UNIVERSIDAD DE GRANADA

Neural Network Hopfield Model

Computational Physics: Degree in Physics

Caleb Porter

July 7, 2023

1 INDEX

2 INTRODUCTION

The Hopfield model was described by Shun'ichi Amari in 1972 and popularized by John Hopfield in 4 1982. Although it is based on the Ising model, a mathematical model of ferromagnetism, the network of Hopfield tries to mathematically simplify a neuron, and thus allow the analysis of the behavior of large-scale neural networks. It characterizes the effect of changes to individual units on a property of neuronal architecture, called energy. The Hopfield network is designed to store patterns so that they can be recovered from cloudy or partial initial states. The network is able to remember the patterns for something we can think of as an energetic surface. On the energetic surface, the patterns are depressions and the configuration is a particle that passes through it. As the system searches for a low energy configuration, the particle falls into depressions, which represent energy minima.3

The network is made up of N x N = N^2 nodes and each of them (i, j) (with i, j = 1, 2, … ,N) can be or active (si,j = +1), or inactive (si,j = 0), thus representing neurons. The so-called energy of the system, here represented using the Hamiltonian formalism, takes the form

$$(s) = ÿ \frac{1}{2} ÿÿÿÿ_{=1 =1 =1 =1} \quad {}_{,j} \quad , \; + \; ÿÿ_{=1 =1} \quad , \qquad (1)$$

Within this equation, we see more properties of the network. The synaptic , represents the weight, connection between the nodes, which are connected to all the other nodes, except with itself. This allows for long-range interactions.

$$, = \{ \begin{matrix} \overline{}2ÿ (ÿ=)(ÿ) , & \text{if } (,) ÿ (,) \\ 1^0 & , \text{if } (,) = (,) \end{matrix} \qquad (2)$$

With

$$= \frac{1}{2ÿÿ_{=1 =1}}$$

This synaptic weight is given in terms of *P* patterns = } where ÿ {0,1}. They are also { symmetrical, that is, , = , · The term *a* in this equation represents the proportion of neurons active in the patterns.

On the other hand, at the , in the energy equation it is called the trigger threshold and is given by end the synaptic weights.

$$, = \frac{1}{2} ÿÿ_{=1 =1} \qquad (3)$$

The *input* received by the nodes *(i,* j) in the network, as we see in the energy equation, is given by the sum of the weights of the active nodes of the network. That, in combination with the trigger threshold, means that the network usually moves towards a state equal to or similar to one of the patterns.

To carry out this model, we are going to use the Metropolis algorithm. The Metropolis algorithm is a Monte Carlo algorithm that uses random numbers to change the state of a system. In this

case, if we choose a node (k, l) randomly, we calculate the quantity ÿ that would be the change in the energy of the system if that node (k, l) changed state. From there, let's calculate (ÿ). ÿ) = min(1, ÿ
If this value p is greater than a number between 0 and 1 chosen at random, the change is made, while, if *p* is less than the random number, the change is not made.

This method goes much faster by using a simplified equation of ÿ = properties ÿ . Using the seen above, one can arrive at the following form, where (k, l) is a random point that would be the only change between H' and H.

ÿ ( ) = (1 ÿ 2) ((ÿ,ÿ , =1, ÿ =1, ÿ) ÿ ,) (2)

3 RESULTS AND DISCUSSION

3.1 SINGLE PATTERN

3.1.1 Random Initial State

Starting our investigation of the model with just one pattern, we will see what results come out for different temperatures. We run the *hopfield.c* program completing 40 Monte Carlo steps and starting from a random initial state of the network. We noticed that the seed was changed between the executions of the program, making the initial random state the same at different temperatures.

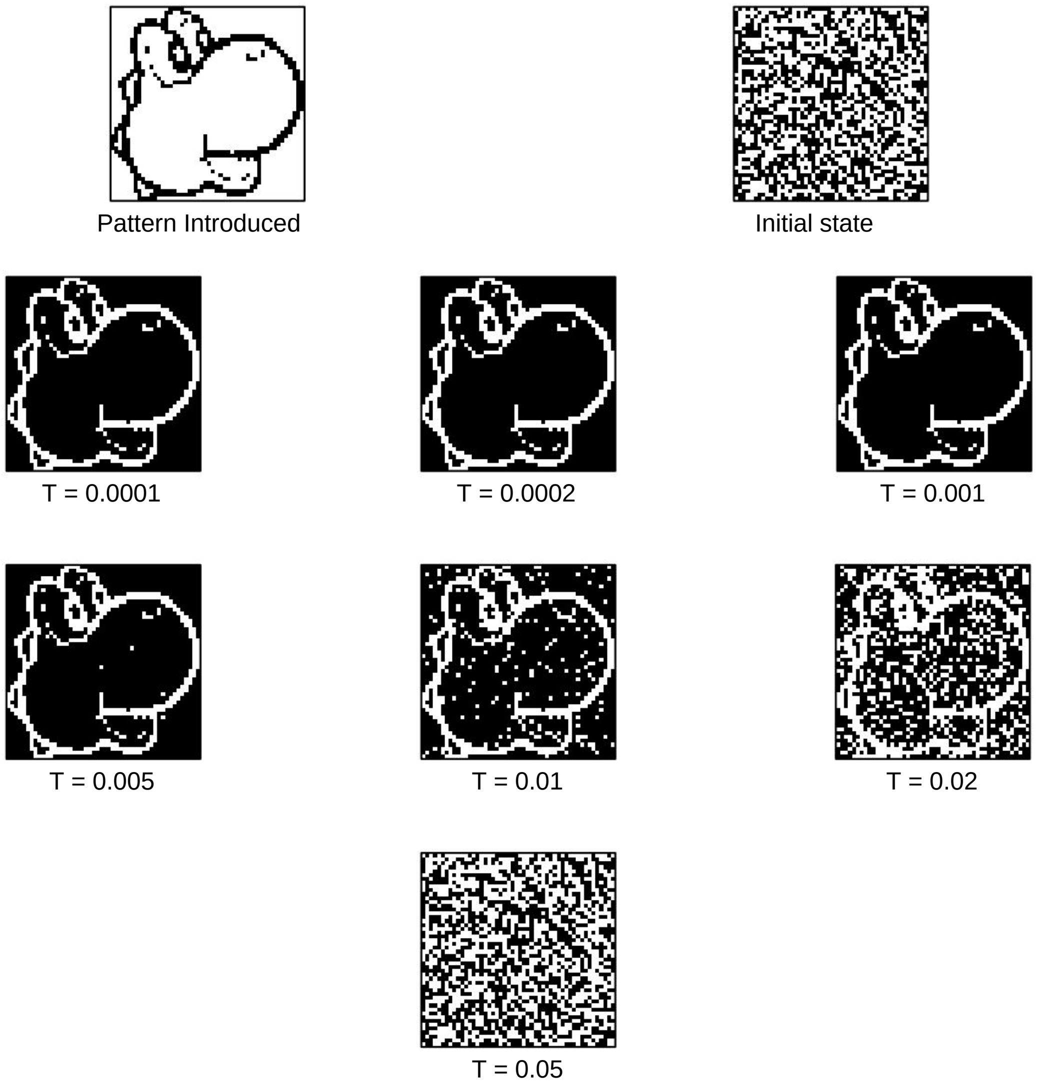


Figure 1: Results after 40 Monte Carlo steps for different temperatures

We already knew that the pattern was a minimum in H, and we hoped that the algorithm would lead towards it. However, in these results we see another energy minimum, the antipattern. The pattern was introduced with the figure in black (zeros) on a white background (ones), and the network leads to the same figure, but with the background in black and the figure white. We can call these states “spurious states” (Wikipedia), and although any minimum of energy that is not a stored pattern is called that, in this case we get the negation of the pattern.

Aside from the spurious states, we can begin to understand the model's dependence on temperature. At lower temperatures, the pattern approximation is very close to perfection. However, when you raise it, the final state moves away from the introduced pattern until it no longer resembles it.

In order to compare this approximation, or convergence, it would be useful to quantify it. We do this by defining the overlap with the pattern as:

$$^{1}(\ ) = \frac{1}{^{2}\ 1(1\ \ddot{y}\ 1)} \ddot{y}\ddot{y}_{=1\ =1}(\ ,\ \ ^{1}\ \ ^{1})(\ \ ,\ \ ^{1})$$

The same program also calculates this value for different temperatures, as seen in the figure 2.

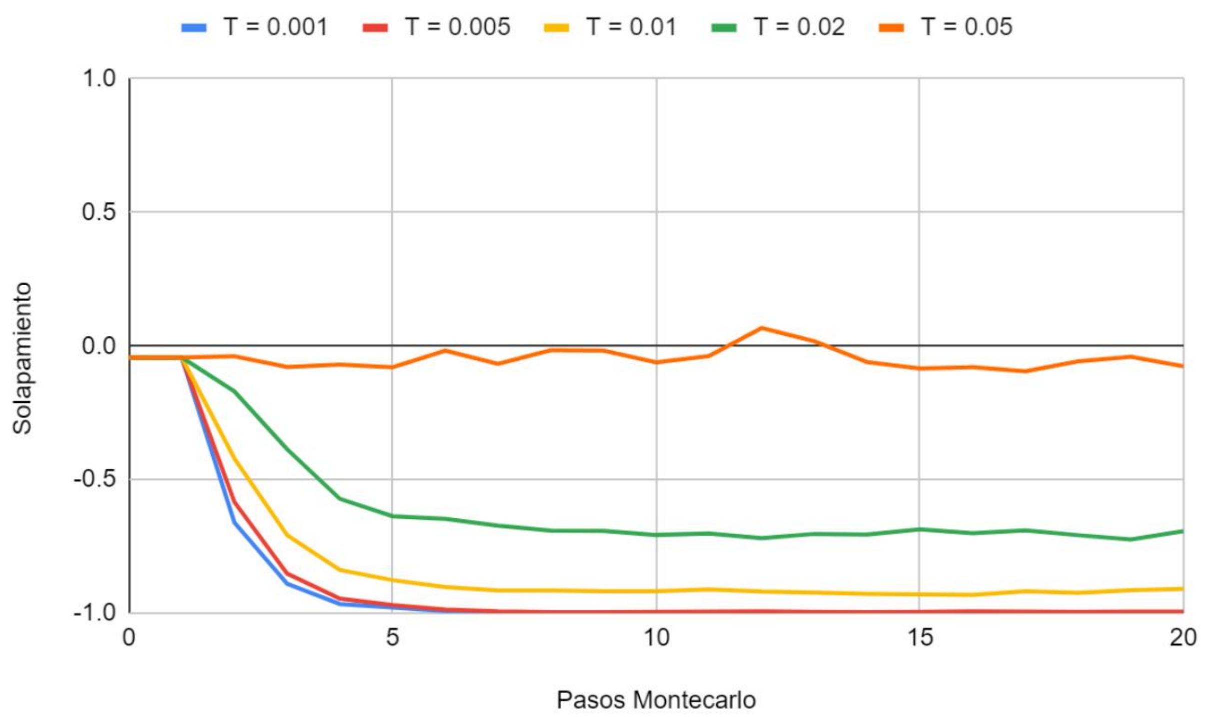


Figure 2 : The overlap as a function of time for different temperatures

We note that the lower temperatures with which we have generated results above are not represented. That is because of their similarity to the temperature T = 0.001, and they all overlap in a similar way. The possible values of the overlap are in the interval [-1,1] with m(s) = 1 when the pattern is recovered perfectly, m(s) = -1 when the anti-pattern is recovered and m(s) = 0 when the final state of the network is disordered compared to the relevant pattern.

Using this knowledge of the overlap, in agreement with the generated images, we see that at low temperatures the antipattern is completely recovered. By raising the temperature, the algorithm remains halfway, oscillating around values that represent a partial recovery of the pattern (or anti-pattern). In the end, at T = 0.05, as before, the final state is unordered, without having gone into a valley in the energy surface.

In the following figure, we have used the program *hopfieldtemp.c* to compare the overlap as a function of temperature, after 20 Monte Carlo steps.

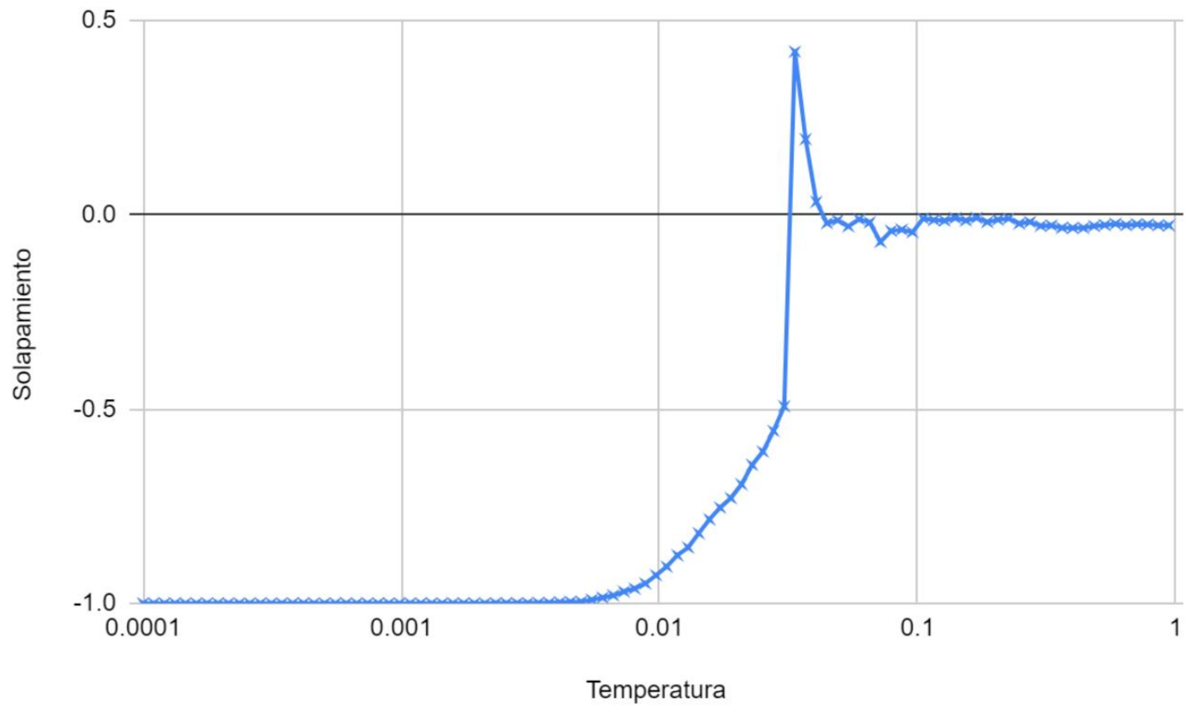


Figure 3: Overlap as a function of time after 20 Monte Carlo steps

In this graph, it is seen that increasing the temperature up to approximately T = 0.009 does not result in a perfect recovery of the antipattern. In the area between T = 0.009 and T = 0.06 the overlap increases until it is very close to zero, a disordered configuration. That is, from T = 0.06, the network no longer recovers the pattern. We also note the peak we see around T = 0.04. Here, the network partially reaches the pattern instead of the antipattern. This may be due to the random nature of the system. As the temperature increases, the possibility of non-downward changes in the energy surface increases. Therefore, even if it starts closer to the antipattern, a random change can push the system towards another valley in the energy surface and thus we reach positive overlap.

3.1.2 Initial state: Deformed pattern In

this section we will see what will happen if we do not start from a random initial state, but from the same pattern that we introduced, but deformed. We will randomly change the state of 30% of the nodes so that the initial network configuration is the introduced pattern warped by ~30%. It is worth highlighting here that we do not discard nodes that are changed once changed, that is, in the algorithm that deforms the pattern, a node can change more than once, making the total deformation a little less than 30%. However, for our purpose, a basic investigation of the model is worth it.

We obtain the following results using the *hopfielddef.c program.*

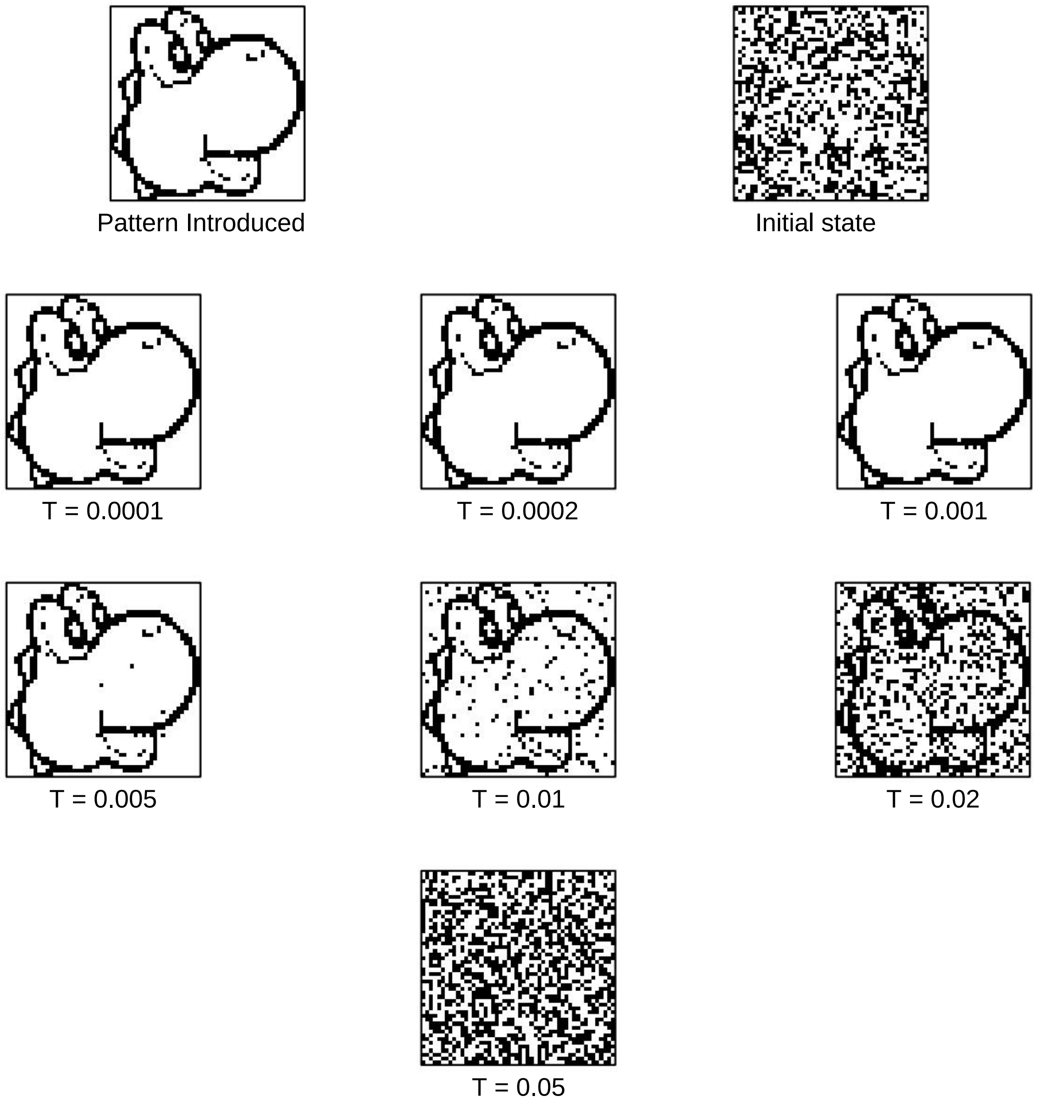


Figure 4: Resulting states from the deformed pattern at various temperatures

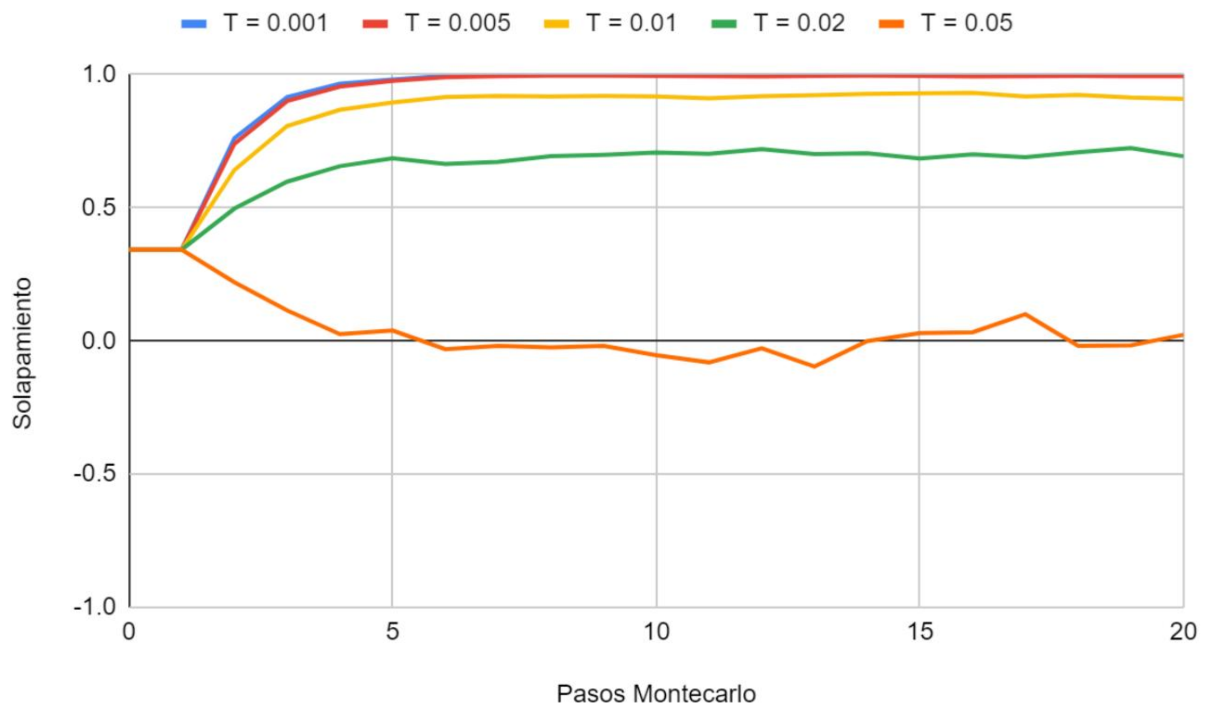


Figure 5: Overlap as a function of Monte Carlo steps starting from the deformed pattern

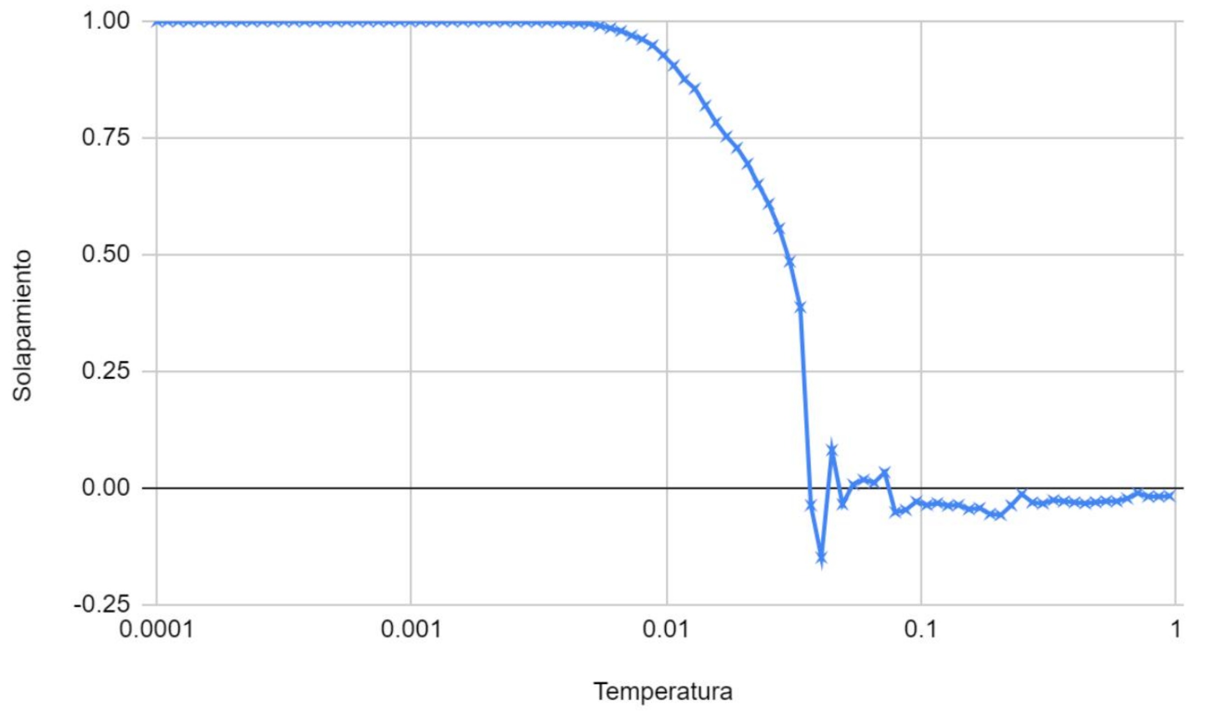


Figure 6: Overlap as a function of temperature, starting from the deformed pattern

Here we see that the system starting from the deformed pattern behaves very similar to the previous one, except that the pattern reaches the spurious state instead. However, the important thing is that even if we start with a configuration close to the pattern, we see that at high temperatures the network still does not recover it.

3.2 VARIOUS PATTERNS

In this section we will investigate what will happen when we introduce several patterns to the Hopfield network at the same time. We will study this in the same way that we have studied the case of the only pattern. However, we must comment that with several patterns *(P),* we will have an equal number of overlaps, which we define just as we did in the first section.

$$( ) = \frac{1}{{}^{2}\quad (1\ ÿ\ )}\ ÿÿ_{=1}(ÿ\ )_{=1}(ÿ\ ),\ = 1, 2, \ldots$$

We are going to use the four patterns in Figure 7, which you will notice that they all have a white background and the letter is in different parts of the frame. We have done it this way so that the patterns are very distinguishable from each other. We can understand why this is necessary by referring to the analogy of the energy surface. The patterns represent depressions in the surface and the net falls into one of them. If the patterns are too similar, the depressions will overlap and you will not be able to visually see what the final state consists of.

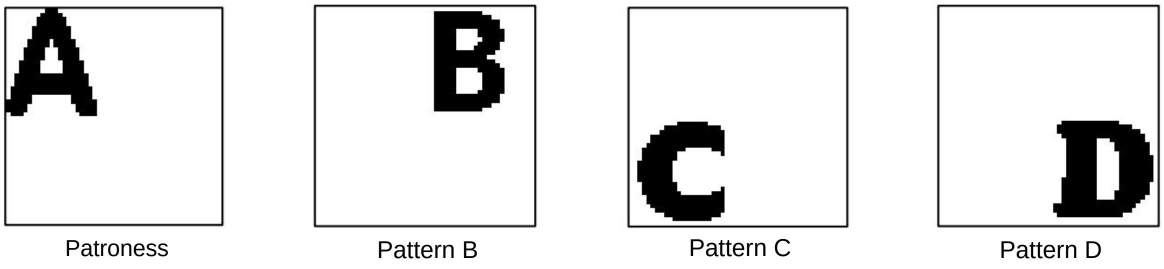


Figure 7: The patterns introduced into the Hopfield network

3.2.1 Random Initial State

In the first test, a random initial condition has been used and 20 Monte Carlo steps have been carried out for different temperatures.

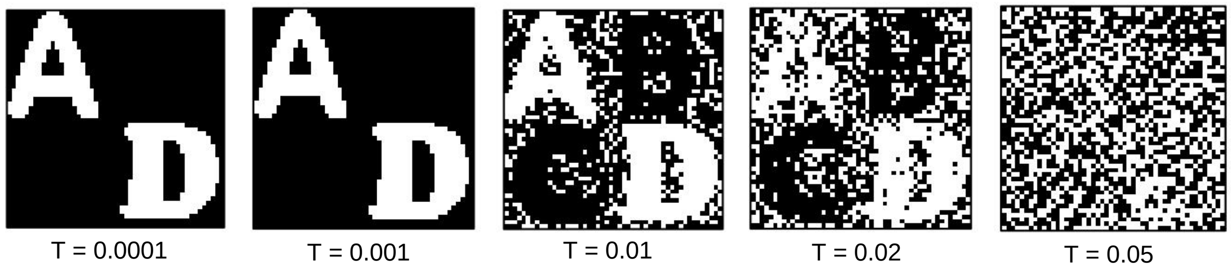


Figure 8: Results starting from a random initial state with various patterns

It is observed that after performing 20 Monte Carlo steps the network ends up in a superposition of the antipatterns of Pattern A and Pattern D, and then, with the increase in temperature, we see that the

Patterns C and D too. Next, the overlap is presented against the number of Monte Carlo steps (at T = 0.0001) and the temperature for each of the patterns (after 20 Monte Carlo steps).

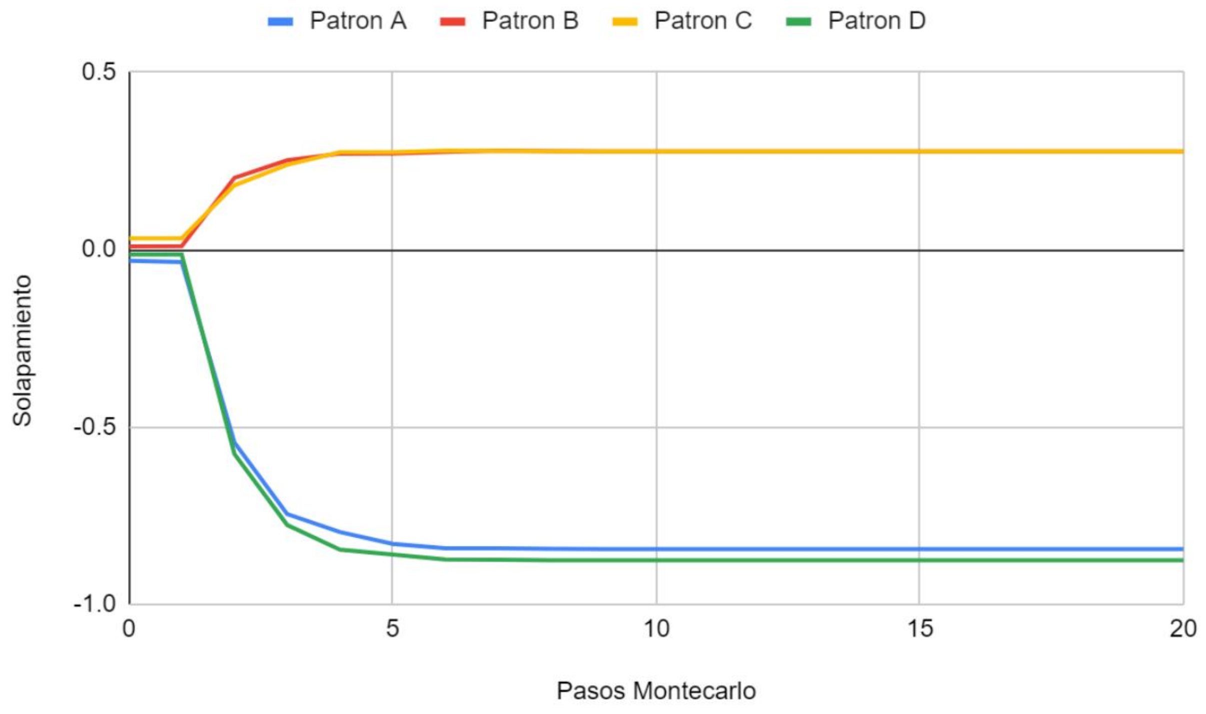


Figure 1: Overlap versus Monte Carlo steps from a random initial state and T = 0.0001

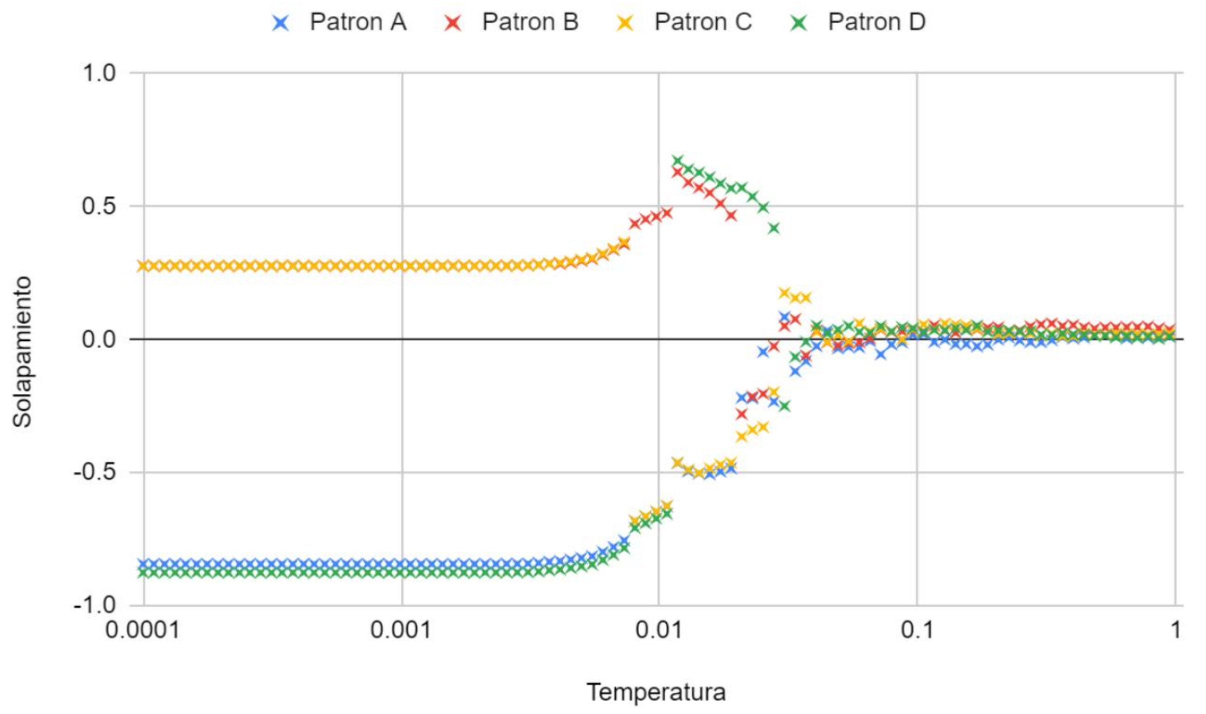
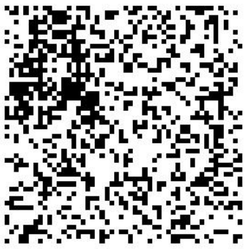


Figure 2: Overlap versus temperature from a random initial state, 20 Monte Carlo steps

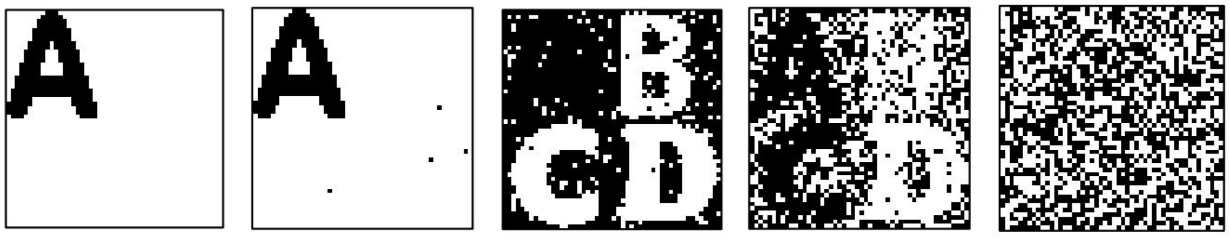
In these figures we can see that the network converges to a state that is the superposition of the stored patterns and that, as in the case of the only pattern, from T = 0.009, we enter a transition zone where the patterns are not recovered perfectly, but they continue to recover. Finally, at high temperatures, there is no recovery.

3.2.2 Initial state: Deformed pattern

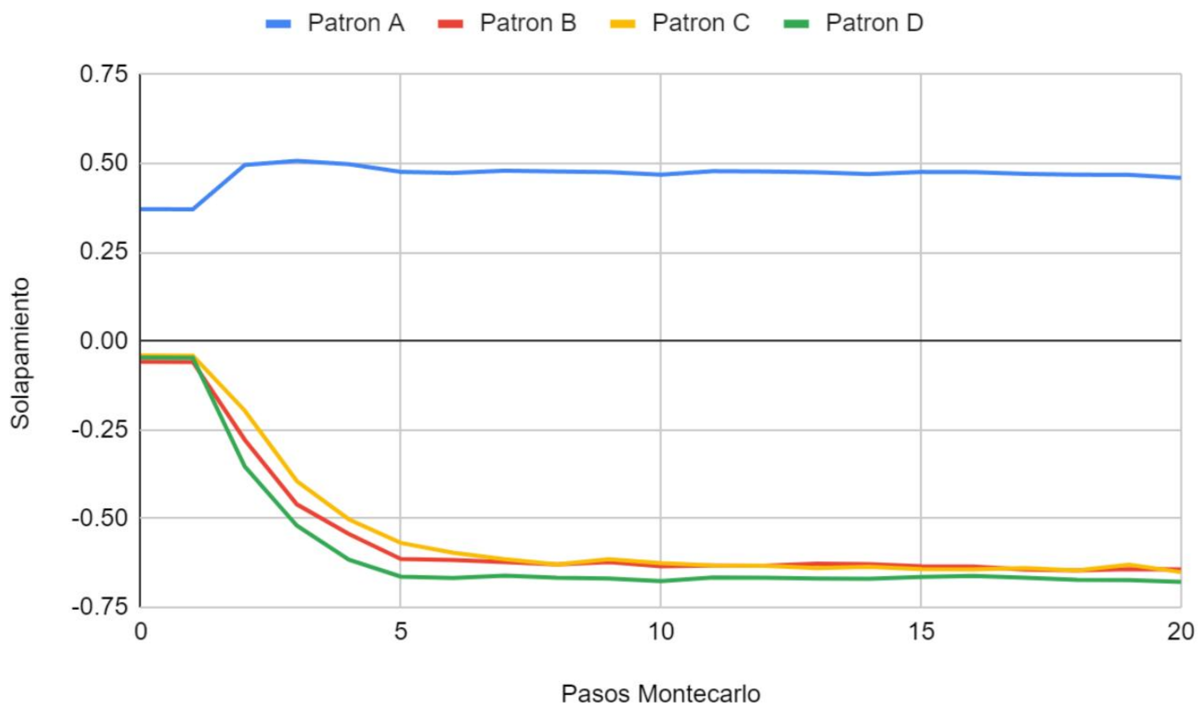
Using the programs *Hopmultdef.c* and *hopmultdeftemp.c* we generate the same as before, but with an initial state of pattern A deformed by 30% (just like the case of a pattern)

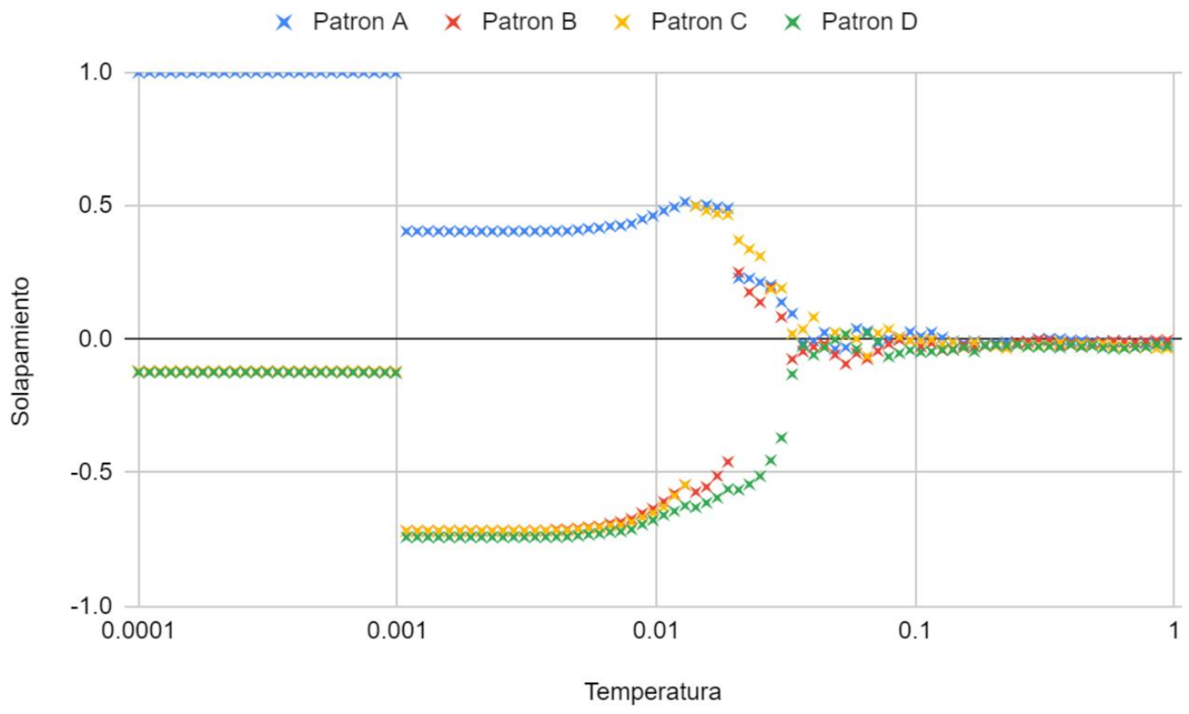


Initial state



T = 0.0001 T = 0.001 T = 0.01 T = 0.02 T = 0.05





The results are expected. At low temperatures, the network generates pattern A, but as the temperature rises, it recovers the other patterns.

3.3 MEMORY AS A FUNCTION OF THE NUMBER OF PATTERNS

In this section we will investigate the relationship between the total number of stored patterns and the system memory. We will do it at a low temperature (T = 0.0001), with a 20 x 20 grid and all patterns will be generated randomly.

We use the program *hopfieldpatrons.c.* This program starts with a pattern and calculates the overlap medium in Monte Carlo steps 20 -40. From there, repeat it, each time with one more pattern up to 100 patterns. We call a pattern remembered if the absolute value of the overlap is greater than 0.75, that is, if | ( )| > 0.75 we can say that μ is remembered by the network. As the number of patterns increases, we will see how many the network is able to remember. Next, we see the figure (INSERT).

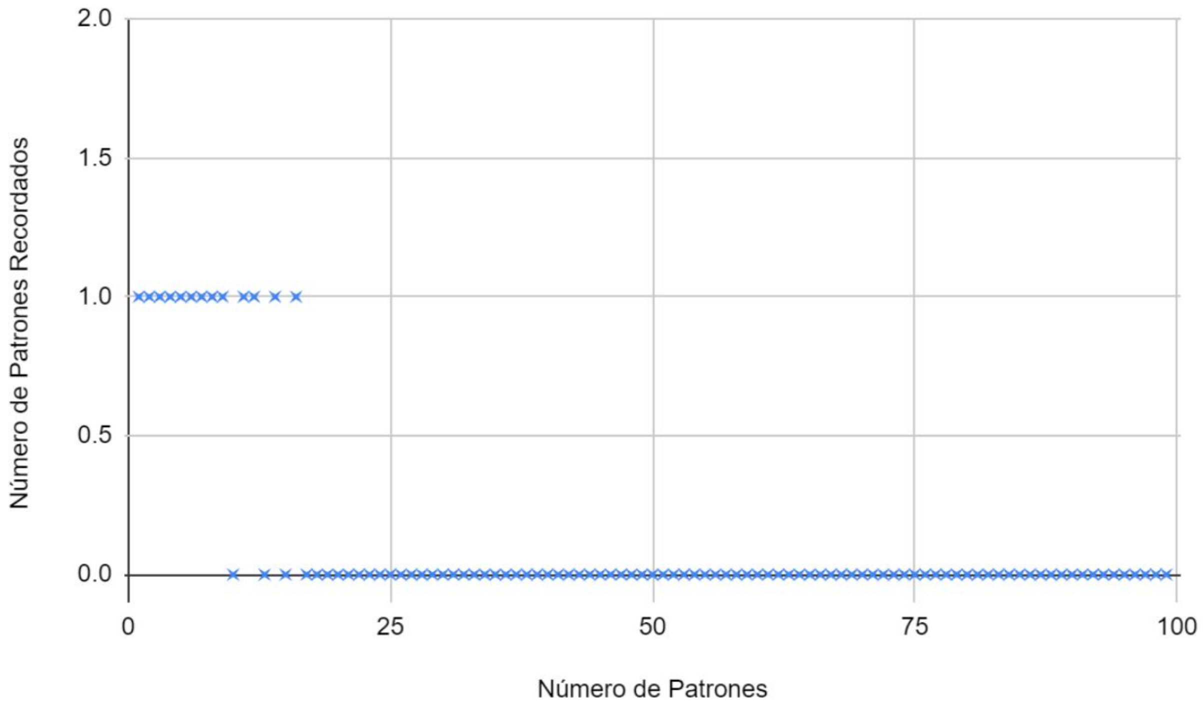


Figure 3: The number of patterns remembered as a function of the number of random patterns, $T = 0.0001$

We see in the figure (INSERT) that the network is only able to remember a single pattern and when it reaches 16 patterns, it is no longer able to remember any. We will also look at the maximum fraction the network can store = / so that all patterns can be ^{2} remembered. Since the only case in which all patterns were remembered was the case of a single pattern, *ac* will be 0.0025 or 1/400.

Finally, we noted that in the previous section the network was capable of remembering several patterns with the absolute value of their overlaps greater than 0.75. However, those patterns were not random patterns and shared a white background, which meant that their letters could appear without lowering the overlap of the other patterns too much. Here, with random patterns, it is very unlikely that we will find patterns similar enough for that to happen.

4 CONCLUSIONS

From these results we can draw some interesting conclusions. First, the Hopfield network is capable of remembering patterns stored at low temperatures. In this, we see that it can remember both the pattern itself and the anti-pattern, or spurious state. However, if the temperature is increased, it loses this ability and only recovers noise. This effect of high temperatures occurs the same when the initial state is very similar to the pattern.

The network is also capable of remembering multiple patterns, but only if they are similar enough, meaning that a single minimum in energy can be considered “remembering” multiple patterns. At higher temperatures, the network arrives at no pattern but at a state between patterns, remembering none but carrying elements of several.

For further research, it would be interesting to continue with these same questions, but varying the size of the network. In the various patterns section, we store four patterns based on the size of the network. This way four figures could fit so that they could all see. With a larger network, recall ability could increase and you would be able to remember more complex information.

5 REFERENCES

1. Hopfield, J.J. (1982). Neural networks and physical systems with emerging collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America, 79(8), 2554–2558.
2. Hopfield, J.J. (1986). Neural networks and physical systems with emerging collective computational abilities. In World Scientific Lecture Notes in Physics (pp. 411–415). WORLD SCIENTIFIC
3. Dennis, Simon (1997). The Hopfield Network: Descent on an Energy Surface. https://staff.itee.uq.edu.au/janetw/cmc/chapters/Hopfield/
4. Wikipedia contributors. (n.d.) Hopfield network. Retrieved July 8, 2023, from Wikipedia, The Free Encyclopedia website: https://en.wikipedia.org/w/index.php?title=Hopfield network&oldid=1023261754