Udacity Self Driving Cars Nanodegree- Traffic Sign Classifier project

Objective:

The objective of this project was to build a Traffic Sign Classifier model using Deep Learning. A dataset of German Traffic Signs was used for developing the model. This dataset had 34799 training set examples, 4410 validation set examples and 12630 testing set examples.

Approach:

The LeNet model from previous lesson was leveraged for this project. This model was developed to build recognize number signs from the MNIST dataset. This model had the following architecture:

Input: 28x28x1 image Layer1:

Conv1: Input: 28x28x1; Output: 28x28x6

Pool1: Input: 28x28x6; Output: 14x14x6

Layer2:

Conv1: Input: 14x14x6; Output: 10x10x16 Pool1:

Input: 10x10x16; Output: 5x5x16 Then, flatten

layer

Layer 3:

Fully connected layer: Output: 120 outputs Layer

4:

Fully connected layer: Output: 84 outputs Layer5:

Fully connected layer: Output: 10 outputs

--

ReLU activation function was used in each of the layers, except the output layer (which used the Softmax function).

The same model was used for this project. Couple minor changes were:

- Input for this project model was a 32x32x3 image; which was then converted to grayscale to get a 32x32x1 image. (Input for the MNIST number recognition was a 28x28x1 image which was then zero padded to convert to a 32x32x1 image as part of Pre-processing).

- Output number of classes for this project model was 43 (for the 43 traffic signs); while for the MNIST numbers recognition example; the output number of classes were 10 (for the 10 numbers 0-9).

Final neural network architecture:

After the above changes, the final neural network architecture used for this project was: Input:

32x32x3 image

As part of Pre-processing step; the images were converted to grayscale and normalized (to have zero mean and variance 1).

Layer1:

Conv1: Input: 32x32x1; Output: 28x28x6

Pool1: Input: 28x28x6; Output: 14x14x6

Layer2:

Conv1: Input: 14x14x6; Output: 10x10x16 Pool1:

Input: 10x10x16; Output: 5x5x16 Then, flatten

layer

Layer 3:

Fully connected layer: Output: 120 outputs Layer

4:

Fully connected layer: Output: 84 outputs

Layer5:

Fully connected layer: Output: 43 outputs

Keep_prob=1.0 was used for dropout; so in effect; dropout regularization was not used.

Results:

As part of Pre-processing step; the images were converted to grayscale and normalized (to have zero mean and variance 1).

The LeNet model with the minor changes as above was trained on the current dataset with the following hyperparameters: learning rate: 0.01; number of epochs: 10 and batch size: 128. It produced decent results (training set accuracy of up to 97% and validation set accuracy up to 91.5%). However, the project required result was a validation set accuracy of at least 93%.
Reducing the learning rate from 0.01 to 0.005 and increasing the number of epochs to 20 produced the minimum desired result:

On Epoch 18; it produced Training set accuracy of 99.5% and Cross Validation set accuracy of 94.6%. Screenshot of the Training run is below.

```
--
Training...

EPOCH 1 ...
Training Accuracy = 0.937

EPOCH 1 ...
Validation Accuracy = 0.880

Model saved EPOCH 2
...
Training Accuracy = 0.973

EPOCH 2 ...
Validation Accuracy = 0.905

Model saved EPOCH 3
...
Training Accuracy = 0.984

EPOCH 3 ...
Validation Accuracy = 0.916

Model saved EPOCH 4
...
Training Accuracy = 0.984

EPOCH 4 ...
Validation Accuracy = 0.916
```

Model saved EPOCH 5
...
Training Accuracy = 0.982

EPOCH 5 ...
Validation Accuracy = 0.923

Model saved EPOCH 6
...
Training Accuracy = 0.986

EPOCH 6 ...
Validation Accuracy = 0.924

Model saved EPOCH 7
...
Training Accuracy = 0.982

EPOCH 7 ...
Validation Accuracy = 0.913

Model saved EPOCH 8
...
Training Accuracy = 0.992

EPOCH 8 ...
Validation Accuracy = 0.929

Model saved EPOCH 9
...
Training Accuracy = 0.987

EPOCH 9 ...
Validation Accuracy = 0.935

Model saved EPOCH
10 ...
Training Accuracy = 0.987

EPOCH 10 ...
Validation Accuracy = 0.923 Model saved

```
EPOCH 11 ...
Training Accuracy = 0.980
```

Validation Accuracy = 0.918 Model saved


```
EPOCH 11 ...
```


```
EPOCH 12 ...
Training Accuracy = 0.983
```

EPOCH 12 ...

Validation Accuracy = 0.915

Model saved EPOCH
13 ...
Training Accuracy = 0.990

EPOCH 13 ...
Validation Accuracy = 0.927

Model saved EPOCH
14 ...
Training Accuracy = 0.989

EPOCH 14 ...
Validation Accuracy = 0.919 Model saved

```
EPOCH 15 ...
Training Accuracy = 0.977
```

Validation Accuracy = 0.919 Model saved

```
EPOCH 15 ...
```

```
EPOCH 16 ...
Training Accuracy = 0.992
```

Validation Accuracy = 0.941 Model saved

```
EPOCH 16 ...
```

EPOCH 17 ...
Training Accuracy = 0.993

EPOCH 17 ...
Validation Accuracy = 0.938

Model saved EPOCH
18 ...
Training Accuracy = 0.995

EPOCH 18 ...
Validation Accuracy = 0.946

Model saved EPOCH
19 ...
Training Accuracy = 0.990

EPOCH 19 ...
Validation Accuracy = 0.928

Model saved EPOCH
20 ...
Training Accuracy = 0.984

EPOCH 20 ...
Validation Accuracy = 0.928 Model saved

--

On the test set within the dataset provided; an accuracy of 91.3% was achieved.

On a test set of 6 test images downloaded from the web; the model produced an accuracy of 83.3%.

Considerations on the performance:

Since the Training set accuracy was high (up to 99%); the model did not have a high Bias problem. Therefore, changing the neural network architecture (for example, to have more layers) or to train longer (more number of epochs) would not helped the performance of the model much; so I did not consider those measures.

The model did have a high Variance problem (Validation set error was 6-9% higher than Training set error). To rectify that, I considered two measures:

- Using Regularization: I tried to use Dropout regularization (with keep_prob values ranging from 0.5 to 0.9); however that seemed to have only worsen the model performance; therefore I decided not to pursue that measure.

- Having more data or data augmentation: This could have helped with the high Variance problem; but since the model was producing the minimum desired result; I decided not to pursue this measure in the interest of time and because I am not adept at data augmentation techniques at this point of time.