**Traffic Sign Classification**

The traffic sign recognition system inside the vehicle plays an important role and could guarantee the safety of human life on the road since it feedbacks road information to the driver in time. The principle of driving assistance systems aiming at road signs recognition is to detect signs, interpret their meaning, then transmit the information to the driver. The convolutional neural network (CNN) can be widely used in traffic sign recognition.

**1.1 Data set Used:**

The data set used in the project is German Traffic Signs data set. Within the data set, there are 43 types of signs in the data. Each image in the dataset is 32 pixels x 32 pixels x 3 Channels (one each for RGB). The data set consists of a total of 34799 training examples, 4410 validation examples, 12630 testing examples.

* randomly adjust brightness of the image using:

*enhancer = ImageEnhance.Brightness(distorted\_image)*

*distorted\_image\_ = enhancer.enhance(random.random())*

* convert image to uint8 array using:

*distorted\_image\_ = np.array(distorted\_image\_, dtype=np.uint8)*

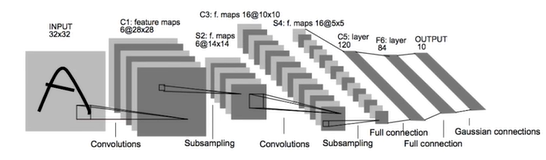
**1.2 Parameters:**

* Rate describes the learning rate of the neural network.
  + rate = 0.001
* Epochs tells how many times to run the training data through our network. More number of epochs the better the data will train, but it will take longer time.
  + EPOCHS = 40
* Batch size determines how may training images to run through the network at a time. Larger the batch size faster the model will train, but the processor may have memory limit on this.
  + BATCH\_SIZE = 4096
* Hyper parameters:

On experimenting got

* + Mu=0
  + Sigma=0.1

**1.3 Architecture:**



Implemented using Lenet architecture. This network take a 32x32 image input. That image goes through a convolutional layer C1, followed by a subsampling or pooling layer S2. There is another pair of convolutional layer and sub sampling layer. Finally, there are 3 fully connected layers including the output layer.

* First convolutional layer:

This layer has a 5x5 filter of input depth of 3 and output depth of 32. We use conv2D function along with the bias.

*conv1\_w = tf.Variable(tf.truncated\_normal(shape=(5,5,3,32), mean=mu, stddev=np.sqrt(2/(5\*5\*3))))*

Using the formula,

Output\_height=(input\_height-filter\_height+1)/stride

Here, (32-5+1)/1=28

So the convolutional layer output is 28x28x6

*conv1\_b = tf.Variable(tf.zeros(32))*

*conv1 = tf.nn.conv2d(x, conv1\_w, strides=[1,1,1,1], padding='VALID') + conv1\_b*

* Then activation is done using ReLu activation function.

*conv1 = tf.nn.relu(conv1)*

* Pooling:

Using 2x2 kernel and 2x2 stride. Which gives an output of 14x14x32.

*conv1 = tf.nn.max\_pool(conv1, ksize=[1,2,2,1],strides=[1,2,2,1], padding='VALID')*

* Second convolutional layer:

Output is 10x10x64

* Second pooling layer:

Output is 5x5x64

* Flatten:

We flatten the above output into a vector. The length of the vector is

5\*5\*64=1600.

*fc0 = flatten(conv2)*

* First Fully Connected Layer:

We pass this vector into a fully connected layer with a width of 120.

* Second Fully connected layer:

Output is of width 84

* Third fully connected layer:

Output is of width 43, which is equal to the number of classes. These ouputs are also known as logits.

**1.4 Training Pipeline:**

* Calculate the cross entropy:

By using tf.nn.softmax\_cross\_entropy\_with\_logits to compare the logits to the ground truth labels and calculate the cross entropy. Cross entropy is a measure of how different the logits are from the ground truth training labels.

* Calculate loss operation:

The tf.reduce\_mean function averages the cross entropy from all training images.

* Optimizer:

AdamOptimizer uses Adam algorithm to minimize the loss function similar to what stochastic gradient descent does. This is where we use the learning rate hyper parameter.

* Minimize:

We use minimize function on the optimizer which uses back propogation to update the network and minimize the training loss.

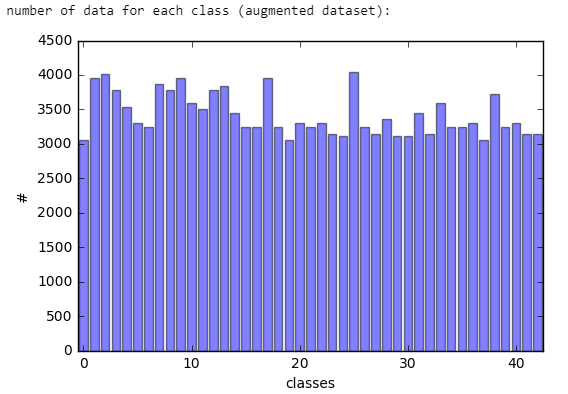
**1.5 Training the model:**

* Created a TensorFlow session and initialized the variables.
* Train over the number of epochs. Epochs are adjusted till the desired accuracy is achieved, without overfitting.
* At the beginning of epoch the training data is shuffled to ensure that the training isn’t biased based on the order of the images.
* Then break the training data into batches and train the model on each batch.
* At the end of each epoch, the model is evaluated using validation data.
* As we train the model, we see that the validation accuracy is really high and stays there, if the hyper parameters are good enough.

**1.6 Augmenting Dataset:**

Here, the training dataset is augmented using pickling technique.

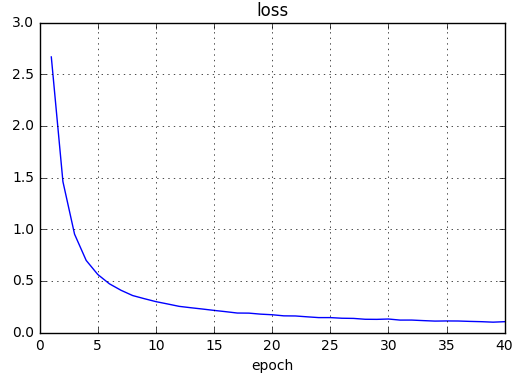
train = pickle.load(f)



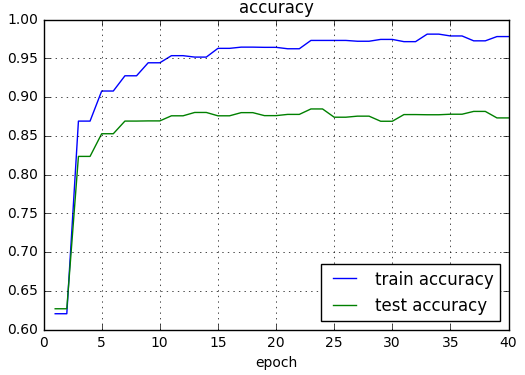
**1.7 Accuracy and Loss:**

Plotting the training and testing accuracy and loss vs epoch graphs.

Loss vs epoch:



Accuracy vs epoch:



**1.8 Results:**

Input:



Output:

Int64

28

* Class number is given as output. Here, class 28 means children crossing.
* Class numbers and their corresponding names are listed below in the table.

|  |  |
| --- | --- |
| **ClassID** | **SignName** |
| 0 | Speed limit (20km/h) |
| 1 | Speed limit (30km/h) |
| 2 | Speed limit (50km/h) |
| 3 | Speed limit (60km/h) |
| 4 | Speed limit (70km/h) |
| 5 | Speed limit (80km/h) |
| 6 | End of speed limit (80km/h) |
| 7 | Speed limit (100km/h) |
| 8 | Speed limit (120km/h) |
| 9 | No passing |
| 10 | No passing for vehicles over 3.5 metric tons |
| 11 | Right-of-way at the next intersection |
| 12 | Priority road |
| 13 | Yield |
| 14 | Stop |
| 15 | No vehicles |
| 16 | Vehicles over 3.5 metric tons prohibited |
| 17 | No entry |
| 18 | General caution |
| 19 | Dangerous curve to the left |
| 20 | Dangerous curve to the right |
| 21 | Double curve |
| 22 | Bumpy road |
| 23 | Slippery road |
| 24 | Road narrows on the right |
| 25 | Road work |
| 26 | Traffic signals |
| 27 | Pedestrians |
| 28 | Children crossing |
| 29 | Bicycles crossing |
| 30 | Beware of ice/snow |
| 31 | Wild animals crossing |
| 32 | End of all speed and passing limits |
| 33 | Turn right ahead |
| 34 | Turn left ahead |
| 35 | Ahead only |
| 36 | Go straight or right |
| 37 | Go straight or left |
| 38 | Keep right |
| 39 | Keep left |
| 40 | Roundabout mandatory |
| 41 | End of no passing |
| 42 | End of no passing by vehicles over 3.5 metric tons |