Build a Traffic Sign Recognition Project

Yuxiang 2017/7/20

I. Project Description

Build a Traffic Sign Recognition Project

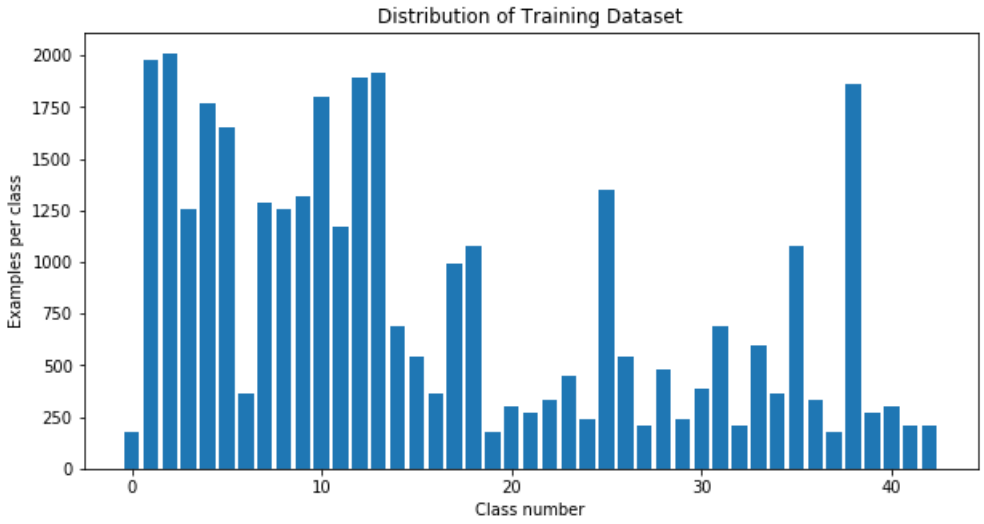
The goals / steps of this project are the following:

- 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

II. Dataset

1. Data exploratory and visualization

```
Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```



III. Method

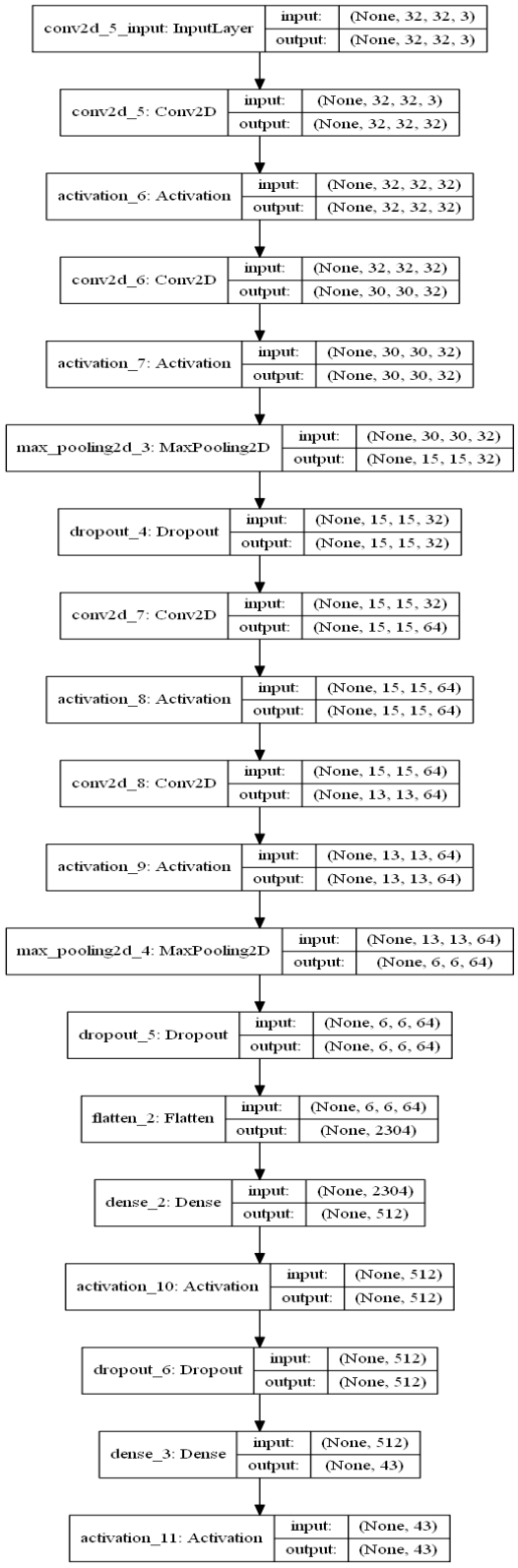
1. Data Preprocess

One-hot encoding: y_train, y_valid, y_test

Augmentation: randomly rotate images in 10 degree; randomly shift images horizontally and vertically 10% of the image width and height.

Normalization: X_train, X_valid, X_test are divided by 255.

2. Model


conv2d_5_input: InputLayer
input: (None, 32, 32, 3)
output: (None, 32, 32, 3)
conv2d_5: Conv2D
input: (None, 32, 32, 3)
output: (None, 32, 32, 32)
activation_6: Activation
input: (None, 32, 32, 32)
output: (None, 32, 32, 32)
conv2d_6: Conv2D
input: (None, 32, 32, 32)
output: (None, 30, 30, 32)
activation_7: Activation
input: (None, 30, 30, 32)
output: (None, 30, 30, 32)
max_pooling2d_3: MaxPooling2D
input: (None, 30, 30, 32)
output: (None, 15, 15, 32)
dropout_4: Dropout
input: (None, 15, 15, 32)
output: (None, 15, 15, 32)
conv2d_7: Conv2D
input: (None, 15, 15, 32)
output: (None, 15, 15, 64)
activation_8: Activation
input: (None, 15, 15, 64)
output: (None, 15, 15, 64)
conv2d_8: Conv2D
input: (None, 15, 15, 64)
output: (None, 13, 13, 64)
activation_9: Activation
input: (None, 13, 13, 64)
output: (None, 13, 13, 64)
max_pooling2d_4: MaxPooling2D
input: (None, 13, 13, 64)
output: (None, 6, 6, 64)
dropout_5: Dropout
input: (None, 6, 6, 64)
output: (None, 6, 6, 64)
flatten_2: Flatten
input: (None, 6, 6, 64)
output: (None, 2304)
dense_2: Dense
input: (None, 2304)
output: (None, 512)
activation_10: Activation
input: (None, 512)
output: (None, 512)
dropout_6: Dropout
input: (None, 512)
output: (None, 512)
dense_3: Dense
input: (None, 512)
output: (None, 43)
activation_11: Activation
input: (None, 43)
output: (None, 43)

This model refers to keras examples on cifar10. This is a simple deep CNN which are suitable to train on the small images dataset. Our dataset is also a small images dataset. The large CNN may requires large input size of the image. So we need to resize the image if we want to use some famous CNN such as Resnet50. The net is a little bit similar as the structure of VGG16 which has been proved as a good CNN.

---

3. Train

Optimizer: adam Batch size: 64 Epochs: 20

IV. Result

Training set accuracy: 97.13%

Validation set accuracy: 98.62%

Test set accuracy: 97.70%

Test on 5 New Images

Accuracy: 100%

The probability of how certain a prediction was made:

100%, 99.15%, 64.41%, 98.55%, 100%

The first sign graph is easy to classify because the sign covers a large portion of this graph. It is clear when we look at it.

The second sign graph has some blur so that it is not clear as well as the first graph, 99.15% can be accepted.

The third sign graph is most difficult one to classify by the model. The sign portion is small and we can see the angle of the picture is not the front. So the quality of this picture is not very good.

The fourth one is easier to classify than the third one. The portion of the sign is small but the color difference is obvious. We can consider it as a not bad picture.

The fifth one is obvious a picture with a good quality. So our model find it easy to classify.