Gerwin de Kruijf	s1063465
Dirren van Vlijmen	s1009852
Olivier Brahma	s1061745

Exercises Week 5: Deep Learning

Plots can be found in the folder ‘plots’. Do not view these plots in dark-mode.
Logs of training for exercise 2 and exercise 4 can be found in ‘logs’.

1.

Default parameters of the three networks

- Optimizer:	Adam
- Loss:	Sparse Categorical CrossEntropy
- Metrics:	Accuracy
- Epochs:	200
- Learning rate:	1e-3

We used the sparse categorical cross entropy because Tensorflow states: Use this cross entropy loss function when there are two or more label classes, which is the case using the MNIST dataset.

Discussion:

0 hidden layers (model_0)
For 0 hidden layers (model_0), we have tried epochs 20,50,100,200 and learning rates 1e-{1,2,3,4,5}. For 200 epochs and a learning rate of 1e-3, the final test accuracy was: **0.2969**. Early stopping would be beneficial since the model is largely overfitting due to the number of epochs. The plot of the learning curve can be found in *ex_1_hiddenlayers_0.png*.
Total params of model_0: 30730.

1 hidden layer (model_1)
For 1 hidden layer (model_1), the ideal learning rate was 1e-4. The final test accuracy was: **0.5141**.
The model largely overfits, since the training accuracy increases while the validation accuracy slightly decreases. Early stopping would therefore be convenient.
The plot of the learning curve can be found in *ex_1_hiddenlayers_1.png*.
Total params of the model: 616610.
We've also tried a learning rate of 1e-3 with epochs of 400 but the training accuracy did not go above 0.6, while the validation accuracy could not get above 0.5.

2 hidden layers (model_2)
For 2 hidden layers (model_2), the learning rate was 1e-4 and epochs were 200.
The plot can be found in *ex_1_hiddenlayers_2.png*.
Total params of the model: 937460.
Early stopping would be very convenient since the validation accuracy decreases rapidly, due to overfitting of the model.
Test accuracy = **0.4962**.
Best validation accuracy = **0.5333**.

2.

Parameters of the network:

- Hidden units:	10000
- Optimizer:	Adam
- Learning rate:	1e-3
- Loss:	Sparse Categorical CrossEntropy
- Metrics:	Accuracy
- Epochs:	1000
- Model:	model_1_large

Model	**Train accuracy**	**Test accuracy**	**Learning rate**	**Params**
Paper	1.0	0.5051	1e-2	-
model_1_large	0.9140	0.5630	1e-4	30830010

Table 1: Comparison of our model with the one layer (512) MLP from the paper: https://arxiv.org/pdf/1611.03530.pdf.

Our model ran for 1000 epochs with a single hidden layer of 10000 units. After 2 hours, it only elapsed 68 epochs, where we stopped the model. Therefore, the results shown in table 1 are from 68 epochs. The complete log can be found in *ex_2_log.log*.

The highest validation accuracy is higher in comparison with the baseline model. However, the model is largely overfitting on the data given that the training accuracy improves while the validation accuracy reaches its maximum around 0.56 and cannot improve on this. The results show that a larger hidden layer improves on this problem.

3.

Activations used: ReLU, Tanh
Optimizers used: Adam, Adagrad, SGD

Activation function	**Test loss**	**Test accuracy**
ReLU	0.8949	0.7075
Tanh	0.9759	0.6762

Table 2: Comparison of different activation functions used in the CNN. Params of the CNN: 122,570.

Plot for the activation function ‘Tanh’ can be found in *ex_3_tanh.png*.
The reason why using Tanh performs worse is because MNIST does not consist of negative data, so ReLU does not do anything because all the values are positive. This means that ReLU makes sure that the network keeps all the information while TanH changes these outputs, which shows that the original information is very important in classifying the digits. The lack of doing anything in this case benefits the network.

Optimizer	**Test loss**	**Test accuracy**
Adam	0.8949	0.7075
Adagrad	1.4249	0.4919
SGD	1.1328	0.6089

Table 3: Comparison of different optimizers for the CNN.

Plots can be found in *ex_3_adagrad.png* and *ex_3_sgd.png*. The reason why Adagrad performs worse than Adam is because Adagrad is based on each parameter having its own learning rate and due to the peculiarities of the algorithm the learning rate is monotonically decreasing, which causes the model to stop learning after a certain time because the learning rate is very low. Besides, Adagrad is mainly used for sparse datasets, which is not the case when working with 60000 data samples.
The reason why SGD performs worse than Adam could be because of its momentum component. When the model becomes close to the goal, momentum is often very high and does not slow down which could cause the model to miss the minima.

4.

Parameters of both networks:

- Epochs: 10
- Learning rate: 1e-3
- Loss: Sparse categorical_crossentropy
- Optimizer: Adam

Model	**Test loss**	**Test accuracy**	**Params**
CNN	0.8949	0.7075	122570
MLP	1.7072	0.3809	123950

Table 4: Comparison of the standard CNN and our MLP model (model_ex_4). The MLP architecture is listed here:

```
model_ex_4 = models.Sequential ([
  layers.Flatten(),
  layers.Dense(40, activation='relu'),
  layers.Dense(20, activation='relu'),
  layers.Dense(10, activation='softmax')   ])
```

A plot of the training of the MLP can be found in *ex_4_MLP_10epochs.png*.
We also trained the MLP for 100 epochs with a learning rate of 1e-4. The results can be found in table 5.

Model	**Test loss**	**Test accuracy**	**Params**	**Learning rate**	**Epochs**
CNN	0.8949	0.7075	122570	1e-3	10
MLP	1.7072	0.3809	123950	1e-3	10
MLP	1.4717	0.4787	123950	1e-4	100

Table 5: Comparison of the standard CNN with our MLP model for two different runs.

A plot of the second run of the MLP model can be found in *ex_4_MLP_100epochs.png*.
The complete training can be found in *ex_4_log.log*.

The CNN clearly outperforms the MLP. This is because the MLP includes too many parameters leading to redundancy and overfitting. CNN is better for image classification due to less parameters, whereas the MLP is fully connected making the network unseemingly large.