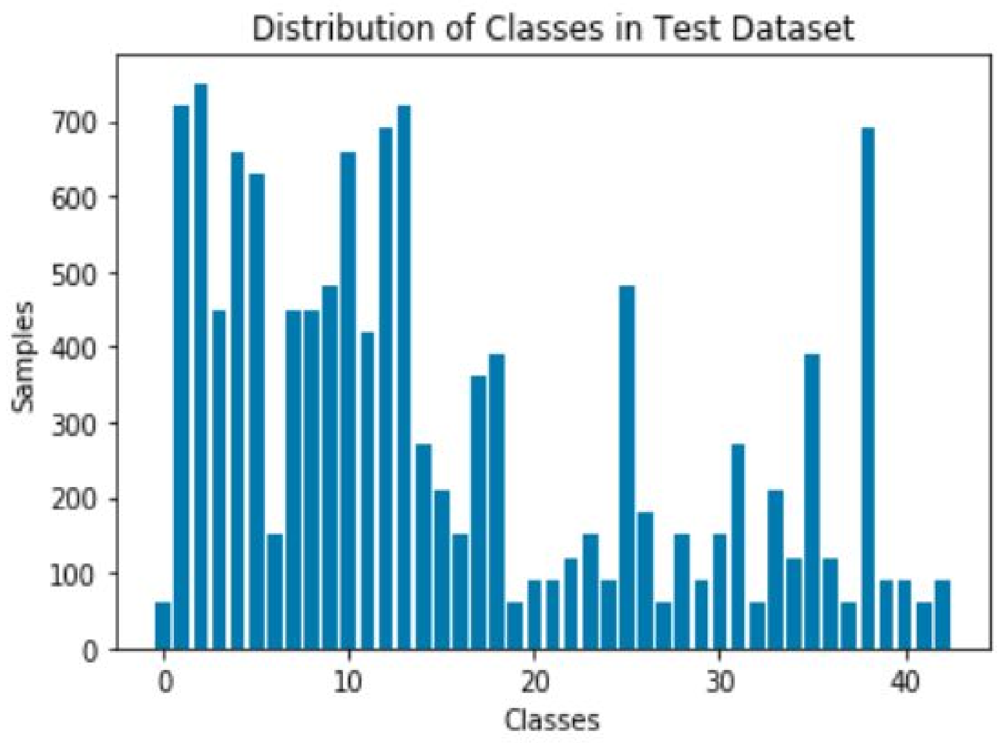
Traffic Sign Classifier : Project 3

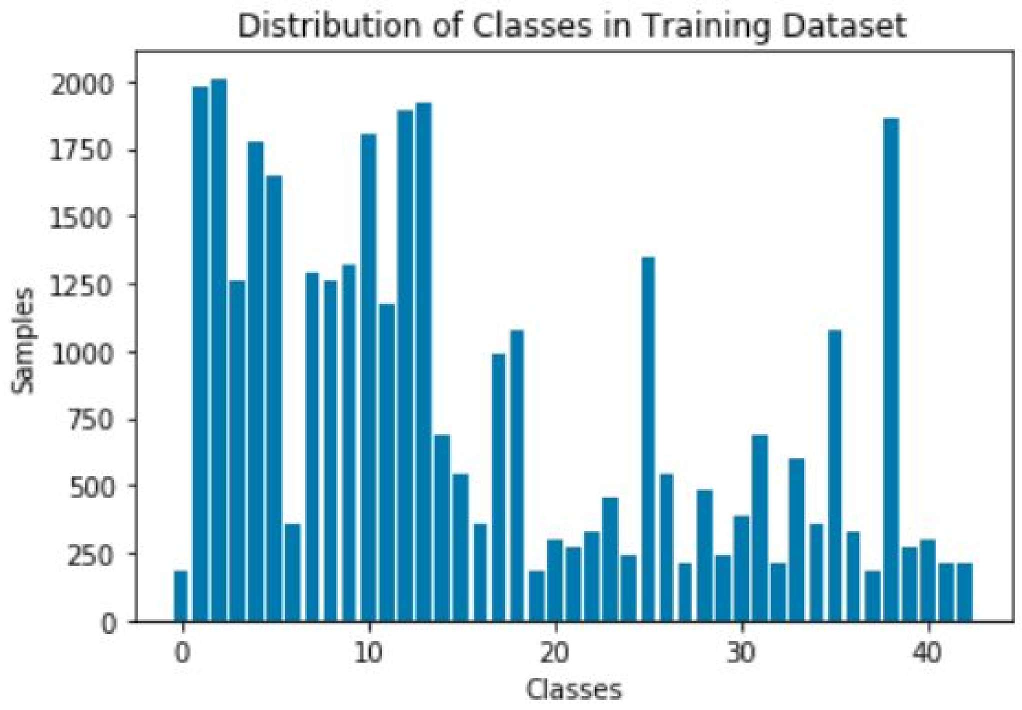
The project aims at classifying German Traffic Signs using Lenet Architecture and aims at an accuracy of atleast 93 percent.

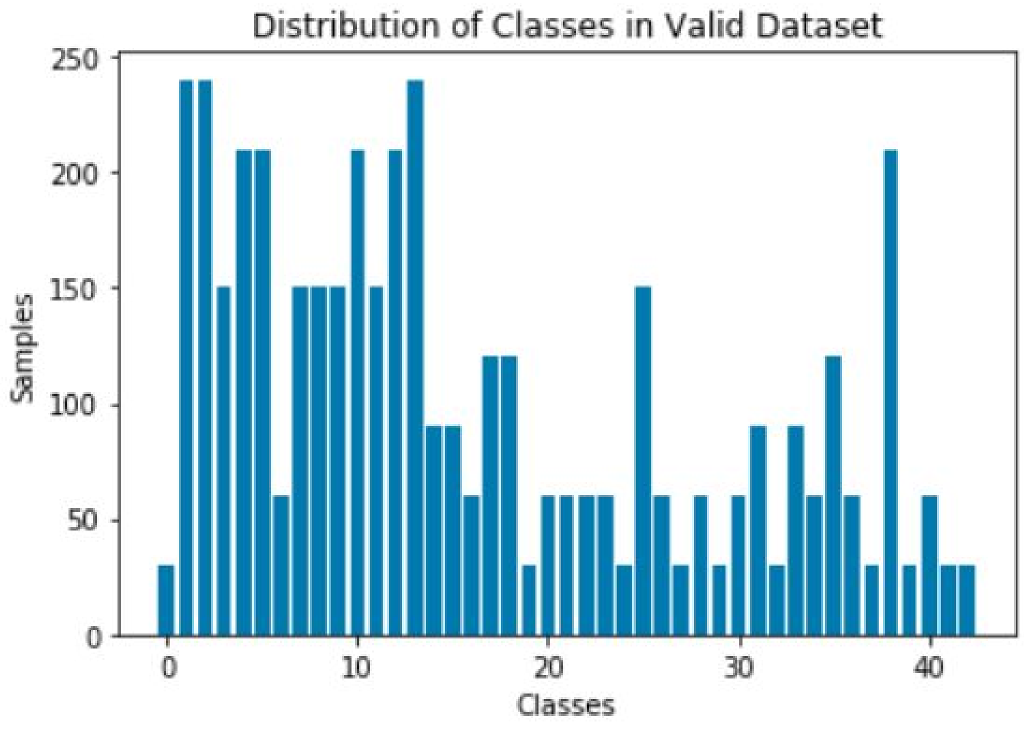
Dataset summary :

The dataset contains,
Number of training examples = 34799
Number of testing examples = 12630
Number of Validation examples= 4410
Image data shape = (32, 32, 3)
Number of classes = 43

Distribution of Classes in the training, Validation and Test Set :







Dataset Visualization :

Preprocessing :

The preprocessing consists of converting the coloured images into grayscale and normalizing the image in the range from [-1,1].

Image = (Image-128)/128

The reason for normalising the images between [-1,1] is when we process the model and the backpropagate the gradients all the features should have similar value, so that gradients do not flow out of control.

Model Architecture:

The model consists of 3 Convolutional layers followed by relu activation and the last conv layer followed by (max-pooling+relu).

This is followed by 4 fully connected layers of shape (256,120) → (120,100) → (100,84) → (84,43). → followed by softmax layer.

The model architecture is described in the table below:

LAYER	DESCRIPTION
Input	Grayscale image of size 32x32x1
Layer 1: Convolutional 5x5	1x1 Stride, padding= ‘valid ’ **Output = 28x28x6**
RELU	Activation Layer
Layer 2: Convolutional 5x5	1x1 Stride, padding= ‘valid ’ **Output = 14x14x10**
RELU	Activation Layer
Layer 3: Convolutional 5x5	1x1 Stride, padding= ‘valid ’ **Output = 8x8x16**
RELU	Activation Layer
Max Pooling	**Output = 4x4x16.**
Fully Connected 1	Input nodes=256, **output nodes = 120**
Fully Connected 2	Input nodes = 120, **output nodes = 100**
Fully Connected 3	Input nodes=100, **output nodes = 84**
Softmax	Input nodes = 84, **output nodes = 43**

Model Training:

I first tried to trained my images on RGB images by just normalising the images between 0-1 but was not expecting good results, thus converted into grayscale image. A little mistake I made when my training accuracy was not improving was using model in training mode while evaluating.
The number of convolutional layers were decided on model (LeNet) taught in the lecture and subsequently modified based on the results. Changing the batch size greatly affected the accuracy.. Also, adding dropout further reduced the accuracy. The model performed best when using the following parameters:

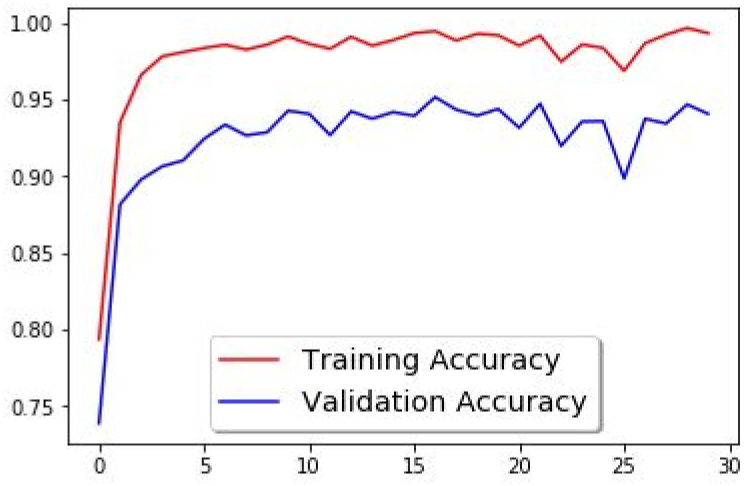
Optimizer: Adam Optimizer
Training Epochs : 30
Batch Size : 512
Learning Rate: 0.009
Loss: Cross_Entropy

The submission describes the approach to finding a solution. Accuracy on the validation set is 0.93 or greater.

Training Accuracy : 99.4

Valid Accuracy : 0.934

Test Accuracy: 0.901



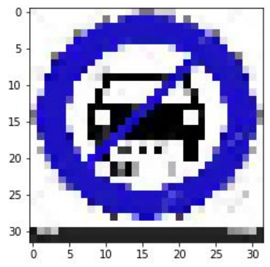
I started with the one model already taught in the class keep the optimizer and the loss same. Images were normalised and converted into grayscale for preprocessing. I tried tweaking with learning rate and epochs which helped me understand the model. Also I visualised some of the filters weight using the Function (outputFeatureMap) in the optional exercise. Adding dropout greatly reduced my model accuracy. I started with a small learning rate and finally the results were perfect at 0.009. Since the images are of very small size the model took only 30 epochs to train and reach a validation accuracy of 93.4%.

Acquiring New Images

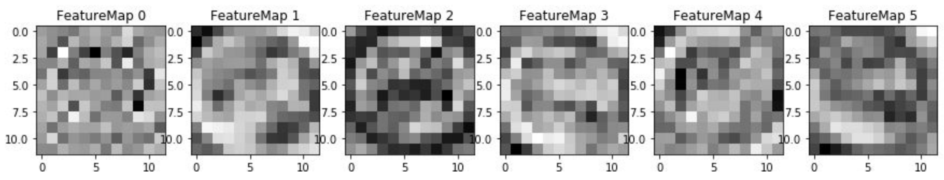
The images for testing the model were downloaded from the web and then preprocessed in the same way as the training images were processed. The image resolution of most of the images was different and it will affect the prediction.

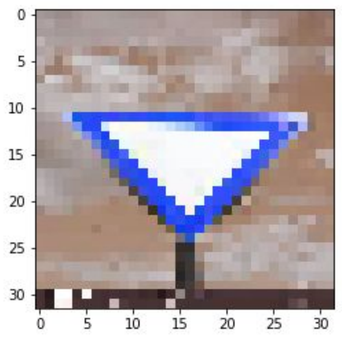
My model performs poorly on the New Images. It is able to detect only the yield class, giving an accuracy of 20%. This is mainly because the images are not from the same set of Database, or the model is not able to generalize well.

When visualizing the feature my model mainly learns the circular features in most of the signs and results in incorrect prediction, but in case of sign1 since its inverted triangle, it is able to detect it perfectly.

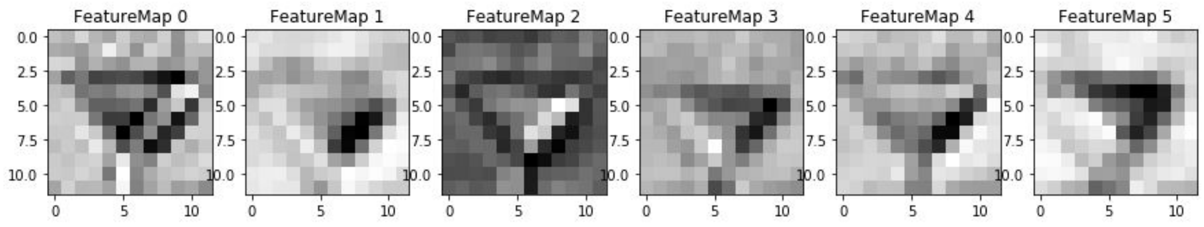


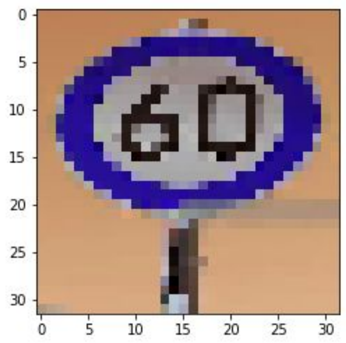
```
INFO:tensorflow:Restoring parameters from ./trafficTest
```



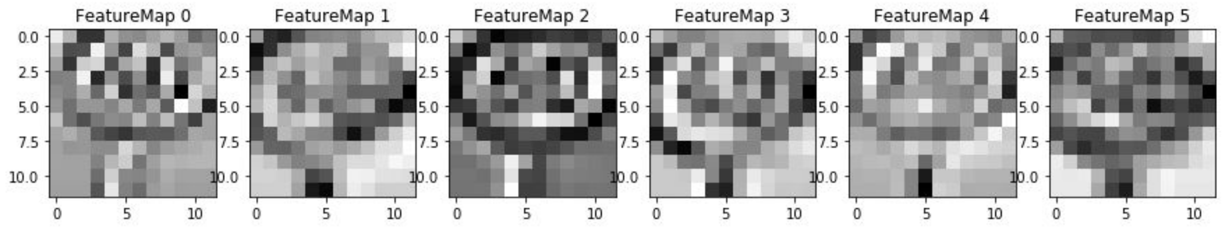


```
INFO:tensorflow:Restoring parameters from ./trafficTest
```



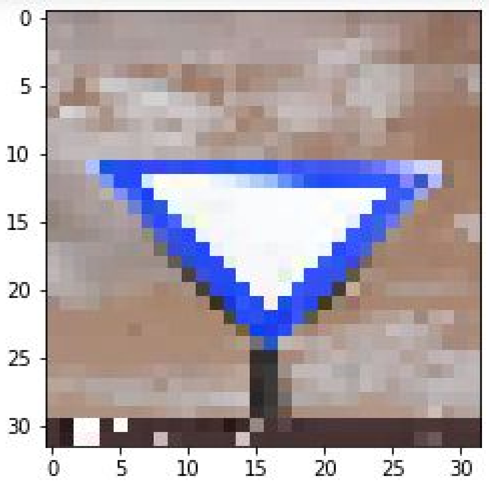


```
INFO:tensorflow:Restoring parameters from ./trafficTest
```



Model Certainty -Softmax Prob:

For some of the class the model is able to predict the class in top 5 softmax probability, but not in most of the case. It is perfectly predicting yield class. (Other predictions are shown in the HTML file).



```
INFO:tensorflow:Restoring parameters from ./trafficTest
Yield with probability score 1.0
Speed limit (20km/h) with probability score 0.0
Speed limit (30km/h) with probability score 0.0
Speed limit (50km/h) with probability score 0.0
Speed limit (60km/h) with probability score 0.0
```