Hopfield Network

BY MARTINO FERRARI

1 Introduction to Hopfield Network

The Hopfield network (HN) is a simple artificial neural network able to memorize information and to restore it even after degradation.

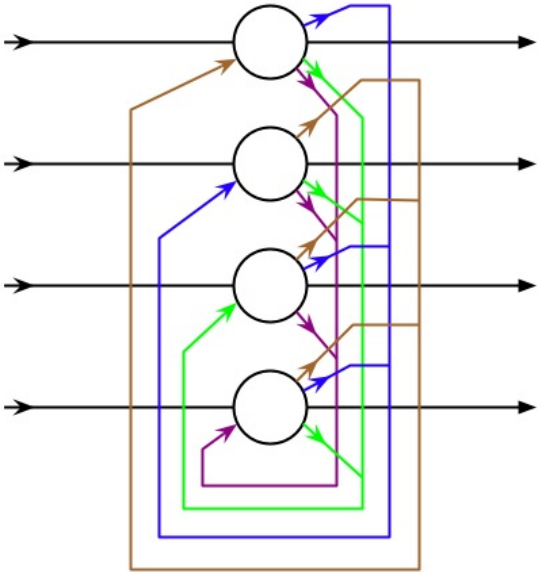

Figure 1. Representation of a 4-neurons Hopfield network (source: Wikipedia.org)

An Hopfield network is composed of N nodes (neurons) all interconnected (as showing in figure 1), the value of a node is binary, $s_i = \pm 1$. The interconnection between two nodes i and j has a certain weight $w_{i,j}$.

The values of the nodes evolve in time until it does not reach a stationary state, in particular the future value of the node i is given by:

$$s_i(t+1) = sgn\left(\sum_{j=1}^{N} w_{i,j} \cdot s_j(t)\right)$$

The weight matrix $W^{\{N \times N\}}$ is the result of the training process, that in the case of the Hopfield Networks is very simple and is done in one iteration.

Given M pattern $(S^1, S^2 ... S^M)$ of size N to memorize the weight $w_{i,j}$ is computed as:

$$w_{i,j} = \frac{1}{N} \sum_{k=1}^{M} s_i^k \cdot s_j^k$$

The goal is to have an highly overfitted training over the given pattern in order to be able to memorize and recognize it.

2 Implementation

In our case we were asked to implement a simple HN able to handle black and white images of 20×20 pixels ($N = 400$) in `python` (in used the version 3).

To do so I choose to use a object oriented approach, representing the HN as a class with different methods for the training and the recognition.

To measure how close the reconstructed image R is to the original O one I implemented two different error function, the simplest is implemented as:

$$e_s(O, R) = \frac{1}{N} \sum_{i=1}^{N} (|o_i - r_i|)$$

The second method however it take in account the fact that the reconstructed image can be the negative of the original (while the semantic of the image is conserved, the values of the vector are not conserved), and is implemented simply as:

$$e_i(O, R) = \min (e_s(O, R), e_s(O, -R))$$

Moreover I have chosen to implement a simple identify function $i(R, S^{\{M\}})$ to identify the reconstructed image R, choosing the closest training image S^i:

$$i(R, S^{\{M\}}) = \arg\min_{S^i} (e_i(S^i, R)|_{i=1}^{M})$$

This information show us how the HN tend to reconstruct the noisy image to a wrong training image S^i when the noise is to strong.

The given training pictures are shown in figure 2.

Figure 2. example images

However to be able to test the HN with a larger training set I added 21 images representing the Italian alphabet, shown in figure 3.

Figure 3. Italian alphabet

3 Results

Before starting to evaluate my HN, I wanted to have some visual feedback, so I choose to train the HN with the first 3 example images and to see if was able to memorize its and then to recognize the images (figure 4).

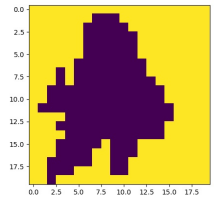

Figure 4. First success of the HN implementation

After succeeding in this first test I choose to see the result of the same operation but using a noisy image as source (I used the given method `Image:add_noise(noiseThreshold)`).

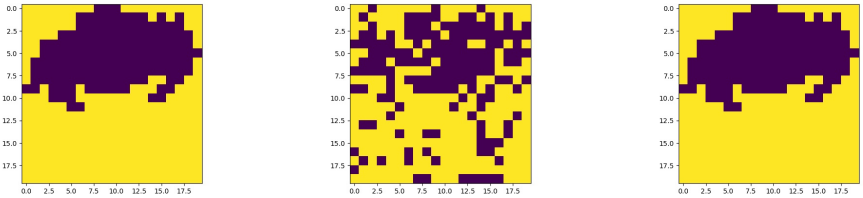

Figure 5. Reconstruction using the HN with 3 different image stored and `noiseThreshold` 0.2

The result of this second test is shown in figure 5, with a `noiseThreshold` of 0.2 the noisy image has significant loss (20% of the pixels has changed), and the HN, trained on 3 images, converge in 6 iterations to the successfully reconstructed image.

To understand better how the noise will affect the reconstruction of the image I choose to visualize the variation of the mean error (using the $e_i(O, R)$ function), error STD and unidentified ratio varying the `noiseThreshold`.

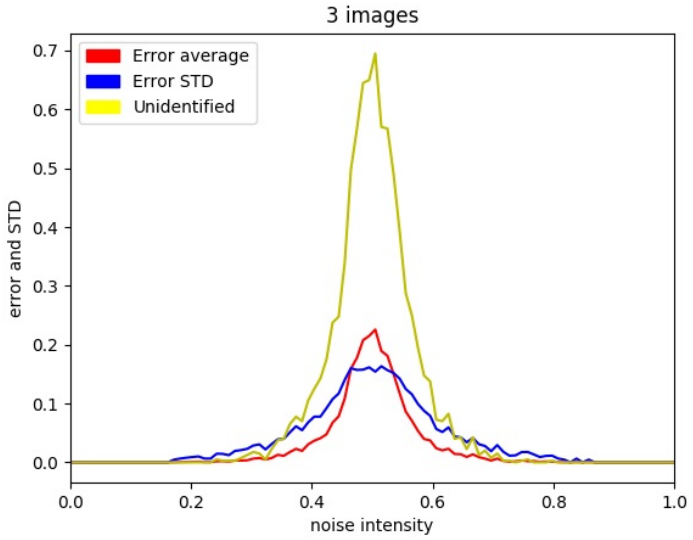

Figure 6. Mean Error, STD and unidentified ratio of the HN varying the `noiseThreshold`

For each noise level I executed 400 reconstruction and I used the result to compute the mean error, the error STD and the unidentified ratio (using the function $i(R, S^{\{M\}})$).

In figure 6 is possible to see that the mean error has some sort of exponential trend while the STD is more smooth, moreover the effect of the noise on the unidentified ratio is even stronger.

Finally, how I was expecting using the $e_i(O, R)$ function to compute the error results in a symmetric plot (a `noiseThreshold` of 0.5 means that half of the pixels have changed, while 1.0 means that all the pixels have changed, resulting in the negative version of the original).

The results of using the standard $e_s(O, R)$ function are shown in the foll wing figure:

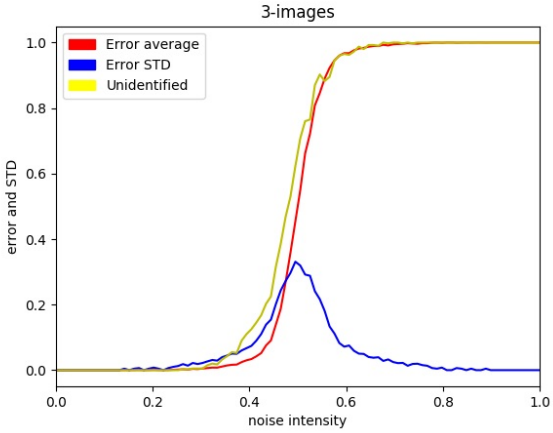

Figure 7. Same analysis of figure 6 but using $e_s()$ as error function

I was also interested to see how the number of memorized images would affect the sensibility to the noise, for this reason I fixed the `noiseThreshold` to 0.2 and using the alphabet series I varied the size M of $S^{\{M\}}$.

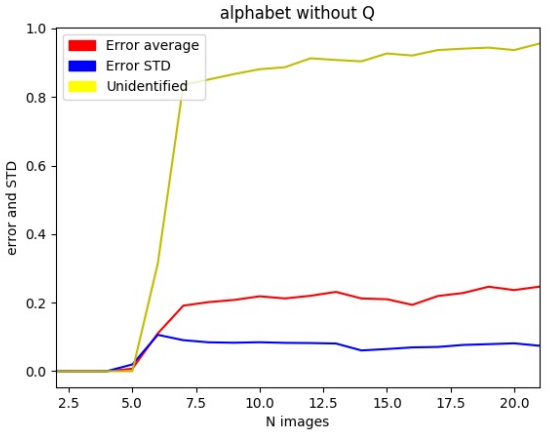

Figure 8. Mean error, error STD and unidentified ratio varying the size of the training set

The results, figure 8 (1000 runs for each point), show a highly non-linear behavior of the reconstruction quality against the size M of the training set. While with few images the mean error is 0 (or close to 0), with more then 5 images the mean error grow quickly, stabilizing around 0.2.

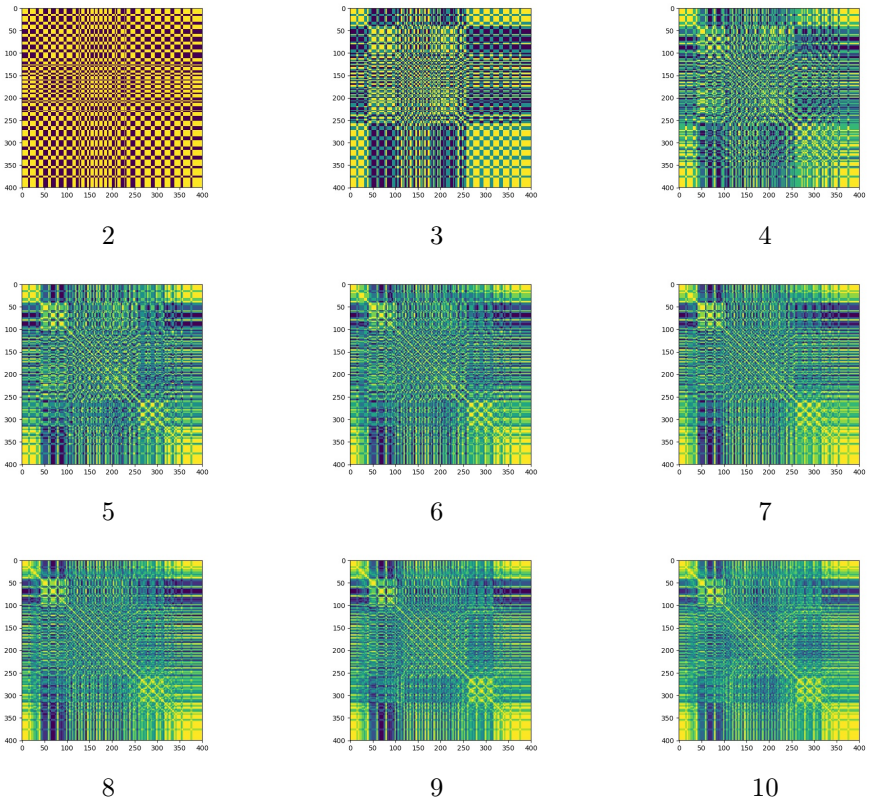

Figure 9. The weight matrix $W^{\{N\times N\}}$ in function of M

In figure 9 is shown the evolution of the weight matrix $W^{\{N\times N\}}$ while increasing the size of the training set M from 2 to 9. At first $W^{\{N\times N\}}$ changes substantially, but soon the matrix saturates and the changes are smaller and smaller. This explain the inability to reconstruct the image correctly as well as the non-linearity of the behavior (until reaching the saturation point the information can be stored without problem).

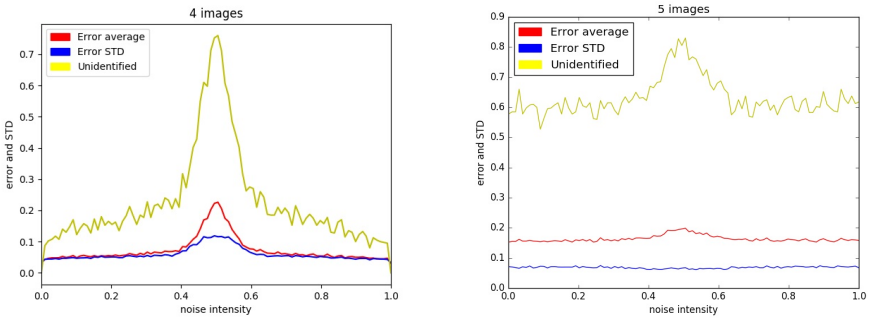

Figure 10. noise intensity analysis with $M = 4$ and 5

Figure 10 confirms the previous observations, showing how the mean error becomes independent from the noise with the increasing of the size of the training set.

To solve this problem it's possible to change the training algorithm adding a support matrix $Q^{\{M\times M\}}$, representing the similarity between the images ($q_{n,m} \in [-1, +1]$):

$$q_{n,m} = \frac{1}{N}\sum_{i=1}^{N} (s_i^n \cdot s_i^m)$$

This information is very useful to adapt the learning to the training set, as very different images are easily recognizable while closest ones are more difficult to distinguish.

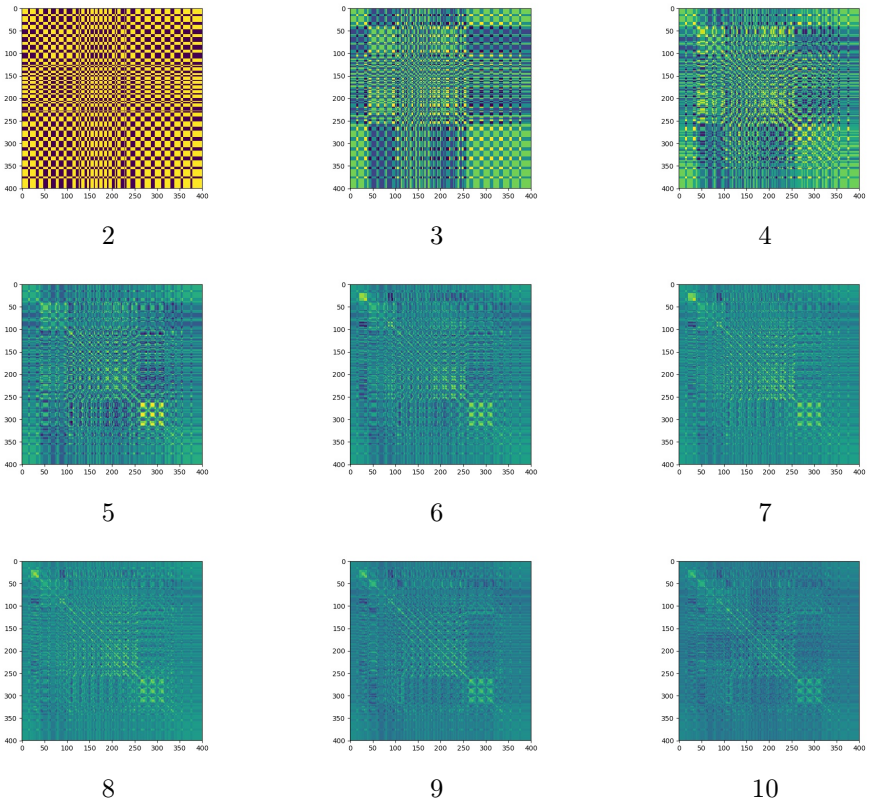

Figure 11. The weight matrix $W^{\{N\times N\}}$ in function of M trained using $Q^{\{M\times M\}}$

With the new algorithm the evolution of $W^{\{N\times N\}}$ is different, the changes between every step are smaller and in this way it does not saturate.

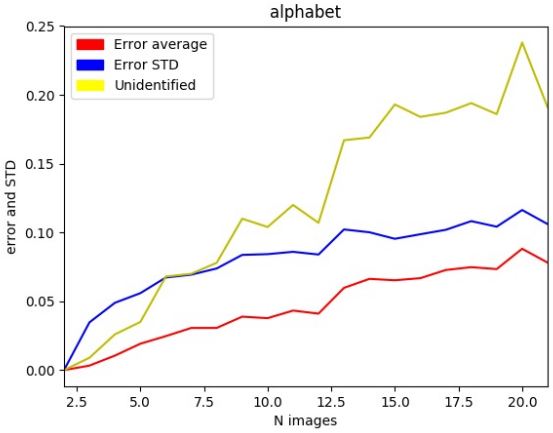

Figure 12. Error, STD and unidentified ratio evolution with the number of images (trained using Q)

For this reason the evolution of the mean error, error STD and unidentified ratio with the number of images is much more smoother even with `noiseThreshold` fixed at 0.4 (before was fixed at 0.2).

The same result can be observed in the noise domain:

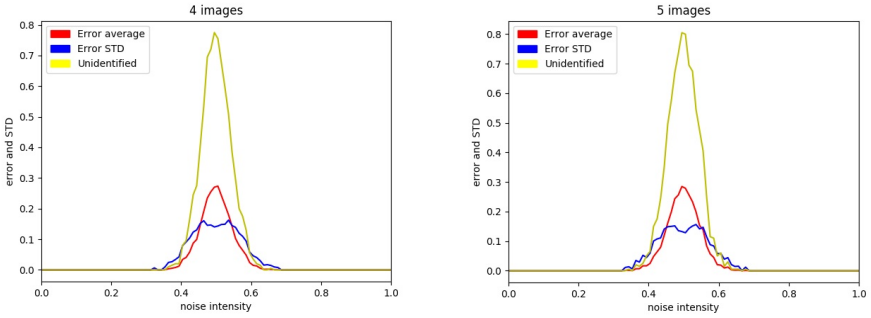

Figure 13. noise intensity analysis with $M = 4$ and 5 (trained using Q)

Even if a small degradation in the results is shown with the increasing of the size of the training set, now the effects are limited and the mean error depends primarily by the noise intensity and not from M anymore.

Even with 21 images the results persists:

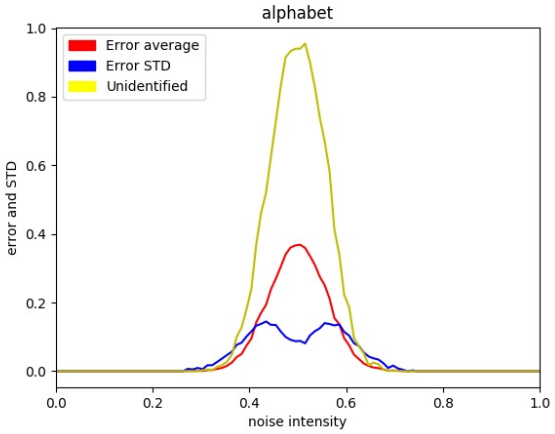

Figure 14. noise intensity analysis with $M = 21$ (trained using Q)

Visually, for the case of 21 images, the reconstruction works very well for `noiseThreshold`<0.35, for example for 0.2:

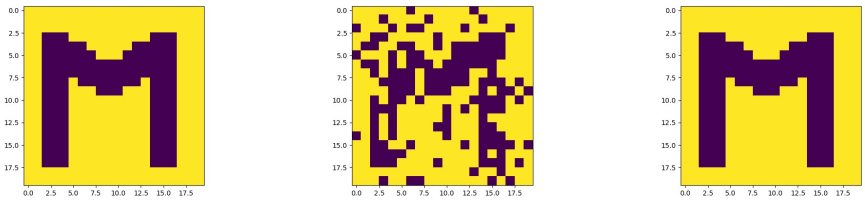

Figure 15. Reconstruction using the HN with 21 different image stored and `noiseThreshold` 0.2

Augmenting the resolution, and so the number of neurons of the HN, will increase even more the results, as the different S^i will have more details and so dissimilarities.

Overall the Hopfield network gave me very good results, especially after adding the support matrix $Q^{\{M \times M\}}$, while being extremely simple and fast. With only 400 neurons it was able to memorize and distinguish more the 20 images, even when severely degraded (`noiseThreshold` of 0.2 represents distortion of 160 of the 400 pixels).

However the HN has several limitation, for example even if the input image is not present in the training set the HN will converge to one of the images of the trainging set (it is not able to reject false inputs) as well as it is not able to recognize any geometric transformation (rotation, resize, etc).