TensorFlow for MNIST data - Kaggle

Goal

The goal in this competition is to develop a model that recognizes a set of handwritten single-digit images.

DataSet

The data for this competition were taken from the MNIST dataset. The MNIST ("Modified National Institute of Standards and Technology") dataset is a classic within the Machine Learning community that has been extensively studied. More detail about the dataset, including Machine Learning algorithms that have been tried on it and their levels of success, can be found at http://yann.lecun.com/exdb/mnist/index.html.

Train.csv

The training dataset consists of 32,000 records with 785 columns.The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

Visually, if we omit the "pixel" prefix, the pixels make up the image like this:

```
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
 |   |   |   | ...  |   |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
```

Test.csv

The test data set, (test.csv), is the same as the training set, except that it does not contain the "label" column. It contains 10,000 test images in the form of pixel vectors.

Output

For each of the 10000 images in the test set, output a single line containing the ImageId and the digit you predict

Model

To understand the dataset, we first try converting the pixel vectors into image form and try printing them so that it will be easy for analysis
Example:

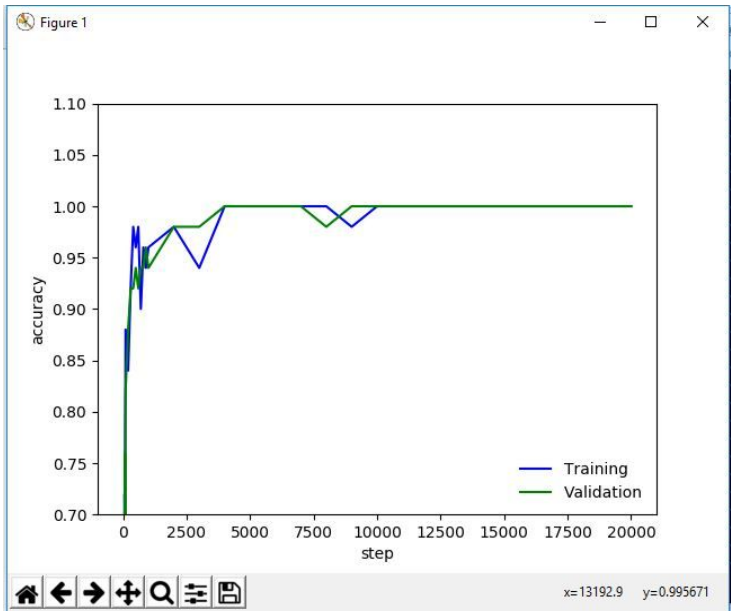
The neural network model has two convolutional layers with pooling in between them, then densely connected layer followed by dropout and lastly readout layer.
The first layer is a convolution, followed by max pooling. The convolution computes 32 features for each 5x5 patch. There is also a bias vector with a component for each output channel. To apply the layer, we reshape the input data to a 4d tensor, with the first dimension corresponding to the number of images, second and third - to image width and height, and the final dimension - to the number of colour channels. After the convolution, pooling reduces the size of the output from 28x28 to 14x14.

The second layer has 64 features for each 5x5 patch.There is also a bias vector with a component for each output channel. Now that the image size is reduced to 7x7, we add a fully-connected layer with 1024 neurones to allow processing on the entire image To prevent overfitting, we apply dropout before the readout layer.Finally, we add a softmax layer, the same one if we use just a simple softmax regression.To evaluate network performance we use cross-entropy and to minimise it ADAM optimiser is used.

To predict values from test data, highest probability is picked from "one-hot vector" indicating that chances of an image being one of the digits are highest. We use stochastic training with mini batches.

Analysis

Accuracy is plotted in the form of graph with respect to the number of iterations



Accuracy of 0.99/1.00 is achieved with this network model

Reference

TensorFlow Deep NN Tutorial