Report – Mat. -Nr. 6614046:

Tutorial 1:

In the first part of tutorial 1 the goal is to set up the back propagation algorithm manually. Back propagation is an optimization method to teach a neural networks.
In the part Optimization there are four different optimizers to try out. The Optimizer "Stochastic Gradient Descent" achieved the best results. The results could be topped by the Momentum optimizer, if the momentum was set higher e.g. 0.99. This is because "*Momentum is like a ball rolling downhill.* T*he ball will gain momentum as it rolls down the hill.*"[1] The larger the momentum is, the faster the Optimizer will reach the the optimum, as seen in the lecture slide 94.
The last part is about regularization. Regularization helps to not overfit your neural network. Overfitting can happen when your neural network is too complex.[2] The two most common principles the "L2 Parameter norm penalty by kernel regularizer" and the Dropout regularizer can be tried out on the neural network.
In the last part, there is also Augmentation presented. Augmentation is getting more data out of the existing dataset to increase the performance of the neural network. The MNIST dataset is a huge dataset of numbers, however it can also help training such big datasets with variances of the dataset.[3]

Tutorial 2:

In the second tutorial, the learning target is to recognize that it is sometimes easier to use a pretrained model than starting from scratch. This is called transfer learning. *This is recommended, when having unstructured data, like image, video and audio data. The advantages are that you can skip extensive training and if you don´t have a big dataset to train on, your neural network can still perform very good.*[4] As seen in the tutorial it is quite simple to prepare the pretrained model and to do feature extraction. It is much faster, than building up a network from scratch and it is possible to get good results with a small dataset and only a few epochs. After four epochs, you get a accuracy of 90,35%.

Tutorial 3:

In tutorial 3 the goal is to build up a U-Net and train this to predict cell boundaries on images. The challenge is, that we only have 30 different images in our training dataset. To increase the dataset, augmentation is used. It is important to make sure that the augmentation of the image and its corresponding mask is done the same way, that the mask still fits the image.

Software Contribution: Task 2 – MNIST (February: Performance-Explainibility-Data Analysis)

The first step was to analyse the dataset. The dataset consists of 60.000 pictures of the digits 0 – 9. In the analysis it was found that there was about the same number of images of each digit in the dataset. This ensures that each digit is trained equally well, that means that there is the same variance of spellings of each digit.

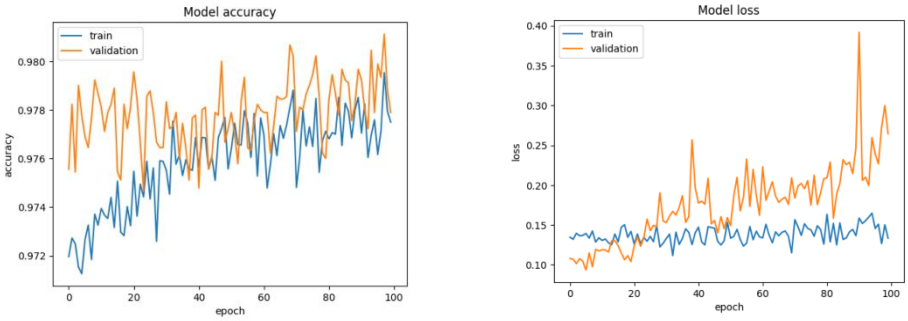
To recognize which design of the neural network and which optimizer is the best for this task, the model is tested with three different designs and three different optimizers. To make the tests comparable some parameters will be set.
The model is trained for 5 epochs, this is an arbitrarily set number. As written in the lecture notes on pages 115 a Regularization method to avoid overfitting is using batches. An optimal batch size is a Power-of-two batch size, because this "match[es] [the] physical processor and improve[s] runtime". Starting with the batch size 32, because we have used this in some tutorials before. The validation split is 15%. The validation split was chosen based on an online research. Resources recommend a split with 10%, 15% or 20%.[5] For the first comparison, the average value (15%) is used. The activation function is ReLU.

Table 1: results of the test

Optimizer / Design	SGD	Momentum (momentum=0,5)	Adam
Without regularization	validation accuracy: 96,81%	validation accuracy: 97,21%	validation accuracy: 97.52%
L2 Parameter norm penalty by kernel regularizer	validation accuracy: 93,67%	validation accuracy: 94,49%	validation accuracy: 94,53%
Dropout	validation accuracy: 95,58%	validation accuracy: 96,88%	validation accuracy: 97,48%

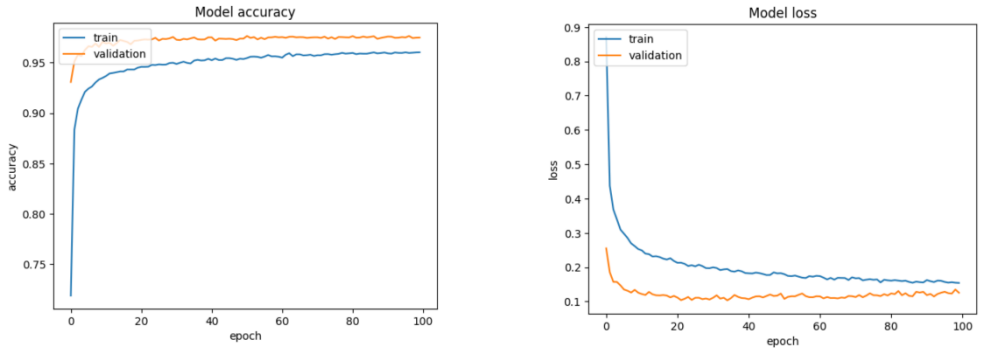
In table 1 it can be seen, that the optimizer "Adam" gets the best results in the validation accuracy. The design "Without regularization" and "Dropout" have almost the same validation accuracy with the optimizer "Adam". Regularization helps you to not overfit your neural network. Due to that I will use the design "Dropout" to not overfit my model in future improvements. In summary it can be said, that the design will be "Dropout" and the Optimizer "Adam".
In a paper from the University of Toronto[6] it was shown, that with a dropout of 0,5 a neural network achieves lower classification error rates than other feather learning methods. It was evaluated partly on the MNIST dataset. That's why the dropout of each layer is set to 0,5.
After those adjustments the model was run for 100 epochs. The results can be seen in picture 1.

Figure 1: Results of training 100 epochs



When looking at the results in figure 1, it is easy to see that the variance is too high. This is because of overfitting. The network picks up to much noise. This can be caused by a too complex network.[7] To reduce the complexity of the network, the number of units of the layers get adjusted from 512(=2^9) to 128(=2^7). This will make the network smaller and less complex.

Figure 2: Training with 100 epochs



In figure 2 it is visible, that the reduced units of the layers improved the results of the neural network. The variance reduced a lot, but is still recognizable in the waviness of the accuracy and the loss. A phenomenon visible in figure 2 is also, that the validation accuracy is higher than the training accuracy. This is due to the dropout, because "*the Dropout layer randomly sets input units to 0 with a frequency of [the given] rate at each step [only] during training time*"[8]. Due to the fact that all neurons are active while validation the neural network can perform much better in validation. Therefore, the dropout will be omitted in future improvements.

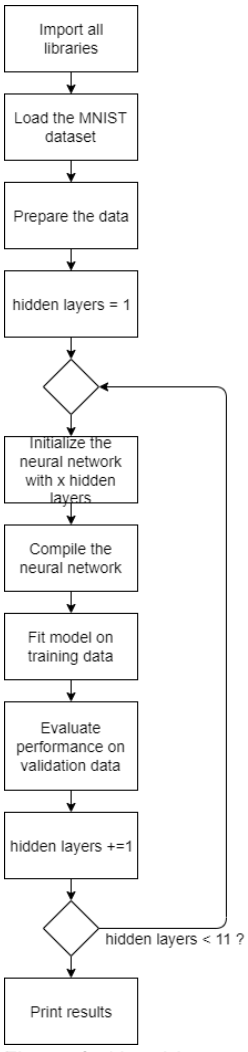


Figure 3: Algorithm to identify optimum number of hidden layers

As described above there is still some variance in the results, this means that the neural network is still too complex for the given task. To optimize the neural network it is important to find the optimum number of hidden layers for it. The algorithm visualized in the diagram on the left (figure 3) varies the number of hidden layers and prints the results for accuracy and loss of each variant. Each layer still has 128 nodes and the ReLU-Function as the activation function. The optimizer of the model is "Adam" and the model is trained for 10 epochs and validated afterwards.

The results of the algorithm can be seen in figure 4. It is quickly visible that with an increasing number of hidden layers the accuracy decreases because the model is getting to complex for the given task. The best accuracy for training has the model with two hidden layers. The same realization can be made in the diagram for the loss of the different number of hidden layers.

In the curve of the model accuracy of the validation data for different number of hidden layers, there is a maximum with five layers. This is an outlier, due to an overfitted model.

To conclude, the optimized model has two hidden layers and its design is without regularization. The optimizer is "Adam". The model accuracy of the training data is above 99%, when the model is trained for 10 epochs. To finish of this research, the model is tested on the test dataset. The accuracy of the test dataset is 97.72%.

In future projects it would be possible to analyse the activation

functions and to implement augmentation. This was due to a limited time not possible.

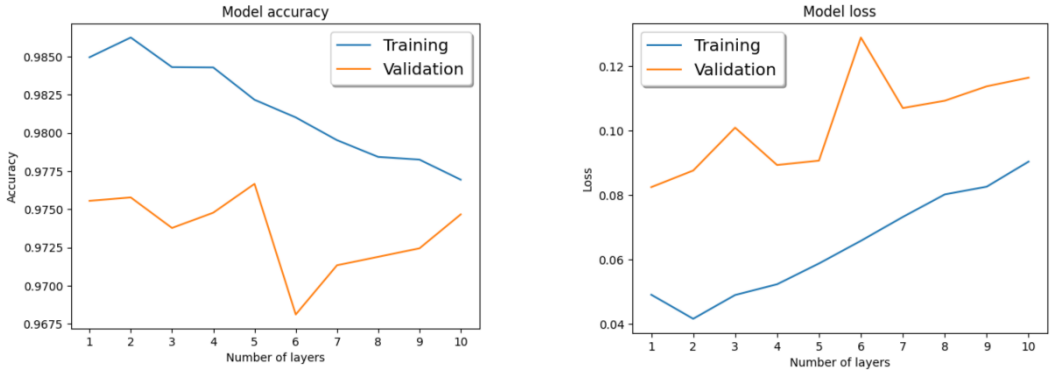


Figure 4: Results of the algorithm

Sources:

[1]: https://medium.com/@vinodhb95/momentum-optimizer-6023aa445e18, 03.12.2022
[2]: https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036, 03.12.2022
[3]: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/, 03.12.2022
[4]: https://datasolut.com/was-ist-transfer-learning/, 04.12.2022
[5]: https://www.v7labs.com/blog/train-validation-test-set, 04.12.2022
[6]: Ba, Lei Jimmy; Frey, Brendan; Adaptive dropout for training deep neural networks; Department of Electrical and Computer Engineering University of Toronto; 2013
[7]: https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036, 04.12.2022
[8]: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout, 09.12.2022