Convolutional Neural Nets: German Traffic Sign Classifier

Vamshidhar Pandrapagada

October 28, 2017

Project Intro

In this project, we use deep neural networks and convolutional neural networks to classify German road traffic signs. The Model is trained and validates so that it can classify traffic sign images using the German Traffic Sign Data set. After the model is trained, it has been tested on the actual images of German traffic signs that are found on the web.

The data set used can be found at German Traffic Sign Dataset

Data Visualization

The Distribution of classes across Train, Test and Validation sets are almost identical. This reflects in the bar charts plotted. Also, a plotting random sample of images (including all labels) gives us a general idea on how the traffic signs are.

Preprocessing Techniques

1. **Data Normaliation:** As for any data-set, image data has been normalized so that the numerical range is between 0 and 1.
2. **Batch Normalization:** As the inputs flow through different layers of the network their distribution change affecting the network's performance. This is called Internal-covariate shift problem. Technique called Batch normalization will center the inputs to zero and normalize them, then scale and shift the result using beta and gamma parameters. This is done just before the activation function of every layer. In order to zero-center and normalize the inputs, the training needs to calculate the input's mean and standard deviation. It does by evaluating the mean and standard deviation of the inputs for every mini batch. (hence the name batch normalization).
   This technique is proven to be very effective. In this case, it gives us a little nudge by improving the accuracy from 93 percent close to ~ 97 percent. The following code was taken from
3. **Image Augmentation**: This is a regularization technique to generate new training images from existing ones to boost the size of the training set. This will not only help the model learn better, but will also reduce over-fitting.
   The idea is to slightly shift, rotate/flip, change brightness/contrast every picture in the training set and add them back to the training set.
   This forces the model to be more tolerant to position and orientation of the key pixels in the image.

Model Design

Pipeline

This architecture used LeNet-5, which is perhaps the most widely known CNN architecture. It was originally designed by Yann LeCun in 1998 and was initially used to classify hand written digits. Below is the design in brief:

1. Input Image: 32 x 32 x 3
2. Convolution 1: Kernel size = 5, Feature maps = 6, Strides = 1, Padding = VALID, Output Image: 28 x 28 x 6
3. Maxpool Layer 1: Kernel size = 2, Strides = 2, Padding = SAME, Output Image: 14 x 14 x 6
4. Convolution 2: Kernel size = 5, Feature maps = 16, Strides = 1, Padding = VALID, Output Image: 10 x 10 x 16
5. Maxpool Layer 2: Kernel size = 2, Strides = 2, Padding = SAME , Output Image: 5 x 5 x 16
6. Flatten : Output = 400 neurons
7. Fully Connected Layer 1: Output = 120 neurons
8. Fully Connected Layer 2: Output = 84 neurons
9. Output Fully Connected Layer: Output = 43 neurons (no of labels)

Convolution 1, Convolution 2, Fully Connected Layer 1 and Fully Connected Layer 2 use batch normalization with relu activation function.

Drop out Regularization:

Drop out regularization was used on Fully Connected Layer 1 and Fully Connected Layer 2.

Hyper Parameters:

1. The number of epochs used: 10
2. Learning Rate: 0.01. Batch normalization has an advantage which will enable us to specify a higher learning rate without compromising on accuracy.
3. Batch size : 128
4. Drop out regularization probability: 0.6

Model Output

With the said hyper-parameters, the training and validation accuracy was close to ~97%. And the test accuracy was ~95%

Test the model on Images from Web

18 German traffic sign images were downloaded from the web. The model was able to classify images correctly ~84% of the time. The errors were due to Position and orientation of the key pixels in the images. This accuracy can be improved by additional data augmentation adding different orientations of the images to the data set. This will help the model adjust to different positions/angles of the pixels in the image leading to a better learning and higher accuracies.

Display predicted probabilities

This visualization (as a horizontal bar graph) will give a better assessment of the top 5 probabilities assigned by the model for each test image.