Alon Ben-Melech 313184178
Tal Berman 318279437

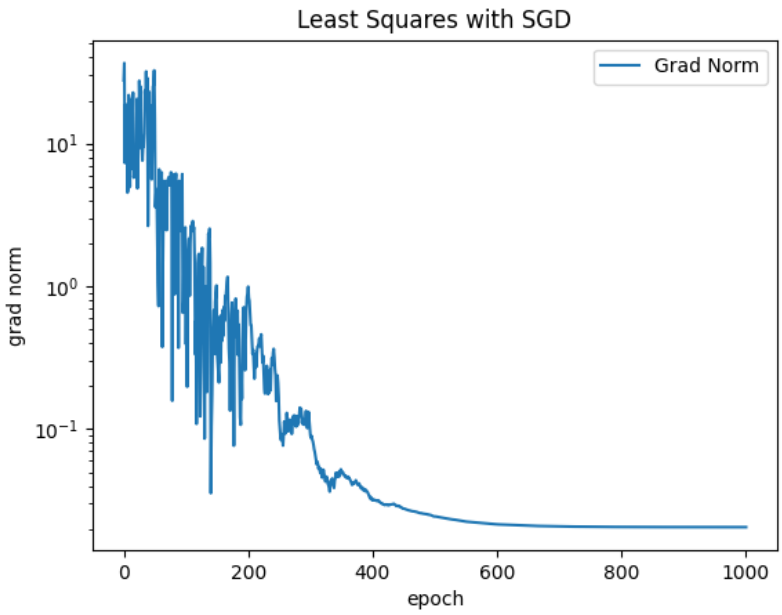
Deep Learning
Assignment 1

2.1.1

2.1.2

We chose X and y to be:

$$X = \begin{pmatrix} 3 & -1 & 1 \\ 1 & -1 & 2 \\ -1 & -3 & 5 \\ 1 & 3 & 4 \end{pmatrix} \quad y = \begin{pmatrix} 13 \\ 20 \\ 0 \\ 14 \end{pmatrix}$$



Alon Ben-Melech 313184178
Tal Berman 318279437

The result from SGD is: [5.34320562 0.50925224 1.96977776]
The result from solving by the normal equations is [5.34172662 0.50719424 1.97122302].

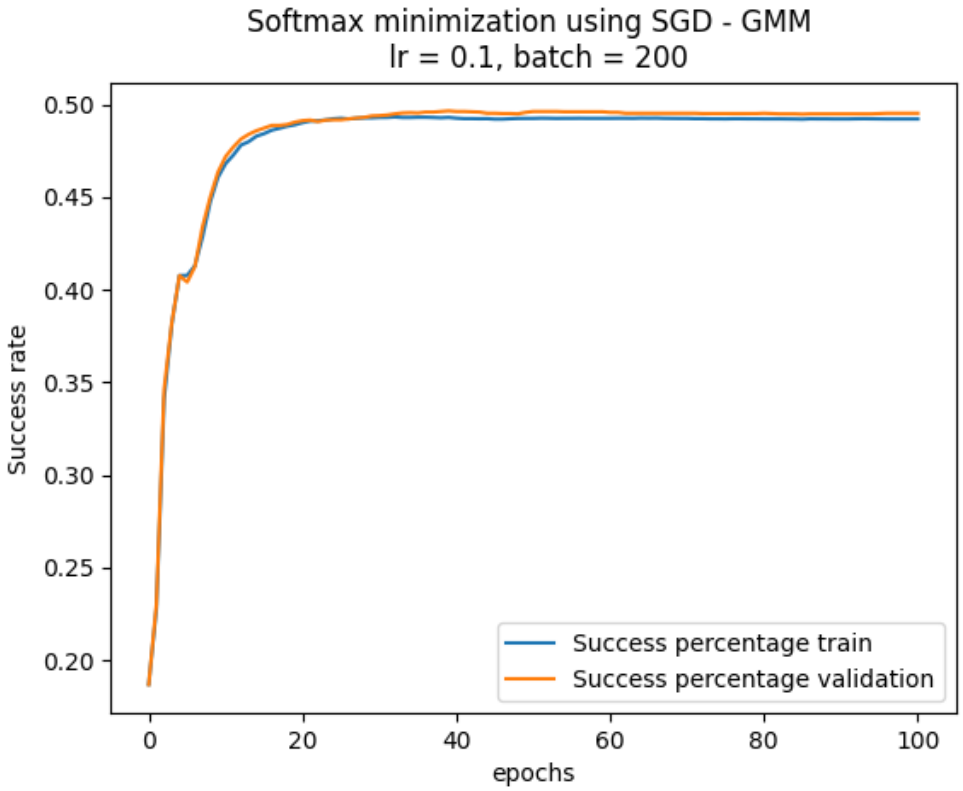
We can see that we got a very close approximation from the SGD algorithm.

2.1.3

GMM:

batch	Learning rate	Training success	Validation success
200	0.1 -after each 50 epochs multiply by 0.1	0.4922	0.49536
20	""	0.49039	0.49376
200	0.7	0.4865	0.48848
20	0.7	0.48848	0.49039

The best result for GMM:

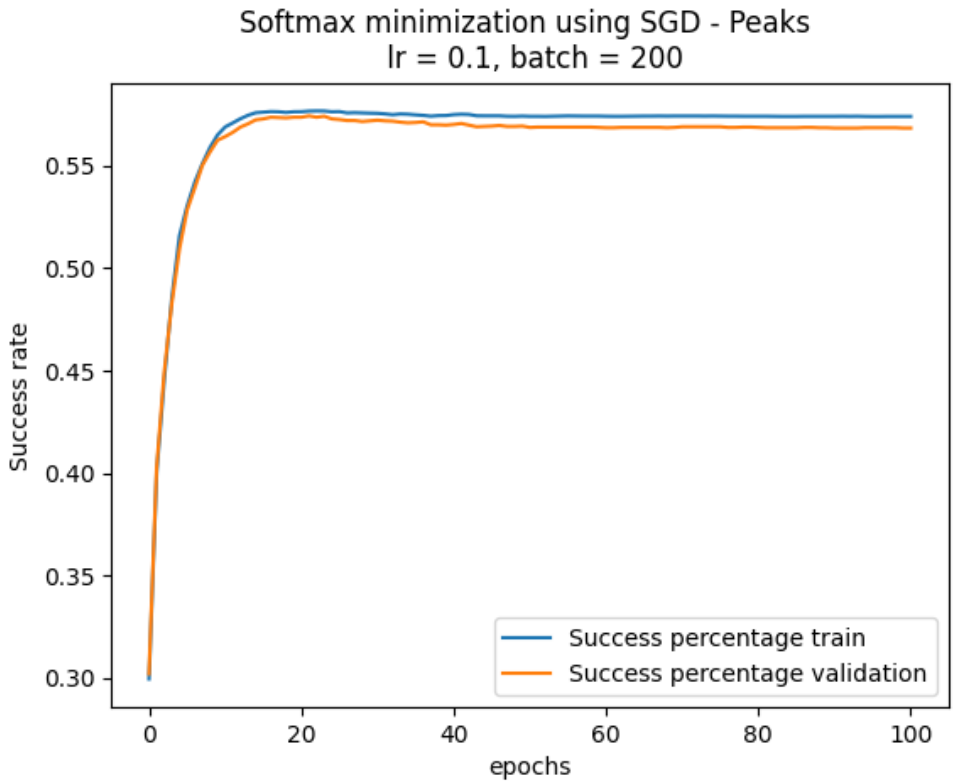


Alon Ben-Melech 313184178
Tal Berman 318279437

Peaks:

batch	Learning rate	Training success	Validation success
200	0.1 -after each 50 epochs multiply by 0.1	0.574	0.5683
20	""	0.57376	0.56704
200	0.7	0.53312	0.52944
20	0.7	0.50104	0.50096

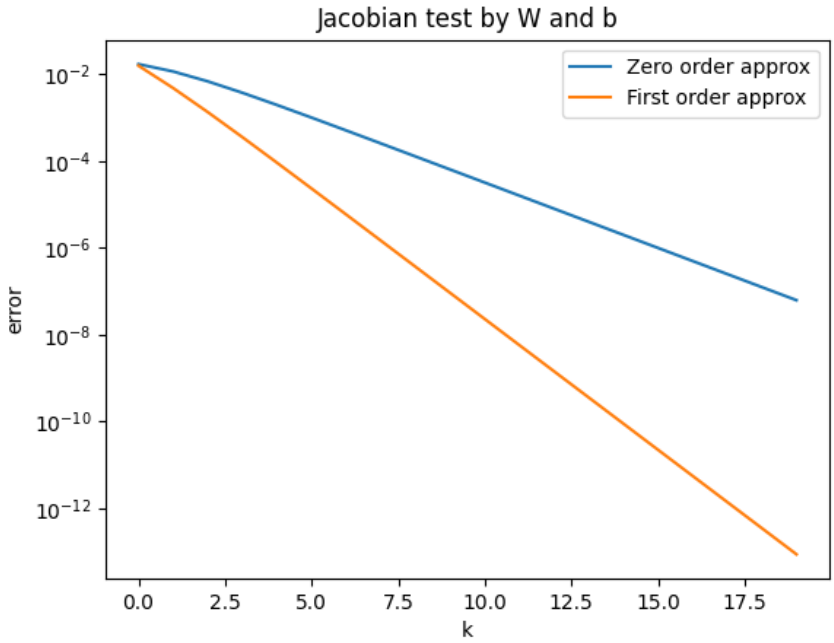
The best result for Peaks:

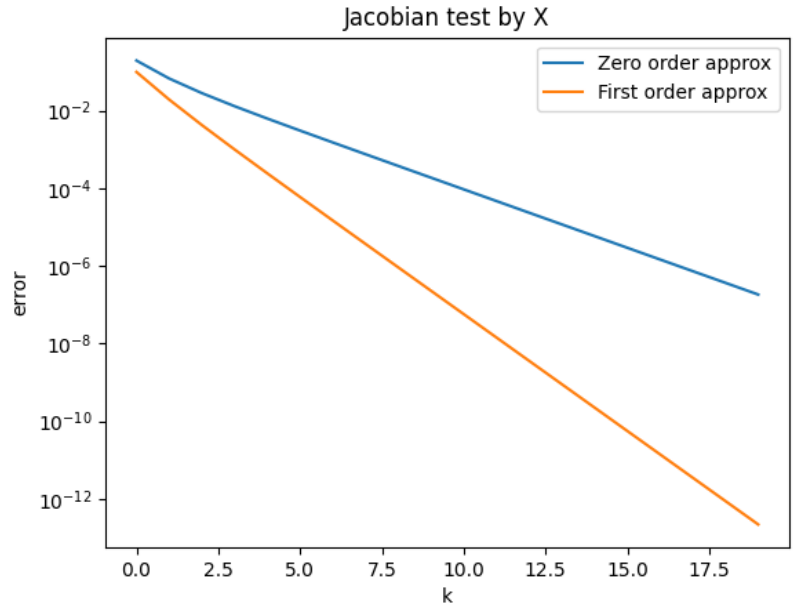


2.2.1

The Jacobian tests for tanh() activation function:

Alon Ben-Melech 313184178
Tal Berman 318279437

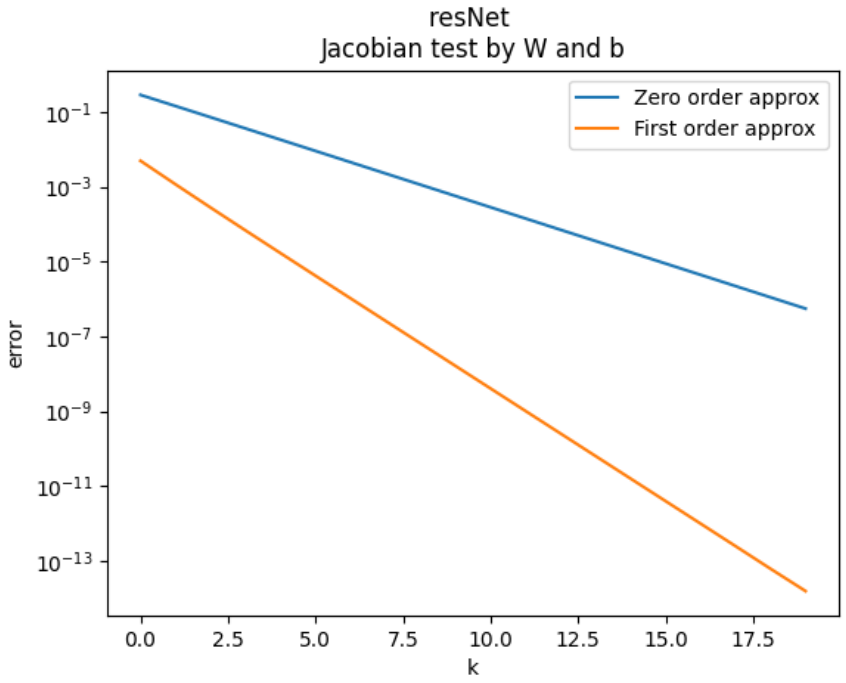


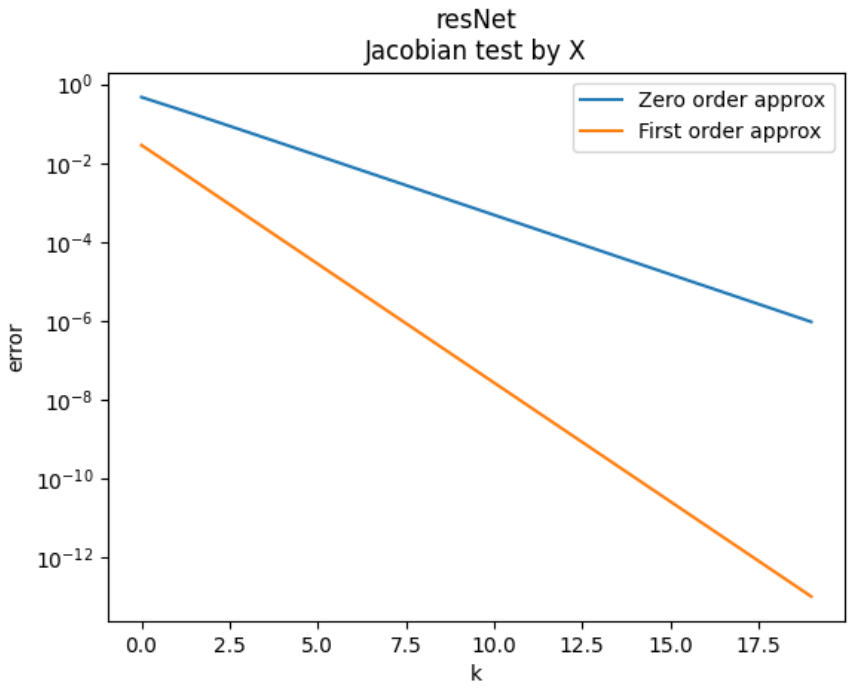


2.2.2

The Jacobian tests for tanh() activation function:

Alon Ben-Melech 313184178
Tal Berman 318279437

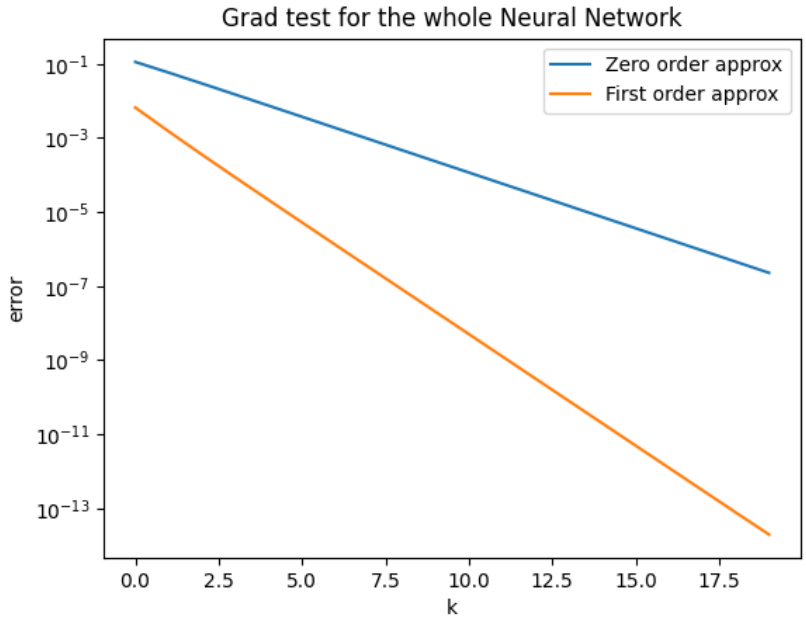


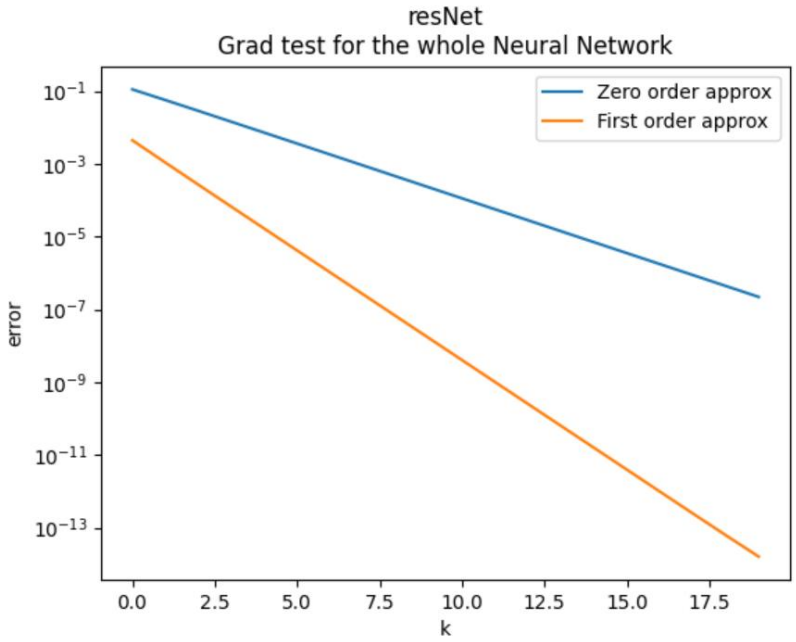


2.2.3

The gradient test for the whole network:

Alon Ben-Melech 313184178
Tal Berman 318279437





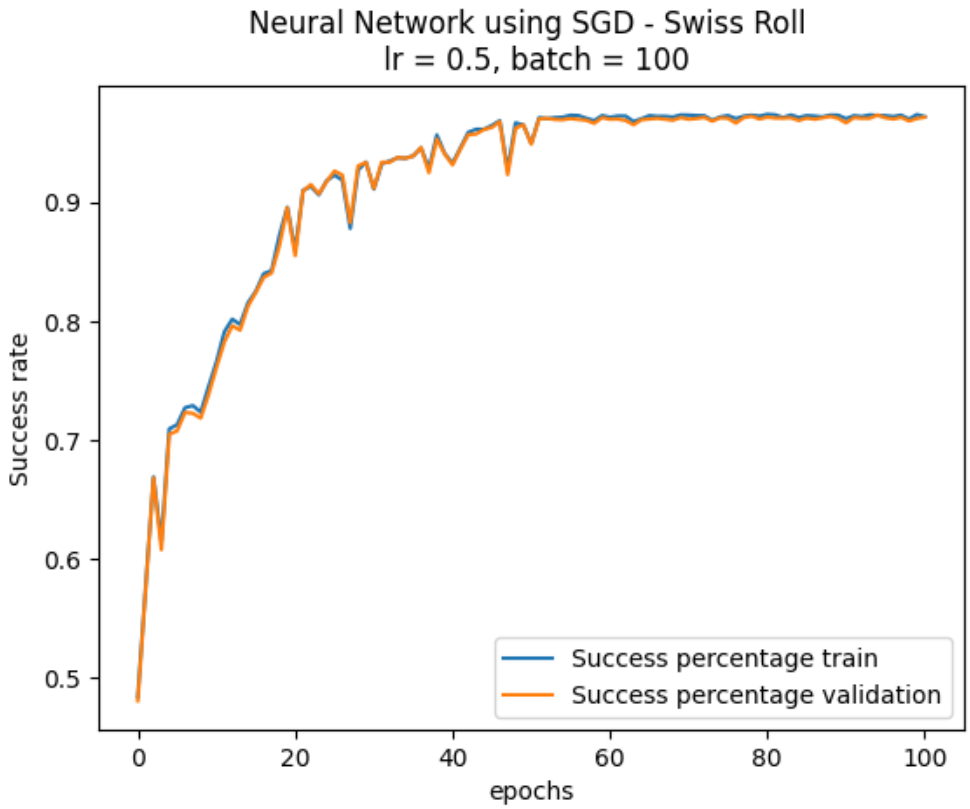
2.2.4:

Neural network

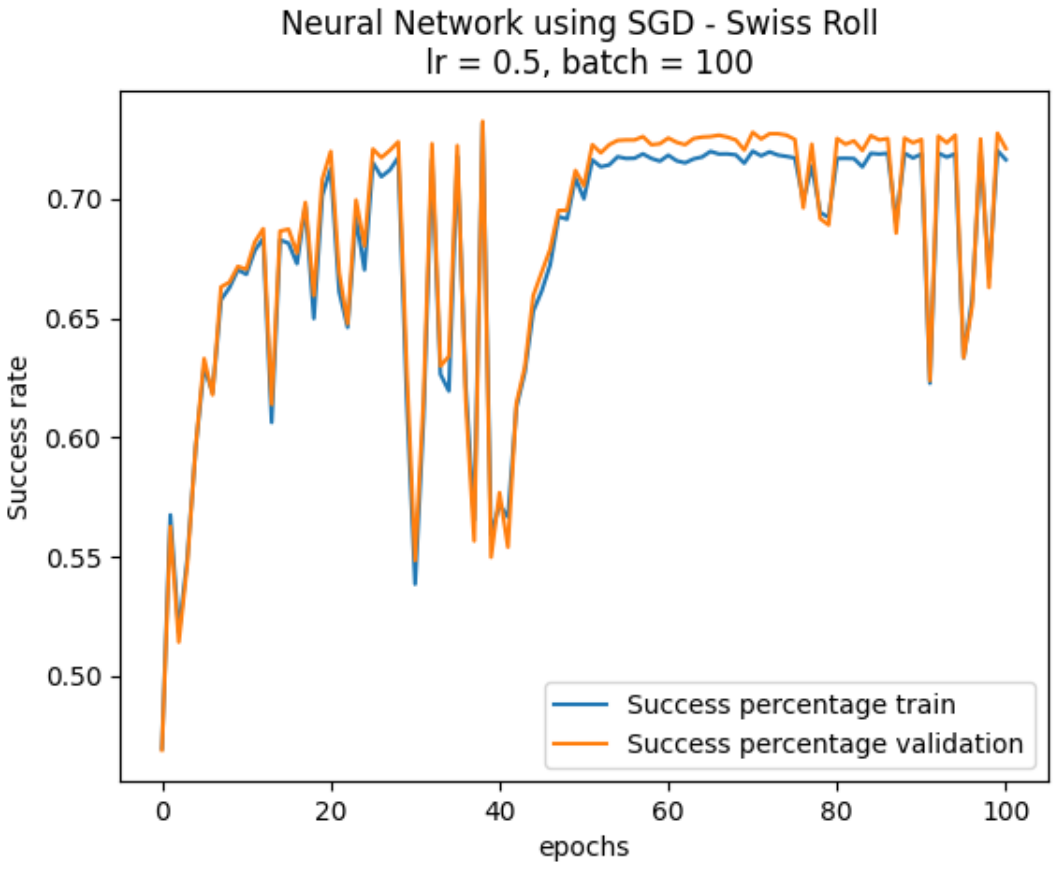
Swiss Roll:

We got our best results with hidden layers of dimensions [10,10,4], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

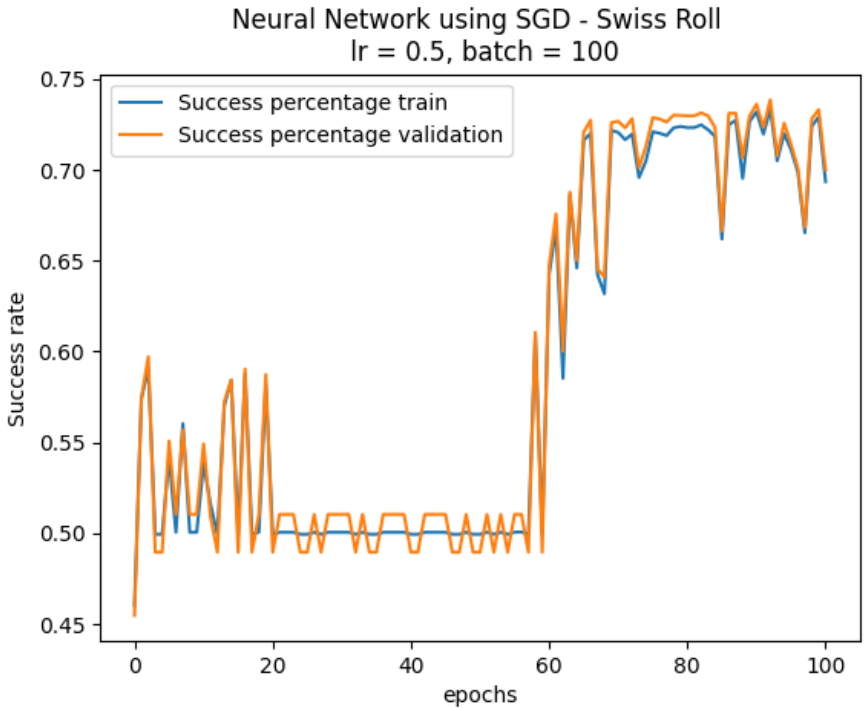


hidden layers of dimensions [3,5,4], learning rate: 0.5, batch: 100:

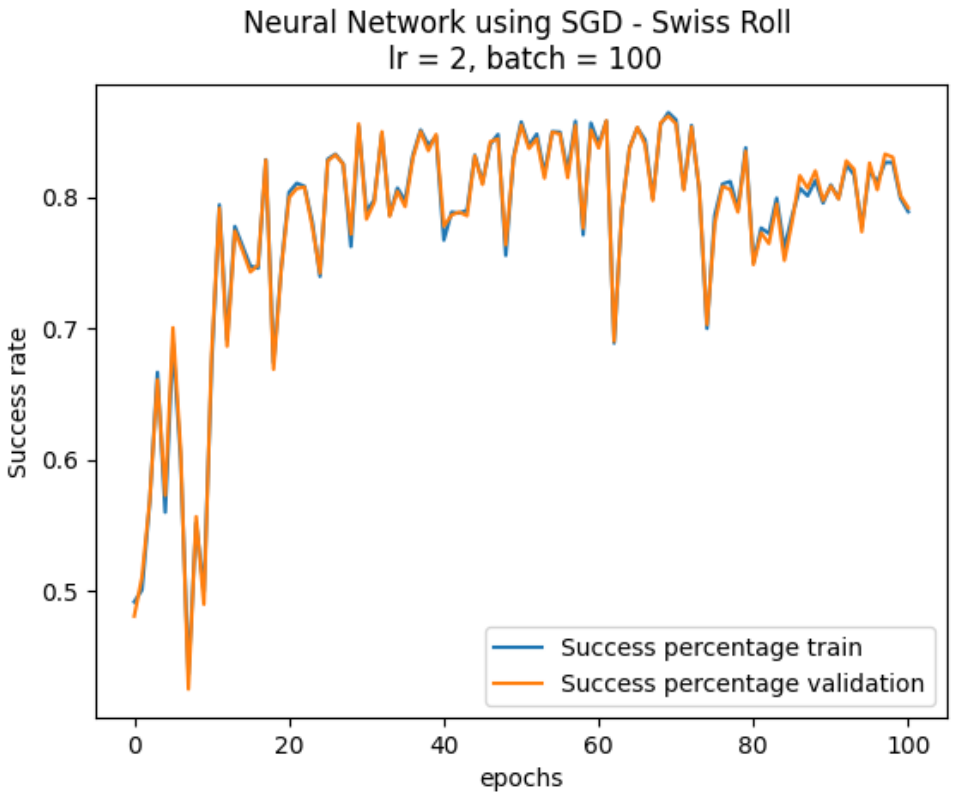


Alon Ben-Melech 313184178
Tal Berman 318279437

hidden layers of dimensions [7,10,10,10,10,10,10,4], learning rate: 0.5, batch: 100:

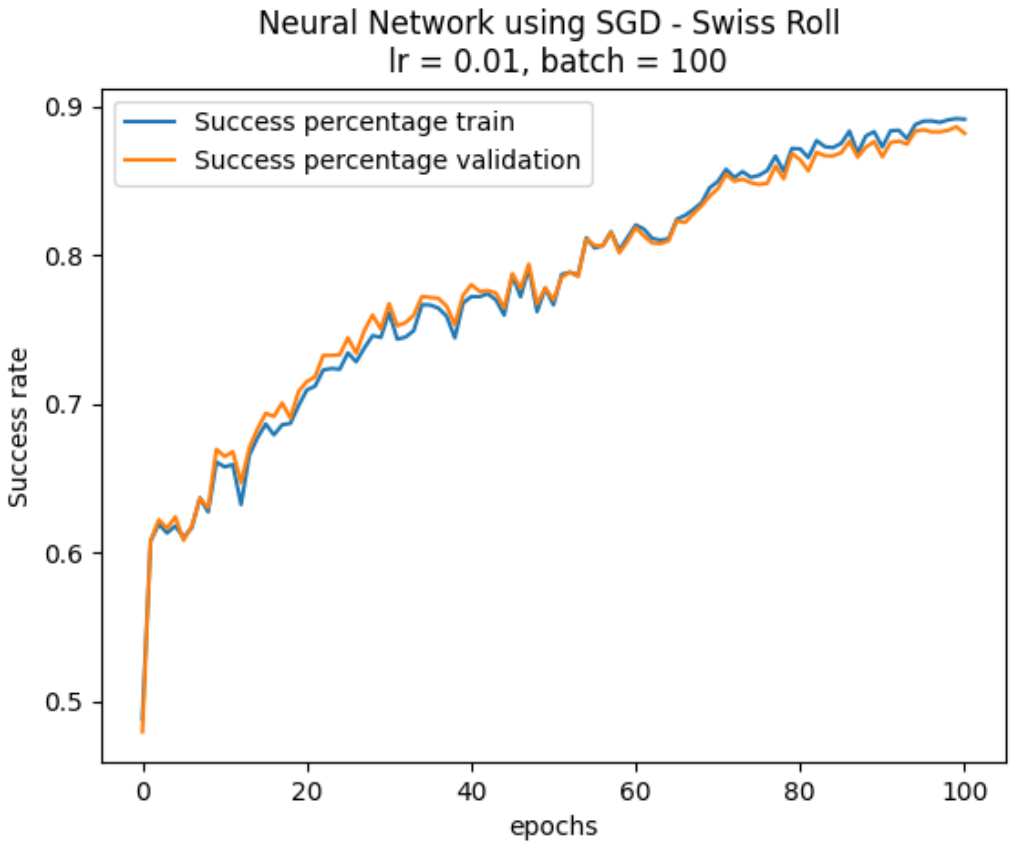


hidden layers of dimensions [7,10 ,4], learning rate: 2, batch: 100:

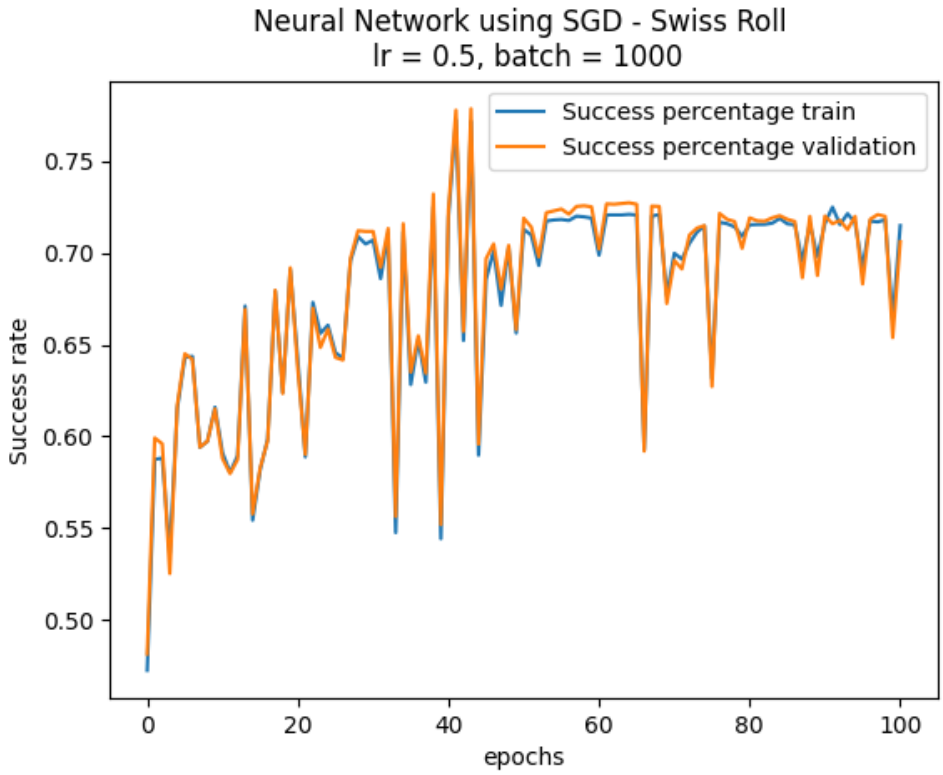


Alon Ben-Melech 313184178
Tal Berman 318279437

hidden layers of dimensions [7,10 ,4], learning rate: 0.01, batch: 100:



hidden layers of dimensions [7,10 ,4], learning rate: 0.5, batch: 1000:



Alon Ben-Melech 313184178
Tal Berman 318279437

Conclusion:

We can see that using too many layers can lead to poor results. A big number of parameters needs a lot of data and epochs to train. We can assume more data and epochs could improve the results for this architecture.

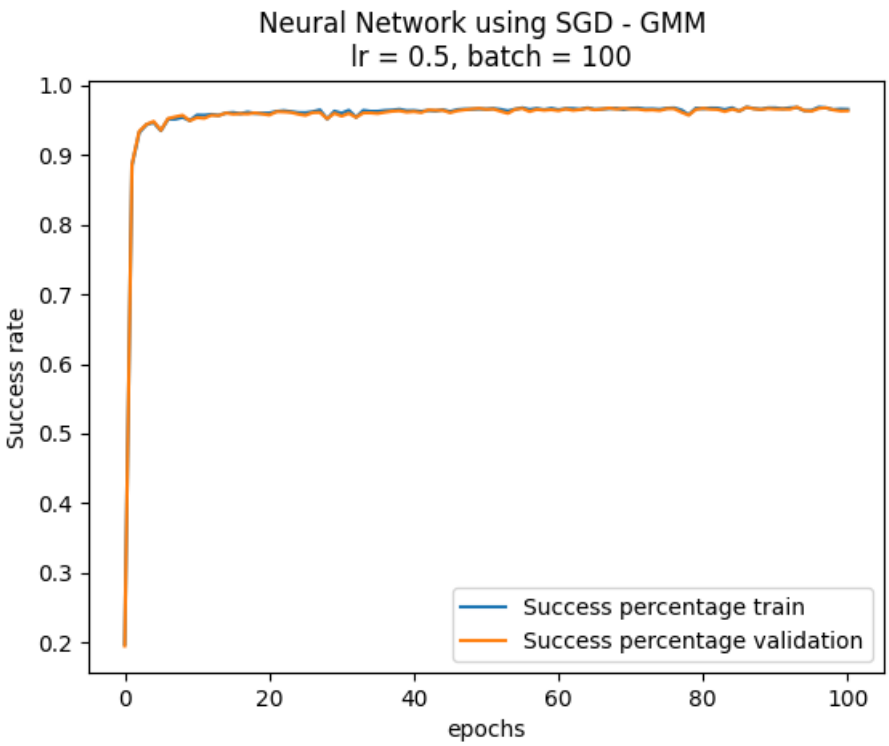
Reducing the dimension by setting the first hidden layer size to 3, got awful results.

As for the learning rate, using a too small learning rate caused slower learning, though the final result is good.
using a too large learning rate caused worst results, because the big jumps make it hard to find a good minimum.

Setting a bigger batch size, led to bad results.

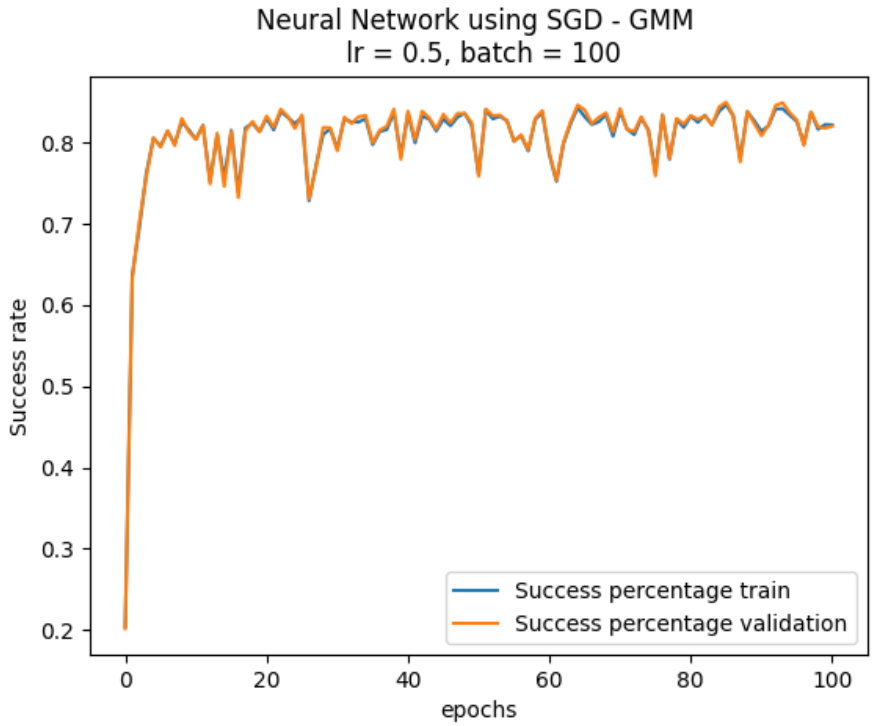
GMM:

We got our best results with hidden layers of dimensions [10,10,4], learning rate: 0.5, batch: 100:

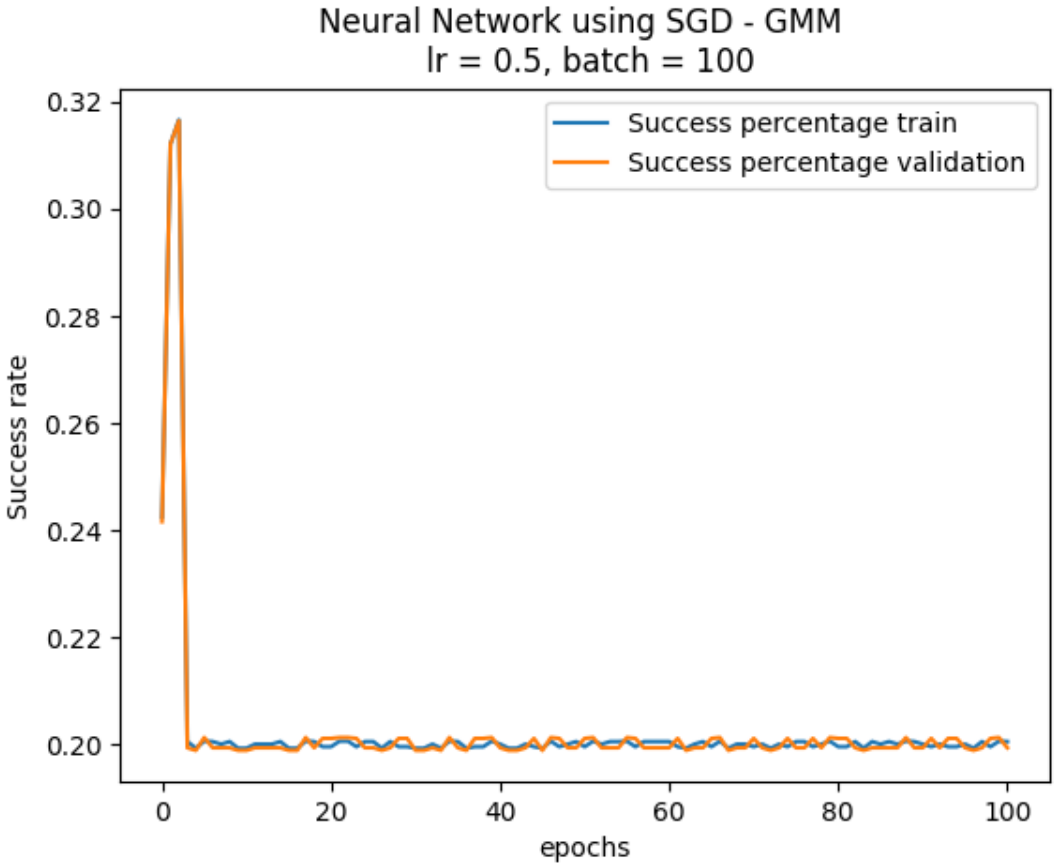


hidden layers of dimensions [3,5,4], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

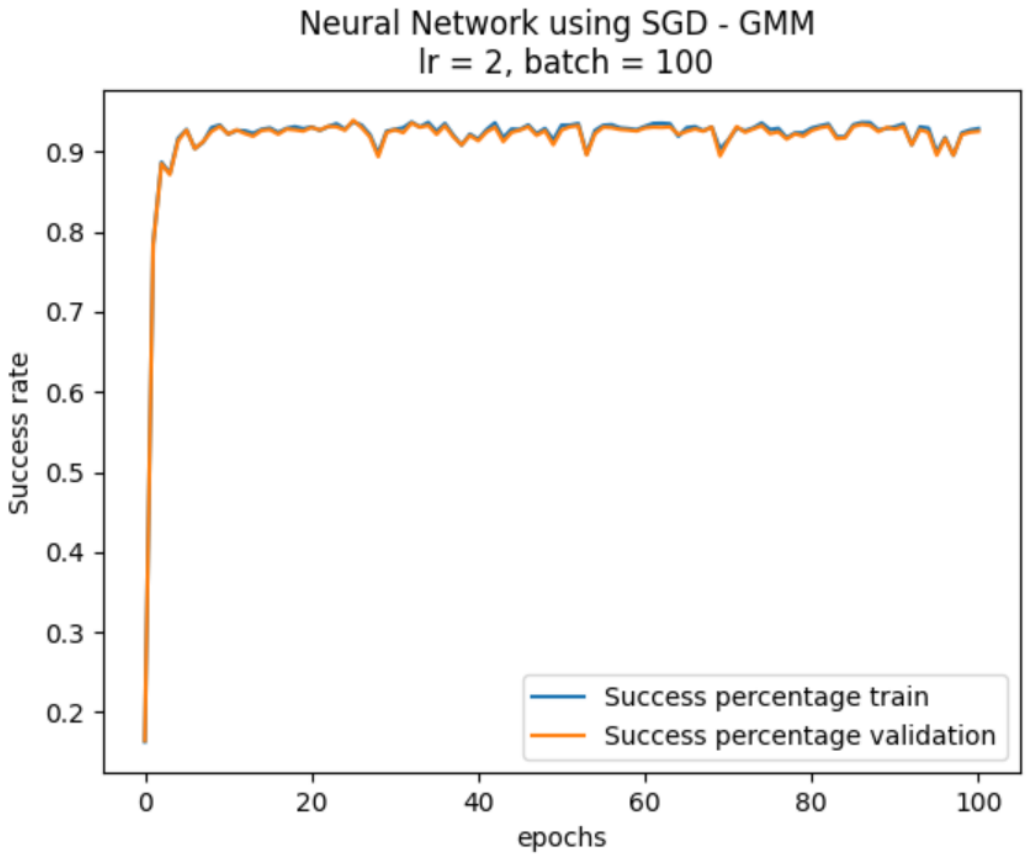


hidden layers of dimensions [7,10,10,10,10,10,10,4], learning rate: 0.5, batch: 100:

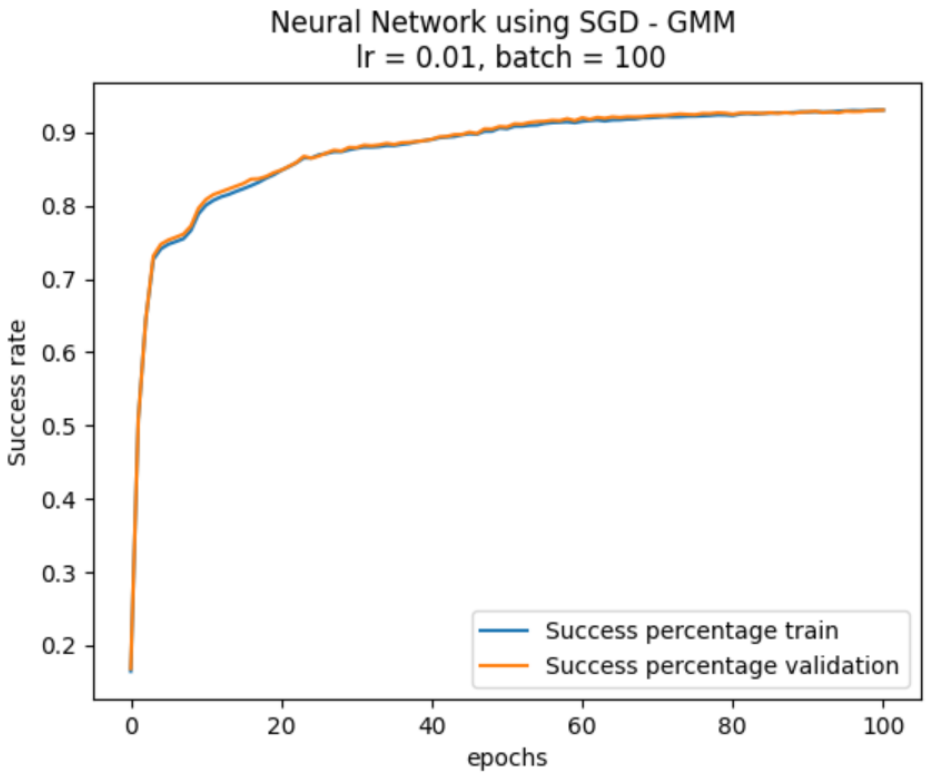


hidden layers of dimensions [7,10 ,4], learning rate: 2, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

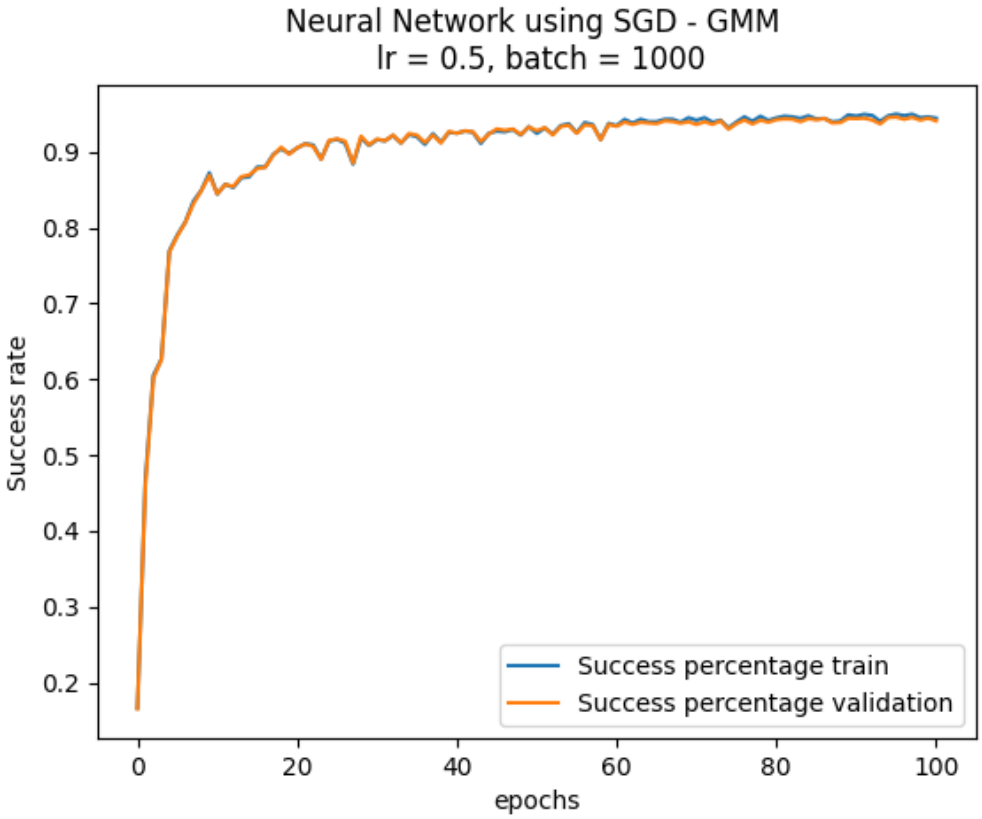


hidden layers of dimensions [7,10 ,4], learning rate: 0.01, batch: 100:



hidden layers of dimensions [7,10 ,4], learning rate: 0.5, batch: 1000:

Alon Ben-Melech 313184178
Tal Berman 318279437



Conclusion:

We can see that using too many layers can lead to very poor results. A big number of parameters needs a lot of data and epochs to train. We can assume more data and epochs could improve the results for this architecture.

Reducing the dimension by setting the first hidden layer size to 3, was not as good as the best result, but not as bad as the same experiment on the swiss roll dataset.

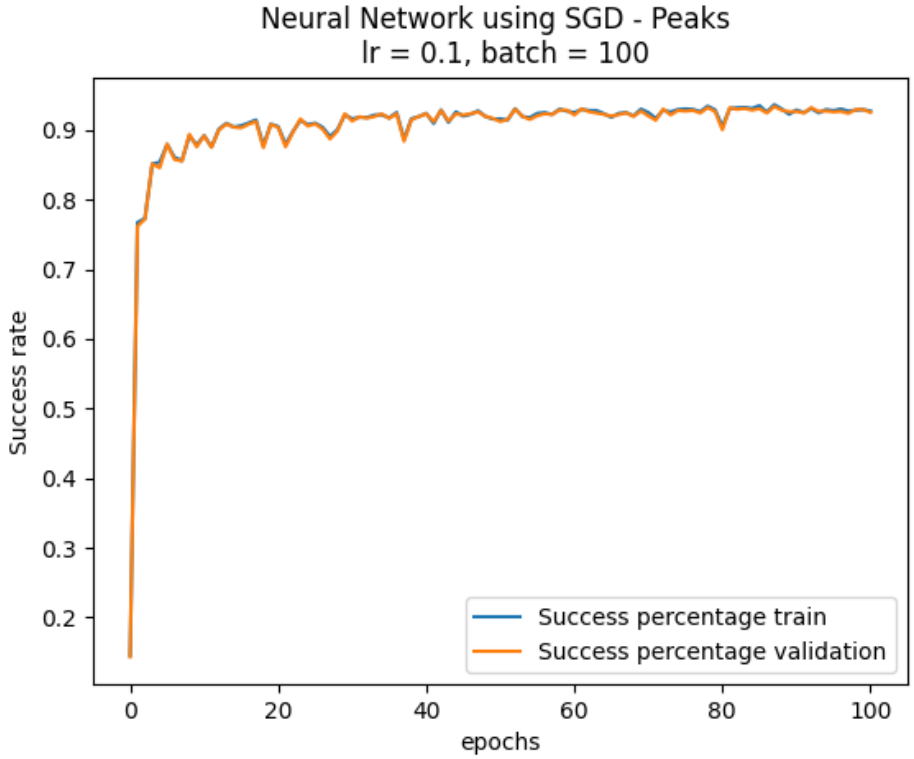
As for the learning rate, using a too small learning rate caused slower learning, though the final result is good.
using a large learning rate gave good results, as opposed to swiss roll in the same experiment. This may be explained because the GMM data set seems to be more easily separable than the Swiss roll.

Setting a bigger batch size, led to slower learning.

Alon Ben-Melech 313184178
Tal Berman 318279437

Peaks

We got our best results with hidden layers of dimensions [10,10,4], learning rate: 0.5, batch: 100:

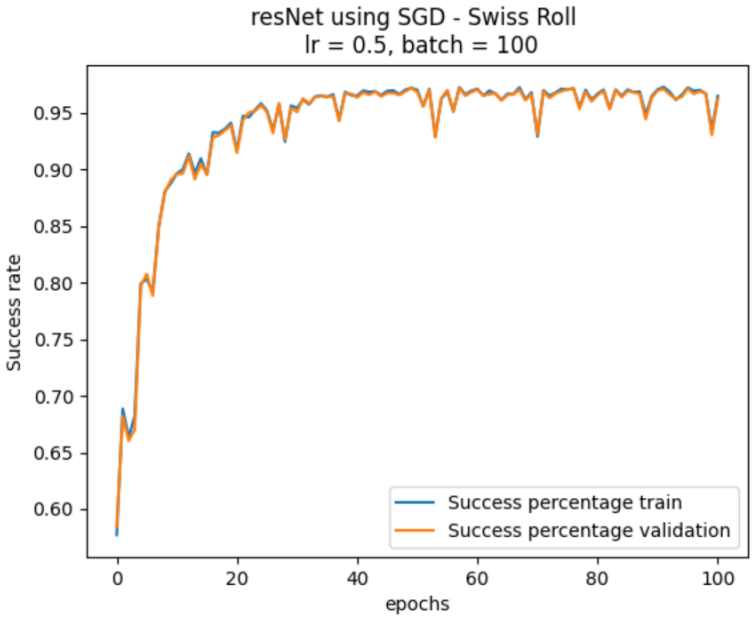


ResNet

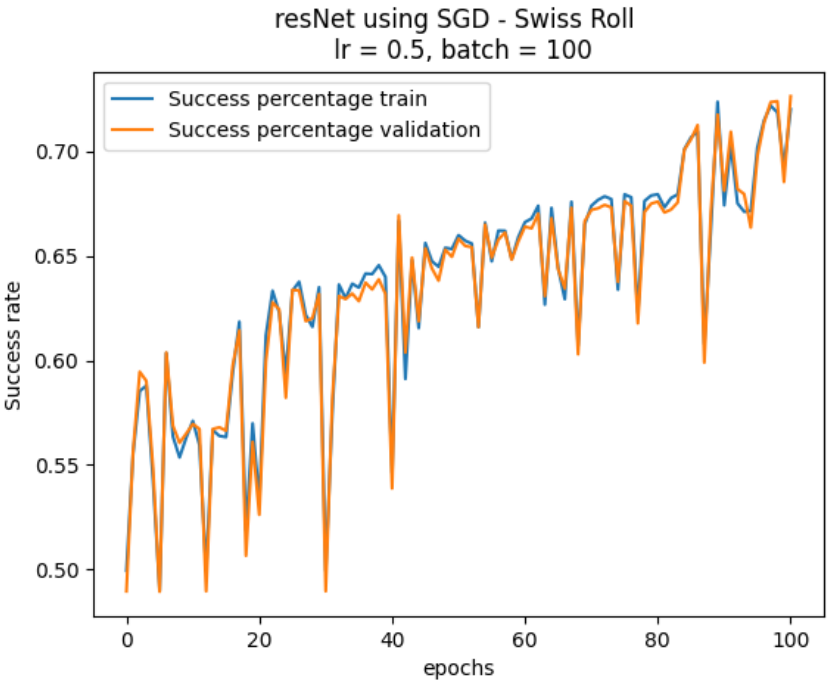
Swiss Roll -

We got our best results with hidden layers of dimensions [10,10,10], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

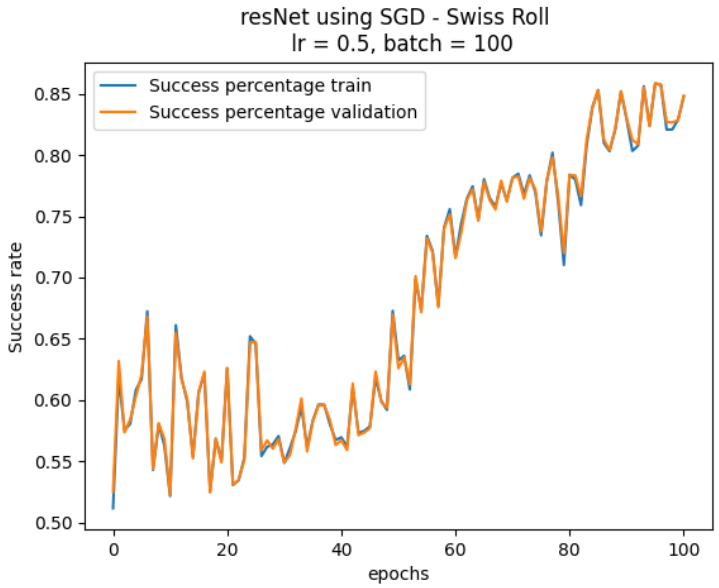


hidden layers of dimensions [3,3,3], learning rate: 0.5, batch: 100:

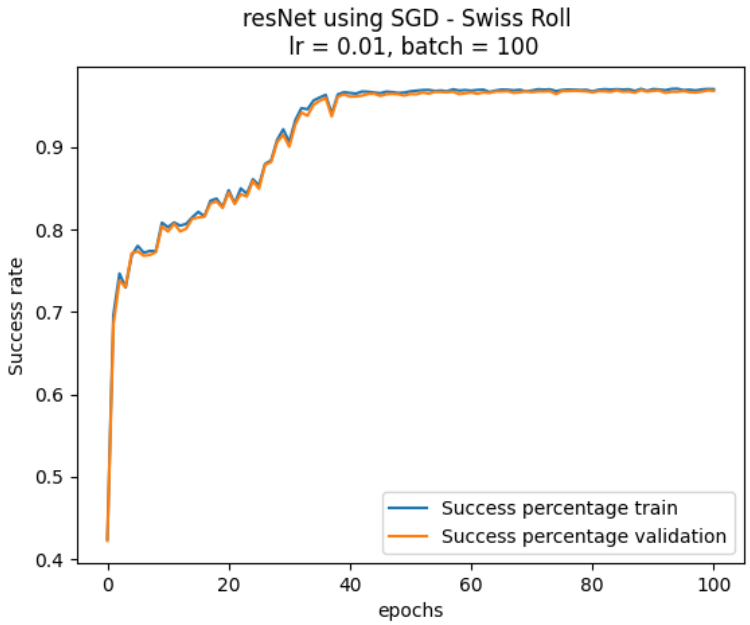


hidden layers of dimensions [10], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437



hidden layers of dimensions [10,10 ,10], learning rate: 0.01, batch: 100:



hidden layers of dimensions [10,10 ,10], learning rate: 0.5, batch: 1000:

Alon Ben-Melech 313184178
Tal Berman 318279437

Conclusion:

Reducing the dimension by setting the hidden layers size to 3, got awful results.

We can see that using fewer layers can lead to slower learning and less accuracy. We can assume that this happens because the hypothesis class is less rich.

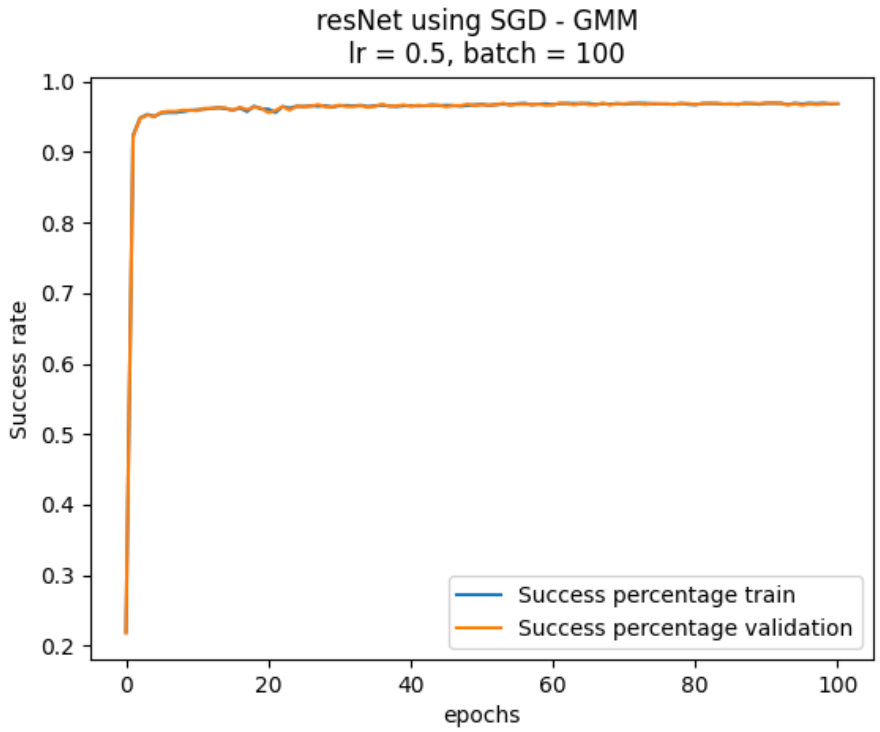
As for the learning rate, using a too small learning rate caused slower learning, though the final result is good.

Setting a bigger batch size, led to slower learning and not as good results as the best parameters.

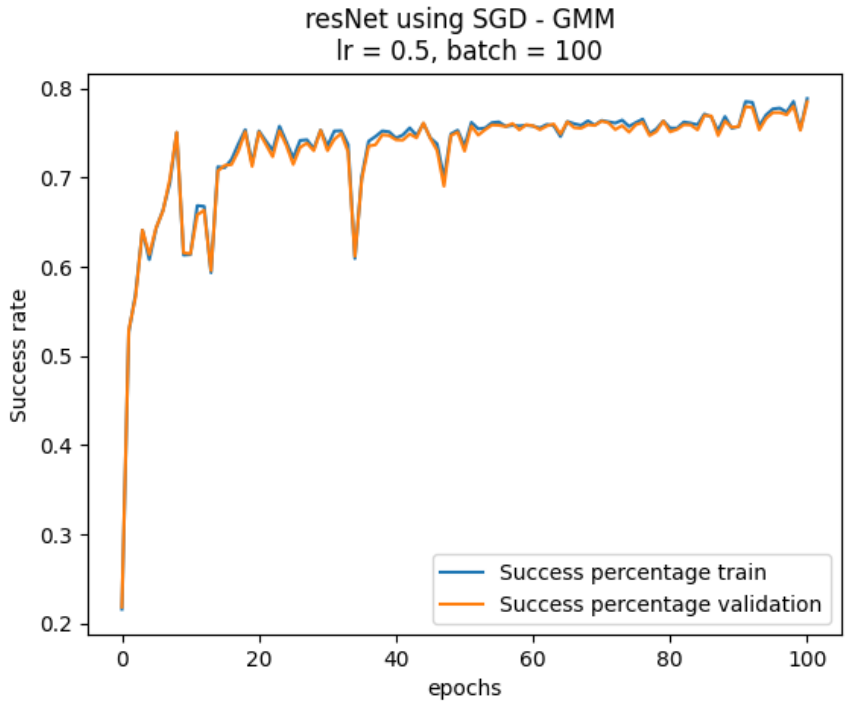
GMM

We got our best results with hidden layers of dimensions [10,10,10], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

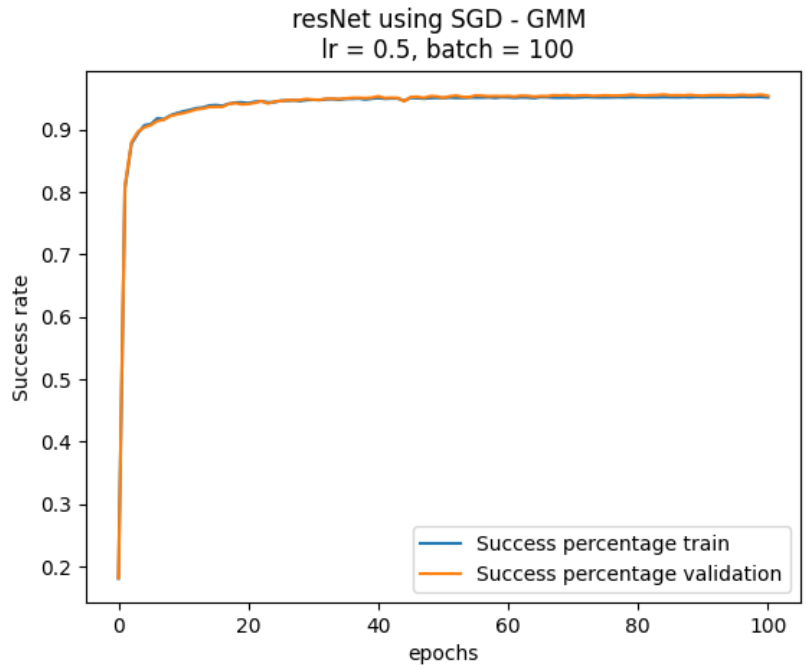


hidden layers of dimensions [3,3,3], learning rate: 0.5, batch: 100:

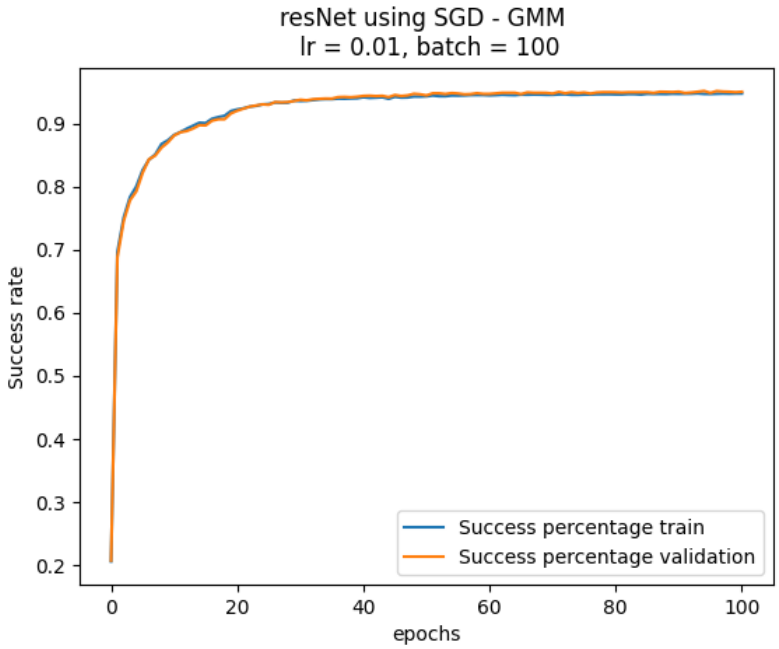


hidden layers of dimensions [10], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

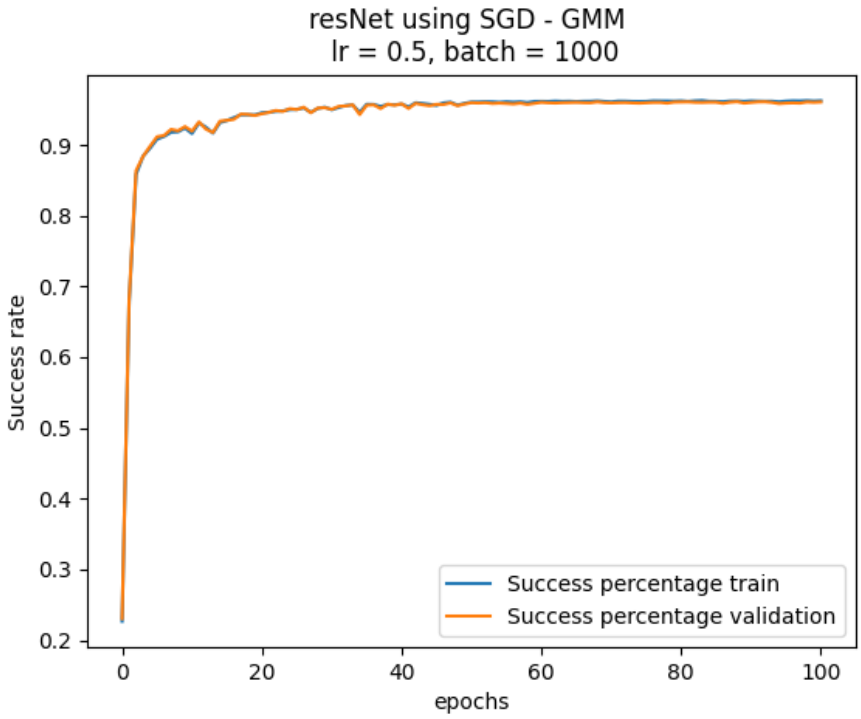


hidden layers of dimensions [10,10 ,10], learning rate: 0.01, batch: 100:



hidden layers of dimensions [10,10 ,10], learning rate: 0.5, batch: 1000:

Alon Ben-Melech 313184178
Tal Berman 318279437



Conclusion:

Reducing the dimension by setting the hidden layers size to 3, was not as good as the best result, but not as bad as the same experiment on the swiss roll dataset.

We can see that using fewer layers can lead to slower learning and less accuracy. but not as slow as on the swiss roll data set.

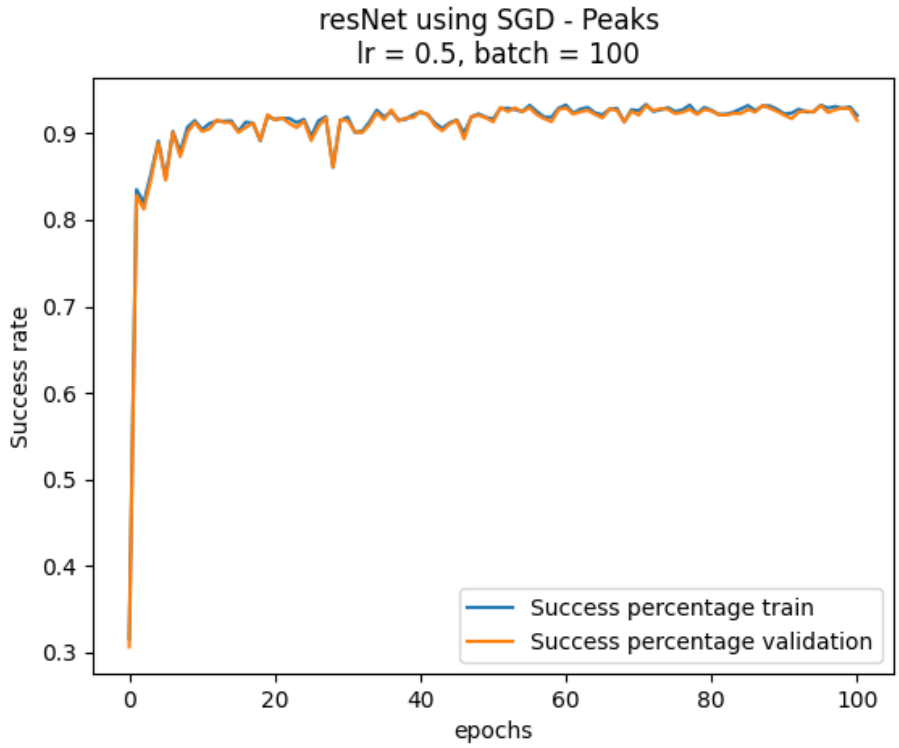
As for the learning rate, using a too small learning rate caused slower learning, though the final result is good.

Setting a bigger batch size, led to a bit slower learning.

Peaks

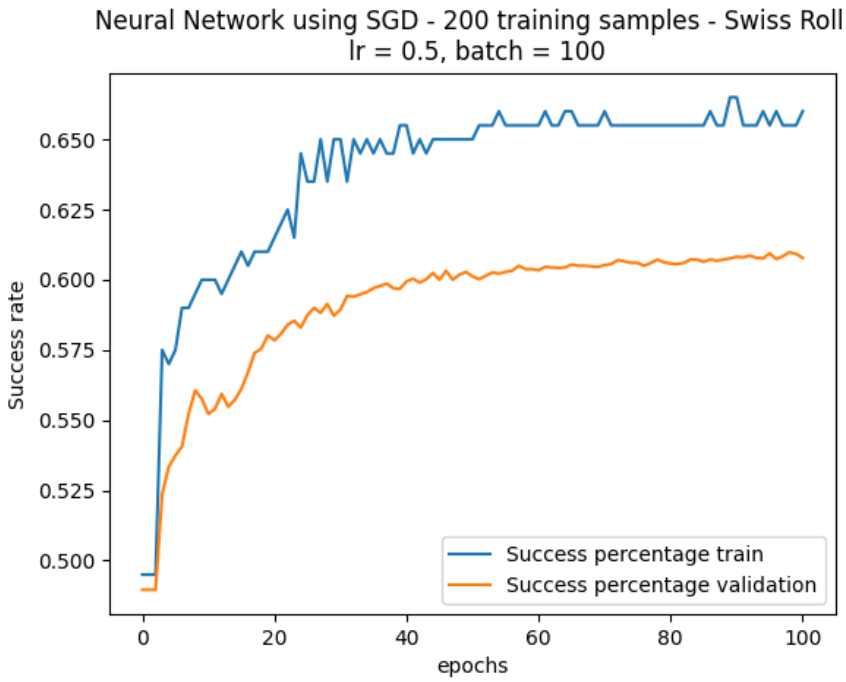
We got our best results with hidden layers of dimensions [10,10,10], learning rate: 0.5, batch: 100:

Alon Ben-Melech 313184178
Tal Berman 318279437

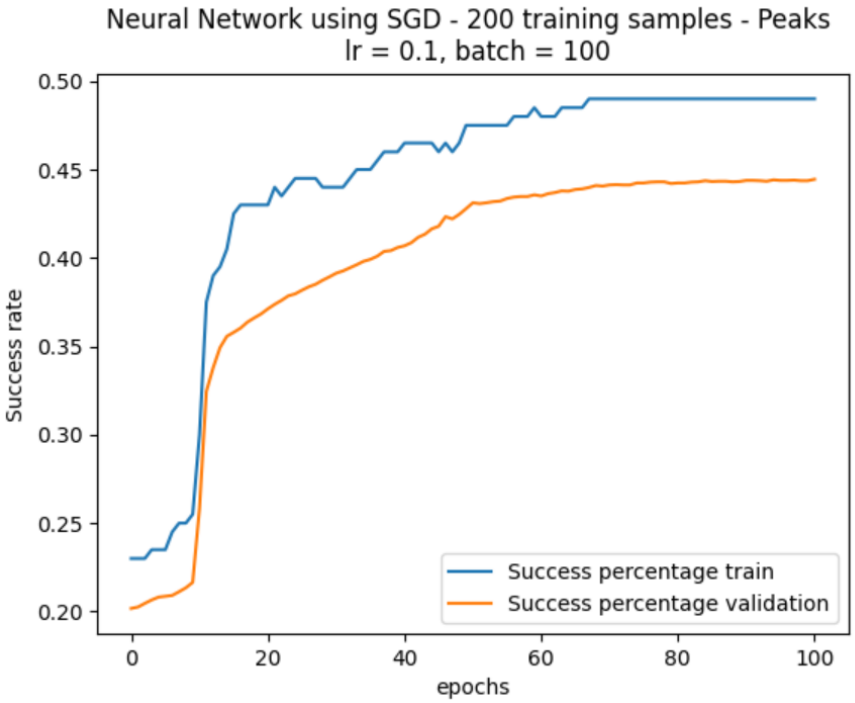


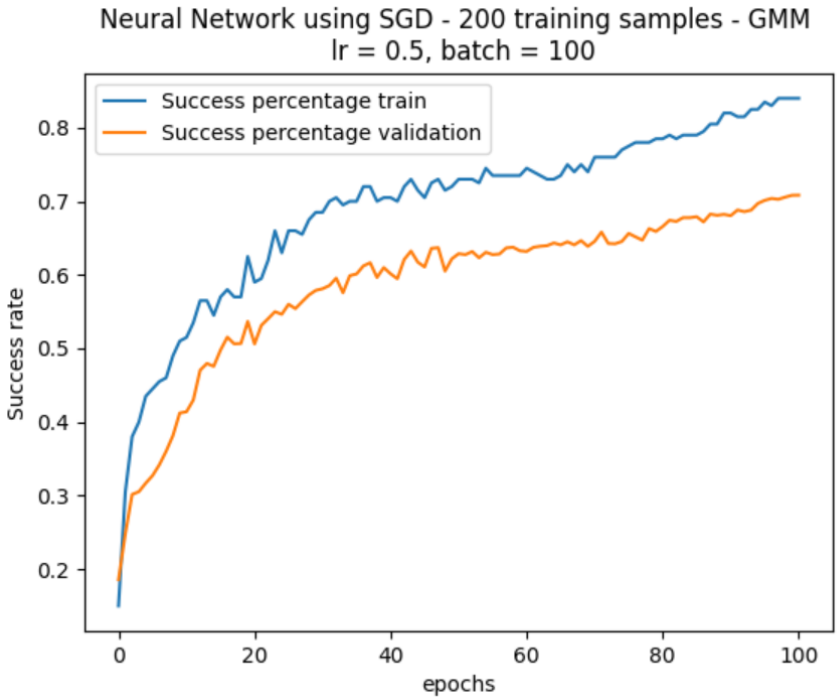
2.2.5

Neural network



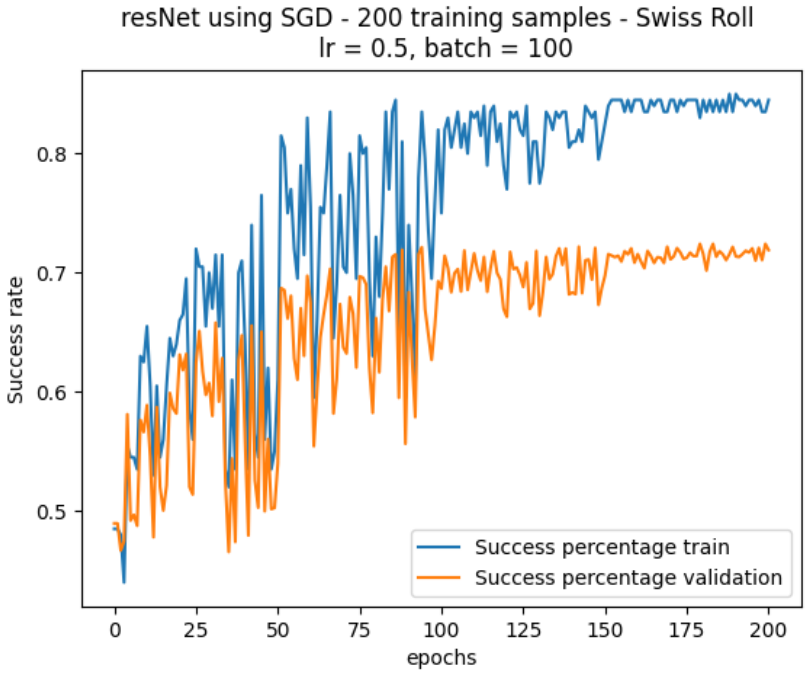
Alon Ben-Melech 313184178
Tal Berman 318279437





Alon Ben-Melech 313184178
Tal Berman 318279437

resNet



Alon Ben-Melech 313184178
Tal Berman 318279437

We can see that for all the runs the train accuracy far exceeds the test accuracy, meaning we are **overfitting** over the data.

In this section we trained our network over only 200 samples.
200 samples are not enough to give a good picture of our distribution, hence when we receive new samples to classify, the probability of this sample being similar to anything we have seen before is significantly lower.