Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goal of this project is to build a neural network based learning model that is able to identify the traffic signs.

The project includes the following main steps:

- Load the data set for German Traffic Signs
- Visualize, analyze and understand the dataset
- Design, train and test a model based on LeNet-5 architecture
- Modify the LeNet-5 architecture to make the new model (LeNet-HI)
- Design, train and test the LeNet-HI model
- Use the LeNet-HI 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. Dataset Summary

Using python and numpy methods the following statistics of the traffic signs data set are extracted:

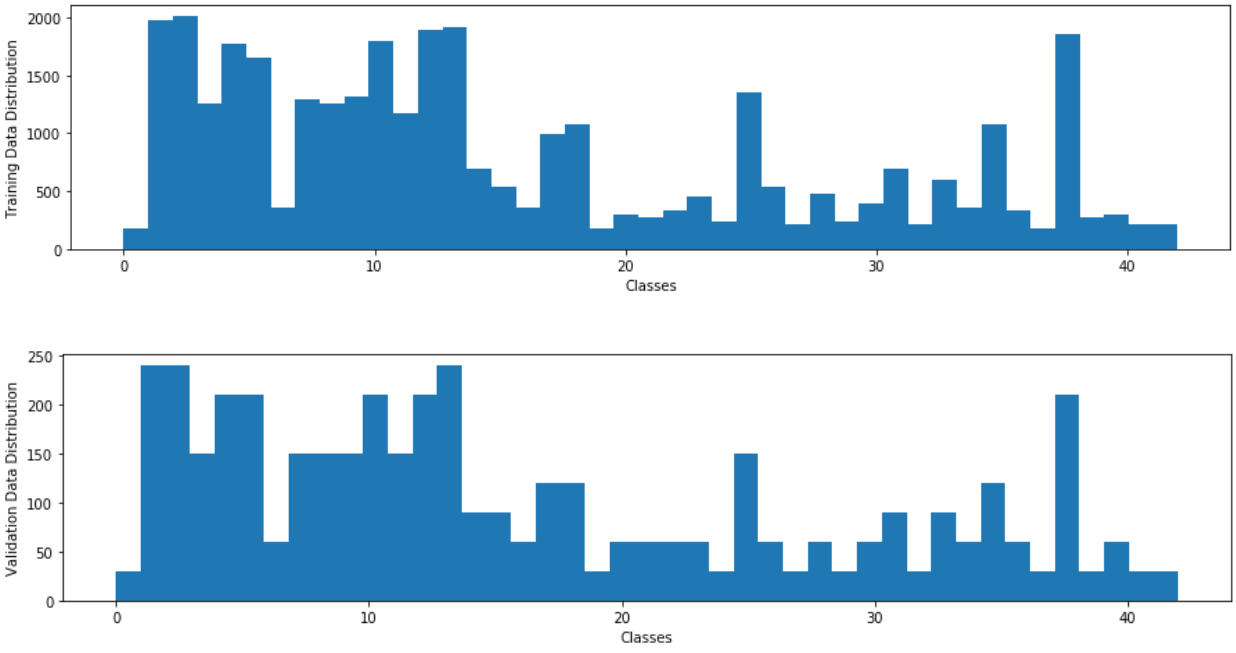
```
# TODO: Number of training examples
n_train = y_train.shape[0]

# TODO: Number of validation examples
n_valid = y_valid.shape[0]

# TODO: Number of testing examp# TODO: Number of training examples
n_train = y_train.shape[0]

# TODO: Number of validation examples
n_valid = y_valid.shape[0]

# TODO: Number of testing examples.
n_test = y_test.shape[0]

# TODO: What's the shape of an traffic sign image?
image_shape = X_test[0].shape

# TODO: How many unique classes/labels there are in the dataset.
classes = []
for x in y_valid:
    if x not in classes:
        classes.append(x)
n_classes = len(classes)

total = n_train + n_valid + n_test
print("Number of training examples = %5d(%2.2f"% (n_train,n_train/total*100),'%)')
print("Number of validation examples = %5d(%2.2f"% (n_valid,n_valid/total*100),'%)')
print("Number of testing examples = %5d(%2.2f"% (n_test,n_test/total*100),'%)')
print("Number of classes =", n_classes)
print("Image Shape: {}".format(X_train[0].shape))
```

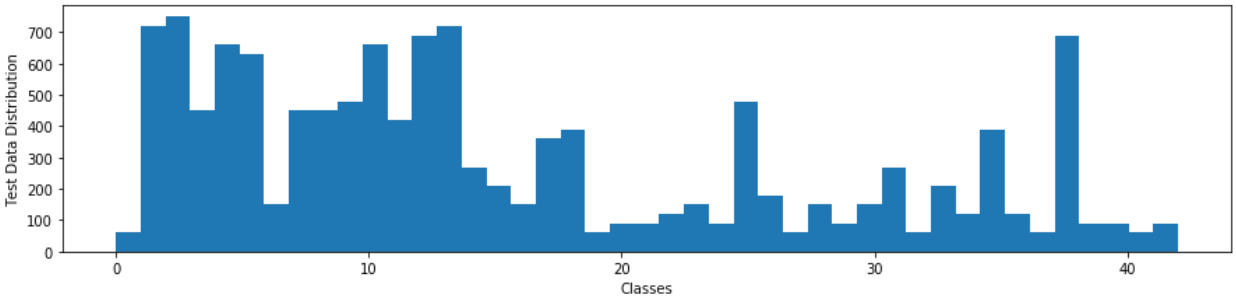
```
Number of training examples = 34799(67.13 %)
Number of validation examples =  4410(8.51 %)
Number of testing examples = 12630(24.36 %)
Number of classes = 43
Image Shape: (32, 32, 3)
```

2. Exploratory visualization of the dataset:

Here is an exploratory visualization of the data set. First randomly some of the pictures are shown:

Then a histogram chart shows how the data are distributed over the classes stretch in the training, validation and testing data sets:





The distribution shows less number of data for labels between 19 to 42. The verification images for the entire stretch will be used to make sure the model works fine for all the labels.

Design and Test a Model Architecture

1. Data Preparation:

As a first step, the images are converted to grayscale to reduce the number of features for the model and make it easier for the network to learn from patterns only (not colors). Here is an example of a traffic sign image before and after grayscaling:

As a last step, I normalized the image data using the formula (pixel-128)/128, to map the data to the range of [-1, 1] instead of [0, 255] with mean value of around zero. This helps training the model because the weights will not be dominated by features with higher range. Here are the summary of the mean values for the training dataset before and after the normalization:

```
mean(X)           mean(X_normal)
82.677589037       -0.354081335648
83.5564273756      -0.347215411128
82.1484603612      -0.358215153428
```

2. Model Architecture:

My final model consisted of the following layers, it's simply a modified version of the LeNet-5 model (with Dropout) which is explained in the next sextions:

Layer	Description
Input	32x32x1 Grayscale image
Convolution	5x5x1, 1 stride, Valid padding, outputs 28x28x6
RELU	
Max pooling	outputs 14x14x6
Convolution	5x5x1, 1 stride, Valid padding, outputs 10x10x16
RELU	
Max pooling	outputs 5x5x16
Flatten	Output 400
Dropout	keep_probe = 0.5
Fully connected	Output =120
RELU	
Fully connected	Output =84
RELU	
Fully connected	Output =43
Softmax	

3. Model Training:

To train the model, I used the AdamOptimizer with learning rate of 0.0009, batch_size of 100, over 70 epochs.

Aside from playing around with the learning rate and batch size to increase the validation set accuracy, a Dropout layer was added to the LeNet-5 architecture:

```
# TODO: Flatten. Input = 5x5x16. Output = 400.
F1 = flatten(P2)

# Dropout
DO = tf.nn.dropout(F1, keep_prob)

# TODO: Layer 3: Fully Connected. Input = 400. Output = 120.
FC1_W = tf.Variable(tf.truncated_normal(shape=(400, 120), mean = mu, stddev = sigma))
FC1_b = tf.Variable(tf.zeros(120))
FC1   = tf.matmul(DO,FC1_W) + FC1_b
# TODO: Activation.
FC1 = tf.nn.relu(FC1)
```

Using LeNet-5 a training accuracy of 94.5% and testing accuracy of 93% is achieved. Hyperparameters are used as below: Epochs = 70, Learning Rate = 0.009, Batch Size = 100. While this performance is good, to make the network more robust and efficient the Dropout technique is used between the convolution layers and fully connected layers in my model (LeNet_HI). Hypothesize is that this will make conv and fully connected layer learn independently and ovoid overfitting; because they are designed to do different tasks in the

model. A keep probability of 0.5 is used in the new model. As we'll see below, with this small modification, the training accuracy of %96 and testing accuracy of %94.5 is achieved (at some runs the numbers were even higher). My final model results were:

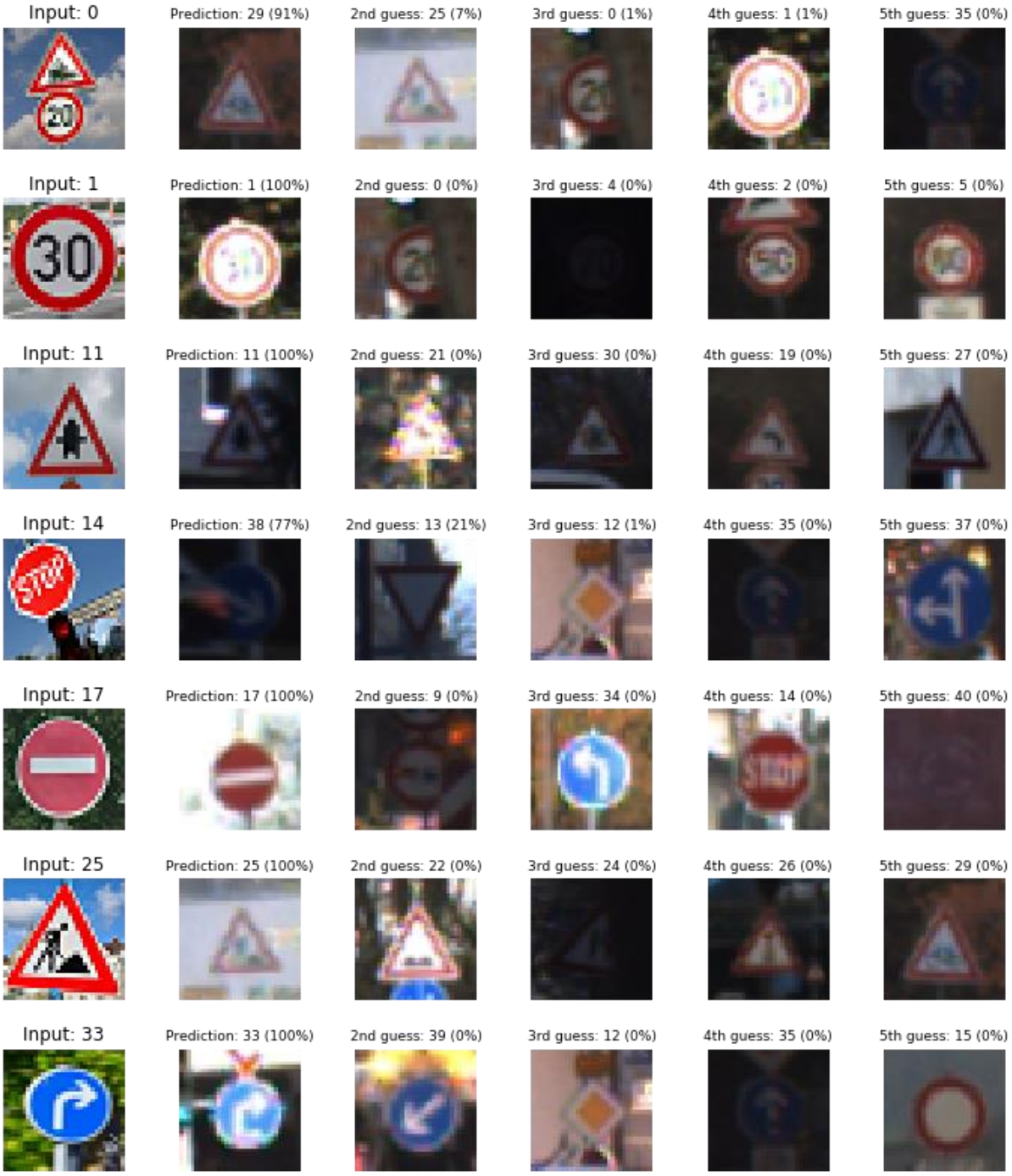
- validation set accuracy of 96%
- test set accuracy of 94.5%

Test a Model on New Images

Here are 7 German traffic signs that I found on the web:

Images 3, 4 and 6 are a little difficult to detect because the signs are a little rotated in the image. Image 7 was not square initially (aspect ratio was not 1) and it was tilted a little backward. Image 1 has 2 sign in it. Image 2 has a busy background which is used to test the robustness of the model.

Using LeNet-HI, the prediction accuracy for this set of 7 images is 71%. The probabilities for 1st to 4th guesses for each image are shown below. The model misclassifies 2 of the seven images: image 1 (with 2 sign in it) and image 4(the image with rotated and tilted stop sign). One reason might be the model has not been trained for these cases. Then more reliable models can be developed by including such versions of the image in the training dataset.



Summary:

A CNN model is developed based on the LeNet-5 model; the new model uses a dropout technique to improve learning performance. The developed model has an acceptable accuracy when used on the testing set (%94.5). However, on more challenging images when for example

the traffic sign is rotated or there are more than 1 sign in the image the model doesn’t provide good performance, such shortcomings can be tackled by using such images in the training set.