Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield Networks

Tristan Perrot, Romain Darous and Mathis Pernin

February 22, 2023

Please be aware of the constraints for this document. The main intention here is that you learn how to select and organise the most relevant information into a concise and coherent report. The upper limit for the number of pages is 6 with fonts and margins comparable to those in this template and no appendices are allowed.
These short reports should be submitted to Canvas by the authors as a team before the lab presentation is made. To claim bonus points the authors should uploaded their short report a day before the bonus point deadline. The report can serve as a support for your lab presentation, though you may put emphasis on different aspects in your oral demonstration in the lab. Below you find some extra instructions in italics. Please remove them and use normal font for your text.

1 Main objectives and scope of the assignment

List here a concise list of your major intended goals, what you planned to do and what you wanted to learn/what problems you were set to address or investigate, e.g.
Our major goals in the assignment were to :

- Outline the fundamental principles governing the operation and functionality of autoassociative networks,
- Describe the training process of Hopfield networks,
- Explain the attractor dynamics in Hopfield networks, illustrating the concept of the energy function associated with these networks,
- Showcase the capabilities of autoassociative networks in pattern completion and noise reduction scenarios,
- Investigate the storage capacity of autoassociative memories, identifying features that contribute to its enhancement and discussing ways to increase it.

Then you can write two or three sentences about the scope, limitations and assumptions made for the lab assignment

Firstly, we'll implement and examine basic Hopfield networks using the Hebbian learning principle commonly used in recurrent networks. We will then build larger networks and look into the key principles of energy and attractors, as well as noise resistance. We will also compare synchronous and asynchronous updates. Finally we will investigate capacity and limitations of Hopfield networks, and study strategies to cope with it.

2 Methods

Mention here in just a couple of sentences what tools you have used, e.g. programming/scripting environment, toolboxes. If you use some unconventional method or introduce a clearly different performance measure, you can briefly mention or define it here.

For this lab, we will use the programming language **Python** and its libraries. More precisely, we will mainly use **torch** to build the Hopfield network. The graphs are displayed using the library **matplotlib.pyplot**. Some results are displayed using the library **pandas** as well.

3 Tasks and Questions - Hopfield Networks

*Make effort to be **concise and to the point** in your story of what you have done, what you have observed and demonstrated, and in your responses to specific questions in the assignment. You should skip less important details and explanations. In addition, you are requested to add a **discussion** about your interpretations/predictions or other thoughts concerned with specific tasks in the assignment. This can boil down to just a few bullet points or a couple of sentences for each section of your results. Overall, structure each Results section as you like. Analogously, feel free to group and combine answers to the questions, even between different experiments, e.g. with noise-free and noisy function approximation, if it makes your story easier to convey.*

*Plan your sections and consider making combined figures with subplots rather than a set of separate figures. **Figures** have to condense information, e.g. there is no point showing a separate plot for generated data and then for a decision boundary, this information can be contained in a single plot. Always carefully describe the axes, legends and add*

meaningful captions. Keep in mind that figures serve as a support for your description of the key findings (it is like storytelling but in technical format and academic style.

Similarly, use ***tables*** *to group relevant results for easier communication but focus on key aspects, do not overdo it. All figures and tables attached in your report must be accompanied by captions and referred to in the text, e.g. "in Fig.X or Table Y one can see ....".*

When you report quantities such as errors or other performance measures, round numbers to a reasonable number of decimal digits (usually 2 or 3 max). Apart from the estimated mean values, obtained as a result of averaging over multiple simulations, always include also ***the second moment****, e.g. standard deviation (S.D.). The same applies to some selected plots where* ***error bars*** *would provide valuable information, especially where conclusive comparisons are drawn.*

3.1 Convergence and attractors

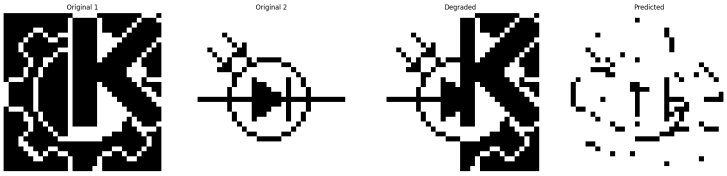
The given distorted patterns did converge to the same attractors `x1`, `x2` and `x3` than before : this means that the basin of attraction of these 3 attractors is large enough to attract 1-bit or 2-bit distorted patterns. After automating the search, we found 16 different attractors in this 8-units network. This means there are lots of others attractors in the network : indeed, when we make the starting patterns even more dissimilar to the stored ones we place them in different basins of attractions and thus get them attracted to a different pattern.

3.2 Sequential Update

In the larger 1024-units networks, we managed to display the patterns as 32x32 images. The first three patterns are stable. When we distort half of the first one, the network still converges to the right stored pattern. Yet when we mix the second and third pattern, we get attracted to a different (unknown) stable point. This was done using synchronous update, the speed of convergence is around 100 iterations, so it converges quite fast.

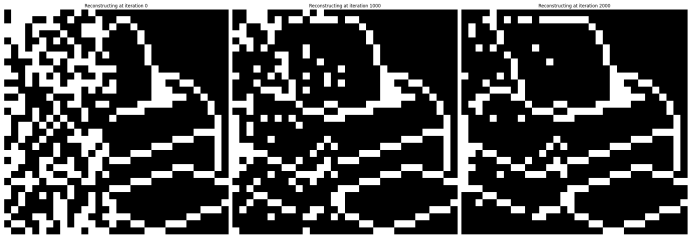
Figur 1: Original patterns

Figur 2: Recalled pattern starting from distorted pattern `p10`

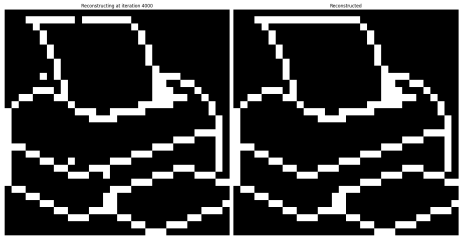


Figur 3: Recalled pattern starting from distorted pattern `p11`

However, when we use asynchronous updates, we obtain different results. Although the convergence is way slower (around 5000 iterations for the first image and 10000 iterations for the second one), we manage to converge to stored patterns for both degraded images. This is because asynchronous update allows the network to explore different paths and not get stuck in a local energy minima.



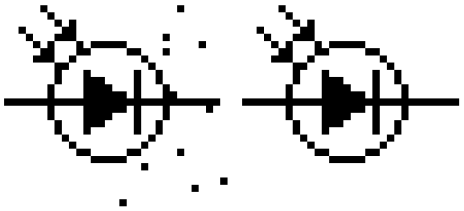
(a) Iteration 0 (b) Iteration 1000 (c) Iteration 2000



(d) Iteration 4000 (e) Iteration 5000

Figur 4: Reconstructing from pattern `p10` with asynchronous update

(a) Iteration 0 (b) Iteration 1000 (c) Iteration 2000



(d) Iteration 4000 (e) Iteration 10000

Figur 5: Reconstructing from pattern `p11` with asynchronous update

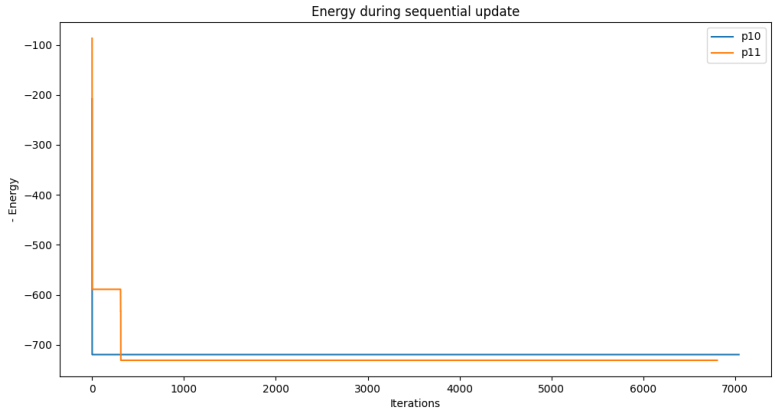
3.3 Energy

We computed the energies of the images and displayed them below. As expected, the energies of the stored patterns (the first three patterns) are lower than those of the last eight patterns which haven't been memorized by the network. We also displayed the evolution of the energy of the state of the network when starting from the two degraded images `p10` and `p11` with sequential update. We observe that the energy decreases very fast at first, but it still takes many more iterations to converge. This means the convergence towards a state very close to the stored one (and energy local minima) is very fast, but the final adjustments to get exactly the stored pattern are longer to happen.

```
Energy at p1 : -719.6953125
Energy at p2 : -682.8203125
Energy at p3 : -731.125
Energy at p4 : -360.240234375
Energy at p5 : -262.9453125
Energy at p6 : -341.6484375
Energy at p7 : -342.865234375
Energy at p8 : -85.7734375
Energy at p9 : -133.755859375
Energy at p10 : -207.990234375
Energy at p11 : -86.75
```

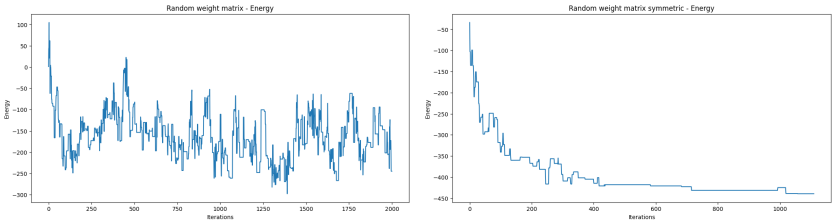
Figur 6: Energies of the images

When we don't initialize the weights by storing any pattern, and initialize them randomly instead, the evolution of the energy is chaotic. The property of the



Figur 7: Energies of the images

decreasing energy after each update doesn't stand anymore, thus the network explore a random across the basins of attraction and never reach and converge to an attractor. However, when we initialize the weight matrix to be a symmetric one, the output is different. In that case, we observe a general decreasing trends and the recall seems to converge. It makes sense as having a symmetric matrix of weights is one of the conditions of convergence of the Hopfield network.



(a) Random weights (b) Random symmetric weights

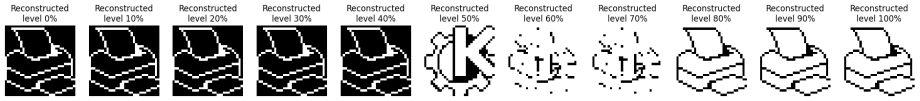
Figur 8: Evolution of the energy of the network state

3.4 Distortion Resistance

When we add progressive noise to stored patterns, we can evaluate the robustness of the network to noise. We didn't notice any difference between the three stored patterns in terms of robustness. When running several simulations, we can easily remove up to 20% of noise, as shown below. When we try to remove 80% or more of noise, we end up with the flipped image as it represents the same state of energy as the stored image but is closer to the original noisy pattern. However, it is harder to remove 30% or 40% and whether we succeed or not depends on the simulation : when we don't succeed, the network converges to an unknown pattern (nevertheless quite close to the targeted pattern). We can also notice that the recall never works well with 50% of noise as it is to far

away from the original clean pattern, but we always converge to another stored pattern.

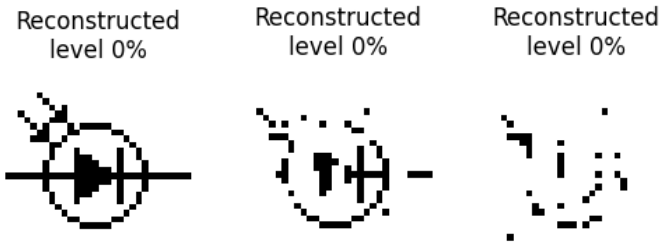
Figur 9: Original noisy patterns : noise goes from 0% to 100%



Figur 10: Reconstructed patterns

3.5 Capacity

When increasing the number of memories stored in the network, we reach its limit capacity. As soon as we store a fourth memory, the network is unable to recall any of the three initial patterns. This is called catastrophic forgetting. Nevertheless, the drop in performance is gradual : the more memories are added, the further the images towards which the network converges are from the image to be recalled and therefore more difficult to recognise.



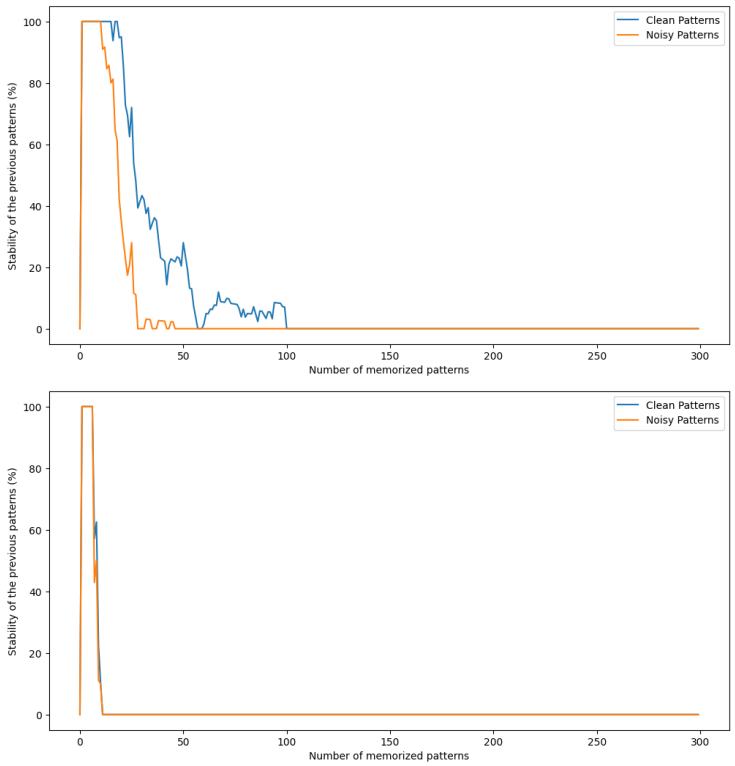
(a) Reconstructed `p3` with 3 stored patterns (b) Reconstructed `p3` with 4 stored patterns (c) Reconstructed `p3` with 7 stored patterns

Figur 11: Reconstructing from pattern `p3` with different storage size

Yet if we take random patterns instead of chosen images, the capacity of the network is way higher. After running several simulations, we noticed that we could store up to 140 patterns in the network before it suffers from catastrophic forgetting (which complies from the rule of a capacity close to $0,138N$). This is a surprising surge of capacity between images and random patterns. We think it could be due to the images used at first being highly biased, contrary to the random patterns that are regularly distributed.

In the next task of the lab, we were asked to build a 100-unit network and observe after each time the network would memorize a new pattern if the previous

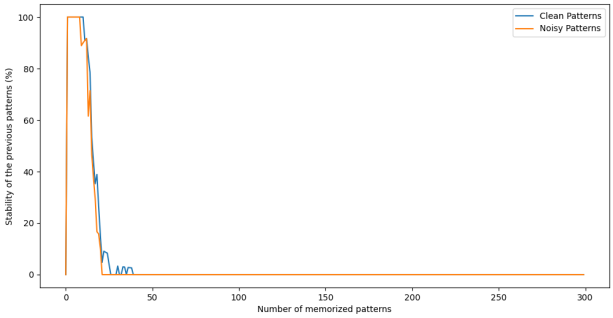
ones are still stored, and further add noise and bias. We displayed below the curves that we obtain, for both unbiased and biased random patterns. It seems our Hopfield Network can learn the patterns very well at first. Nevertheless, as excpected we observe the phenomena of catastrophic forgetting : the curve presents a drop in memory around 14 stored patterns for the unbiased random patterns, and even sooner with the biased ones. Indeed we believe that the biased patterns are more complex thus they require more neurons to be accurately represented. Finally, if we remove self-connections and thus simplify the architecture, we observe that the clean patterns and noisy patterns curves look the same.



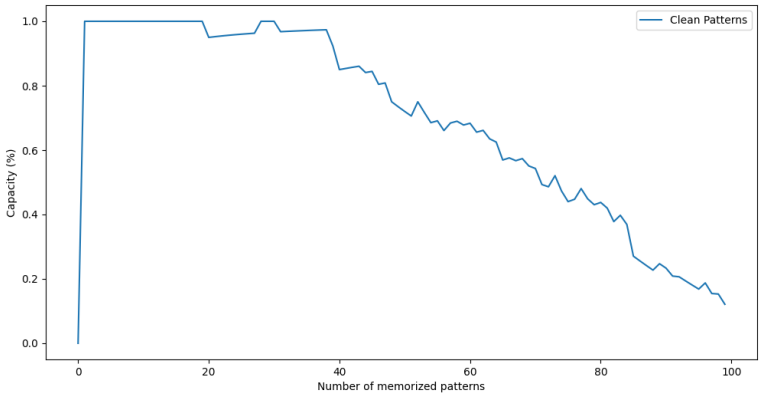
Figur 12: % of stable points with 300 unbiased and biased random patterns

3.6 Sparse Patterns

In this task, we were asked to modify the weight matrix and the update rule according to the activity of the data in order to mitigate the effects the bias in the data on the memorizing. We displayed our results below. As expected, catastrophic forgetting is less present than before, and the network is able to store more biased random pattern than in the previous section (approximately 40, against 10 previously).



Figur 13: % of stable points with no self-connections



4 Final remarks *(max 0.5 page)*

Please share your final reflections on the lab, its content and your own learning. Which parts of the lab assignment did you find confusing or not necessarily helping in understanding important concepts and which parts you have found interesting and relevant to your learning experience?
Here you can also formulate your opinion, interpretation or speculation about some of the simulation outcomes. Please add any follow-up questions that you might have regarding the lab tasks and the results you have produced.

In this lab, we explored various aspects of Hopfield networks, including convergence, attractors, sequential updates, energy dynamics, distortion resistance, and storage capacity. The distorted patterns consistently converged to attractors, showcasing the network's robustness to noise up to a certain threshold. Sequential updates provided insights into energy evolution and the impact of symmetric weight matrices, revealing regular oscillations. Catastrophic forgetting became apparent as the number of stored memories increased, with differences observed between biased and random patterns. However, challenges were

encountered in implementing a 100-unit network for memory observation, and attempts to mitigate bias effects in sparse data did not yield expected improvements. Despite these challenges, the study shed light on both the strengths and complexities of Hopfield networks in pattern storage and retrieval.