Submitted by: Lior Fuks Matriculation-Nr. 4251285

Deep Learning Lab Course Exercise 2

Implementation of a Convolutional Neural Networks using TensorFlow

Introduction

This exercise task is to implement a convolutional neural network using the TensorFlow Library under python. The convolutional neural network is composed of two convolutional layers with a 16 3x3 filter and a stride of 1. Each of the layers is followed by a ReLU activation function and a max-pooling layer. Following the convolutional layers a fully connected layer composed of 128 units is implemented with a softmax layer for classification. In each iteration, the network is optimized by minimizing the cross-entropy loss using stochastic gradient decent.
The convolutional neural network is used as a classification tool for the MNIST database. The MNIST database contains a set of images of hand written digit. This database is composed of a training set of 60000 images and a testing set of 10000 images.
TensorFlow is the main library used in order to implement the convolutional neural network. It was developed by Google and it is used commonly for machine learning applications.
This exercise aim is to get familiar with the TensorFlow environment, train and test a convolutional neural network using different hyper-parameters and comparing its performance on GPU and CPU.

Results

Validation accuracy in relation to learning rate

In the following graph, the validation accuracy for each epoch is plotted for different learning rates, while keeping the other parameters constant. Each learning rate was running for 20 epochs and was using a batch size of 50. The used learning rates are: 0.1, 0.01, 0.001, 0.0001.

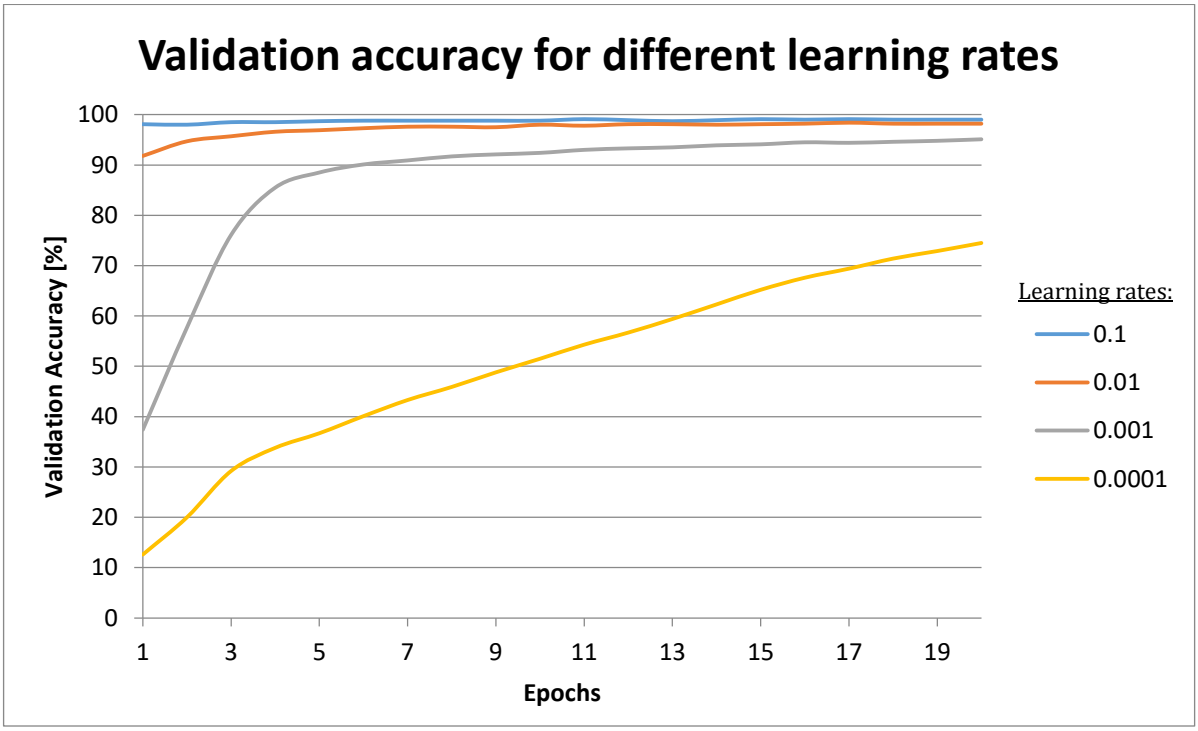

Figure 1: Validation accuracy for different learning rates.

GPU and CPU performance comparison

The performance of the convolutional neural network is tested for different filter sizes on both CPU and GPU is shown in the following table. The CNN was using a batch size of 50 and a learning rate of 0.1 for 20 epochs.

Filter size	GPU runtime [s]	CPU runtime [s]	Number of parameters
8	57	701	52258
16	70	916	104250
32	89	1677	211690
64	143	4304	440384
128	258	-	953098
256	572	-	2199690

Table 1: CPU and GPU runtime for given filter size.

In the following graph, the run time is plotted in correlation to the number of parameters for both CPU and GPU.

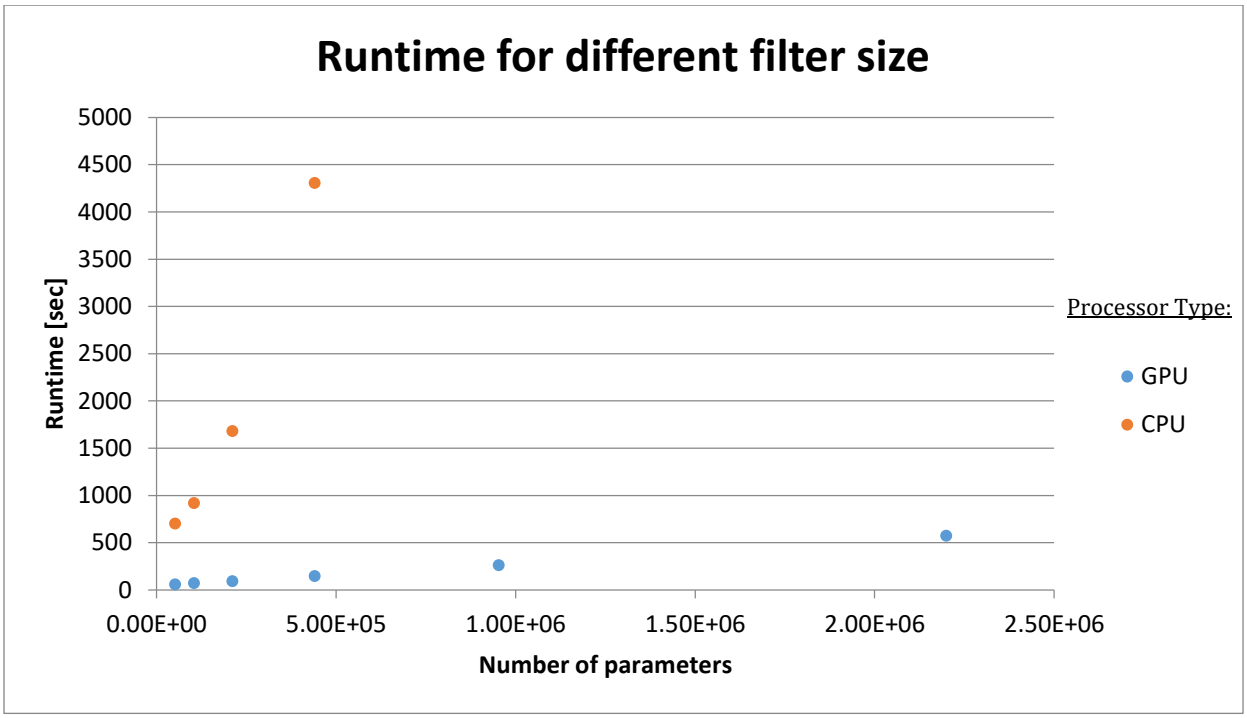

Figure 2: Runtime for given filter sizes for GPU and CPU

Discussion:

As seen in Figure 1, we observe that the learning rates of 0.1, 0.01, and 0.001 converge to nearly 100% validation accuracy. In addition, given enough epochs, the learning rate of 0.0001 will converge as well.
These results are suitable to the fact that a big learning rate might over shoot and not converge to a local minimum, while a small learning rate might take a long time to converge.

From Figure 2, one can see that, as expected, the GPU has a vastly better runtime compare to the CPU for the same number of parameters. In addition, a big filter size has a higher computational runtime compare to a smaller filter size.