Hopfield Networks: A Brain-Inspired Model for Associative Memory

Ingrid Corobana
ingrid.corobana@s.unibuc.ro
Implementation, experiments

Cosmin Glod
cosmin.glod@s.unibuc.ro
Mathematical analysis, presentation

Irina Moise
irina.moise@s.unibuc.ro
Literature review, visualization

Archaeology of Intelligent Machines — January 2026

Abstract

This project explores Hopfield Networks through the lens of biological and physical analogies, presenting them as artificial "energy landscapes" where memories are stable valleys, mirroring protein folding dynamics and brain attractor states. We implement a Hopfield network from first principles, conduct experiments on capacity, noise robustness, and spurious attractors, and connect the classical model to modern extensions used in transformers.

1 Introduction

1.1 Motivation

The human brain performs remarkable feats of memory: we can recall a complete song from just a few notes, recognize a face from a partial glimpse, and fill in missing details of past experiences. These capabilities emerge from billions of neurons connected by trillions of synapses, yet the underlying computational principles can be captured by surprisingly simple models.

Hopfield Networks, introduced by John Hopfield in 1982 [1], provide one such model. They demonstrate how a system of simple binary units, connected by symmetric weights and updated according to local rules, can exhibit emergent properties of associative memory. In 2024, Hopfield and collaborators were awarded the Nobel Prize in Physics for this work, recognizing its profound impact on both neuroscience and artificial intelligence.

1.2 Project Philosophy: Brain-Inspired Analogies

We will base the narrative of our project on biological and physical analogies, following recent popular explanations that use **protein folding** and **brain-inspired energy landscapes** as conceptual frameworks [3, 4].

Key Analogy: Just as proteins explore a high-dimensional configuration space and "roll downhill" to settle into low-energy folded states, Hopfield networks navigate a state space where stored memories correspond to energy minima (attractor states). Network dynamics drive the system toward these attractors, enabling memory retrieval from partial or noisy inputs.

In our implementation, each step (data representation, Hebbian learning, network dynamics, memory retrieval) will be presented both in mathematical/code form and using corresponding brain analogies (neurons, synapses, attractor states).

1.3 Contributions Per Member

- **Ingrid Corobana:** Led implementation of the core `HopfieldNetwork` class in Python, designed and executed capacity/noise experiments, created the image retrieval notebook with Simpsons character faces, wrote project documentation and README files.
- **Cosmin Glod:** Derived mathematical foundations (energy function proofs, Hebbian rule derivation, pseudo-inverse learning rule), created the modern theory notebook connecting classical Hopfield to transformer attention, prepared the Beamer presentation slides.
- **Irina Moise:** Conducted literature review on Hopfield's original 1982 paper and modern extensions (Ramsauer et al. 2020), created visualizations (energy landscapes, pattern similarity matrices, 3D energy surfaces), documented biological analogies throughout.

1.4 What Each Member Learned

Ingrid: "I learned how simple local update rules can produce emergent global behavior. Implementing the network from scratch gave me deep appreciation for how biological constraints (symmetric weights, no self-connections) lead to useful computational properties. In the future, I want to explore continuous Hopfield networks and their connection to modern attention mechanisms."

Cosmin: "Deriving the pseudo-inverse rule helped me understand why orthogonalization matters for memory capacity. The connection to transformers was surprising—attention is just modern Hopfield retrieval! I want to learn more about energy-based models and contrastive learning."

Irina: "Reading Hopfield's original paper showed me how physics concepts (energy, thermodynamics, spin glasses) apply to neural computation. The protein folding analogy made the mathematics intuitive. I want to explore Boltzmann machines and restricted Boltzmann machines next."

2 Background: What Has Been Done Before

2.1 Hopfield Networks (1982)

Hopfield Networks are fully-connected recurrent neural networks where:

- **Neurons**: Binary units $s_i \in \{-1, +1\}$ representing firing/silent states
- **Synapses**: Symmetric weights $w_{ij} = w_{ji}$ learned via Hebbian rule
- **Dynamics**: Asynchronous updates minimize an energy function
- **Memory**: Stored patterns become stable fixed points (attractors)

The energy function is defined as:

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j - \sum_i \theta_i s_i \tag{1}$$

where θ_i are neuron thresholds (often set to zero).

Hebbian Learning Rule:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu, \quad i \neq j, \quad w_{ii} = 0 \tag{2}$$

where $\boldsymbol{\xi}^\mu$ are the P patterns to be stored, and N is the number of neurons.

Capacity: For random uncorrelated patterns, the network can reliably store approximately $0.138N$ patterns [1, 5].

2.2 Modern Extensions

Recent work has generalized Hopfield networks to achieve exponential capacity:

- **Dense Associative Memory** (Krotov & Hopfield, 2016): Higher-order interactions
- **Modern Hopfield Networks** (Ramsauer et al., 2020): Connection to attention mechanisms in transformers [2]

These modern variants demonstrate that the core principles of energy-based associative memory remain relevant in contemporary deep learning architectures.

3 Approach

3.1 Repository and Tools

- **GitHub Repository:** `https://github.com/dirgnic/Hopfield_Networks`
- **Google Colab Notebooks:**
  - Project Description: `https://colab.research.google.com/drive/1ijqbNkXiPpae_7u4anCMlk6SQ_0sjzTM`
  - Main Implementation: `https://colab.research.google.com/drive/18eRvSoLZ_0654VFzdFsTjFCBSbpGeku7`
- **Software Tools:** Python 3.9+, NumPy, Matplotlib, scikit-learn (for PCA visualization), PIL (for image processing)
- **Hardware:** Standard laptop CPU (no GPU required)
- **Processing Time:** All experiments complete in under 1 minute on a standard laptop

3.2 Biological Inspiration: From Proteins to Neurons

Act I — Energy Landscapes in Nature

We begin by drawing an analogy to protein folding:

- A protein chain explores a vast configuration space
- Energy landscapes guide the folding process
- Low-energy valleys correspond to stable folded states

- Nature solves optimization by ”rolling downhill”

Transition to Neural Systems:

- Replace protein configurations with neural firing patterns
- Energy valleys become stored memories (attractors)
- Brain dynamics = descent toward familiar states

3.3 Implementation from First Principles

Act II — Building a Hopfield Network

3.3.1 Step 1: Neurons and Patterns

Implementation:

- Represent neuron states as $s_i \in \{-1, +1\}$
- Patterns are vectors $\boldsymbol{\xi} \in \{-1, +1\}^N$
- Example: 10×10 binary images (flattened to $N = 100$ neurons)

Brain Analogy:

- Each element = a neuron (firing or silent)
- A pattern = a global brain state snapshot

3.3.2 Step 2: Hebbian Learning

Implementation:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu, \quad i \neq j \tag{3}$$

Brain Analogy:

- Hebb’s principle: ”Cells that fire together, wire together”
- Synaptic plasticity strengthens co-active connections

3.3.3 Step 3: Network Dynamics

Update Rule:

$$s_i^{(t+1)} = \text{sign}\left(\sum_j w_{ij} s_j^{(t)} - \theta_i \right) \tag{4}$$

Energy Minimization:
Asynchronous updates guarantee $\Delta E \leq 0$, so the system converges to a local minimum.

Brain Analogy:

- Each neuron ”listens” to synaptic inputs
- Positive input $\rightarrow$ fire; negative $\rightarrow$ stay silent
- System collectively settles into consistent configuration

3.3.4 Step 4: Memory Retrieval

Implementation:

1. Add noise to a stored pattern (flip random bits)
2. Initialize network with noisy input
3. Run update rule until convergence
4. Check if final state matches original pattern

Brain Analogy:

- "Hear a few notes, recall the whole song"
- Partial cue triggers complete memory
- Network "fills in" missing information

3.4 Experiments and Evaluation

Act III — Testing the Model

3.4.1 Data and Exploratory Analysis

- **Data**: Synthetic 10×10 binary letter images (A, B, C, D, E)
- **EDA**:
  - Visualize stored patterns
  - Compute Hamming distances between patterns
  - Plot pattern similarity matrix (normalized overlap)
  - Interpretation: High similarity $\rightarrow$ overlapping energy valleys $\rightarrow$ reduced capacity

3.4.2 Evaluation Metrics

1. **Retrieval Accuracy**: Fraction of noisy inputs correctly recovered
2. **Noise Robustness**: Accuracy vs. noise level (10%, 20%, 30%, etc.)
3. **Capacity**: Number of patterns stored before performance collapse
4. **Convergence**: Number of iterations to reach stable state
5. **Energy Trajectory**: Visualization of energy minimization during retrieval

3.4.3 Capacity Experiment

Procedure:

- Store $P = 1, 2, \ldots, 30$ random patterns
- For each P, measure retrieval accuracy with 10% noise
- Plot accuracy vs. P
- Compare to theoretical limit ($\sim 0.138N$)

Brain Analogy:
When too many memories are crammed into one energy landscape, valleys merge and the network creates "false memories" (spurious attractors).

3.4.4 Spurious Attractors

Procedure:

- Store patterns near/above capacity
- Initialize with random states
- Check if network converges to non-stored patterns
- Visualize spurious attractors

Brain Analogy:
Like confabulation in human memory — the brain creates stable but incorrect memories by blending real experiences.

3.5 Modern Connection: Transformers

Briefly mention that modern Hopfield networks generalize the classical model and are mathematically equivalent to attention mechanisms in transformers [2]. This connects our project to cutting-edge AI research.

4 Limitations

1. **Limited Storage Capacity:** The classical Hopfield network can only store approximately $0.138N$ patterns reliably. For our 100-neuron network ($N = 100$), this means only $\sim$14 memories—far too few for practical applications like image databases.
2. **Binary Patterns Only:** Our implementation uses binary states $\{-1, +1\}$, which cannot directly represent continuous or grayscale data. Real images must be binarized, losing significant information. Modern continuous Hopfield networks address this but were not implemented.
3. **Spurious Attractors:** The network creates "false memories"—stable states that don't correspond to any stored pattern. These include mixture states (blends of stored patterns) and spin-glass states. We observed these experimentally but did not implement methods to suppress them.
4. **No Temporal Sequences:** Classical Hopfield networks store static patterns only. They cannot learn or reproduce temporal sequences like melodies, motor actions, or video frames. Extensions like asymmetric Hopfield networks exist but were beyond our scope.

5. **Scalability Issues:** The weight matrix grows as $O(N^2)$, making large networks memory-intensive. We tested up to $N = 1000$ neurons but did not explore sparse connectivity or hierarchical architectures that might improve scaling.

6. **Correlated Pattern Interference:** Highly similar patterns (e.g., letters "O" and "Q") interfere with each other, reducing effective capacity below the theoretical limit. While the pseudo-inverse rule helps, it requires $O(N^3)$ matrix inversion.

7. **Synthetic Data Only:** Our experiments use synthetic letter patterns and preprocessed cartoon faces (Simpsons). We did not evaluate performance on natural images, handwritten digits (MNIST), or noisy real-world sensor data.

8. **Incomplete Modern Theory Coverage:** While we discuss the connection to transformers, we did not implement the continuous modern Hopfield network from Ramsauer et al. (2020) that achieves exponential capacity.

5 Conclusions and Future Work

5.1 Reflections

What could we have done differently?

- We could have implemented the modern continuous Hopfield network with exponential capacity, which would have made the transformer connection more concrete
- More systematic comparison between Hebbian and pseudo-inverse learning rules across different pattern correlation levels
- GPU implementation using PyTorch for larger-scale experiments
- Interactive web demo using JavaScript/WebGL for real-time visualization

How could this project be improved?

- Implement attention-based modern Hopfield retrieval from Ramsauer et al. (2020)
- Add stochastic updates (Boltzmann machine) to escape local minima
- Test on real image datasets (MNIST, CIFAR-10) with proper preprocessing
- Explore sparse connectivity patterns inspired by biological neural networks
- Add temporal sequence learning with asymmetric weights

Did we enjoy this project?

Yes! The elegant connection between physics, neuroscience, and computer science made this fascinating. Seeing the network "recall" a complete pattern from a corrupted input feels almost magical, yet it emerges from simple mathematical rules. The 2024 Nobel Prize recognition made us appreciate how foundational these ideas are to modern AI.

What did we learn?

- Energy-based models provide powerful frameworks for understanding neural computation
- Simple local update rules can produce complex emergent behavior (content-addressable memory)

- Historical ideas from 1982 remain directly relevant to cutting-edge AI (transformers use Hopfield-like retrieval)
- The importance of combining mathematical analysis with empirical experiments
- How to structure a research project: literature review → implementation → experiments → analysis

5.2 Suggestions for Future AMI Projects

- Provide a curated list of computational resources (Colab Pro, cloud GPU credits) for students wanting to scale up
- Include intermediate checkpoints/milestones to encourage incremental progress
- Brief LaTeX tutorial session for students unfamiliar with academic typesetting
- More class time for project discussions and peer feedback
- A shared repository of baseline implementations students can build upon

6 References

References

[1] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), 2554-2558.

[2] Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., ... & Hochreiter, S. (2020). Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*.

[3] "A Brain-Inspired Algorithm For Memory" (YouTube): `https://www.youtube.com/watch?v=1WPJdAW-sFo`

[4] "Hopfield Networks Explained" (YouTube): `https://www.youtube.com/watch?v=piF6D6CQxUw`

[5] Wikipedia: Hopfield Network. `https://en.wikipedia.org/wiki/Hopfield_network`