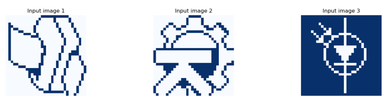
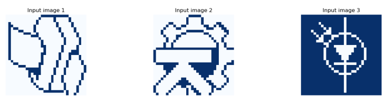
0.1 Sequential Update

For the first task, the network converges perfectly as can be shown by the input and output images:



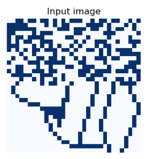
(a) Input into the Hopfield Network



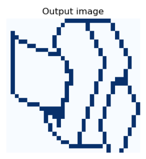
(b) Input into the Hopfield Network

Figure 1: The input and output into the network

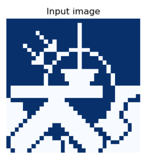
For pattern 10 and 11 however, it is only able to recall pattern 10 as can be seen by the following plots: When performing random, asynchronous updates, the convergence time is



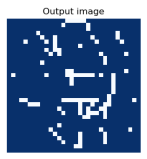
(a) Input pattern 10



(b) Output from pattern 10



(c) Input pattern 11



(d) Output pattern 11

Figure 2: Pattern 10 and 11 and its respective output

slower, however, it successfully recalls both pattern 10 and 11 to an acceptable degree, which the Hopfield network with synchronous update could not do. The stopping condition in this case was to stop the running if no changes in the x-vector happened for 100 consecutive asynchronous updates. From the images, it is clear to see that the pattern is unveiled step by step at a consistent pace. The images below show the plots for every 500th asynchronous update.

0.2 Energy

When considering our three attractors for p1, p2, and p3, we can see that the energy for these lie somewhere around -1400, and when inspecting the distorted patterns p10 and p11 we notice that they are located in areas with much higher energy (-400 and -200 respectively), since they have not yet reached a stable attractor. When we recall these distorted patterns we can see that they are quickly decreasing and reaches a minimum after some iterations, confirming that as the network recalls the correct stored pattern, the energy decreases.

When we initialize the network with random weights, we do no longer have any well defined attractors corresponding to meaningful patterns, this could lead to the network cycling through random patterns and never converge, or possibly converge to a spurious attractor. If we however enforce the weight matrix to be symmetric, we are guaranteed to converge to a stable state as long as we are using asynchronous updates, see fig 5.

0.3 Distortion Resistance

For this assignment, noise was applied on a bit by bit basis for patterns 1,2 and 3, with every permutation of each of the pattern was tested on. All the pattern stopped to converge to the stored memory when around 450 bits out of 1024 had been flipped, which suggest that they have a similar robustness, although they deviate slightly on how much noise they can tolerate. Another interesting phenomena for this experiment was that all the patterns, irrespective of noise, only needs a single step recall to converge, even though they converge to a different pattern than the original. The table below show the Hamming distance from the original pattern that each pattern could tolerate, before starting to converge to a different pattern than the original signal:

Pattern 1	457
Pattern 2	467
Pattern 3	424

Table 1: The Hamming distance each pattern could withstand before converging to the wrong pattern

0.4 Capacity

The network can successfully store three memories with perfect recall, however, when a fourth pattern is introduced, the network can not perfectly recall a single image as it could before. When storing four patterns, it is however possible to see traces of what the images should be. When introducing a fifth pattern, the memory becomes completely corrupted and it is hard to discern what input pattern each output pattern corresponds to. This phenomena is called catastrophic forgetting effect and often occurs in Hopfield networks.

For randomly generated the performance significantly improves. The network can store three additional noisy patterns on top of patterns one, two and three. After that, the performance starts to deteriorate, although the decline is much less steep as it still can recall some of the patterns perfectly. The difference between the random patterns and the actual images is that all the images are actual images and not noise, and thus they should be correlated in what their bit values are for certain pixels. This means that the images will not be close to orthogonal, which leads to cross-talk between the patterns. The images with random noise does not have any correlation with each other, which probably means that they are closer to being linearly independent and thus introduces less cross talk, which leads to a higher memory capacity. From the figures in fig (10) it is possible to see that the memory capacity deteriorates significantly quicker for the case where the input vectors are noisy, compared to the case where they are the same as the weights. Theoretically, the memory capacity is defined as $0.138N$ where N is the amount of nodes in the network. Thus, in this case the memory should be around 14 patterns as the network that the experiments were ran on had 100 nodes.

With a non-zero diagonal and a skewed dataset, the following plot is derived:

0.5 Sparse Patterns

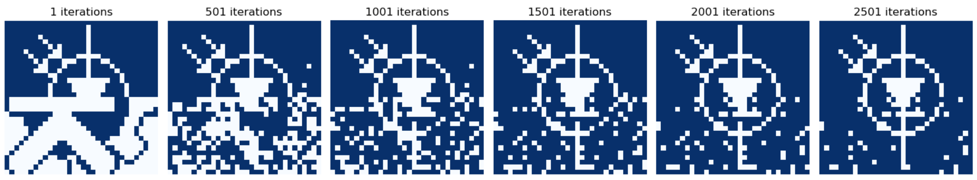
When training our network on sparse binary patterns, we increase the likelihood that stored patterns are more orthogonal and less correlated. This reduces interference during recall, allowing the network to better separate different stored patterns and ultimately increasing its capacity. While it is intuitive to conclude that a lower sparsity level (ρ) leads to higher capacity, this does not guarantee stable recall. In fact, the network's performance remains highly dependent on the threshold θ, which controls neuron activation. An improper choice of θ can either suppress recall (if too high) or increase pattern interference (if too low), making effective storage and retrieval of patterns unreliable.

1 Final remarks

This assignment provided a valuable opportunity to explore and analyze the workings of the Hopfield network. We examined how network memory capacity is influenced by various factors and identified ways to optimize it. Adjusting parameters, modifying the update algorithm, and promoting sparsity in input patterns were found to be effective strategies for improving performance. This study has deepened our understanding of associative memory and the practical limitations of Hopfield networks.

3.2_development_pattern_10.png

(a) Development of the image generated from pattern 10 every 500th iteration.



(b) Development of the image generated from pattern 11 every 500th iteration.

Figure 3: The images generated by the network with asynchronous updates every 500th iteration

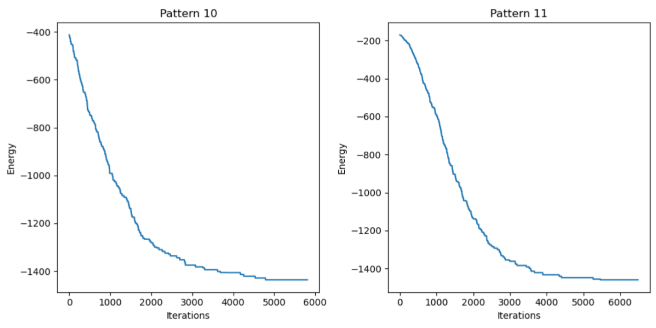


Figure 4: Energy decrease for p10 and p11

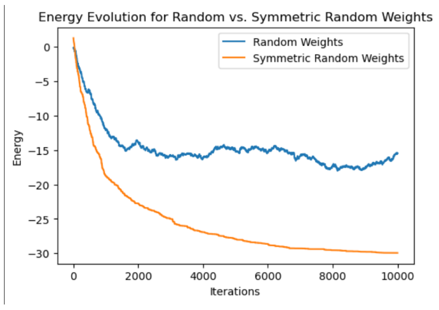


Figure 5: Randomly initialized weights with and without symmetric weight matrix

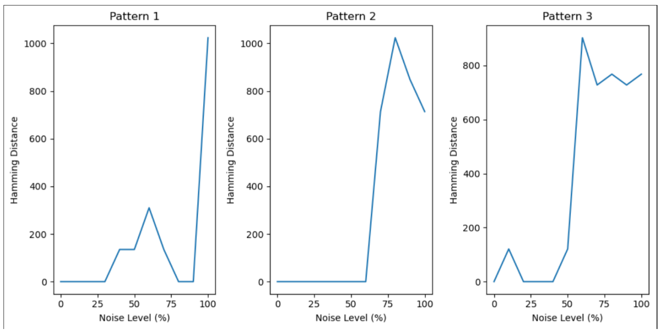
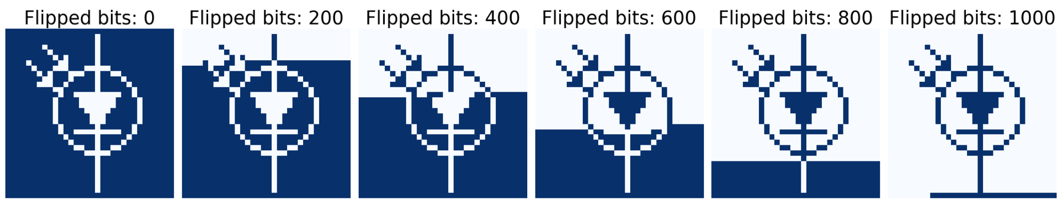
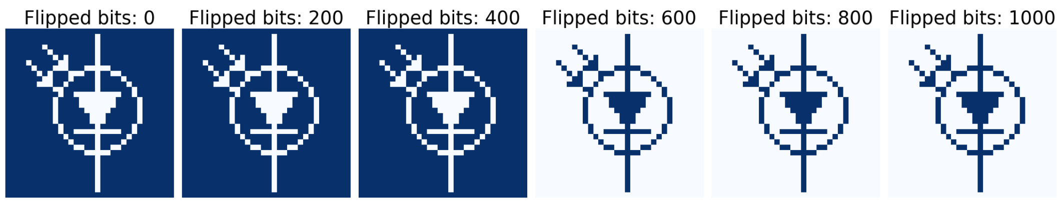


Figure 6: Level of noise applied to pattern 1, 2, and 3 with the number incorrect bits (hamming distance) from the true pattern

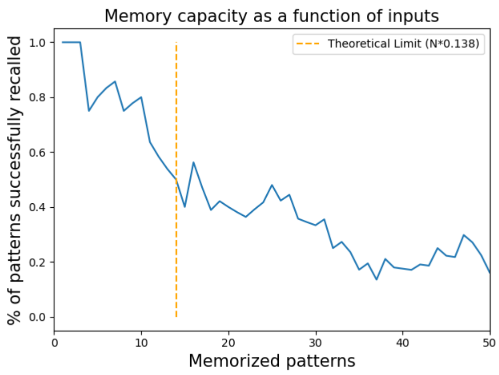


(a) Input pattern 3 with different noise levels

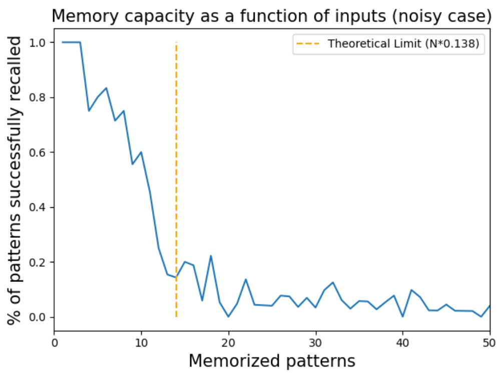


(b) Output with different noise levels

Figure 7: Input and output of network fed with pattern 3 with different noise levels



(a) Memory capacity



(b) Memory capacity with noise

Figure 8: Comparison of memory capacity with and without noise

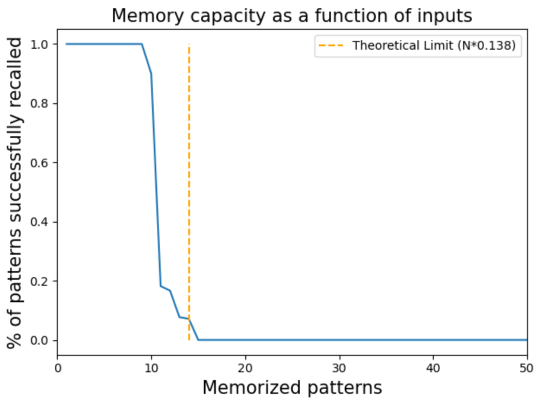
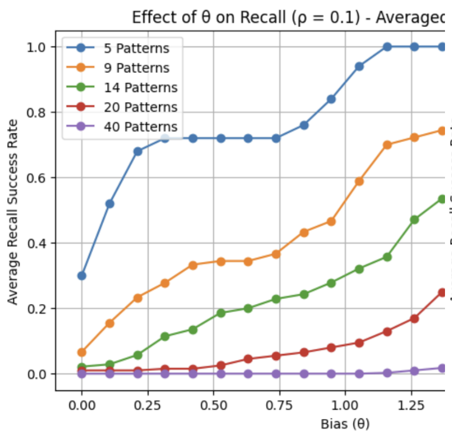
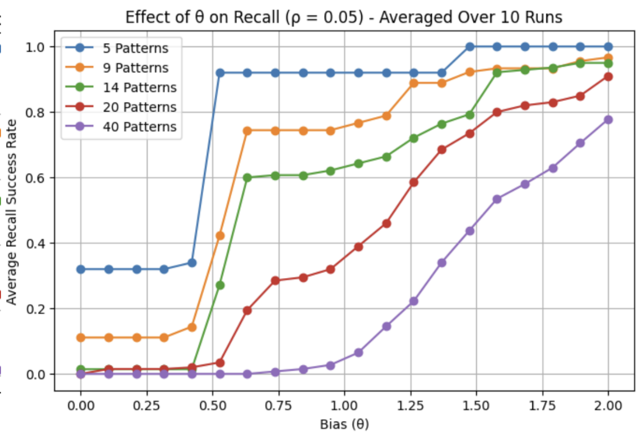


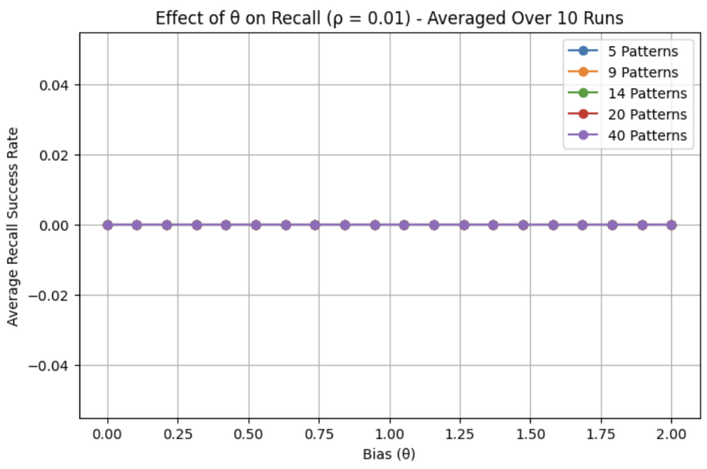
Figure 9: Caption



(a) Rho = 0.1



(b) Rho = 0.05



(c) Rho = 0.01

Figure 10: Effect of theta on recall, with different rho values and stored patterns