Deep Learning Lab – Assingment 2

Guilherme Alessandri Alvim Miotto – 4653503

November 13, 2018

This short report describes the results obtained on the second assignment of the Deep Learning Lab course. In this assignment, we had to implement a Convolutional Neural Network (CNN) using Tensorflow and experiment with different hyper-parameter combinations. First we performed two manual searches; one for learning rate and another for filter size. Afterwards, we did an automated random search for multiple hyper-parameters simultaneously. The problem at hand is classification of handwritten digits (MNIST dataset). The following sections show the results obtained.

1 Baseline network

The network we used was comprised of six main layers. Two convolutionals followed by MaxPoolings and two fully connected layers. Table 1 describes the chain of layers used in the network. The inputs of convolutional layers are padded (padding=SAME). And the input of the first fully connected layer is flattened.

#	Type	Size	Stride	Activation
1	Convol	3x3 filter	1x1	ReLU
2	MaxPool	2x2 pool	2x2	—
3	Convol	3x3 filter	1x1	ReLU
4	MaxPool	2x2 pool	2x2	—
5	Dense	128 units	—	ReLU
6	Dense	10 units	—	Softmax

Table 1: Layers of the baseline network

The optimizer used was Stochastic Gradient Descent with learning rate of 0.001. Training was done in batches of 128 samples for a total of 12 epochs. Using this setup, accuracy on the validation dataset was 89,0%. Other results are show on Table 2.

Metric	Value
Training loss	0.444
Training accuracy	87.16%
Validation accuracy	89.02%

Table 2: Baseline network results

2 Manual search

We performed two independent searches for hyper-parameters. The first one for learning rate, where the the following values were tested : 0.1, 0.01, 0.001 and 0.0001. Figure 1 shows the evolution of the validation accuracy along the epochs for each learning rate. In this example, the highest learning rate (0.1) gave the best results (validation accuracy of 98,48%). However this result should be interpreted with care; high learning rates may harm convergence or even cause divergence. From the figure we can also see that too small learning rates may result in a very slow convergence, being 12 epochs not enough to get a good validation performance.

After experimenting with the learning rates, we did the same for filter size. The following sizes were tested: 1, 3, 5 and 7. All filters are squares. Figure 2 shows the evolution of the validation accuracy along the epochs for each filter size. We can observe that a filter of 5x5 was the one that gave best results (validation accuracy of 90,82%).

Manually searching for hyper-parameters is valuable in order to gain a better "feeling" of the network at hand and of deep learning in general. Nevertheless, this can be a very tedious and time consuming activity. In this context, an automatic hyper-parameter optimization tool is very desirable.

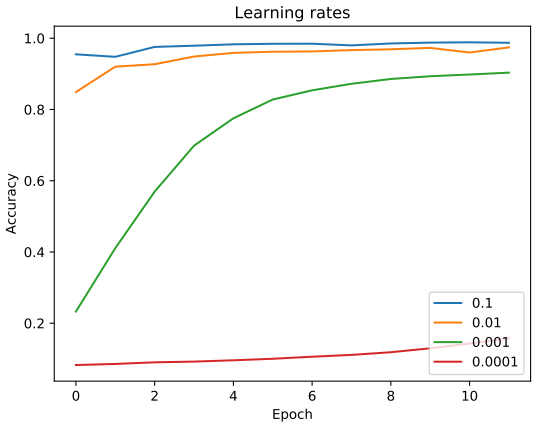

Figure 1: Validation accuracy evolution along the epochs for different learning rates

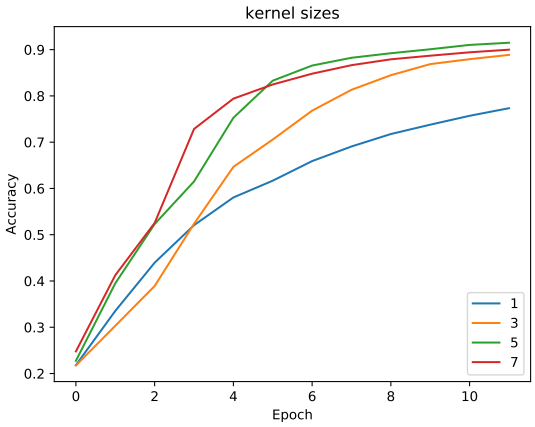

Figure 2: Validation accuracy evolution along the epochs for different filter sizes

3 Automatic random search

We used the package HpBandSter to randomly search for good combinations of hyperparameters. Table 3 show which hyper-parameters were included in the search. Search length was 50 iterations using a budget of 6 epochs per function evaluation.

Hyperpar.	Interval	Sampling
Learning rate	$[10^{-4}, 10^{-1}]$	Logarithmic
Batch size	$[16, 128]$	Logarithmic
Num of filters	$[2^3, 2^6]$	Logarithmic
Filter size	$[3, 5]$	Uniform

Table 3: Hyper-parameters included in the random search

The search results are shown on Figure 3 and Table 4. The figure plots the validation accuracy of the best model obtained at each iteration. Parameters and performance of the overall best model are shown on the table. The best model achieved test accuracy of 98.9%, which substantialy higher than the validation accuracy of the baseline model.

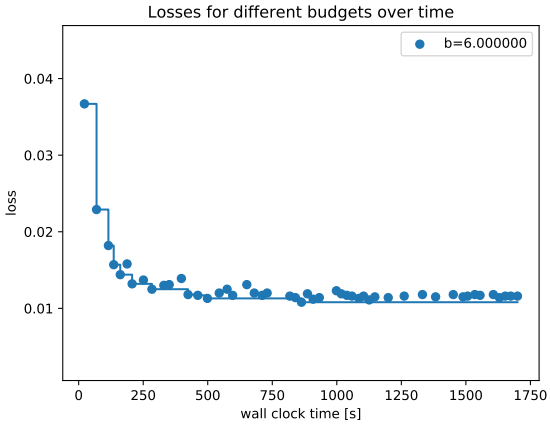

Figure 3: Validation performance of the best found configuration after each random search iteration

Metric	Value
Learning rate	0.037
Batch size	73
Number of filters	15
Filter size	5
Training loss	0.0056
Training accuracy	99.86%
Test accuracy	**98.88%**

Table 4: Parameters and results of the best configuration found by the random search