# The Memory Capacity of Neural Networks

### **Course:** Seminar/Project Machine Learning & Neuroinformatics/Brain-Computer Interfacing (708.415)

### **Author:** Christoph Rieger

### **Student number:** 01530103

### **Date:** 04.02.2023

# Introduction

This project aims to investigate the memory capacity of neural networks depending on their size and architecture. Randomly generated input and output patterns are handed to the network to be memorized. Two different network architectures, as well as two different definitions of when a pattern is learned correctly, are analyzed.

# Methods

### Data

The input vectors x and target vectors y are binary vectors of size N. The number of these vectors is defined by the dataset size DS. Different values for DS were tried for each network, until the maximum number of memorizable patterns was determined. All input and target vectors for each DS together yield the matrices X and Y.   
The vectors x and y have a sparsity s which was chosen as 0.1. This means that 10% of the bits of x and y were active with values of one and the rest of the bits were inactive with values of zero. Both vectors were generated randomly, but no duplicate vectors with the same bits were allowed within the input and output matrix respectively.

### Network

The first network used was a 1-Layer fully connected network and will be called simple network from now on. It has N input neurons and N output neurons. The input and output layer are fully connected. The activation function used was the sigmoid function, which scales the outputs between zero and one. This behavior is desired because the target vector’s values lie between those values. For an arbitrary value a it is defined as

(1)

After the application of the activation function the network’s prediction/output z is obtained. The prediction and the target vector are then handed over to the loss function. In this experiment the network is performing a binary classification task between y and z for each single bit. Thus, Binary cross-entropy (BCE) was chosen because it is designed for such classification tasks. For a batch size of one it is given by

(2)

As optimizer stochastic gradient descent and ADAM were tested. After evaluation ADAM was chosen as optimizer for this experiment because it was converging faster than stochastic gradient descent with no apparent drawback.

The described “simple“ network was later expanded by a recurrence between the output neurons. The training process now consisted of two steps through time, while using the same input vector. The first step was identical to the first network, the input goes to the N input neurons and is passed via the fully connected layer to the N output neurons.   
After that, in the second step, the input was again given to the input neurons of the network. Additionally, the output neurons of step 1 were connected to the output neurons of step 2, creating a recurrent layer with trainable weights. These recurring connections were initialized fully connected between the two output layers, but all weights of connections between the same output neurons (e.g.: output neuron 1 of step 1 to output neuron 1 of step 2) were always kept at zero.

At last the recurrent network was expanded once more by adding a third step, which functioned in the same way as step 2, taking the original input vector as input and connecting the output neurons of step two to the output neurons of step 3. This resulted in three total steps with the same input vector.

The learning rate used for training was 0.01 for all networks except the recurrent network with 3 steps. For the recurrent network with three steps the learning rate had to be reduced to 0.001 because the loss was not converging.

### Loss and classification

In the evaluation phase each bit of the network’s prediction z had to be cast to either one or zero. This was done by setting all outputs greater or equal than 0.5 to 1 and all smaller than 0.5 to 0.

Two different definitions of when a pattern counts as correctly memorized by the network were analyzed. The first one only counted a pattern as memorized if the output vector is equal to the target vector.

To create a more lenient definition a second definition was introduced. It calculated the number of bits to ignore #bti as

. (3)

During each training step output neurons with the worst loss were disabled. This was done #bti times for the group of output neurons that had a target of 1 and another #bti times for the group that had a target of 0. The neurons were “disabled” by setting their loss to 0. The idea behind this procedure was that stochastically the input and output patterns could overlap many of their active bits, making it more difficult for the network to 100% correctly classify those overlapping patterns. The bits to ignore are supposed to counteract this stochastic increased difficulty of memorization.  
Due to this disabling of output neurons, and thus preventing them from learning, the validation paradigm also had to be adjusted. Instead of requiring all the bits of the output and target vector to be equal, #bti of the active group and #bti of the inactive group were allowed to be unequal.

### Analyzed networks

To summarize, four different networks were trained and analyzed.

1. simple network
2. simple network with the custom loss function
3. recurrent network that took two steps with the same input data and with custom loss function
4. recurrent network that took three steps with the same input data and with custom loss function

### Determination of the maximum memorizable patterns

To determine the maximum number of memorizable patterns the appropriate DS for the network had to be searched. It was not possible to simply take a DS much bigger than the maximum of memorizable patterns, because for those the network was not trainable properly and it performed poorly.  
Instead the biggest dataset a network can still memorize perfectly was searched. After finding it the DS was increased by 100 until the biggest DS was found where the network could still memorize more than 90% of the dataset.

### Calculation of mean and standard deviation of the memorized patterns

The simple network without a recurrent layer was trained five times with each set of parameters and the mean and standard deviation of the memorized patterns was calculated over those five runs. For the more complex recurrent network only three runs were performed, to speed up the training process. The maximum amount of memorized patterns used as result was taken from the best performing run.

# Results

The results of the experiment for all four networks are given in Table 1 and 2. The accuracy is calculated as maximum number of memorized patterns divided by the dataset size.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | N | | | | |
| 100 | 200 | 300 | 400 | 500 |
| Simple network | Maximum number of memorized patterns | 283 | 580 | 897 | 1194 | 1494 |
| Accuracy  [%] | 94.33 | 96.67 | 99.67 | 99.5 | 99.6 |
| Mean of memorized patterns | 243 | 550 | 883.8 | 1165.6 | 1486.8 |
| Standard deviation of memorized patterns | 23.95 | 32 | 11.95 | 23.56 | 7.93 |
| Simple network with custom loss function | Maximum number of memorized patterns | 337 | 694 | 1077 | 1580 | 2008 |
| Accuracy  [%] | 96.29 | 99.14 | 97.91 | 98.75 | 95.62 |
| Mean of memorized patterns | 305 | 674.8 | 1046 | 1395.2 | 1904 |
| Standard deviation of memorized patterns | 24 | 20.7 | 26.56 | 169 | 141.7 |

Table 1: Results for the simple network, once with basic loss and once with custom loss

Table 2: Results for the recurrent network, once with 2 steps and once with 3 steps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | N | | | |
| 50 | 100 | 150 | 200 |
| Recurrent network  (2 steps) | Maximum number of memorized patterns | 534 | 920 | 1241 | 1549 |
| Accuracy  [%] | 97.09 | 92 | 95.46 | 96.81 |
| Mean of memorized patterns | 524.8 | 902.2 | 1228 | 1539.33 |
| Standard deviation of memorized patterns | 6.18 | 12.5 | 9.27 | 7.13 |
| Recurrent network  (3 steps) | Maximum number of memorized patterns | 589 | 1219 | 1528 | 1950 |
| Accuracy  [%] | 98.17 | 93.77 | 95.5 | 92.86 |
| Mean of memorized patterns | 577.66 | 1199 | 1503 | 1929.33 |
| Standard deviation of memorized patterns | 13.3 | 14.51 | 17.72 | 14.7 |

In Figure 1 the number of memorized patterns depending on N of each network can be seen.

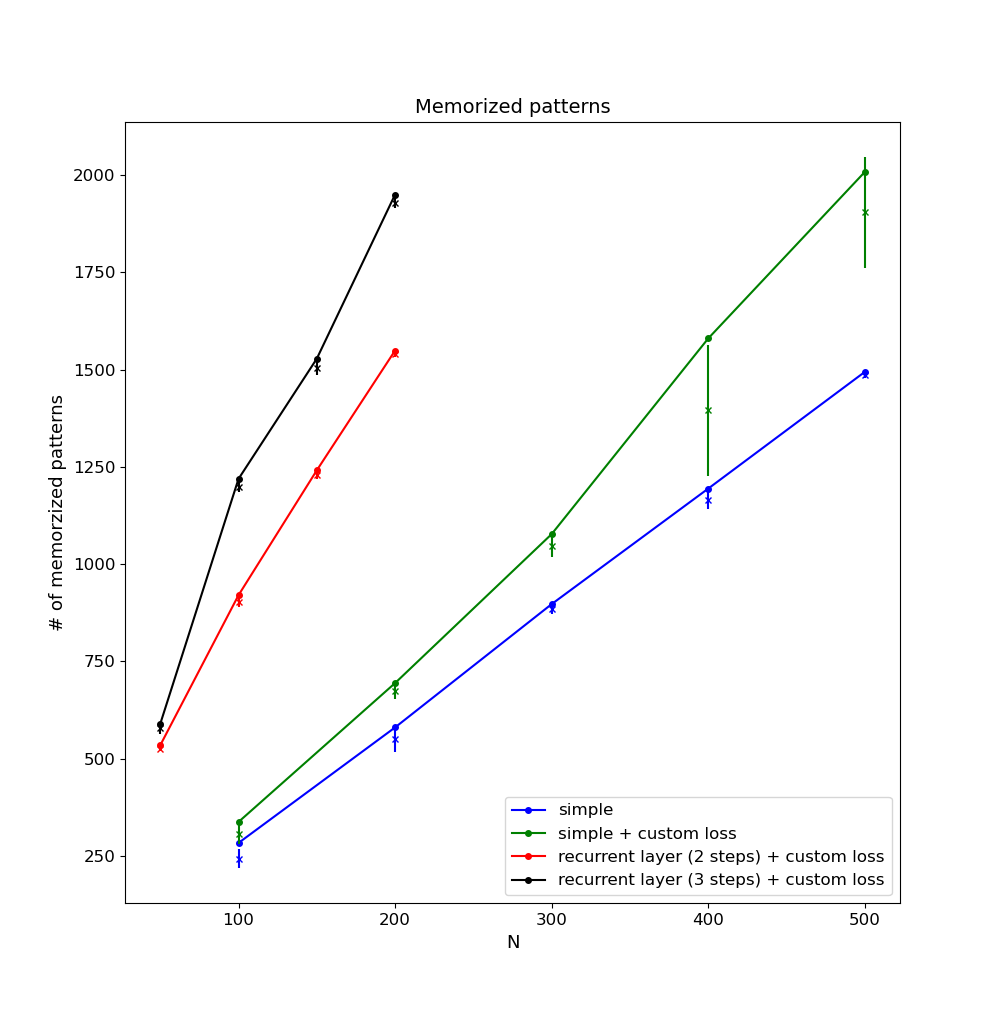


Figure 1: Number of patterns memorized of the 3 different networks. The datapoints of the graphs are the maximum number of memorized patterns of all runs. Marked with “x” is the mean of the runs with the standard deviation as a vertical line.

The number of memorized patterns per N, depending on N, for each network can be seen in Figure 2.

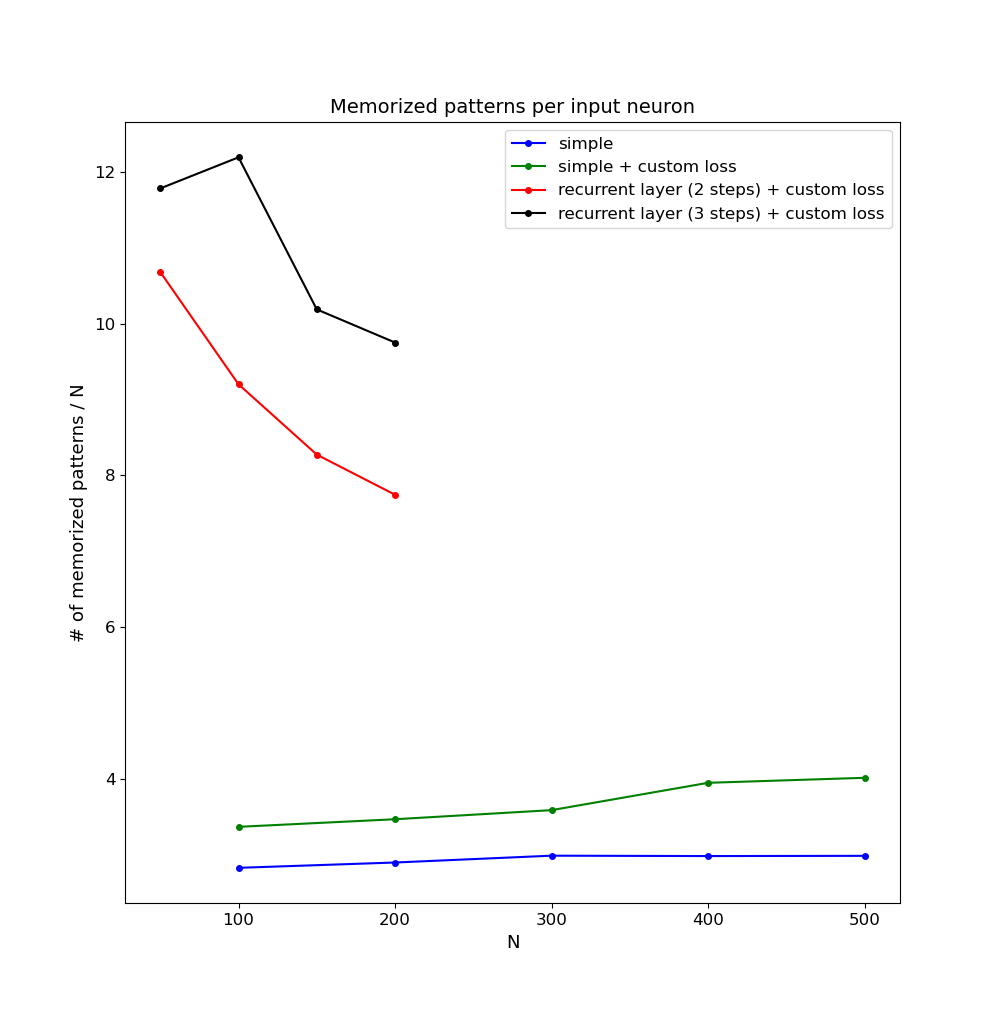


Figure 2: Number of memorized patterns per N.

The number of memorized patterns per N squared, depending on N, for each network can be seen in Figure 3.

## 

Figure 3: Number of memorized patterns per N squared.

# Discussion

In Figure 1 the simple network performed the worst. Its slope was constant, for each added neuron about three more patterns could be memorized.  
By introducing the custom loss function to the simple network it was able to memorize more patterns, the bits to ignore acting like a positive offset. Furthermore, in Figure 2 it can be seen that it was able to memorize more patterns per N, the higher N became.  
When looking at Figure 1 again the most complex network with custom loss and a recurrent layer performed the best. Its performance could be improved even further by performing more recurrence steps, or by adding additional layers. In Figure 2 the memorized patterns per N of the most complex network is larger than of the other two networks. However, it is declining with rising N, in contrary to the other two networks which performed better or the same with rising N. This behavior is most likely due to the more difficult training of the complex network. The loss of the two fully connected networks decreased with each training epoch, making the training process straight forward. In contrast the loss of the recurrent network increased and decreased many times during the training process. Because of that the recurrent network was much harder to train and that could explain the decreasing memorized patterns per N of it.

Something that must be considered, is the fact that with rising N not only did the network get more neurons, but also the input and target vectors got larger and more complex. Because of that the network was not able to memorize N2 more patterns with rising N, as one might expect. The observed behavior of the memorized patterns with regard to N2 can be seen in Figure 3. If the network could memorize N2 patterns the graphs would have a constant slope of one. This is not the case however and all graphs trend towards zero with rising N.