

Traffic Sign Recognition

Lingyi

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

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

You're reading it!

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 len() function to calculate summary statistics of the traffic signs data set:

```

print("Number of training examples =", n_train)
print("Number of testing examples =", n_test)
print("Number of validation examples =", n_validation)
print("Image data shape =", image_shape)
print("Number of classes =", n_classes)

```

```

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

```

Visualization of the data for different label:

Visulizaition of Data by different type

Speed limit (20km/h)



Speed limit (30km/h)

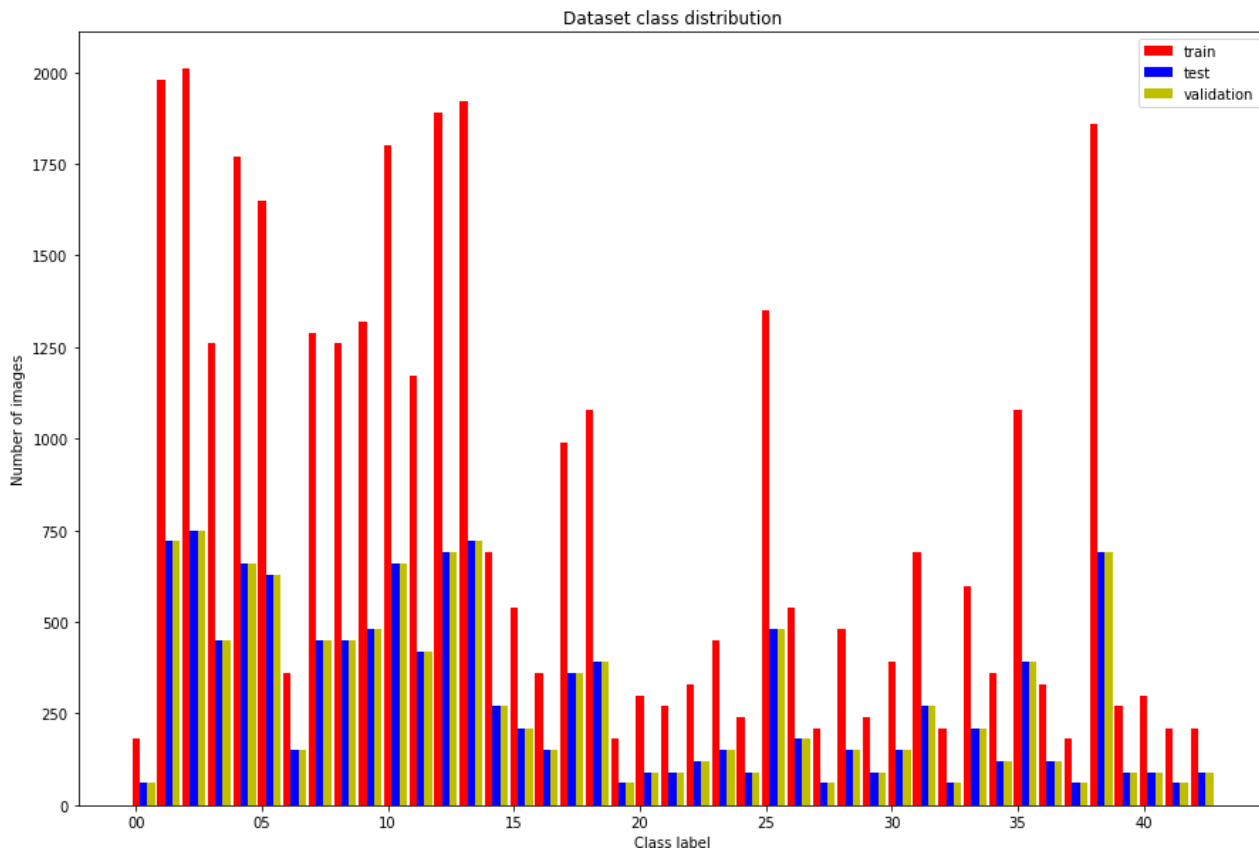


Speed limit (50km/h)



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 sets distribution.



Design and Test a Model Architecture

1. Describe how you preprocessed the image data.

```
def preproImg(img):

    image = cv2.split(cv2.cvtColor(img, cv2.COLOR_RGB2YUV))[0]

    image = cv2.equalizeHist(image)

    mini, maxi = np.min(image), np.max(image)
    image = (image - mini) / (maxi - mini) * 2 - 1

    image = np.expand_dims(image, axis=2)

    return image

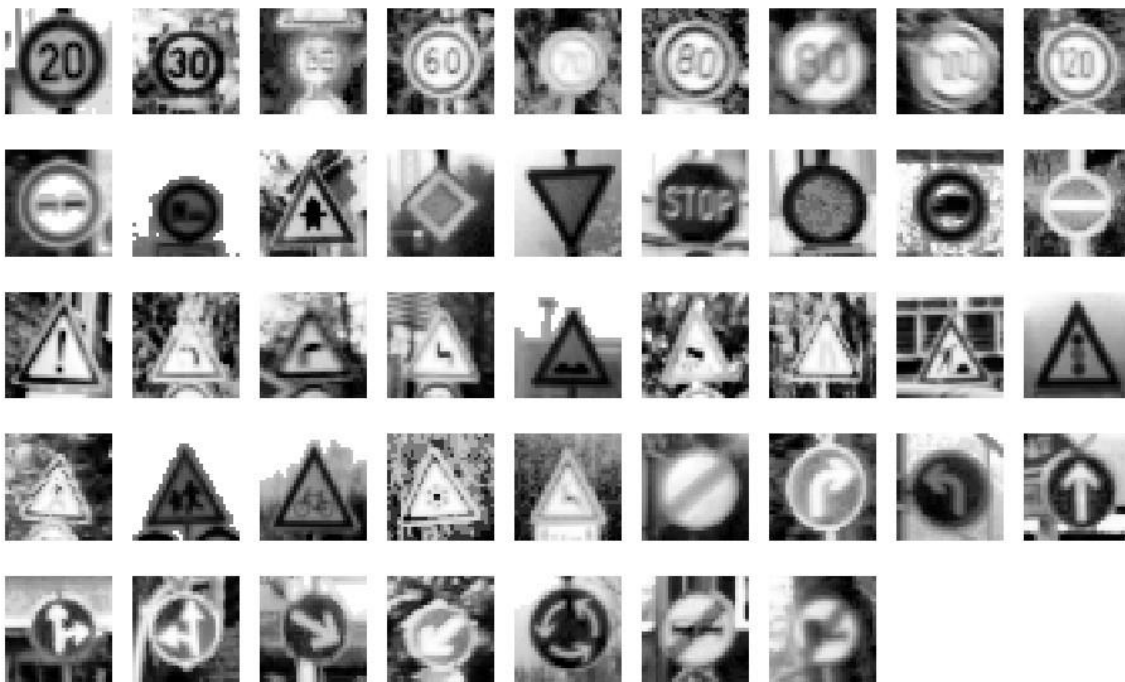
def preproData(data):

    return np.array([preproImg(im) for im in data])
```

Here I decided to convert the images to grayscale as previous experiments show using gray images produces better results than using color images. Histogram equalization was applied to improve visibility of the signs and normalization with zero mean was used to facilitate the convergence of the optimizer during training.

Visualization after the pre-processing.

Visulization of the data after preprocessing in grayscale



2. Describe what your final model architecture looks like

Input: 32x32x1 images

1st: 3x3x12 kernel, stride 1, convolution with ReLU activation, output 30x30x12

2nd: 3x3x24 kernel, stride 1, convolution with ReLU activation, output 28x28x4

Max pooling, output 14x14x24

3rd: 5x5x36 kernel, stride1, convolution with ReLU activation, output 10x10x36

4th: 5x5x48 kernel, stride 1, convolution with ReLU activation, output 6x6x48

Max pooling, output 3x3x48

5th: Flatten, concatenate 14x14x24 and 3x3x48, output 5136

6th: First fully connected with 512 neurons and dropout to reduce variance, output 512

7th: Second fully connected with 256 neurons and dropout to reduce variance, output 256

Final Output: 43 fully connected layer.

3. Describe how you trained your model.

batch_size = 32

keep_prob = 0.7

The maximum number of epochs is 30, and if up to 5 epochs there is no change in the accuracy, the training will be forced to stop.

```
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=one_hot_y)
loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
training_operation = optimizer.minimize(loss_operation)
```

So first I process the data to 32x32x1 images and then I input the training data to the network, getting the logits. And use `tf.nn.softmax_cross_entropy_with_logits` to get the cross entropy. I use `AdamOptimizer` to train my model, with a learning rate of 0.0001. In each training epoch, I will shuffle the training data of the single epoch first, and then feed them into the model. After that I will test the accuracy with the epoch's training data and validation data. And for the two fully connected layers, I keep a keep probability as 0.7 for drop out operation.

4. Training results

```
EPOCH 29 ...
Training Accuracy = 1.000
Validation Accuracy = 0.972
```

```
Training finished after 5 epochs without improvement
```

test accuracy

```
with tf.Session() as sess:
    saver.restore(sess, tf.train.latest_checkpoint('.'))
    print('Test accuracy: {:.3f}%'.format(evaluate(x_te, y_test) * 100))
```

```
INFO:tensorflow:Restoring parameters from ./model.ckpt
Test accuracy: 95.154%
```

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report.



2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set.

The accuracy is 80% for the 5 new images. It is 100% sometimes, depends on the training model. This means my model or training parameters could have been improved.











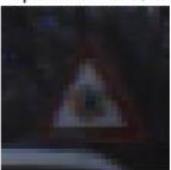












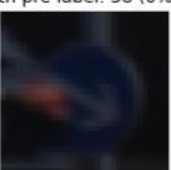





3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction.

Image quality that may affect the classification:

- Resolution: it may affect the accuracy for prediction, for example, the speed limit of 30 and 80, they are similar if the resolution is not high, the model may produce wrong type.
- Brightness: if the images' brightness is very low, the image may be hard to find the features.

etc.

The top 5 softmax probabilities for each image along with the sign type of each probability are shown.

sign 	max pre label: 5 (82%) 	2nd pred label: 1 (18%) 	3rd pre label: 31 (0%) 	4th pre label: 37 (0%) 	5th pre label: 18 (0%) 
sign 	max pre label: 13 (100%) 	2nd pred label: 21 (0%) 	3rd pre label: 34 (0%) 	4th pre label: 30 (0%) 	5th pre label: 2 (0%) 
sign 	max pre label: 14 (80%) 	2nd pred label: 3 (12%) 	3rd pre label: 1 (7%) 	4th pre label: 2 (1%) 	5th pre label: 34 (0%) 
sign 	max pre label: 15 (99%) 	2nd pred label: 12 (1%) 	3rd pre label: 13 (0%) 	4th pre label: 14 (0%) 	5th pre label: 38 (0%) 
sign 	max pre label: 17 (100%) 	2nd pred label: 33 (0%) 	3rd pre label: 9 (0%) 	4th pre label: 14 (0%) 	5th pre label: 34 (0%) 