Deep Learning Lab 2018

Srdjan Milojevic

Exercise 2

1 Overview of the task

The main goal of this exercise is implementing a scaled-down version of LeNet and testing various hyperparameters using Hpbandster. With a successful completion of this exercise a student is expected to be familiar with Tensorflow and its functionalities for implementing a neural network. Also, a student should understand how to set up and conduct an experiment.

2 Implementing a CNN in Tensorflow

Following the instructions, the author implements a convolutional neural network and tests its performance using the default parameters. Because it was previously experimentally concluded that a dropout regularization technique successfully reduces overfittng (in exercise 1), it is also implemented in the created CNN. On Figure 1 we can see the validation performance.

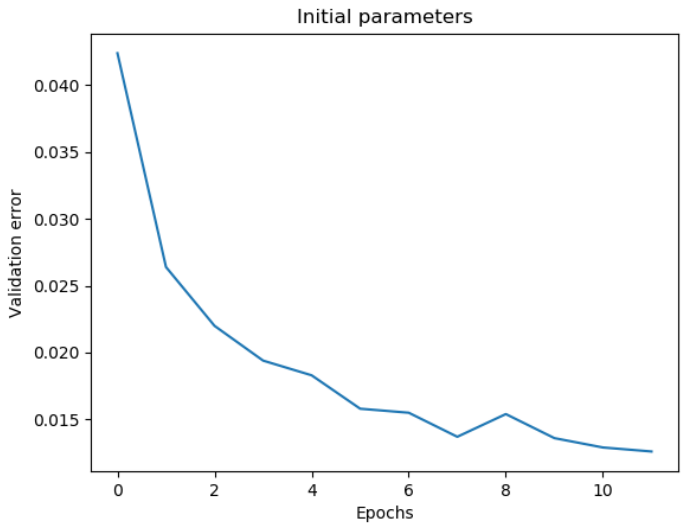

Figure 1: Learning curves of the realized neural network.

3 Results and Conclusion

On the Figure 2 the effects of the learning rate to the CNN are shown. It is visible that bigger values of learning rates give worse results because the optimizer finds a local minimum and the big learning step hinders the network from finding a better solution.

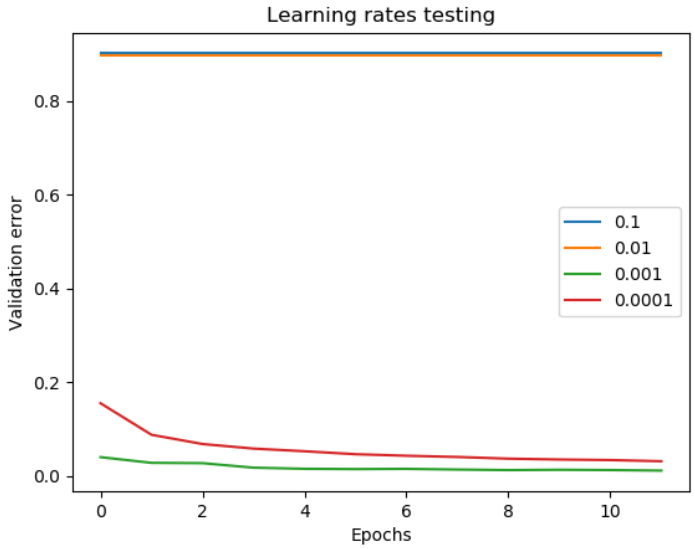

Figure 2: Learning curves of the CNN with different learning rates.

On the other hand, choosing the size of filter depends on the type of key features in the dataset. A smaller one focuses more on the details, but could also provide too much information. Contrary to that, a bigger kernel size could overlook important features. Looking at Figure 3 it is clear that in the current setup 1x1 kernel leads to confusion due to more information.

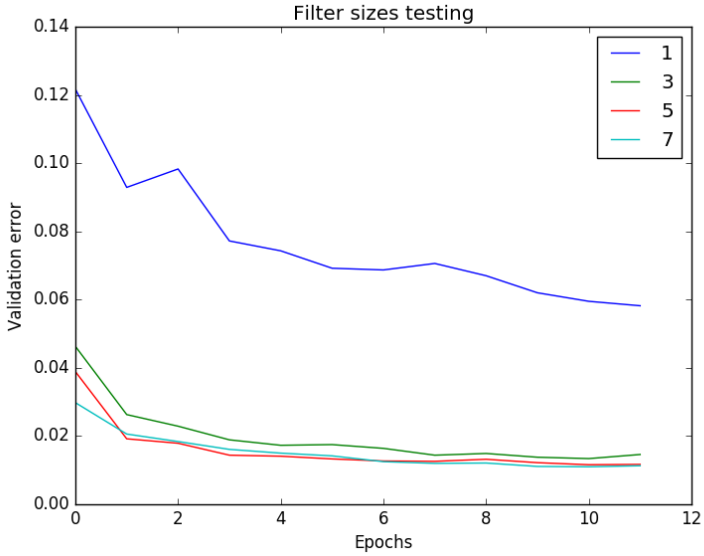

Figure 3: Learning curves of the CNN with various filter sizes.

Finally, using the adapted *random_search.py* the author fine-tunes the hyperparameters. This is achieved by creating a configuration space with the ranges of values for each parameter. Afterwards, an object of class *MyWorker* runs and waits for incoming configurations to evaluate. At the end, the user gets information about the optimizer run e.g. statistics about the best configuration. Retraining the CNN with the determined optimal parameters it reaches a test accuracy of 0.9898% in only 6 training epochs. Additionally, on Figure 4 an insight to the progress of random search is visualized.

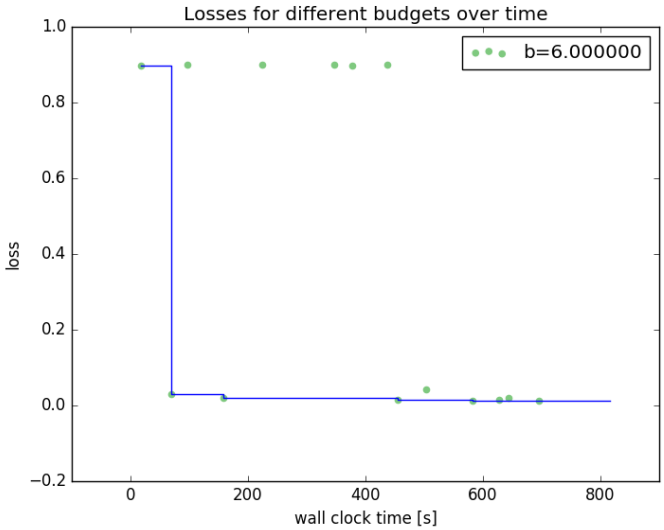

Figure 4: Progress of random search.

The values returned by the random search optimizer are:

```
{
 "batch\_size": 106,
 "num\_filters": 17,
 "filter\_size": 5,
 "lr": 0.0024045739453498866
 }
```

The final code can be found here.

Disclaimer: The author uses only the given number of epochs (12 and 6) for computational reasons.