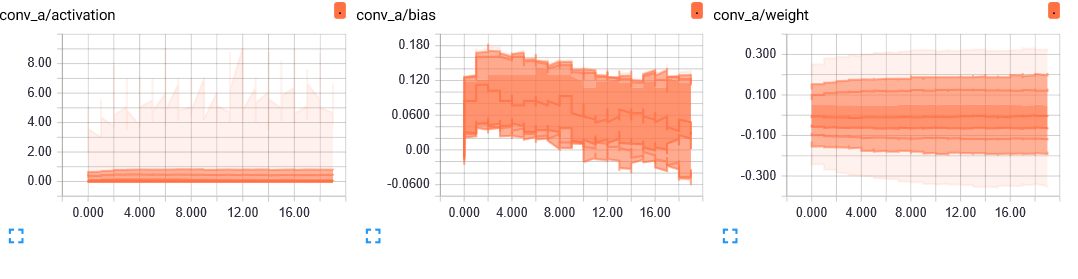
| **Probability** | **Prediction** |
| --- | --- |
| 1.00000000e+00 | Yield | |
| 8.12878022e-27 | Priority Road | |
| 2.33300721e-27 | No Vehicles | |
| 1.74369268e-27 | Keep left | |
| 1.43084134e-27 | Children crossing | |

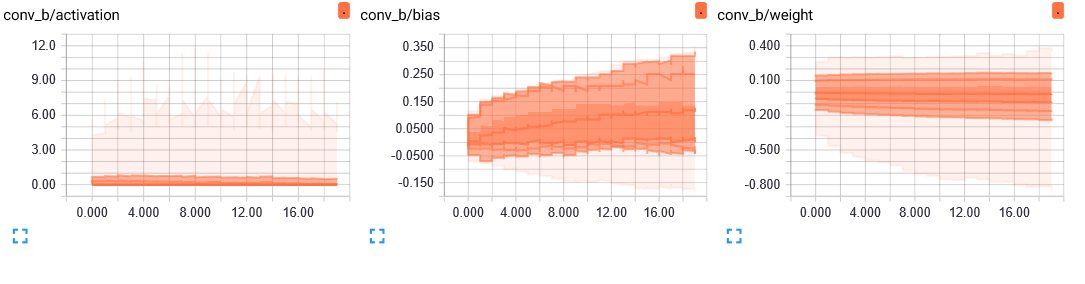
**Tensorboard Traning O/P:**

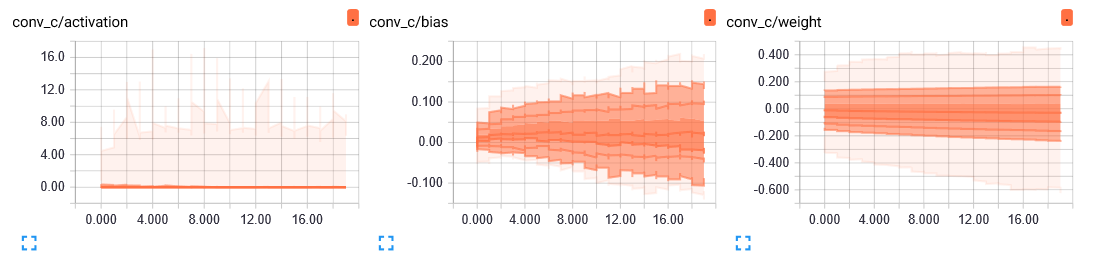


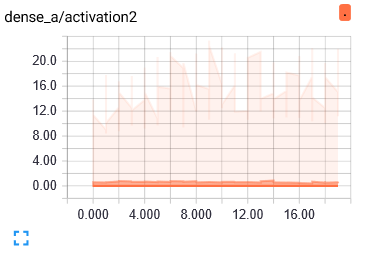


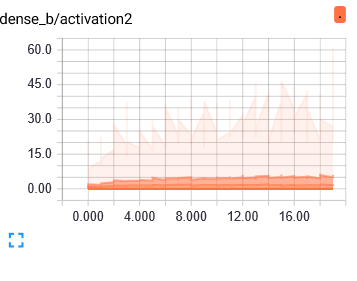
**Distribution:**

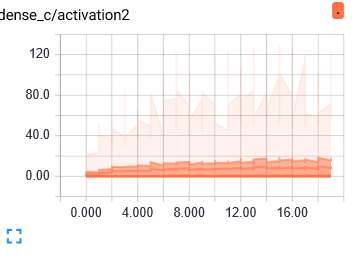


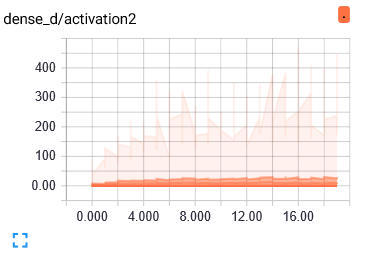












**Histrigroms:**

