Report.md

Traffic Sign Classification

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. Loading the Data

At first we need to load data from training and testing files. Then splitted the data to get testing set as 20% of training data. So training data inludes:

Number of training examples = 27839

Number of testing examples = 12630

Number of validating examples = 6960

Number of classes = 43

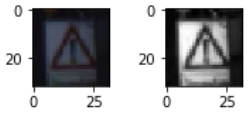
And each image has shape (32, 32, 3)

2. Preprocessing pictures

For preprocessing pictures I used **2 methods:**

-normalize image data following (pixel - 128)/ 128 equation

-converting RGB image to grayscale



We have good principles:

-We want our variables have 0 mean and equal variance. Well conditioned data provides better results and makes easier to find solution. In picture pixels are between [0:255]. I substracted 128 and divided by 128, it doesn't change the content but it's easier for the optimization.

Grayscaled image decrease number of parameters, decrease image depth. We don't need to learn color of signs. The color is not a factor now, I just need to classify signs and it's faster/ easier to classify pictures ignoring color scheme.

Each image becomes to be (32, 32, 1)

2. Model Architecture

My LeNet architecture contains:

1. Convolutional Layer with input: (32, 32, 1), shape: (5, 5, 1, 6), strides: (1, 1, 1, 1) with padding:'VALID'. The output: (28,28,6) Then I used ReLu activation function and POOLing operatiion with kernel size=(1, 2, 2, 1], strides=(1,2,2,1) with padding:'VALID'
2. Second layer is also Convolutional with characteristics:shape=(5, 5, 6, 16),strides=(1, 1, 1, 1]),padding:'VALID' with ReLu function after and POOLing operation also Input = 10x10x16. Output = 5x5x16
3. Then I made vector flatten with output of 400 points

4. Next Layer is fully connected with Input = 400. Output = 120, ReLu activation and then dropout function for Regularization and have increased number of epochs
5. Layer 4 is also fully connected with Input = 120. Output = 84,ReLu activation and then dropout function for Regularization
6. Layer 5 fully connected with Input = 84. Output = 43 and returns logits, calculated using formula tf.matmul(fc2, fc3_W) + fc3_b

Finding solution:

-I use convolutional networks for implementing sharing weights idea throw the image. This method allows us don't care where in the image traffic sign is.

-I use ReLu activation function instead of sigmoid function to cut not matter details of picture which return me negative output.

-Then between conv layers I put max pooling operation for getting max perceptron value in shape(2x2). Compressing 2x2 to 1 pixel with maximum value allow to optimize network, if we've already got some properties in previous conv. layer we could cut some information for increasing the speed and avoiding network overfitting.

-After I've added dropout operation with keep prob=0.5 between fully connected layers to avoid overfitting neural network, then I've choosen keep_prob to 0.8 and the best result I've got using keep_prob=0.7 throw big set of experiments.

This regularization method gave me better accuracy than just standard LeNet architecture.

Let's talk about hiper parameters:

I've spend a lot of time to choose optimal. When I've added dropout operations between fully connected layers I've decided that I need to increase number of epochs and count of epochs now is 15. We know good rule: "Keep calm and lower Learning rate". At first I've tried to increase learning rate expecting to get train network faster with small amount of loss, but practice says that lower rate gives better results, so now learning rate is 0.001 I did all experiments on my personal PC and when I've increased batch size I've got lack of sources, so the optimal size was like in LeNet Lab=128 as I did on the previous lesson. And for optimizer I used AdamOptimizer as we did on LeNet Lab.

So:

Learning rate is: 0.001

For Training I used keep_prob=0.7 and for Validation keep_prob=1.0

Count of Epochs=15

Batches size=128

Optimizer: Adam

3. Training Results

EPOCH 1 ... Validation Accuracy = 0.637

EPOCH 2 ... Validation Accuracy = 0.847

EPOCH 3 ... Validation Accuracy = 0.902

EPOCH 4 ... Validation Accuracy = 0.937

EPOCH 5 ... Validation Accuracy = 0.951

EPOCH 6 ... Validation Accuracy = 0.958

EPOCH 7 ... Validation Accuracy = 0.966

EPOCH 8 ... Validation Accuracy = 0.970

EPOCH 9 ... Validation Accuracy = 0.973

EPOCH 10 ... Validation Accuracy = 0.976

EPOCH 11 ... Validation Accuracy = 0.973

EPOCH 12 ... Validation Accuracy = 0.978

EPOCH 13 ... Validation Accuracy = 0.979

EPOCH 14 ... Validation Accuracy = 0.981

EPOCH 15 ... Validation Accuracy = 0.980

Test Accuracy = 0.923

And this results meet all requirements for task

4. Testing Model on New Images

I've downloaded 5 pictures from the internet (GTS folder) and reshaped them to 32x32 size:

© Can Stock Photo

Then made for them 2 operations:

-normalize image data following (pixel - 128)/ 128 equation

-converting RGB image to grayscale

Prediction results are not so bad as I think:

[12 3 33 25 34]

4 of them are correct (12,3,25,34) so accuracy= 80.0 % In comparison of test Accuracy very good result

5. Softmax Probabilities

Softmax probabilities look-like this:

[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0.99900001 0. 0.001 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.001 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.991 0. 0. 0. 0. 0. 0. 0.007 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.004 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.99199998 0.003 0. 0. 0. 0. 0. 0.001 0.]]

Top 5 Probabilities:

[1. , 0. , 0. , 0. , 0.]

[0.99900001, 0.001 , 0. , 0. , 0.]

[0.991 , 0.007 , 0.001 , 0. , 0.]

[1. , 0. , 0. , 0. , 0.]

[0.99199998, 0.004 , 0.003 , 0.001 , 0.]

Predictions:

[12, 0, 1, 2, 3]

[3, 5, 0, 1, 2]

[33, 40, 9, 0, 1]

[25, 0, 1, 2, 3]

[34, 20, 35, 41, 0]

That means that for first prediction 12- probability=1.0 and this is correct answer, for second prediction 3: probability= 0.999 and this is correct answer, for third prediction 33 probability=0.991 and answer is not correct, four prediction is also correct with value=25 and probability=1.0, for the last prediction the probability= 0.991 and the ansver is correct.

I think that my network couldn't make valid prediction for second picture because i need to train it more using a lot of transformations for data