Traffic Sign Recognition

Writeup

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

---

1. Basic summary of the data set.

The training, validation and testing data are provided and stored in each pickled file. Each pickled data is a dictionary with 4 key/value pairs:

- 'features' is a 4D array containing raw pixel data of the traffic sign images, (number of examples, width, height, channels).
- 'labels' is a 1D array containing the label/class id of the traffic sign. The file signnames.csv contains id -> name mappings for each Id.
- 'sizes' is a list containing tuples, (width, height) representing the the original width and height the image.
- 'coords' is a list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image.

After loading the traffic sign data into Python (code cell [1]), and extracting 'features' and 'labels' from the pickled data, I got training, validation, and testing images as:

- `img_train: (34799, 32, 32, 3)`
- `img_valid: (4410, 32, 32, 3)`
- `img_test: (12630, 32, 32, 3)`

In the above data tensors, the first dimension is the number of samples, and the second to fourth dimension are the sizes of each sample. That being said, each image is 32x32 color images (3 RGB color channels).

Thus, I can get the number of training, validation, testing images by extracting first dimension of each tensor (code cell [2]):

```
n_train = img_train.shape[0]
```

```
n_test = img_test.shape[0]
```

The shape of an traffic sign image is dim `2` to dim `4` of the `img_train` tensor:

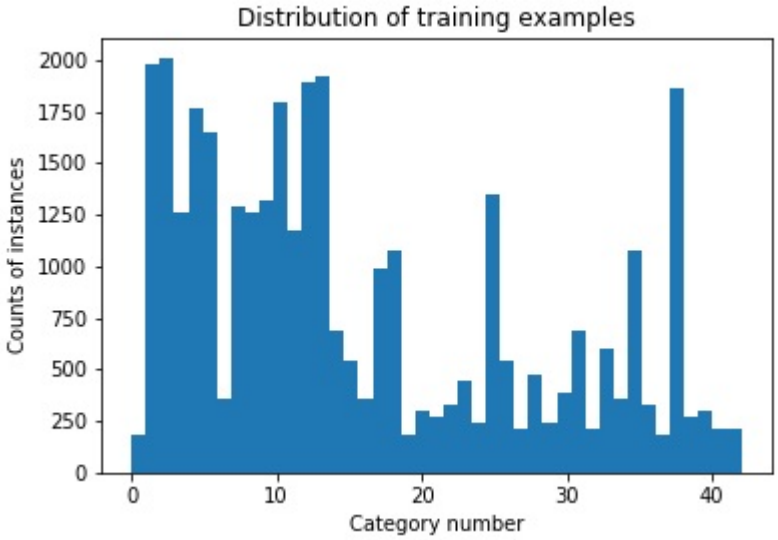
```
image_shape = img_train.shape[1:]
```

The number of classes in the data set is number of unique labels in `y_train`, which can be computed as:

```
n_classes = len(set(y_train))
```

2. Exploratory visualization of the dataset.

To have a sense of how many classes and how the classes are distributed in the dataset, a histogram is plotted to visualize the distribution of all classes (code cell [6]). As the histrogram shows, each class is not evenly distributed. Some classes have `1250~2000` examples while most classes in category `20-43` have `~250` examples.

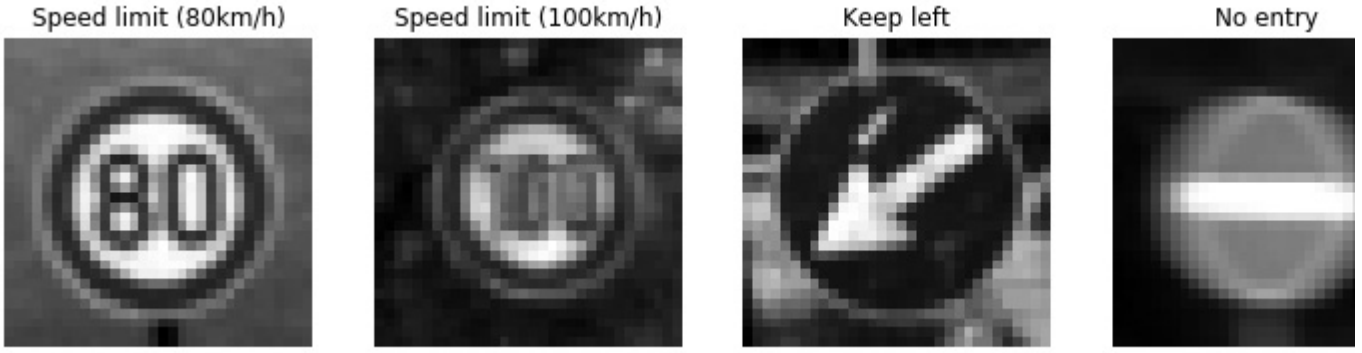


Design and Test a Model Architecture

1. Image pre-processing (converting to grayscale and normalization).

As the original images shows, raw data are cropped images of traffic signs, with a very low spatial resolution (`32x32`). The samples are in color space, and illumination condition is largely varied across classes. Some traffic signs has very well lighting condition while some are very poorly illuminated (code cell [7]).

To reduce sensitivity in color, each color image is converted into gray-scale. Also, to compensate variance in illumination, following gray-scale images are normalized into range of [0,1], using Min-Max scaling (code cell [7]).



2. Training, validation and testing data.

The training, validation and testing data are provided and stored in each pickled file. As mentioned above, training set has 34799 images, validation set has 4410 images and test set has 12630 images(code cell [1]).

3. Final model architecture.

The model consists of 2 layers of convolution and 3 layers of feed-forward networks. Each layer of convolution is with 5x5 spatial filters, followed by a RELU activation and a Max pooling in 2x2 neighborhood. The filter depth in layer 1 and layer 2 are 6 and 12. Since traffic signs are of spatial size of 32x32, 5x5 convolution kernal is a good balance between extracting useful features without being too computational intensive. Most of traffic signs are in centered part of the images, so valid padding is used for convolutions.

In the feed-forward network, the number of neurons decrease to approximately a half to the size of it's previous layer. In summary, the architecture of the entire model is as follows:

Layer	Description
Input	32x32x3 RGB image
Convolution1 5x5x6	valid padding, output: 28x28x6

Layer	Description
RELU	
Max pooling	2x2 neiborhood, output: 14x14x6
Convolution2 5x5x12	valid padding, output: 10x10x12
RELU	
Max pooling	2x2 neiborhood, output: 5x5x12
Flatten	output: 300
Fully connected 1	300x150, output: 150
Fully connected 2	150x84, output: 150
Fully connected 3	84x43, output: 43
Softmax	output: probabilities

---

4. Train the convolution network.

For network training, a Adam optimizer is used for minimizing the training loss. The loss is computed as cross-entropy of softmax probabilities and traning labels. The Adam optimizer uses moving averages to update parameters, to prevent stuck in local optima. The algorithm uses an adaptive step size for learning, and thus converges to this step size without fine tuning. As a trade-off, the model requires more computation and memory parameters in each training step. The initial learning rate is 0.001.

5. Mini-batch feeding and network convergence

An iterative approach is used to train the network for convergence. In each epoch, the whole data set is divided into mini-batches before feeding to the network. This makes training faster and scalable. The batch size is 128. A validation accuracy is monitored at each iteration. As output of code cell [10] shows, the validation accuracy is increasing from Epoch 1 to epoch 45. After epoch 45, the validation accuracy changes downward. That means 45 is a good epoch number to train the network sufficiently without overfitting.

The model is very similar to LeNet. LeNet has successfully used for OCR problem for MNIST data. Since traffic sign data is also of very low resolution, a 2 layer convolution followed by a 3 layer feedforward network is chosen. The final model results were

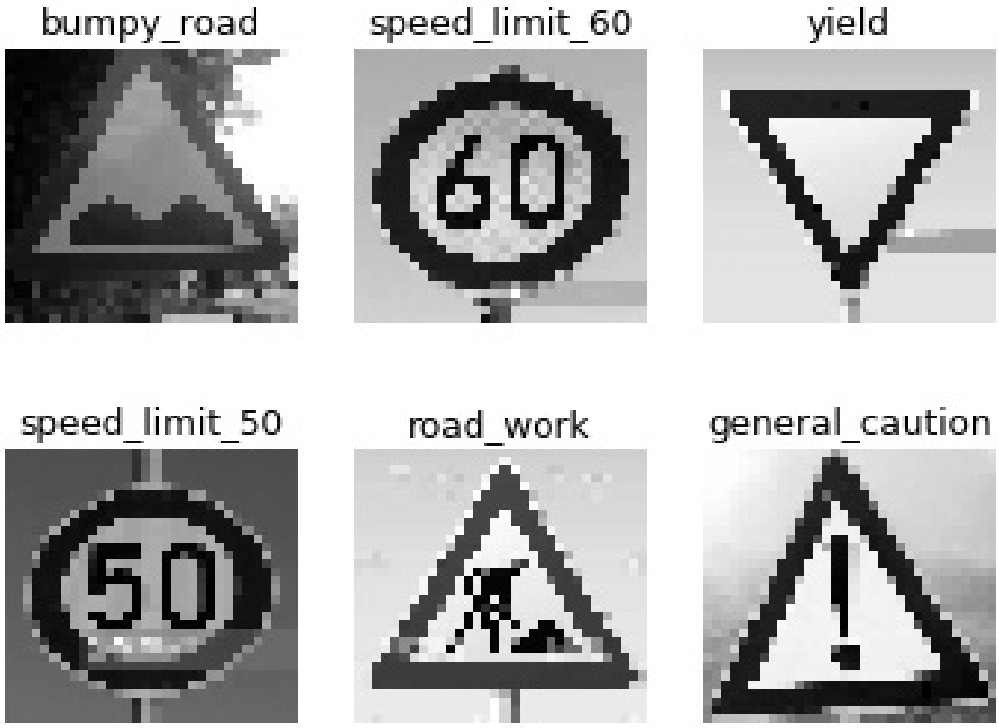
- training set accuracy: 100.00%
- validation set accuracy: 92.90%
- test set accuracy: 91.81%

Test a Model on New Images

1. Choose 9 German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are 9 German traffic signs that I found on the web:

After converting to grayscale and normalize pixel values, the output test images looks like:



Although original images are of high relolution and pretty straight-forward, the preprocessed images might be difficult to classify since they are greatly down-sampled. "Children crossing" and "Stop" sign images looks most challenging.

2. Predictions on new traffic signs.

After feeding these test images into the network, here are the results of the prediction:

Truth	Prediction
Children Crossing	Children crossing
No vehicles	No vehicles
Stop	No vehicles
Bumpy road	Turn left ahead
Speed limit 60	Speed limit (60km/h)
Yield	Yield
Speed limit 50	Speed limit (50km/h)
Road work	Road work
General caution	General caution

Out of 9 images, 7 of them are correctly classified, which gives an accuracy of 77.78%. It's less than what we got on test set (91%), but it's a very small dataset and 1 misclassified images cause the accuracy to drop by 11%. Surprisingly, children crossing image is correctly classified even though is heavly down-sampled. "Stop" sign is misclassified as "No vehicles" and "Bumpy road" is misclassified as "Turn left". I guess both have curves in the image center, which is confusing to the network.

3. Softmax probabilities for each prediction.

The code for making predictions on my final model is located in the code cell [16].

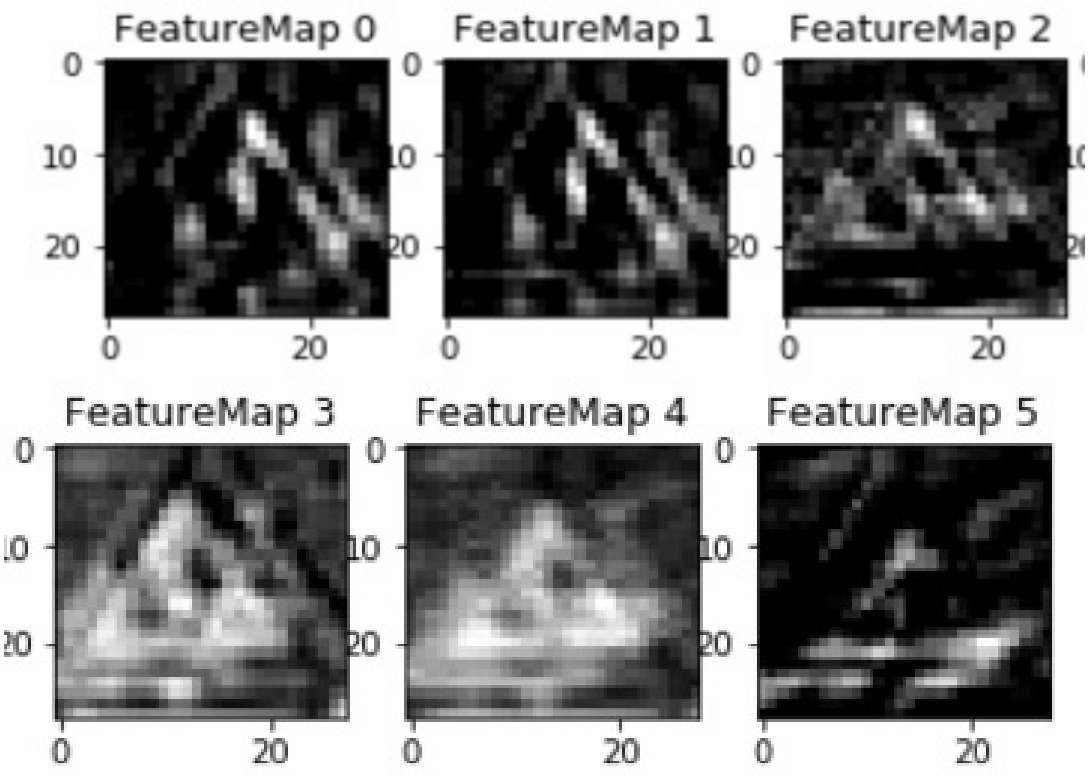
For each prediction, the newtork is very confident (the corresponding probability being 1 and all else being 0). For example, the top five soft max probabilities were

Probability	Prediction
1.0	Children crossing
0.0	U-turn
0.0	Yield
0.0	Bumpy Road
0.0	Slippery Road

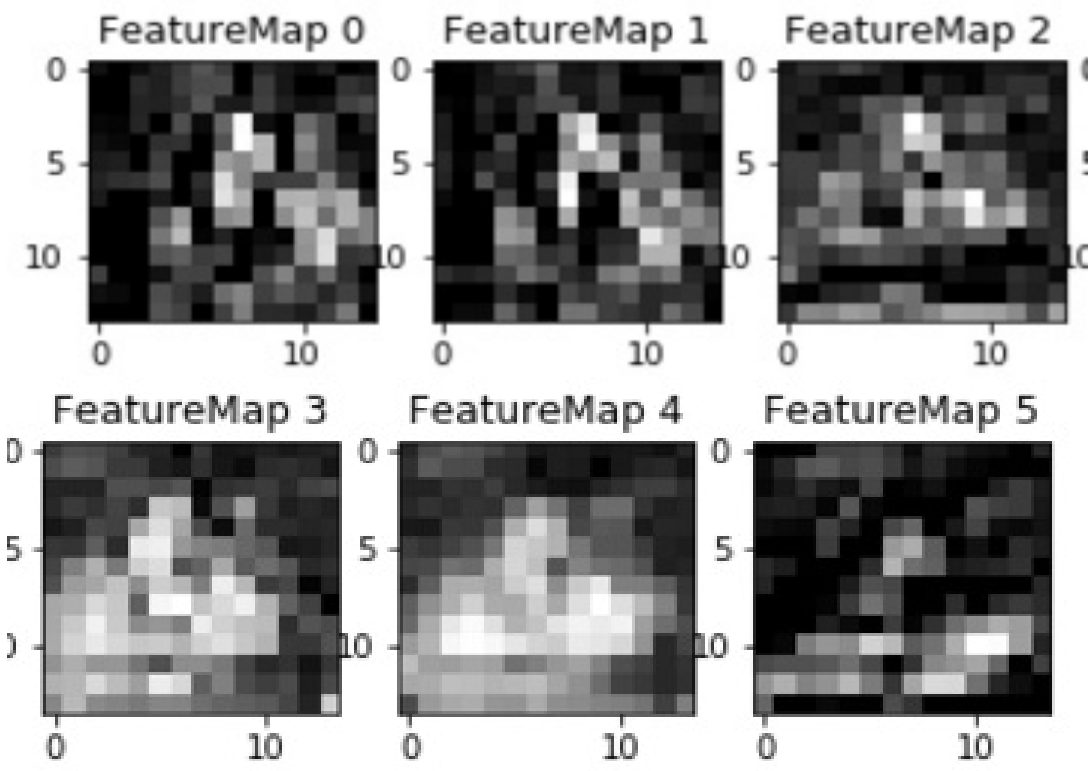
This means that the model is very well trained. Some methods to prevent overfitting might be helpful to improve the network.

3. Feature maps

In layer C1, the feature map picks up main edges of the input image. Each feature map detected edges of different orientations, in combination all the feature maps learned the main shape of the traffic sign (a triangle).



Layer S1 feature maps are down-sampled version of C1 feature maps. Layer C2 neurons fires in a geometric shape very similar to the grayscale input image. Layer S2 is a down-sampled version of layer C2 features.


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5