IL2230 Lab 2

Linnéa Ridderström, Sofie Franzén, Daniel Saber Tehrani, Silvia Barrett

December 2021

1 Exploration

In the exploration phase, the code from the tutorial of pytorch was changed in different ways to investigate how certain factors affect the accuracy of the network. The factors that were considered were filter map size, number of convolutional layers, and the non-linear function used.

Table 1 shows the performance when changing the filter map size of layer 1 (left) and layer 3 (right), both of these where tested separately. None of the sizes had much affect on the network's performance. This is because it only runs under two epochs and therefore the number of feature maps won't increase the accuracy. If the epochs would increase we would also see an increase and bigger difference in accuracy for all combinations of feature maps.

More convolutional layers were added to the CNN system. The filter size was increased for each added convolution layer, while every other parameter like epochs was static. The result is displayed in figure 1. Noted is that, with increased number of layers, the number of epochs must be increased as the system needs more time to train to gain sufficient accuracy. This is because of the vanishing gradient problem that the added layers create. Therefore, the number of epochs were also increased, but seemingly not enough to increase the accuracy by much. In all cases, the accuracy drops with added layers.

A experiment was conducted where three different non activation functions where tested for the CNN system. The conclusion that can draw from the result displayed in figure 1 is that the the tanh and ReLU functions handle the presented task equally well while the Sigmoid suffers heavy from the gradient problem and is no better than random with it's classification accuracy of 10%. The measured classification accuracy is low overall and can be increased by increasing the number of epochs, which gives the system more time to train. For example with 25 epochs, using the Sigmoid function yields an accuracy of 42%.

If we do not increase the number of epochs, none of the factors increase the classification accuracy. Instead, the increased feature maps does not affect the classification accuracy and increased layers has the an negative effect with it's

decrease of classification accuracy. Also, choosing the wrong non linear activation function, for example sigmoid, will also have devastating results for the classification accuracy. If we increase the number of epochs the classification accuracy will probably increase for all factors.

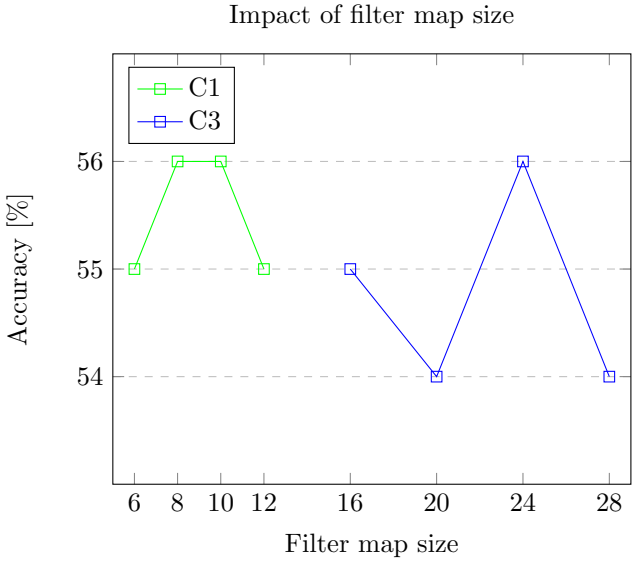

Figure 1: Accuracy as a function of the size of the filter maps for C1 and C3. When one layer was varied, the other was frozen at its baseline value. The tests were run with 2 and 10 epochs.

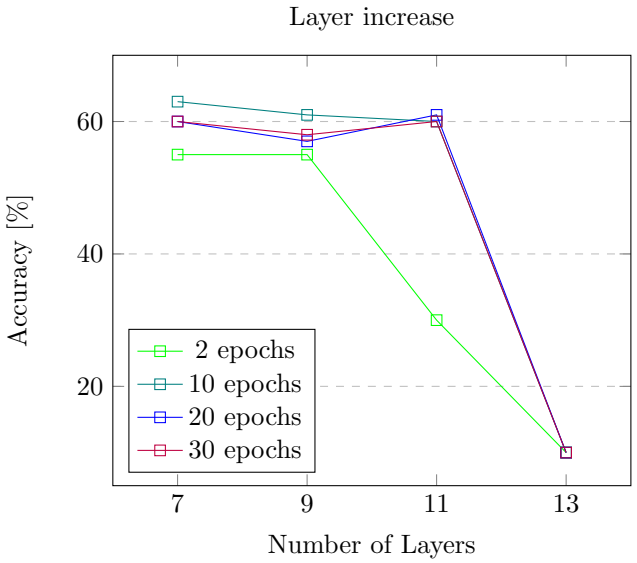

Figure 2: Accuracy as a function of the number of layers. The original filter sizes were used and two convolutional layers were added each step. All cases were tested with different amounts of epochs as well.

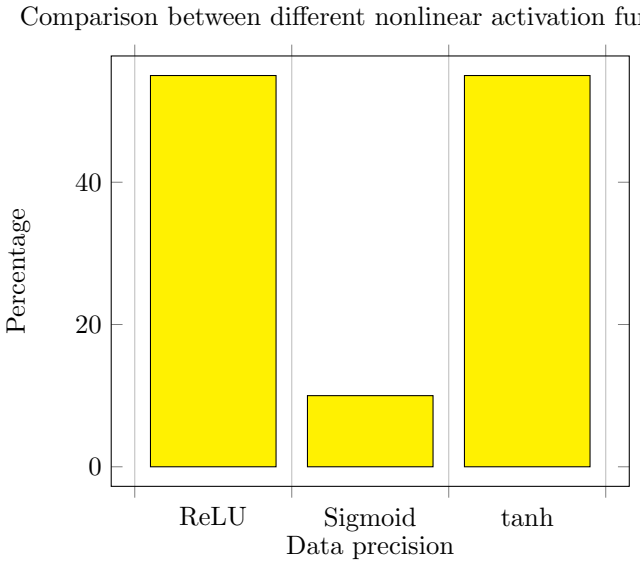

Figure 3: Accuracy for the different non-linear functions when used on the base case of 7 layers, 2 epochs and original filter sizes.

2 Handwritten digit recognition

In this part of the lab we considered an MLP with 1 hidden layer (MLP1), and an MLP with 2 hidden layers (MLP2). The number of neurons for the layers were altered and the results are presented in figure 4 and table 1. MLP1 was constructed with two linear layers, where the first one takes the 28×28 image as its input sample. MLP2 was constructed as an extension of MLP1, it has three linear layers and, as with MLP1, the first layer takes the 28×28 image as its input sample.

In figure 4 we see that the accuracy for small number of neurons, between 30 and 90, increases with the number of neurons. As the number of neurons approaches 90 we observe a change in the graph's form. From this value on there is no longer an increase in the accuracy but rather a waveform with highs and lows around 99 percents accuracy.

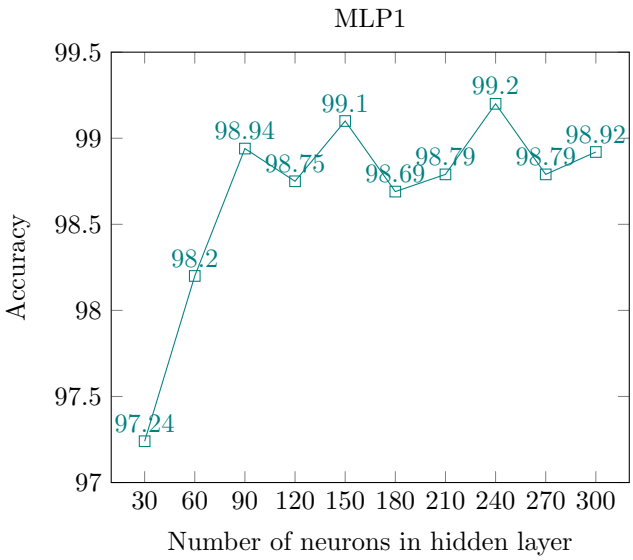

Figure 4: Accuracy of MLP1 as a function of the number of neurons in the hidden layer. 10 epochs were used during training.

Table 1: Number of neurons in the hidden layers of the MLP2 and how it affects accuracy. 10 epochs were used during training

Number of neurons in hidden layer		
1	**2**	**Accuracy**
100	100	99.04%
100	200	98.64%
100	300	99.19%
200	100	98.63%
200	200	99.10%
200	300	99.00%
300	100	99.10%
300	200	99.33%
300	300	99.41%

In table 1 we see the accuracy as a result of different values of neurons in each hidden layer. We do note that the highest accuracy, 99.41 percent, is given for 300 neurons in each of the two layers. This is greater than the highest accuracy for MLP1, which we found yo be 99.2 percent.

From this we can draw the conclusion that in order to increase the accuracy for a linear neural network we can either increase the number of neurons if the number is small to begin with, or change the structure and add more layers if the numbers of neurons is already big. The line seems to be at 90 numbers of neurons in the hidden layer.

We then consider the LeNet-5 model. The problem with LeNet-5 is that the MNIST dataset image have the dimensions 28×28, while LeNet-5 takes in the dimensions 32×32. The solution to that is to straight up transform the dimensions. After that the structure is built designed on how a LeNet-5 architecture is with convolutional layers, average pooling and linearizations.

Table 2: Performance comparison between MLP1, MLP1, and LeNet. 10 epochs were used during training.

	Classification Accuracy (%)	**Run-time (ms)**	**Memory Consumption (Kbit)**
MLP1	99.16	0.656	13.88
MLP2	99.26	0.887	47.19
LeNet-5	99.64	7.719	629.68

In table 2 we can see that LeNet has best accuracy between the three tested methods but it also has the highest run time and required memory usage. But we would still say that the increase in accuracy is more important then the saving of memory and time. Therefore according to us we think LeNet-5 is the better method.