Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield networks

Widad Farhat, Xiaolei Liu and Saly Al Masri

February 18, 2025

Main objectives and scope of the assignment

Our major goals in the assignment were:

- to implement and understand the principles of Hopfield networks as autoassociative memory systems
- to investigate pattern completion and noise reduction capabilities in Hopfield networks
- to analyze network dynamics through energy functions and convergence properties
- to explore storage capacity limitations and methods to improve them
- to examine the performance of both synchronous (Little model) and asynchronous (sequential) updates

The scope of this assignment was limited to discrete Hopfield networks using the Hebbian learning rule. We focused on bipolar (-1,1) and binary (0,1) representations, particularly examining small-scale networks (8 neurons) and scaling up to larger networks (1024 neurons) for pattern recognition. Our investigation assumed symmetric weight matrices and did not explore continuous-valued networks or alternative learning rules.

Methods

We implemented our Hopfield network simulations using Google Colab as our primary development environment. The implementation relied on matrix operations for efficient computation of weights and state updates.

1 Convergence and Attractors

1.1 Testing Pattern Recall with Noise

In this section, we tested the Hopfield network's ability to recall stored patterns from noisy inputs. We defined three distorted versions of the original patterns:

- **x1d**: A one-bit error version of **x1**
- **x2d**: A two-bit error version of **x2**
- **x3d**: A two-bit error version of **x3**

We trained the network using the original patterns and then tested the recall with the distorted inputs. The results showed that the network successfully recalled the original patterns despite the noise.

This demonstrates that the Hopfield network can effectively correct small errors in the input patterns and converge to the correct stored patterns.

1.2 Checking the Number of Attractors

Next, we investigated how many attractors exist in the network. An attractor is a stable state that the network converges to when presented with an input. We generated all possible 8-bit binary patterns and tested them to see which patterns the network would converge to. The results showed that there are **14 attractors** in the network.

This means that in addition to the three stored patterns, there are 11 other stable states (spurious attractors) that the network can converge to. These spurious attractors are not part of the original training set and can be seen as "false memories" stored by the network.

1.3 Investigating Heavily Distorted Inputs

We then tested the network's ability to handle heavily distorted inputs, where more than half of the bits were flipped. We defined new patterns with 5-bit errors:

```
patterns_d2 = np.array([
    [1,  1, -1,  1, -1, -1, -1,  1],  # x1d2
    [1,  1,  1,  1,  1,  1, -1, -1],  # x2d2
    [1, -1, -1,  1,  1,  1, -1,  1]   # x3d2
])
```

The results showed that the network failed to recall the original patterns when the input was heavily distorted.

This indicates that the network's ability to correct errors is limited. When the input is too dissimilar from the stored patterns, the network may converge to a spurious attractor or fail to recall the correct pattern.

1.4 Visualizing Attractor Occurrences

We also visualized the distribution of attractor occurrences in the network. The bar plot below shows the frequency of each attractor: This visualization helps us

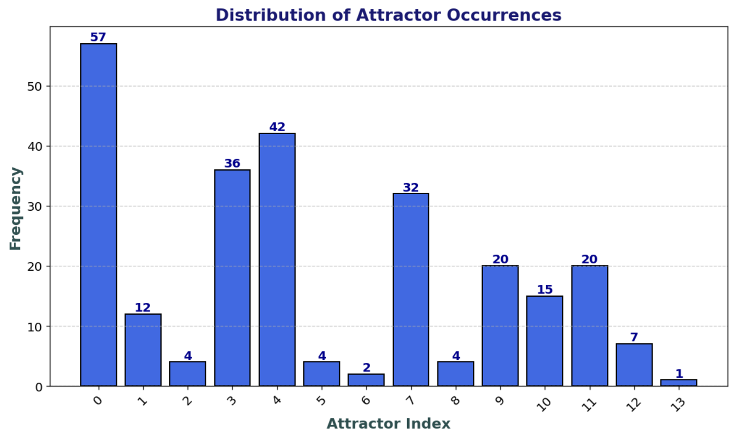


Figure 1: Distribution of attractor occurrences in the Hopfield network. The x-axis shows the attractor index, and the y-axis shows how many initial states converge to each attractor.

understand how often each attractor is reached when the network is presented with random inputs.

2 Sequential Update

2.1 Loading and Visualizing Patterns

In this section, we switched to a larger 1024-neuron network and worked with image patterns stored in the pict.dat file. We loaded the data and reshaped it into 1024-dimensional patterns, We then visualized the first three patterns as 32×32 images:

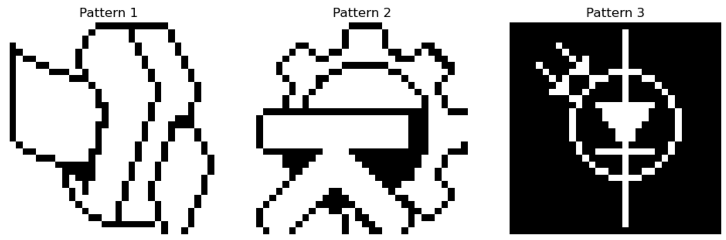


Figure 2

2.2 Training the Network

We trained the Hopfield network on the first three patterns (p1, p2, and p3) and checked if the patterns remained stable after retrieval, The results confirmed that the patterns were stable.

2.3 Pattern Completion

We tested the network's ability to complete degraded patterns. We used p10 (a degraded version of p1) and p11 (a mixture of p2 and p3) as inputs,The results showed that the network was able to recover the original patterns from the degraded inputs, demonstrating its ability to perform pattern completion.

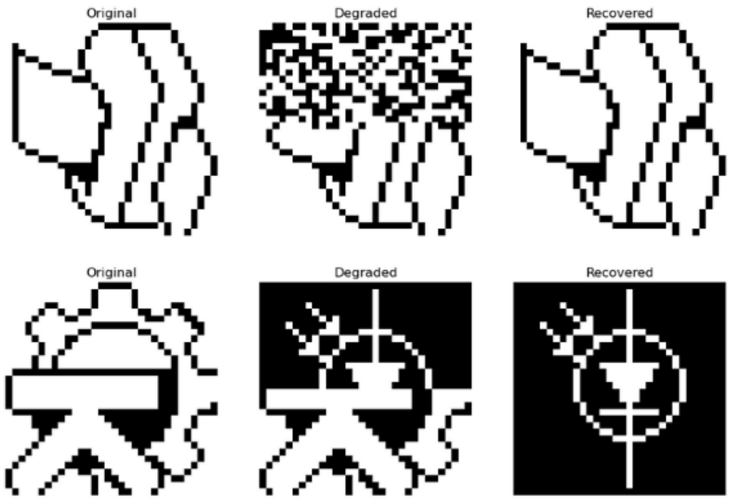


Figure 3

2.4 Sequential Update with Random Unit Selection

Finally, we tested the network's behavior when units were updated sequentially in random order. The following figure showed how the network converges to one stable state in one update, witch is a pathological pattern.

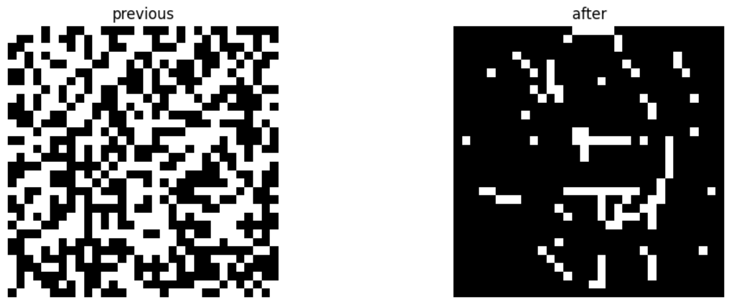


Figure 4: Pattern recovered from randomness

3 Energy

In this section, we explored the concept of energy in Hopfield networks. The energy function is a crucial aspect of Hopfield networks, as it helps us understand the network's convergence behavior. The energy function is defined as:

$$E = -\sum_i \sum_j w_{ij} x_i x_j$$

This energy function decreases as the network updates its states, ensuring that the network converges to a stable state (an attractor). Below, we discuss the results of the experiments related to energy.

3.1 Energy at Different Attractors

We calculated the energy at each of the stored patterns (attractors) in the network. The energy values for the attractors **p1** to **p9** were calculated and analyzed.

Observations:

- The energy values for **p1**, **p2**, and **p3** are lower than those for **p4** to **p9**, indicating that the first three patterns are more stable attractors.
- The energy values for **p4** to **p9** are identical, suggesting that these patterns may not be as distinct or stable as the first three.

3.2 Energy at Distorted Patterns

We also calculated the energy for distorted patterns **p10** and **p11**.

Observations:

- The energy of **p10** matches the energy of **p1**, indicating that the network successfully restored **p10** to the original pattern **p1**.
- The energy of **p11** is close to the energy of **p2**, suggesting that the network partially restored **p11** but did not fully converge to **p2**.
- This confirms that the network can correct distortions and converge to the correct attractor, but the success depends on the level of noise.

3.3 Energy Evolution During Sequential Updates

We tracked the energy changes during sequential updates for pattern **p11**.

Observations:

- The energy decreases rapidly in the first iteration, indicating that the network quickly corrects the distorted pattern.
- After the first iteration, the energy stabilizes at -1462.25, which is the energy of the attractor **p3**.
- This demonstrates that the network converges to a stable state (attractor) after a few iterations.

Visualization:

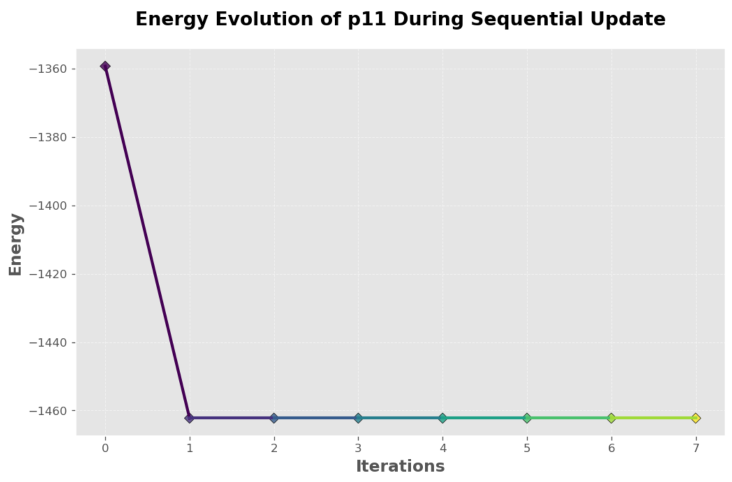


Figure 5: Energy Evolution of p11 During Sequential Update

3.4 Random Weight Matrix

We generated a weight matrix with normally distributed random numbers and observed the energy evolution during sequential updates.

Observations:

- The energy fluctuates significantly and does not converge to a stable value.
- This behavior is expected because the random weight matrix does not correspond to any meaningful attractors.
- The network fails to stabilize, and the energy does not decrease monotonically.

Visualization:

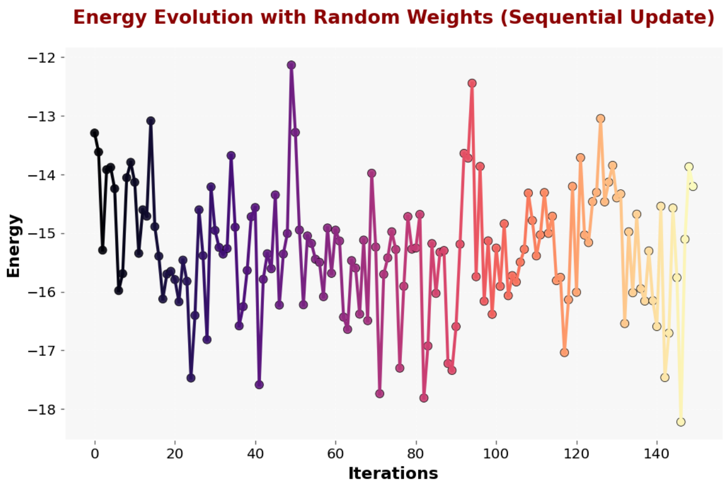


Figure 6: Energy Evolution with Random Weights (Sequential Update)

3.5 Symmetric Weight Matrix

Finally, we made the weight matrix symmetric by setting $w = 0.5 \times (w + w^T)$ and observed the energy evolution.

Observations:

- With a symmetric weight matrix, the energy decreases monotonically, and the network converges to a stable state.
- This behavior is consistent with the theoretical properties of Hopfield networks, where symmetric weights ensure convergence to a local minimum of the energy function.

Visualization:

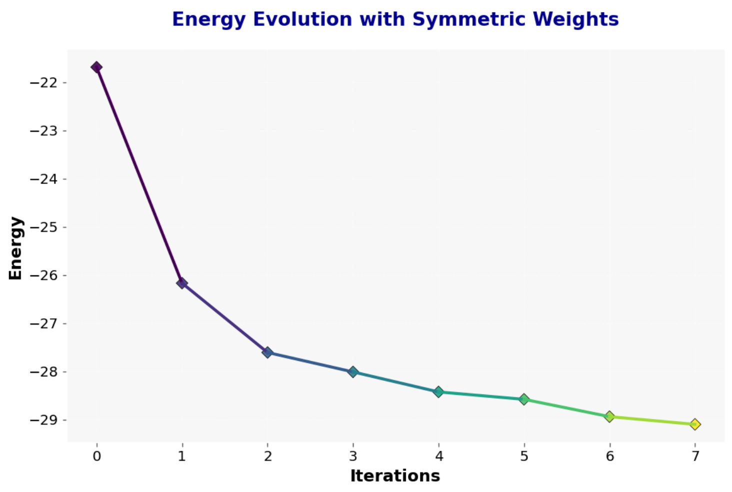


Figure 7: Energy Evolution with Symmetric Weights (Sequential Update)

3.6 Summary

In this section, we explored the energy dynamics of Hopfield networks. We observed that:

- The energy at attractors corresponds to stable states, and distorted patterns converge to these attractors.
- The energy decreases monotonically during sequential updates, ensuring convergence.
- Random weight matrices lead to unstable energy fluctuations, while symmetric weight matrices ensure stable convergence.

These experiments confirm the importance of the energy function in understanding the convergence behavior of Hopfield networks.

4 Distortion Resistance

In this section, we investigated the Hopfield network's ability to resist noise and distortion. The goal was to determine how much noise the network can handle before failing to restore the original patterns. We also examined whether there are differences in noise tolerance between the three attractors (**p1**, **p2**, and **p3**) and whether the network always converges to the correct attractor. Additionally, we explored whether extra iterations beyond a single-step recall improve the network's performance.

4.1 How Much Noise Can Be Removed?

We tested the network's ability to restore patterns with varying levels of noise, ranging from 0% to 100%.

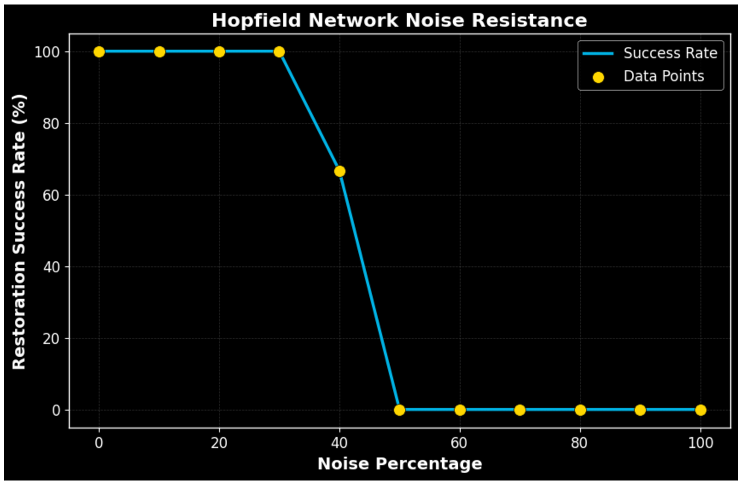


Figure 8: Network Performance vs. Noise Level

Observations:

- **0% to 30% Noise:** The network successfully restored all three patterns (**p1**, **p2**, and **p3**) with 100% accuracy.
- **40% Noise:** The network failed to restore **p1**, but **p2** and **p3** were still successfully restored.
- **50% Noise and Above:** The network failed to restore any of the patterns, indicating that the noise level was too high for the network to correct.

Conclusion: The Hopfield network can handle up to **30% noise** with perfect restoration. Beyond this threshold, the network's ability to restore patterns deteriorates, and by **50% noise**, it fails completely.

4.2 Noise Tolerance of Different Attractors

We examined whether the three attractors (**p1**, **p2**, and **p3**) have different levels of noise tolerance.

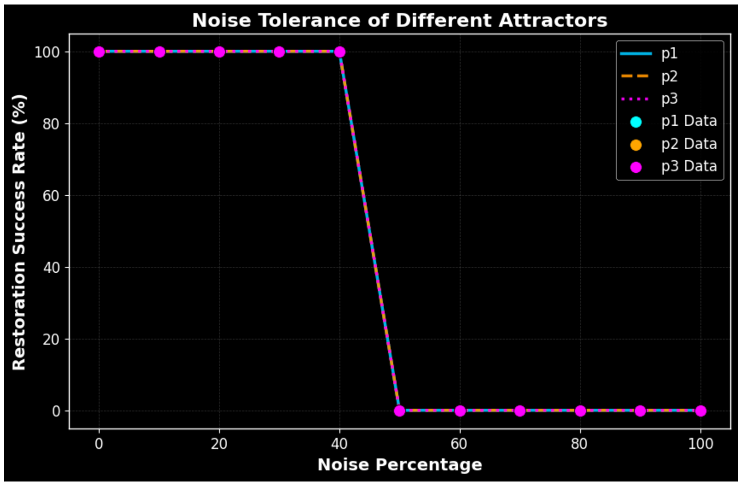


Figure 9: Noise Tolerance Comparison Between Attractors

Observations:

- **p1 and p3:** These patterns showed similar noise tolerance, with successful restoration up to **40% noise**.
- **p2:** This pattern was slightly less robust, failing at **40% noise** while **p1** and **p3** were still successfully restored.

Conclusion: There is a slight difference in noise tolerance between the attractors. **p1** and **p3** are more robust to noise compared to **p2**.

4.3 Network Convergence to Correct Attractor

We tested whether the network always converges to the correct attractor when presented with noisy patterns. The results demonstrated varying behavior at different noise levels:

Observations:

- **0% to 30% Noise:** The network always converged to the correct attractor.
- **40% Noise:** The network failed to restore **p1**, but it did not converge to the wrong attractor.
- **50% Noise:** The network failed to restore **p1**, and **p2** and **p3** converged to the wrong attractors.
- **60% Noise and Above:** The network failed to restore any of the patterns, and it did not converge to any of the stored attractors.

Conclusion: The network reliably converges to the correct attractor for noise levels up to **30%**. Beyond this, the network may fail to restore the pattern or converge to the wrong attractor.

4.4 Impact of Extra Iterations

We tested whether extra iterations beyond a single-step recall improve the network's performance.

Observations:

- The energy decreases rapidly in the first iteration and stabilizes thereafter.
- Extra iterations do not significantly improve the network's performance, as the energy reaches its minimum after the first iteration.

Conclusion: Extra iterations beyond the first do not provide significant benefits in terms of energy minimization or pattern restoration.

4.5 Number of Attractors

We investigated the number of attractors in the network by testing a large number of random patterns.

Observations:

- The network has **14 attractors**, which include the three stored patterns (**p1**, **p2**, and **p3**) and **11 spurious attractors**.

- Spurious attractors are stable states that do not correspond to any of the stored patterns.

Conclusion: The Hopfield network can store multiple patterns, but it also creates spurious attractors, which can lead to incorrect pattern recall.

4.6 Summary

In this section, we explored the Hopfield network's resistance to noise and distortion. Key findings include:

- The network can handle up to **30% noise** with perfect restoration.
- **p1** and **p3** are more robust to noise compared to **p2**.
- The network reliably converges to the correct attractor for noise levels up to **30%**.
- Extra iterations beyond the first do not significantly improve performance.
- The network has **14 attractors**, including spurious attractors that can lead to incorrect pattern recall.

These results highlight the strengths and limitations of Hopfield networks in handling noisy inputs and restoring stored patterns.

5 Capacity

When only three patterns are added, the model initially achieves a recall accuracy of 100%. However, as the noise level increases to approximately 40-50%, the accuracy drops drastically, reflecting the expected impact of noise on performance.

With four patterns, recall accuracy changes significantly—none of the patterns maintain a perfect recall accuracy, indicating that the number of stored patterns starts to affect overall performance.

As more patterns are introduced (five or six), the results deteriorate further, suggesting that the model struggles with increased pattern complexity. Interestingly, it was observed that pattern P3 had the highest accuracy, highlighting a unique behavior in recall performance.

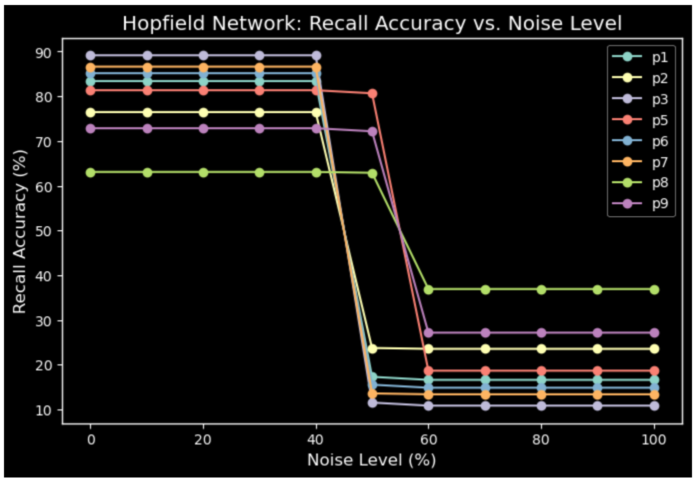


Figure 10: Relationship between the number of stored patterns and recall accuracy. The plot illustrates whether the drop in performance occurs gradually or abruptly as the storage limit is approached.

The impact of pattern shape on recall accuracy varies significantly between random and structured patterns. When using a random pattern with a small shape, the recall accuracy initially starts at 100% but decreases with high variance, indicating sensitivity to noise and instability in retrieval performance. In contrast, random patterns with a larger shape improve performance, as recall accuracy starts at 100% but decreases more steeply, eventually converging to zero.

Furthermore, capacity differs between random patterns and structured images due to the correlation between patterns. Images, for example, exhibit strong pixel correlations, where adjacent pixels tend to have similar values. This reduces the orthogonality of patterns, leading to greater interference in the weight matrix. As a result, the effective capacity for storing and recalling patterns is lower for structured images compared to purely random patterns.

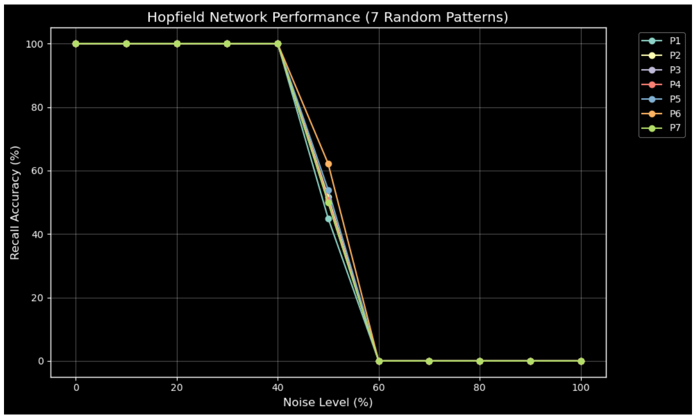


Figure 11: Comparison of recall performance between storing random patterns and image-based patterns. The figure examines whether more patterns can be stored efficiently when using random inputs.

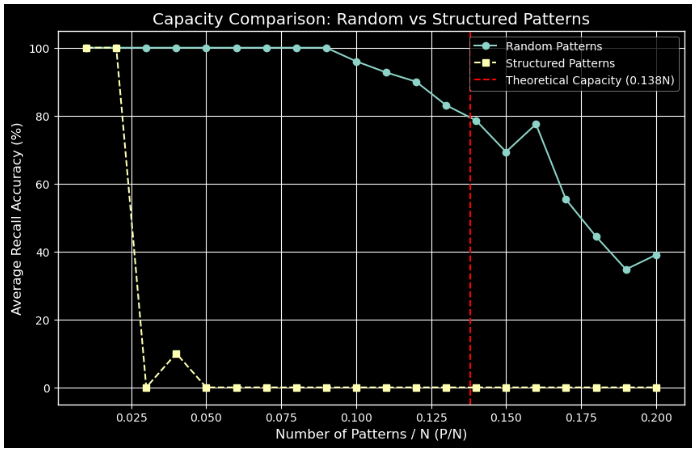


Figure 12: Impact of pattern correlations on storage capacity. The figure compares the theoretical Hopfield network capacity of $0.138N$ with experimental results for random and image-based patterns, highlighting the effects of correlation.

Increasing the noise level directly reduces the stable pattern count, with the noisy recall curve initially following the non-noisy curve when the number of stored patterns is low. However, as more patterns are added, the noisy curve declines drastically compared to the non-noisy one, highlighting the network's decreasing ability to correct for noise. When the network size is significantly smaller than the number of stored patterns ($N = 10 \ll 300$), both curves exhibit high variance and fail to converge. For a slightly larger network ($N = 100 < 300$), convergence occurs rapidly, reaching zero for both curves. However, when the network size approaches or exceeds the number of patterns ($N = 300-500 \geq 300$), convergence is slower, taking longer to reach zero.

This behavior aligns with the capacity limitations of associative networks. When overloaded with patterns, the network struggles to retrieve correct patterns, often converging to spurious states instead. The presence of self-connections (w_{ii}) exacerbates this issue by making the network less robust to noise, potentially creating misleading indications of higher capacity. Removing self-connections can improve noise removal by reducing spurious states, even though it may slightly lower the raw storage capacity. Thus, while networks with self-connections might seem to store more patterns, their effective recall performance under noisy conditions is reduced, reinforcing the trade-off between memory capacity and robustness to noise.

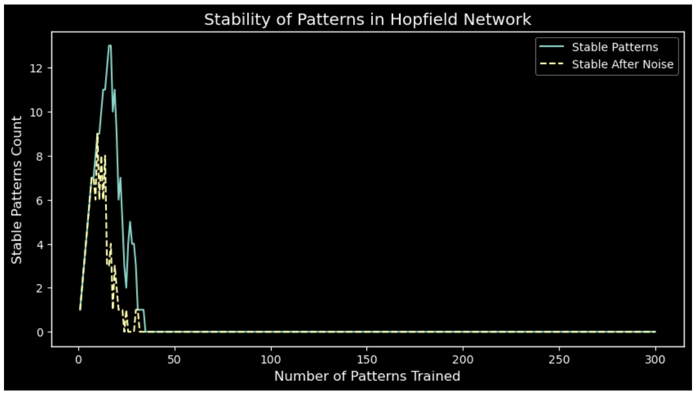
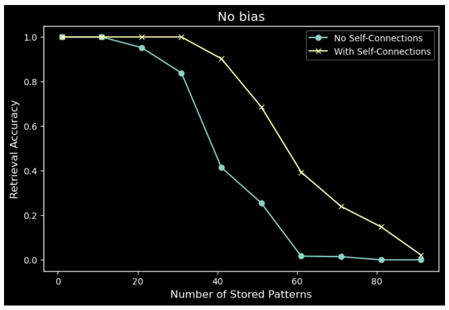
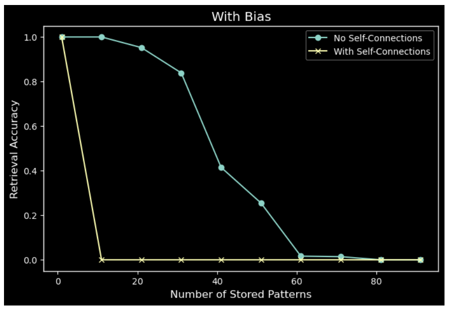


Figure 13: Impact of learned patterns on stability and noisy recall. The figure shows how stability changes with more stored patterns and the effect of noise on convergence.

Biasing patterns to contain more +1 values affects network capacity by increasing correlations between stored patterns. Skewing the distribution, making positive values more likely and leading to imbalanced patterns. Higher average activity results in greater overlap between patterns, reducing their orthogonality and lowering effective capacity below $0.138N$. This effect is similar to structured image patterns, where inherent biases further limit capacity compared to fully random, unbiased patterns.



(a) Without a bias



(b) With a bias

Figure 14: Analysis of Hopfield network capacity. The left figure examines the self-connection and no self-connections without bias, while the right figure explores how biasing patterns affects recall performance and capacity.

6 Sparse

Higher sparsity, represented by a lower activity ratio ρ, necessitates a lower bias threshold θ to optimize performance. Adjusting θ according to ρ enhances recall accuracy and storage efficiency. Sparse patterns with extremely low activity levels ($\rho = 0.01$) benefit from lower bias values ($\theta = 0.025$), whereas higher activity levels require slightly increased bias thresholds to maintain performance.

The capacity of a Hopfield network is influenced by both sparsity (ρ) and bias (θ). Lower activity levels in the stored patterns require smaller bias values to ensure efficient storage. However, bias selection must be carefully tuned, as excessively low or high values degrade storage performance. Increased sparsity improves pattern orthogonality, which enhances the network's ability to store a greater number of patterns effectively.

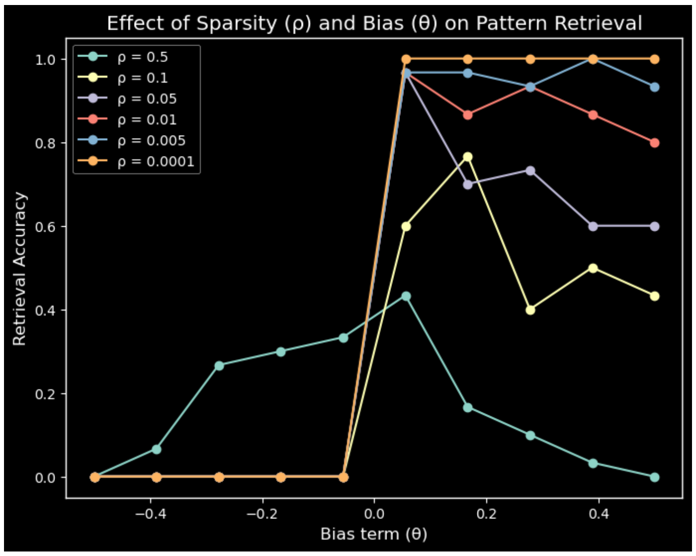


Figure 15: Effect of sparsity and bias on storage capacity with different bias values (θ) when generating sparse patterns with 10% activity, as well as for even sparser cases ($\rho = 0.05$ and $\rho = 0.01$).

7 Conclusion

This study examined Hopfield networks as autoassociative memory systems, demonstrating their ability to recall stored patterns and correct minor distortions. However, performance declines with increased noise and pattern complexity, with storage capacity limited to approximately $0.138N$. Higher sparsity improved storage efficiency by enhancing pattern orthogonality, while structured patterns exhibited lower capacity due to increased correlations. Removing self-connections improved noise tolerance but slightly reduced capacity. These findings highlight the trade-offs in Hopfield networks, emphasizing their constraints in practical applications.