**Intro to deep learning report**

For original model we obtained accuracy of **96,8 %** and a test loss (Categorical cross-entropy used) of **31,8**. The first task was to increase the size and depth of the inner layers and examine the resulting models' accuracies. The following table represents the results we got for the inner layer sizes tested. Overlooking epochs and batch sizes (as in the tasks we were instructed to experiment with different epochs and batch sizes across all variations), all other hyperparameters were kept the same as in the original models.

| Model | Inner Layer Sizes | Batch Size | Epochs | Test Accuracy | Test Loss |
|---|---|---|---|---|---|
| 1 | [256, 128] | 32 | 2 | 0.947 | 43.32 |
| 2 | [256, 128, 64] | 32 | 2 | 0.967 | 23.45 |
| 3 | [256, 128] | 32 | 5 | 0.870 | 205.84 |
| 4 | [256, 128, 64] | 32 | 5 | 0.929 | 96.01 |
| 5 | [256, 128] | 32 | 10 | 0.909 | 187.53 |
| 6 | [256, 128, 64] | 32 | 10 | 0.945 | 145.31 |
| 7 | [256, 128] | 64 | 2 | 0.957 | 31.06 |
| 8 | [256, 128, 64] | 64 | 2 | 0.972 | 19.16 |
| 9 | [256, 128] | 64 | 5 | 0.965 | 32.62 |
| 10 | [256, 128, 64] | 64 | 5 | 0.950 | 59.15 |
| 11 | [256, 128] | 64 | 10 | 0.942 | 86.74 |
| 12 | [256, 128, 64] | 64 | 10 | 0.825 | 435.09 |

From the values we can see that based on accuracy the models performed quite well. Epochs seem to have significant impact on the model's performance as for same inner layer structure we get very different loss values and accuracies. In the 12$^{th}$ model the model is already most likely overfitting as the test loss is so high and the accuracy has clearly decreased. Additionally, with this hyperparameter tuning we already obtain a better accuracy and test loss values with model (8$^{th}$ model).

Next, we examine the impacts of changing the activation function in the inner layers. The activation function used for the original model was relu. In this part we tested the model with following activation functions: softmax, sigmoid, tanh. The following table represents the results we got from this test.

| Model | Activation | Batch Size | Epochs | Test Accuracy | Test Loss |
|---|---|---|---|---|---|
| 1 | Softmax | 32 | 2 | 0.662 | 0.78 |
| 2 | Sigmoid | 32 | 2 | 0.946 | 0.17 |
| 3 | Tanh | 32 | 2 | 0.915 | 0.30 |
| 4 | Softmax | 64 | 2 | 0.577 | 1.09 |
| 5 | Sigmoid | 64 | 2 | 0.950 | 0.16 |
| 6 | Tanh | 64 | 2 | 0.945 | 0.17 |
| 7 | Softmax | 32 | 5 | 0.721 | 1.02 |
| 8 | Sigmoid | 32 | 5 | 0.944 | 0.19 |
| 9 | Tanh | 32 | 5 | 0.862 | 0.52 |
| 10 | Softmax | 64 | 5 | 0.588 | 1.63 |
| 11 | Sigmoid | 64 | 5 | 0.938 | 0.19 |
| 12 | Tanh | 64 | 5 | 0.946 | 0.20 |
| 13 | Softmax | 32 | 10 | 0.684 | 2.60 |
| 14 | Sigmoid | 32 | 10 | 0.948 | 0.22 |
| 15 | Tanh | 32 | 10 | 0.883 | 0.59 |
| 16 | Softmax | 64 | 10 | 0.337 | 3.92 |
| 17 | Sigmoid | 64 | 10 | 0.930 | 0.25 |
| 18 | Tanh | 64 | 10 | 0.953 | 0.18 |

Changing relu to other activation functions clearly had a large impact on the test loss; the values are significantly lower than. Softmax has really low accuracies which most likely caused by the fact that softmax produces floating probability predictions and accuracy does not correspond best to the performance of softmax predictions; test loss value does indicate that the model should be performing relatively well. As we get a lot better test loss values with this experiment, we can conclude that we should consider using some other activation function in the inner layers than relu.

In the next part we examine the combined affect of changing the inner layer sizes and depths with different activation functions. The activation functions and the inner layer sizes and depths are the same as in the previous parts, we just combine these tests. The results can be seen in the following table:

| Model | Inner Layer Sizes | Activation | Batch Size | Epochs | Test Accur | Test Loss |
|---|---|---|---|---|---|---|
| 1 | [256, 128] | Softmax | 32 | 2 | 0.768 | 0.58 |
| 2 | [256, 128] | Sigmoid | 32 | 2 | 0.970 | 0.09 |
| 3 | [256, 128] | Tanh | 32 | 2 | 0.844 | 0.55 |
| 4 | [256, 128, 64] | Softmax | 32 | 2 | 0.310 | 1.43 |
| 5 | [256, 128, 64] | Sigmoid | 32 | 2 | 0.940 | 0.19 |
| 6 | [256, 128, 64] | Tanh | 32 | 2 | 0.902 | 0.35 |
| 7 | [256, 128] | Softmax | 64 | 2 | 0.405 | 1.22 |
| 8 | [256, 128] | Sigmoid | 64 | 2 | 0.966 | 0.11 |
| 9 | [256, 128] | Tanh | 64 | 2 | 0.951 | 0.17 |
| 10 | [256, 128, 64] | Softmax | 64 | 2 | 0.310 | 1.46 |
| 11 | [256, 128, 64] | Sigmoid | 64 | 2 | 0.972 | 0.09 |
| 12 | [256, 128, 64] | Tanh | 64 | 2 | 0.820 | 0.68 |
| 13 | [256, 128] | Softmax | 32 | 5 | 0.712 | 0.99 |
| 14 | [256, 128] | Sigmoid | 32 | 5 | 0.93 | 0.25 |
| 15 | [256, 128] | Tanh | 32 | 5 | 0.691 | 1.95 |
| 16 | [256, 128, 64] | Softmax | 32 | 5 | 0.756 | 1.12 |
| 17 | [256, 128, 64] | Sigmoid | 32 | 5 | 0.955 | 0.16 |
| 18 | [256, 128, 64] | Tanh | 32 | 5 | 0.913 | 0.32 |
| 19 | [256, 128] | Softmax | 64 | 5 | 0.735 | 0.67 |
| 20 | [256, 128] | Sigmoid | 64 | 5 | 0.976 | 0.08 |
| 21 | [256, 128] | Tanh | 64 | 5 | 0.932 | 0.25 |
| 22 | [256, 128, 64] | Softmax | 64 | 5 | 0.489 | 1.17 |
| 23 | [256, 128, 64] | Sigmoid | 64 | 5 | 0.956 | 0.15 |
| 24 | [256, 128, 64] | Tanh | 64 | 5 | 0.9 | 0.37 |
| 25 | [256, 128] | Softmax | 32 | 10 | 0.449 | 4.42 |
| 26 | [256, 128] | Sigmoid | 32 | 10 | 0.956 | 0.17 |
| 27 | [256, 128] | Tanh | 32 | 10 | 0.883 | 0.53 |
| 28 | [256, 128, 64] | Softmax | 32 | 10 | 0.538 | 2.37 |
| 29 | [256, 128, 64] | Sigmoid | 32 | 10 | 0.916 | 0.42 |
| 30 | [256, 128, 64] | Tanh | 32 | 10 | 0.843 | 0.98 |
| 31 | [256, 128] | Softmax | 64 | 10 | 0.521 | 2.59 |
| 32 | [256, 128] | Sigmoid | 64 | 10 | 0.964 | 0.13 |
| 33 | [256, 128] | Tanh | 64 | 10 | 0.922 | 0.33 |
| 34 | [256, 128, 64] | Softmax | 64 | 10 | 0.567 | 0.97 |
| 35 | [256, 128, 64] | Sigmoid | 64 | 10 | 0.961 | 0.16 |
| 36 | [256, 128, 64] | Tanh | 64 | 10 | 0.9 | 0.45 |

Again, we get very low test loss values which verifies our observations from the previous part; relu should be changed to some other activation function. We also get the highest accuracy yet, combined with very low test loss ($20^{th}$ model).

In the last hyperparameter testing part we test the impact of changing the optimizer and learning rate. The inner layer architecture is the same as in the original model as well as the activation function. The optimizer used for the original model was adam. As I wanted to test different learning rates with the adam optimizer without making the runtime too long at this point, only other optimizer tested is SGD (stochastic gradient descent). The below table represents the results we got from this test:

| Model | Optimizer | Learning Rate | Batch Size | Epochs | Test Accuracy | Test Loss |
|---|---|---|---|---|---|---|
| 1 | Adam | 0.001 | 32 | 2 | 0.9559 | 31.9789 |
| 2 | Adam | 0.01 | 32 | 2 | 0.9631 | 46.7357 |
| 3 | Adam | 0.1 | 32 | 2 | 0.1030 | 3.5978 |
| 4 | SGD | 0.001 | 32 | 2 | 0.8889 | 38.1795 |
| 5 | SGD | 0.01 | 32 | 2 | 0.9400 | 26.9257 |
| 6 | SGD | 0.1 | 32 | 2 | 0.9734 | 18.4387 |
| 7 | Adam | 0.001 | 64 | 2 | 0.9572 | 24.0497 |
| 8 | Adam | 0.01 | 64 | 2 | 0.9635 | 38.2071 |
| 9 | Adam | 0.1 | 64 | 2 | 0.1073 | 665.5694 |
| 10 | SGD | 0.001 | 64 | 2 | 0.8023 | 33.2287 |
| 11 | SGD | 0.01 | 64 | 2 | 0.9320 | 29.8881 |
| 12 | SGD | 0.1 | 64 | 2 | 0.9736 | 12.0171 |
| 13 | Adam | 0.001 | 32 | 5 | 0.9352 | 85.6303 |
| 14 | Adam | 0.01 | 32 | 5 | 0.951 | 81.0061 |
| 15 | Adam | 0.1 | 32 | 5 | 0.1368 | 11032.0732 |
| 16 | SGD | 0.001 | 32 | 5 | 0.9165 | 35.7277 |
| 17 | SGD | 0.01 | 32 | 5 | 0.9653 | 15.5511 |
| 18 | SGD | 0.1 | 32 | 5 | 0.9712 | 28.525 |
| 19 | Adam | 0.001 | 64 | 5 | 0.9694 | 25.1107 |
| 20 | Adam | 0.01 | 64 | 5 | 0.954 | 71.2262 |
| 21 | Adam | 0.1 | 64 | 5 | 0.1135 | 18277.4141 |
| 22 | SGD | 0.001 | 64 | 5 | 0.8905 | 37.7249 |
| 23 | SGD | 0.01 | 64 | 5 | 0.9588 | 18.4686 |
| 24 | SGD | 0.1 | 64 | 5 | 0.975 | 17.4872 |
| 26 | Adam | 0.01 | 32 | 10 | 0.9798 | 66.2185 |
| 27 | Adam | 0.1 | 32 | 10 | 0.101 | 19307.1191 |
| 28 | SGD | 0.001 | 32 | 10 | 0.9277 | 31.1182 |
| 29 | SGD | 0.01 | 32 | 10 | 0.9744 | 12.5546 |
| 30 | SGD | 0.1 | 32 | 10 | 0.9692 | 37.3403 |
| 31 | Adam | 0.001 | 64 | 10 | 0.9474 | 77.0032 |
| 32 | Adam | 0.01 | 64 | 10 | 0.942 | 163.7097 |
| 33 | Adam | 0.1 | 64 | 10 | 0.0979 | 11259.1504 |
| 34 | SGD | 0.001 | 64 | 10 | 0.9129 | 36.9093 |
| 35 | SGD | 0.01 | 64 | 10 | 0.9644 | 17.2982 |
| 36 | SGD | 0.1 | 64 | 10 | 0.984 | 13.4324 |

As we used relu again as the activation function, we can see that the test loss scores have increased significantly. Interestingly we get very different accuracy results from this part: the Adam optimizer performs poorly with a learning rate of 0.1, whereas SGD works very well in contrast. Based on the

result of this test we should consider using SGD as our optimizer as it seems to perform very well with all hyperparameter combinations.

Finally, we tested the impact of removing the convolution-, and max-pooling layers. The hyperparameters used were otherwise the same as in the original model; the only difference was the layer architecture. Surprisingly we got very good scores with this model; we obtained accuracy of 97,2 % with a loss value of 15,65 . The most probable reason for this is that the data is actually quite simple, and the neural network with hidden layers only is capable of capturing the patterns of the data.