### Question 1

Describe how you preprocessed the data. Why did you choose that technique?

**Answer:**

* **Pre-processing Technique**: Normalized image using **adaptive histogram equalization**:
* **Reason**
  + The pixel values would be normalized in the range of 0-1 which would help with gradient decent.
  + Histogram Equalization would reduce the impact of varying contrast and illumination in images and thus make the model more robust to the same.

### Question 2

Describe how you set up the training, validation and testing data for your model.***Optional***: If you generated additional data, how did you generate the data? Why did you generate the data? What are the differences in the new dataset (with generated data) from the original dataset?

**Answer:**  
The provided testing data is kept untouched, and the final model related matrix would be reported using this unseen data.

The provided training data had been divided as follow:

* **Training Data:** 70% of original training data had been used for training.
* **Validation Data:** 21% of original training data. This would be used for checking model performance while training.
* **Test Data:** (different from provided test data) This would be used to fine tune the model, and check how the model performs on unseen data before running the model on provided test data set.

**Additional Data**: The training data defined above was augmented to obtain 50 samples per class. The additional data was generated to increase samples for training, which in turn would make training more robust. The newly generated dataset includes normalized original data, and newly generated data (by randomly introducing rotation, x-translation, y-translation, sheer factor and zoom factor to the original data). The new dataset is balanced and had already been pre-processed for normalizing contrast and illumination.

### Question 3

What does your final architecture look like? (Type of model, layers, sizes, connectivity, etc.) For reference on how to build a deep neural network using TensorFlow, see[*Deep Neural Network in TensorFlow*](https://classroom.udacity.com/nanodegrees/nd013/parts/fbf77062-5703-404e-b60c-95b78b2f3f9e/modules/6df7ae49-c61c-4bb2-a23e-6527e69209ec/lessons/b516a270-8600-4f93-a0a3-20dfeabe5da6/concepts/83a3a2a2-a9bd-4b7b-95b0-eb924ab14432)from the classroom.

**Answer:**

Modified LeNet Architecture  
Input (32x32x3) -> Convolutional 5x5x6 -> ReLu -> MaxPool (stride of 2) -> Convolutional 5x5x16 -> ReLu -> MaxPool (stride of 2) -> Flatten (400) -> Fully Connected (120 units) -> Fully Connected (84 units) -> Fully Connected (43 units, output)

The above code block have been commented and shows how the shape of the data varies across the layers.

### Question 4

How did you train your model? (Type of optimizer, batch size, epochs, hyperparameters, etc.)

**Answer:**

* AdamOptimizer for managing learning rate
* Batch Size: 128
* Epochs: 30
* Variables initialized with normal distribution (mean=0, std dev=0.1)
* Biases initialized with zeros

### Question 5

What approach did you take in coming up with a solution to this problem? It may have been a process of trial and error, 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 this is suitable for the current problem.

**Answer:**

1. Preproceesed data by applying adaptive histogram equalization to make training immune to varying contrast and illumination.
2. Augmented training data by randomly introducing rotation, horizontal and vertical shift, and scaling.
3. Trained with LeNet Architecture as the problem description is similar to image classification problem on which LeNet was applied. The model had been tweaked to handle RGB channel and 43 output classes. Dropouts were not introduced as the model did not overfit.
4. Decided batch size and epoch by using ***generated*** validation and test set

### Question 6

Choose five candidate images of traffic signs and provide them in the report. Are there any particular qualities of the image(s) that might make classification difficult? It could be helpful to plot the images in the notebook.

**Answer:**  
The images were taken from [this PDF](http://www.adcidl.com/pdf/Germany-Road-Traffic-Signs.pdf). As the images are of Germany Traffic Sign itself, it should not be difficult for model to predict.

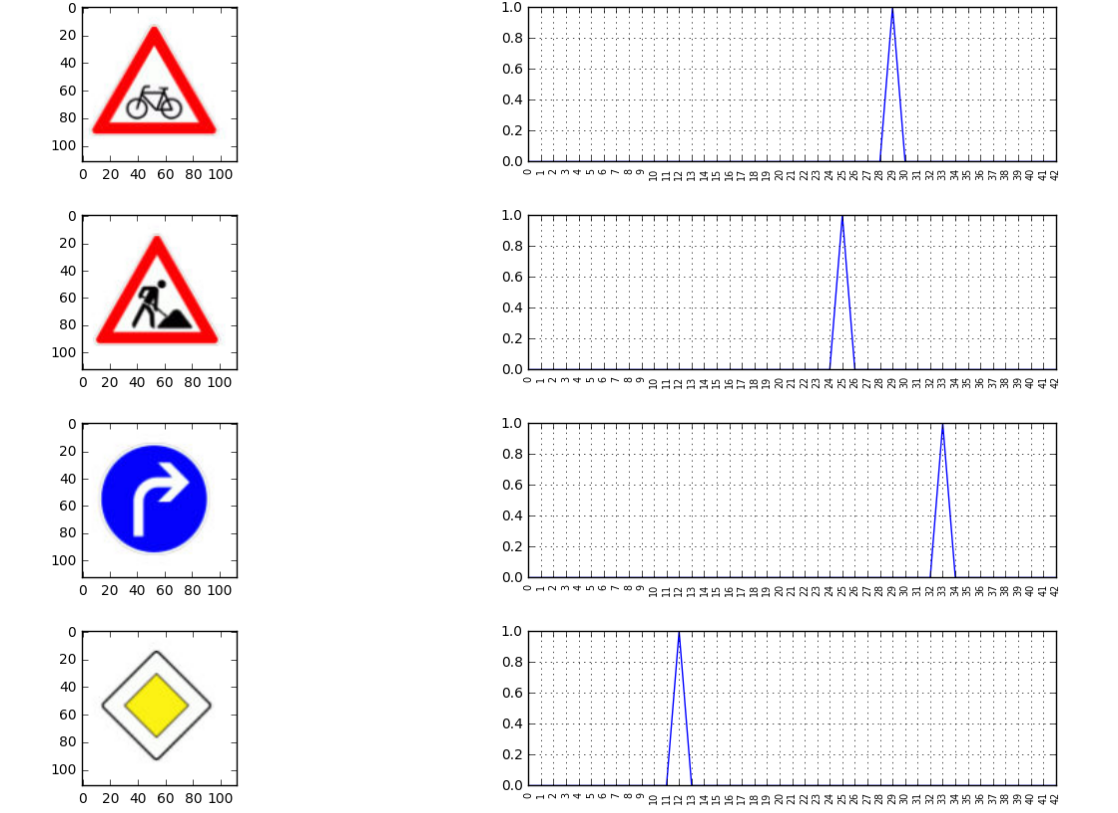
### Question 7

Is your model able to perform equally well on captured pictures when compared to testing on the dataset? The simplest way to do this check the accuracy of the predictions. For example, if the model predicted 1 out of 5 signs correctly, it's 20% accurate.

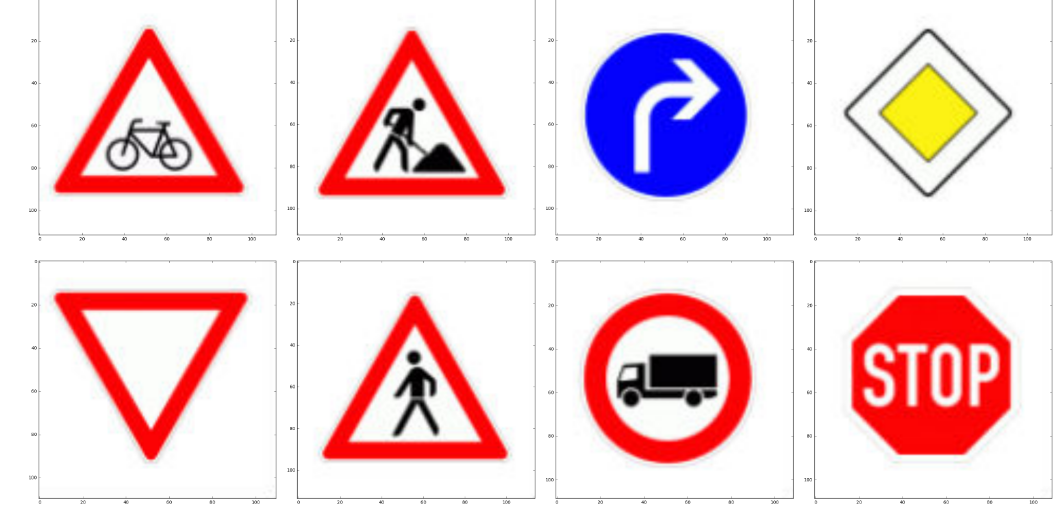
***NOTE:***You could check the accuracy manually by usingTraffic*signnames.csv*(same directory). This file has a mapping from the class id (0-42) to the corresponding sign name. So, you could take the class id the model outputs, lookup the name inTraffic*signnames.csv* and see if it matches the sign from the image.

**Answer:**  
The model is able to predict correctly on newly provided images with accuracy of 100%.

**Softmax probability output as below –**



### Q 7 ) Test a Model on New Images , output as below

****

