Physical models of living systems, homework 6

Hopfield Networks

Gloria Isotton

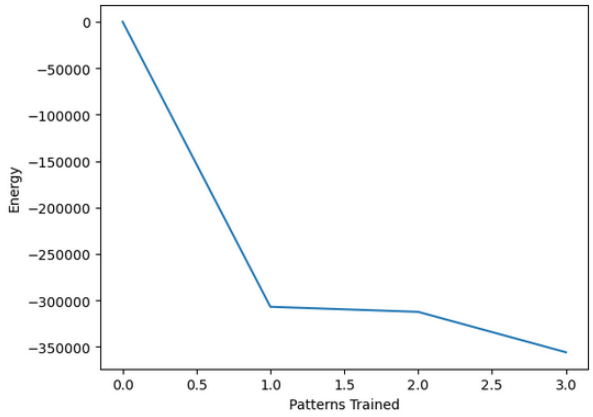
December 23, 2023

Introduction

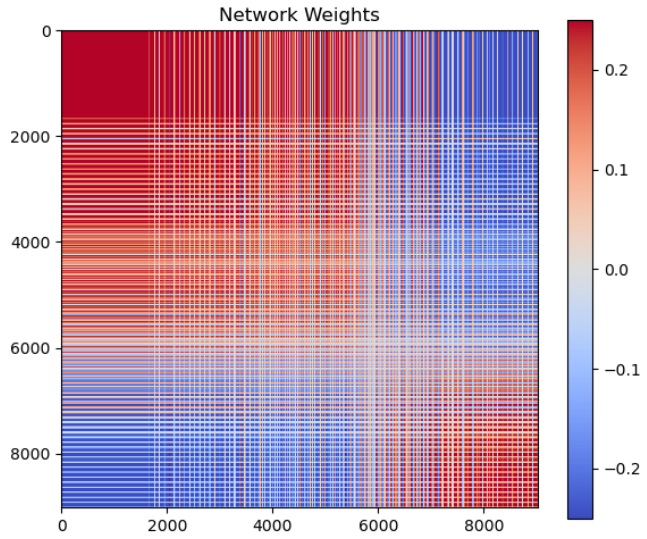
At its essence, a Hopfield Network is a computational model designed to restore undamaged data from corrupted versions of the same data. Introduced by Hopfield in 1982, this neural network works as a network with associative memory. To illustrate this mechanism, consider a situation where you catch a whiff of a familiar fragrance and immediately reminisce about a particular vacation by the sea. In this scenario, associative memory entails receiving an input, such as the scent triggering a fragment of a memory stored in your brain and subsequently retrieving an output that encapsulates the entirety of the stored memory, recounting the details of that seaside vacation. When the network receives an input, i.e., it is initialized to a certain state, the nodes within the network begin updating, eventually converging to a state corresponding to a previously stored pattern. The learning algorithm effectively encodes a specific pattern in the network by adjusting the weights. In classical Hopfield Networks these patterns are polar (binary), meaning $x_i \in (\{-1, 1\})^d$. Hopfield networks assign a scalar value known as *energy*, denoted by E, to each state within the network. This term is named after its tendency to decrease or remain constant during the update of network units. Moreover, through iterative updating, the network eventually reaches a state that represents a local minimum in the energy function. The training process of a Hopfield network involves the reduction of energy in states that the network should "remember". This enables the network to work as a content addressable memory system, wherein the network converges to a "remembered" state even when provided with only a partial input. Consequently, the network can recover from a distorted input to reach the trained state that bears the closest resemblance to the given input.

Training

I relied on a precompiled Python code implementing a basic asynchronous Hopfield network architecture with 95x95 neurons. This means that only one neuron is updated at a time during each iteration of the loop. To ensure accurate restoration of patterns, I configured the network to use 1000 timesteps. I considered essentially four distinct images for the training: one featuring a cat, another displaying the Batman logo and two photographs of an individual. The initial steps involved reading the available images, converting them into binary patterns and resizing them uniformly. Subsequently, I proceeded with the training of the network. Hereafter, I observed and analyzed the general behavior of the energy dynamics and the resultant weights.



(a) Energy varying pattern. As the network learns more patterns, the energy tends to decrease.



(b) Heatmap of the Network Weights: input neurons vs output neurons. Some patterns are present.

The second step consists in concentrating on one of the images used for training (the cat) and in attempting to obscure it with a mask (made of all ones). By varying the quantity of corrupted pixels at each iteration, I applied the algorithm to restore the original pattern. As visible from figure 2, the comparison between the fraction of corrupted pixels and the accuracy of the restored pattern reveals that, as expected, the more the image is corrupted, the lower the resemblance to the given input becomes.
In a similar way I applied the algorithm to restore the original pattern to a series of corrupted batman logo pictures. I trained the network with images featuring diverse corruptions (both in level and types of corruption), including:

1. an image of the Batman logo with a missing black circle around it;
2. another version of the Batman logo with pixels inverted and a missing circle;
3. an image with salt and pepper noise;
4. a picture missing of Batman, containing only the circle.

In all scenarios, the restored patterns demonstrate a reasonably good accuracy, except for the second case where almost all pixels remain inverted relative to each other, as shown in figure 3.

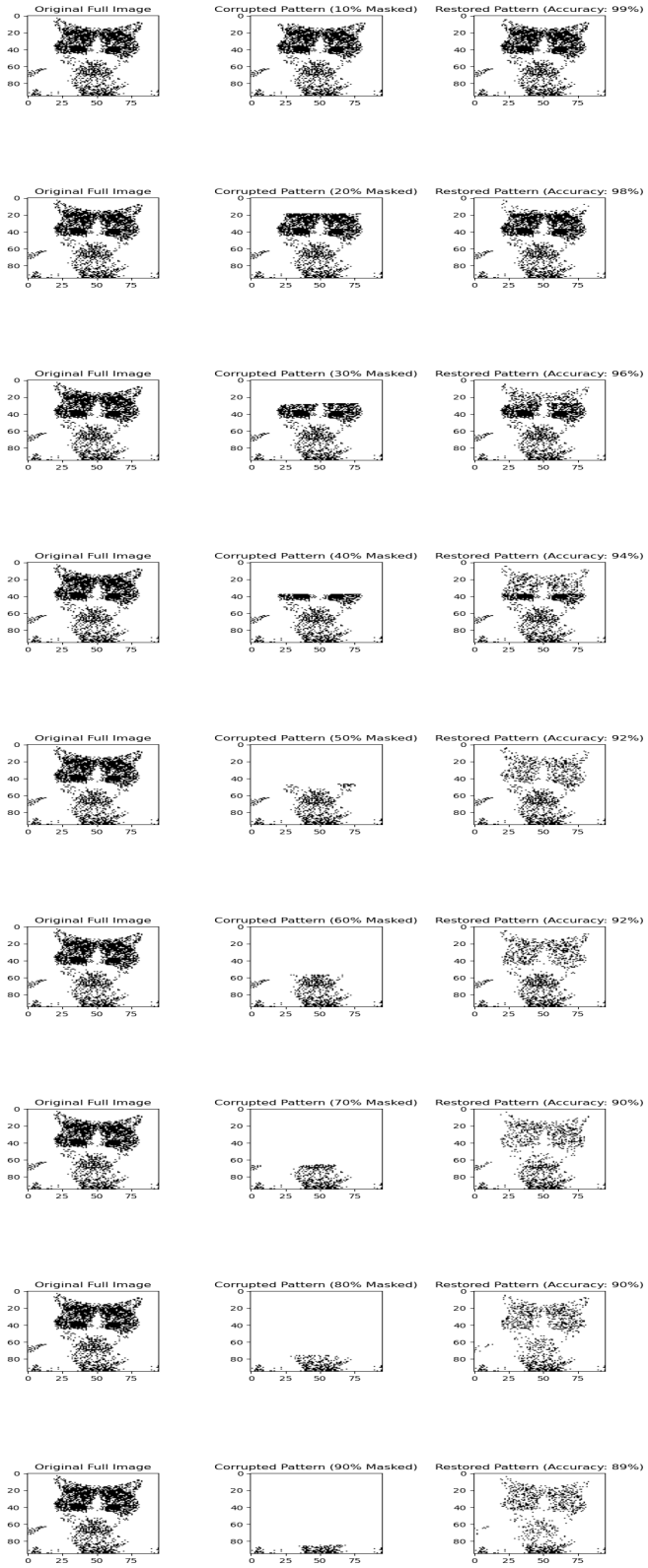


Figure 2: Cat pattern exemples.

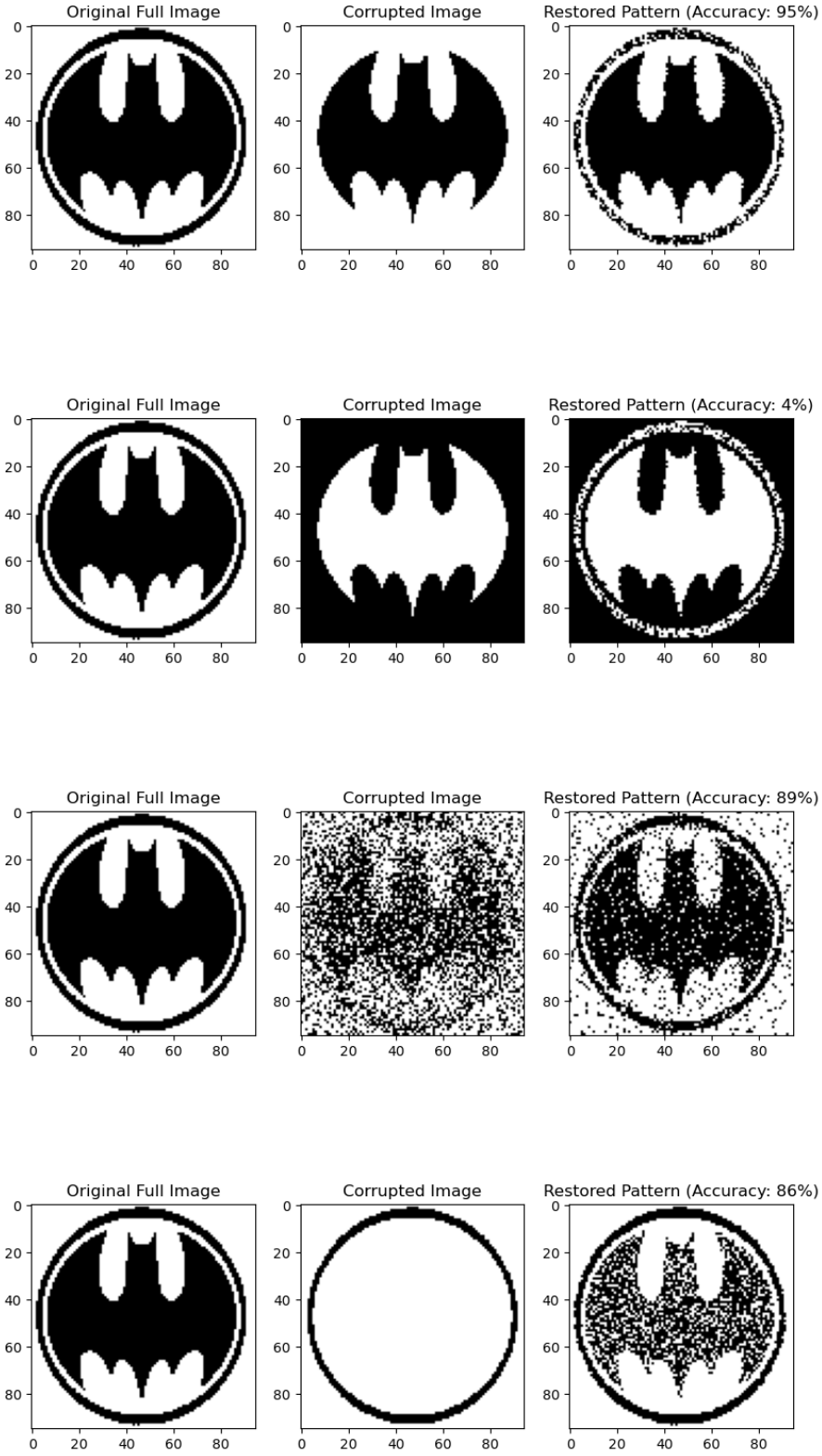


Figure 3: batman logo pattern exemples.