

Carl Ridnert (940325-0112)  
Óttar Guðmundsson (910302-0450)

### **Lab assignment 2 -Hopfield Networks**

#### **5.0 Hebbian learning and Hopeld memory**

The network managed to store all patterns, generating this matrix

```
[ [ 3.  1. -1.  3.  1. -1.  3. -1.]
  [ 1.  3.  1.  1. -1.  1.  1.  1.]
  [-1.  1.  3. -1.  1. -1. -1.  3.]
  [ 3.  1. -1.  3.  1. -1.  3. -1.]
  [ 1. -1.  1.  1.  3. -3.  1.  1.]
  [-1.  1. -1. -1. -3.  3. -1. -1.]
  [ 3.  1. -1.  3.  1. -1.  3. -1.]
  [-1.  1.  3. -1.  1. -1. -1.  3.]]
```

We know it worked, since feeding the pattern into the network we get the same pattern outcome

#### **5.1 Convergence and attractors**

The network managed to converge x1d in 2 tries. For x2d and x3d it managed to converge in 3 rounds. By searching for all possible combinations of one's and zero's we found that there were 16 attractors, thus we found another 13.

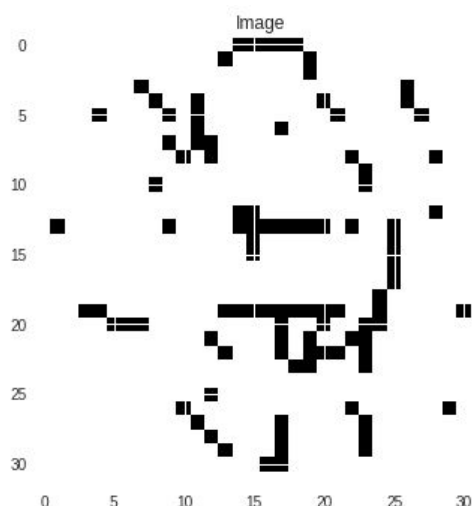
```
[0 0 0 0 0 1 0 0]
[0 0 0 0 1 0 0 0]
[0 0 1 0 0 0 0 1]
[0 0 1 0 0 1 0 1]
[0 0 1 0 1 0 0 1]
[0 1 1 0 0 1 0 1]
[1 0 0 1 0 1 1 0]
[1 0 0 1 1 0 1 0]
[1 0 1 1 1 0 1 1]
[1 1 0 1 0 0 1 0]
[1 1 0 1 0 1 1 0]
[1 1 0 1 1 0 1 0]
[1 1 1 1 0 1 1 1]
[1 1 1 1 1 0 1 1]
```

By changing the number of nodes corrupted, they will all end up in any one of those fixpoints.

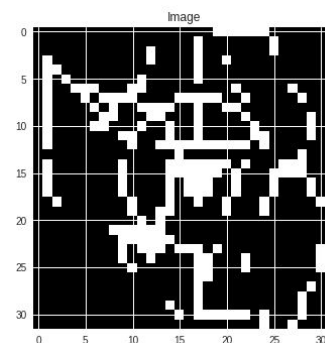
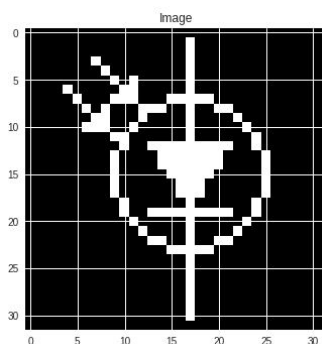
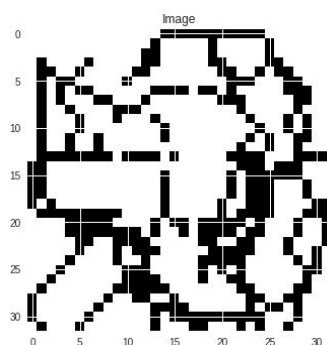
Carl Ridnert (940325-0112)  
 Óttar Guðmundsson (910302-0450)

## 5.2 Sequential Update

We found that the network could recognize all three patterns P1 P2 and P3 however, for the degraded patterns the network could recognize pattern p10 as patterns p1 after 2 iterations. Pattern p1 however did not converge to any of the patterns p2 or p3 but to some other attractor that looked as follows.



Clearly this is not something desirable. Convergence was very fast (2 steps) and now we feed some random input into the system and this means that convergence probably won't happen at all because there are  $2^{1024}$  combinations of pictures. Every hundredth iteration we plotted the pictures and some samples are shown below. The network almost captures the pictures for some inputs and in general the output of the network for random inputs seems to be combinations of the attractors.



Carl Ridnert (940325-0112)

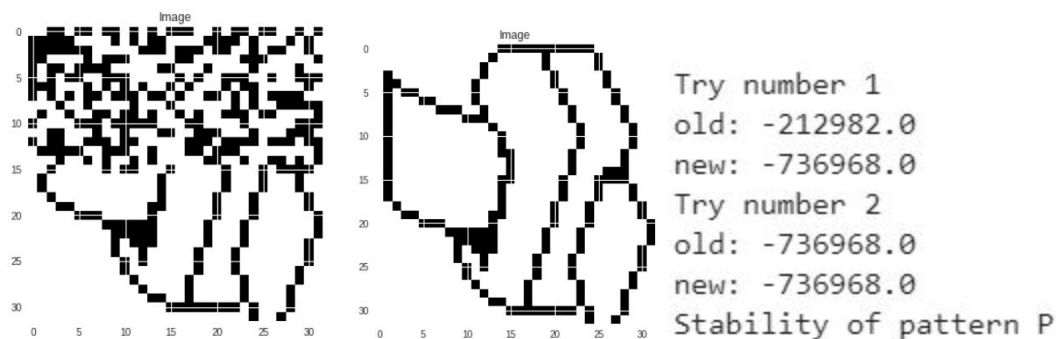
Óttar Guðmundsson (910302-0450)

### 5.3 Energy

The energy function can be expressed in  $0.5 * \text{activation} * \text{weights} * \text{activation}^T$  in linear algebra. We calculated the energy at the trained attractors and then the distorted patterns

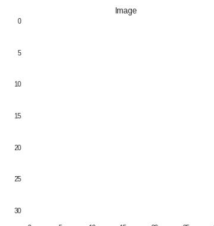
Pattern	Energy
p1	-736.968
p2	-699.208
p3	-748.672
p10	-212.982
p11	-88.832

The attractors have way higher energy (note: less is more in Hopfield networks) then the distorted patterns.



By setting the weights as a random numbers the network doesn't make any progress. It stabilizes at a fairly low energy level with either all nodes as one's or zero's.

Try number 1  
old: -736968.0  
new: -659536.0  
Try number 2  
old: -659536.0  
new: -659536.0  
Stability of pattern P1



Making the weight matrix symmetric we end up getting the same weight array. Since the matrix is correlational (mirrored at  $x_{ii}$ ), transposing it gives us the same matrix as before. So by adding a transposed version to the original is just the same as multiplying it by two. Then we half it and get the same results.

Note that if the matrix is not correlated we will not get the same array.

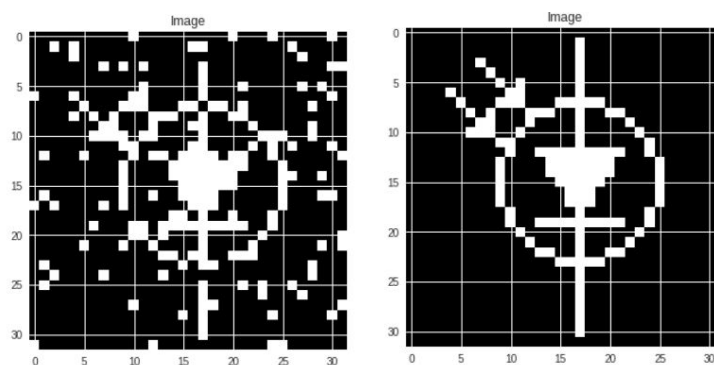
Carl Ridnert (940325-0112)  
Óttar Guðmundsson (910302-0450)

Correlated array	Not Correlated array
[[9 3 5]	[[9 1 2]
[3 9 6]	[3 9 4]
[5 6 9]]	[5 6 9]]
Transposed	Transposed
[[9 3 5]	[[9 3 5]
[3 9 6]	[1 9 6]
[5 6 9]]	[2 4 9]]
Apply formula	Apply formula
[[9. 3. 5.]	[[9. 2. 3.5]
[3. 9. 6.]	[2. 9. 5. ]
[5. 6. 9.]]	[3.5 5. 9. ]]

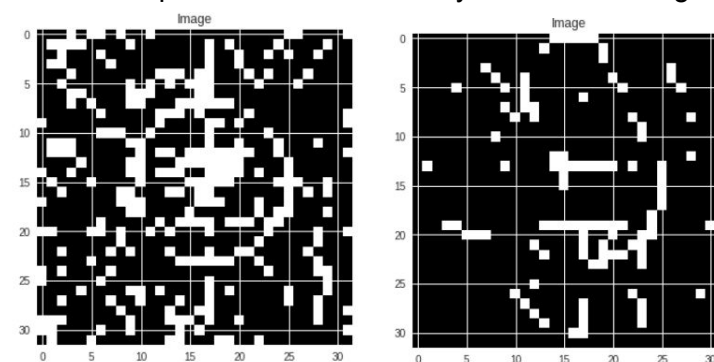
Changing the randomized weight matrix didn't change anything at all. We still ended up in the same state with the same energy

## 5.4 Distortion Resistance

Testing it on p3 we can see that that it tolerates 100 corrupted nodes but adding another 100 result in no convergence.



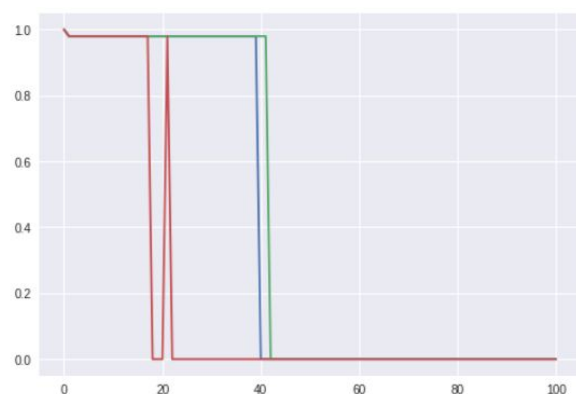
100 corrupted nodes successfully restore the image



200 corrupted nodes fails at restoring the image

To get the results we want, we iterated through all patterns with, scaling from 0% to 100% noise in each image. The results varied a bit but our flip function selected a random nodes from the image, and in a few cases one nodes was flipped two times.

Carl Ridnert (940325-0112)  
 Óttar Guðmundsson (910302-0450)

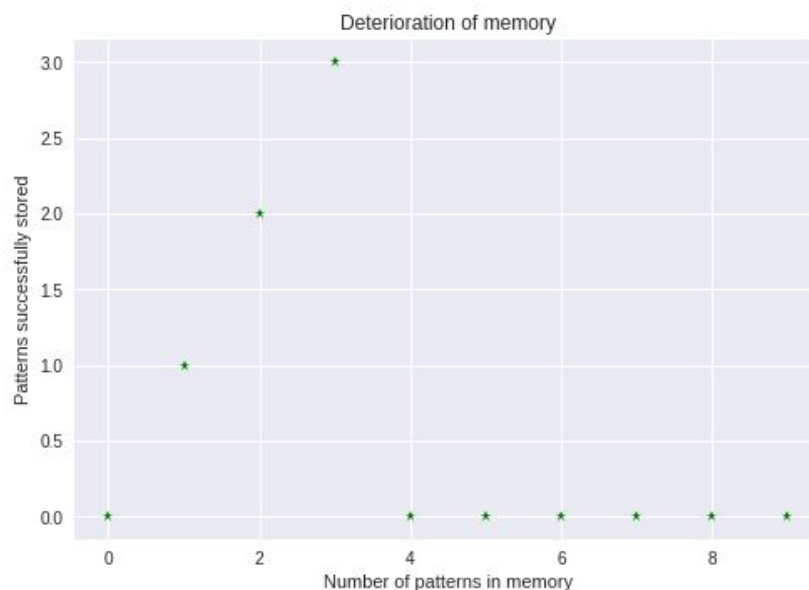


First broken pattern of  
 p1 is at 40% (409.6 nodes)  
 p2 is at 42% (430.08 nodes)  
 p3 is at 18% (184.32 nodes)

For some reason, patterns p1 and p2 manage to survive a little longer than p3. We are not sure why but maybe it is because in the former two the difference in the images is more spread out while the form of p3 is in one place.

### 5.5 Capacity

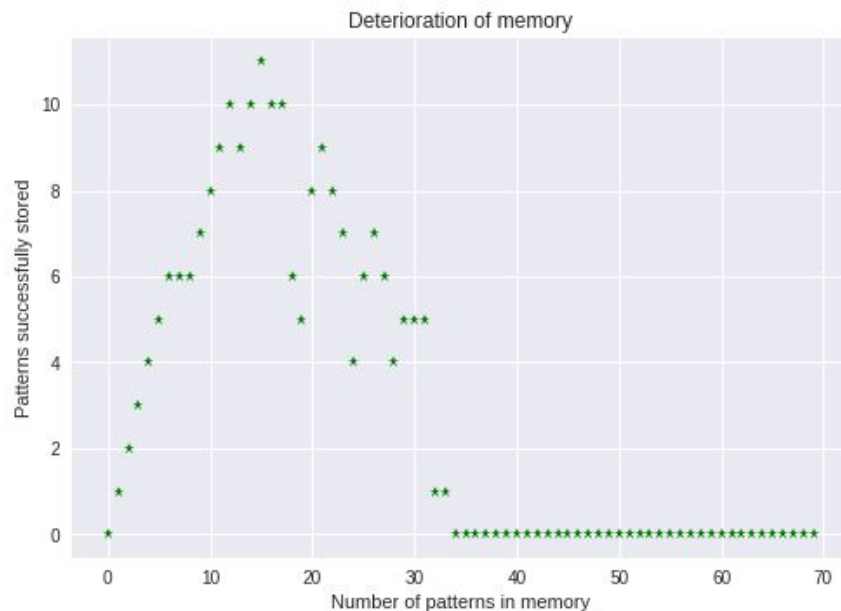
The memory deteriorates very fast, we can only store the three first images whereafter the performance drops to the point where no image can be stored, this is depicted below.



When generating random patterns we can store more, we aborted the loop after 20 patterns were successfully stored. We suspect that we might be able to store up to 130-140 patterns for this network.

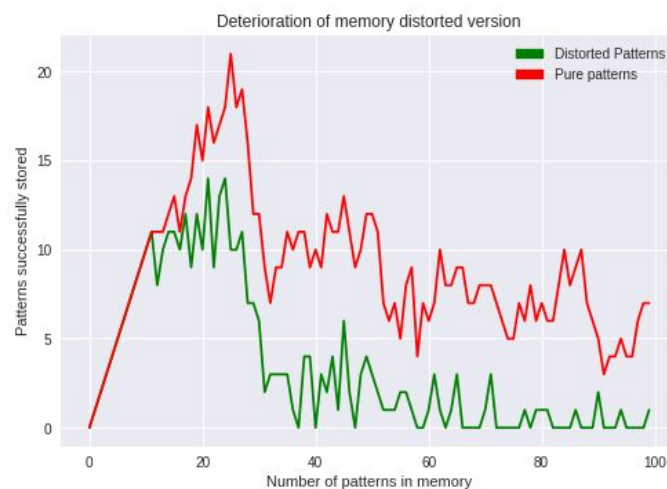
Carl Ridnert (940325-0112)  
 Óttar Guðmundsson (910302-0450)

When creating 300 random patterns in a 100-node network and then successively adding them to the weight matrix we can see from the plot below that the number of patterns the network can remember decays rapidly after about 13 patterns in the matrix.



We note that  $0.138 \cdot 100 = 13.8$  so this result is quite expected.

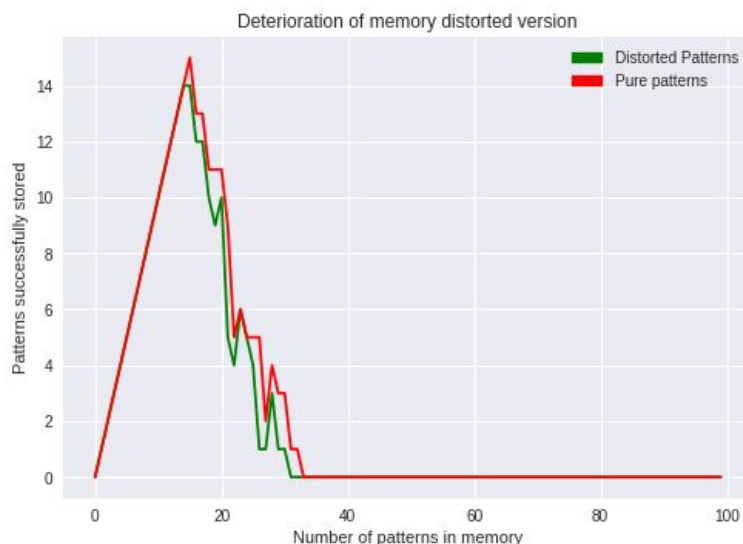
Next we generate some noisy patterns where 4 values in the patterns are changed. The convergence to the corresponding pure patterns are checked, the results are visible below.



We see that for larger values of numbers of patterns stored in the weight matrix, the recognition performance is very different for the distorted patterns and in fact a lot worse.

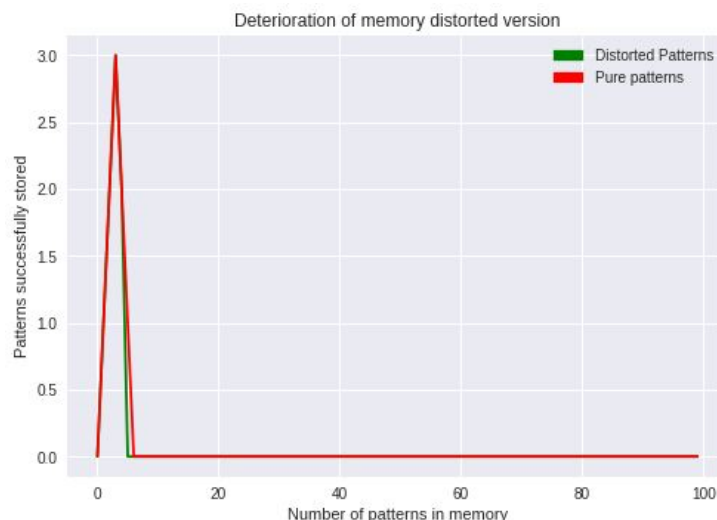
Carl Ridnert (940325-0112)  
 Óttar Guðmundsson (910302-0450)

Next we will remove the self-connections in the weight matrix, the results after doing this is shown in the next plot.



One can see that the network performs almost the same for the distorted patterns but that the network cannot remember as much as previously. In this sense the change in the self-activation in the network has a noise-canceling effect. The maximum number of retrievable patterns in this case are around 15, almost the same as for the distorted patterns when self-activations were present but worse than the pure patterns in that case.

Finally we bias the patterns to contain more +1 values, this has the effect of drastically reducing the number of patterns stored as can be seen in the plot below.



The network seems to have problems storing patterns where there is some kind of order, just as the pictures have some clusters of same values, by adding the bias we obtain the same kind of clustering.

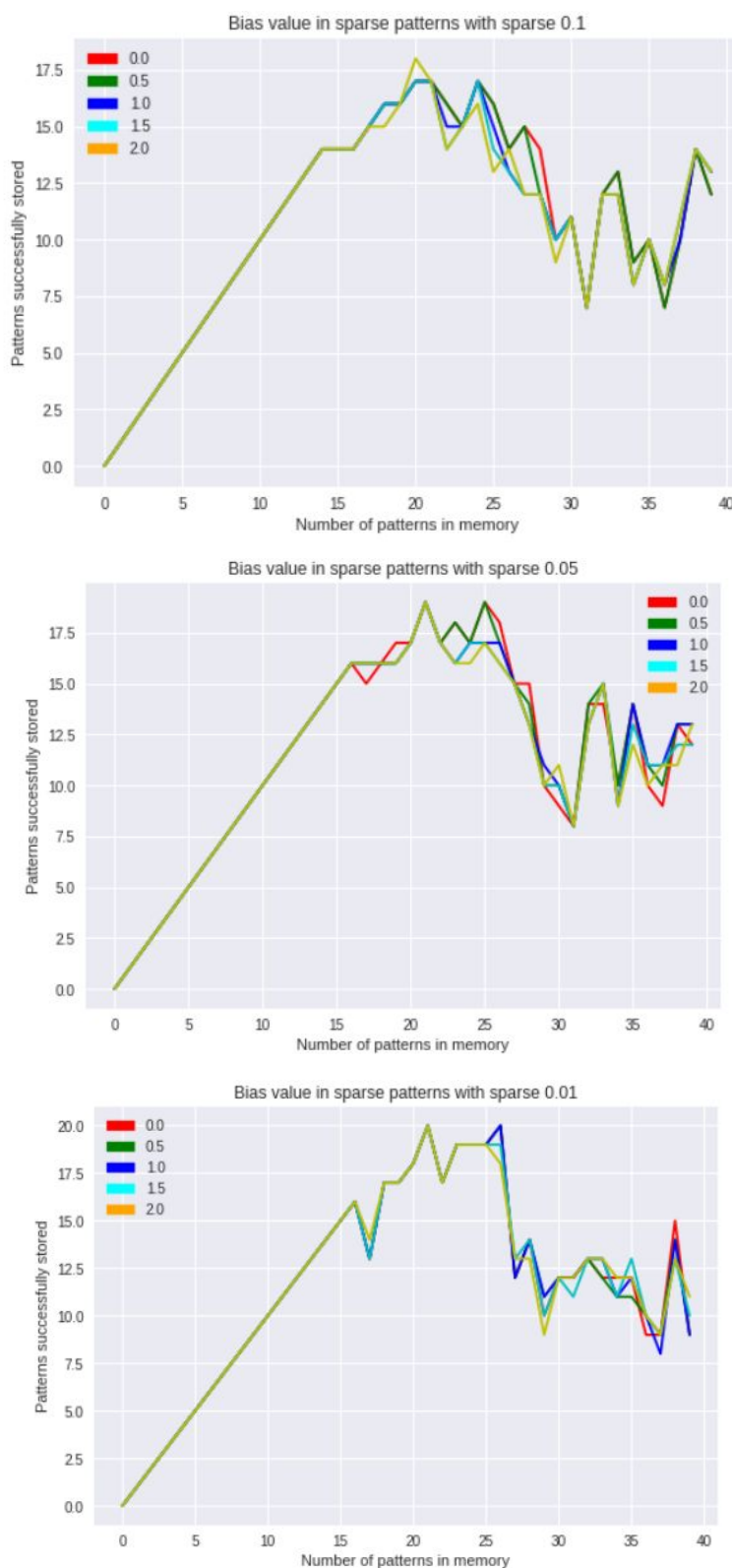
Carl Ridnert (940325-0112)

Óttar Guðmundsson (910302-0450)

## 5.6 Sparse Patterns

For the sparse patterns we looped through 5 different bias terms, ranging from 0 to 2.

### Self connected





Carl Ridnert (940325-0112)  
 Óttar Guðmundsson (910302-0450)

### Not self connected

