Deep Learning Lab Course
Second Exercise

Aaron Klein

Due: November 20, 2017

The goal of this exercise is to get familiar with Tensorflow by implementing a small convolutional neural network (CNN) and training it on the MNIST dataset. At the end, send us a small report (1 or 2 pages) with the below described plots and your interpretations of them.

1 Implementing a CNN in Tensorflow

First, we will implement a scaled-down version of LeNet which we have discussed during the lecture. You can reuse your code from the previous example to load MNIST. Our CNN consists of two convolutional layers (16 3×3 filters and a stride of 1), each followed by ReLU activations and a max pooling layer. After the convolution layers we add a fully connected layer with 128 units and a softmax layer to do the classification. We train the network by optimizing the cross-entropy loss with stochastic gradient descent. Make sure that you save the validation performance after each epoch, i.e the learning curve, since we are going to need them for the experiments in the next section.

2 Changing the Learning Rate

After implementing our neural networks and making sure that it works correctly, we will have a look on the effect of the learning rate on the network's performance. Try the following values for the learning rate: $\{0.1, 0.01, 0.001, 0.0001\}$, save the results (validation performance after each epoch) and plot all learning curves in the same figure. Which conclusions could be drawn from this figure? Which value for the learning rate works best?

3 Runtime

In the second experiment, we will study how the training time or runtime of convolutional neural network changes with the number of units. Train your neural network on a GPU with the following number of filters: $\{8, 16, 32, 64, 128, 256\}$, measure the runtime (by using python's build-in function time.time()) and compute the total number of parameters. Generate a scatter plot with the number of parameters on the x-axis and on the runtime on the y-axis. Each filter size should be one point in this scatter plot. How does the number of parameters change if you increase the number of filters? How does the runtime change? At last, run the same experiments again on a CPU (with only the following number of filters: $\{8, 16, 32, 64\}$). How does the plot change?