
**Savitribai Phule Pune University**
**Gokhale Education Society's**

**R. H. Sapat College of Engineering, Management Studies and Research,**
**Nashik - 422 005, (M.S.), INDIA**

**DEPARTMENT OF COMPUTER ENGINEERING**
**Third Year Computer Engineering**
**Year 2023– 2024**

**Roll No:** 52

**Name of Student:** Sonawane Atharva Hemant

**Mobile No.:** +91 8484842280

**official-Mail ID:** 21_atharva.sonawane@ges-coengg.org

**Seminar Title:** Neuromorphic Hopfield Network

**Seminar Guide :** Mrs. V.S.Nikam

**Area of the Seminar:** Neuromorphic computing.

## Abstract

Neuromorphic systems, inspired by how our brains work, offer fresh and creative solutions for artificial intelligence. They imitate the way our brains use neurons and connections to process information. Many scientists are putting a lot of effort into creating models, methods, and systems based on this idea. Some have even shown that certain algorithms, like the Hopfield algorithm, can work well in big projects. This paper gives a detailed overview of the Hopfield algorithm and how it could be used in new research. In the end, we talk about potential future uses and offer suggestions for developers who want to use this model in their own AI projects. This should help them understand how it works and how to apply it effectively.



**INDEX TERMS :** Neuromorphic computing, neuro-inspired model, Hopfield algorithm, artificial intelligence.

# Introduction

A Neuromorphic Hopfield Network (NHN) is like a special computer network inspired by how our brains work. It's based on an idea from John Hopfield in 1982. NHNs try to copy some of the brain's abilities to remember things and recognize patterns. They use simple on-off signals and calculate things based on energy. NHNs are built to remember information based on what it is, not where it's stored. This is similar to how our brains remember things based on context.

NHNs have a key feature: every neuron is connected to every other neuron in a balanced way. This is important for their memory abilities. The connections between neurons start random and the network learns by adjusting them during training. The Hopfield rule helps with this, making the NHN settle into stable states that match stored patterns.

In real life, NHNs are used in many areas. They help with remembering and finding information, solving problems, and even in making computer systems that act like brains. But they do have some limits, like how much they can remember and the chance of getting mixed up. Still, they're really useful for tasks that need memory and pattern recognition.

Mixing neuromorphic ideas with the Hopfield network in creating an NHN tries to get the best of both worlds: energy-efficient computing inspired by the brain, along with the strong memory and pattern skills of Hopfield networks. This combination could be a powerful tool for many different tasks, especially those needing efficient memory and pattern skills.

# Method/Algorithm

Creating a Neuromorphic Hopfield Network (NHN) involves teaching it to remember and recognize patterns using specific steps. One commonly used method is called the Hopfield update rule.

**Step 1: Getting Ready**

- Start by setting up the NHN with special on-off neurons. These neurons can be either 'on' (+1) or 'off' (-1).

**Step 2: Setting Up Connections**

- Decide how strong the connections between neurons should be. These connections represent how well the neurons communicate with each other. You can do this randomly or using a specific method.

**Step 3: Teaching Patterns**

- Train the NHN to remember patterns by adjusting the connection strengths. For each pattern you want it to remember:
- Change the pattern into a special form made of zeros and ones.
- Adjust the connection strengths using a specific calculation involving the pattern and some matrices.

**Step 4: Remembering Patterns**

- To bring back a pattern, give the NHN a starting pattern. This can be a complete pattern or a slightly messy version of one you taught it.
- Keep updating the neuron states until they stop changing. This means the network has remembered the pattern.

**Step 5: Checking the Result**

- The NHN will end up in a stable state, which means it remembers the pattern. This stable state shows the remembered pattern.

**Step 6: Handling Mistakes**

- Make sure the remembered pattern is one of the ones you taught the NHN. If not, it might be a mistake.
- Use methods like adding some random changes to the starting pattern or using special measures to fix mistakes.

**Step 7: Trying Different Patterns**

- You can do this process with many different patterns by starting with different ones.

**Step 8: Checking and Stopping**

- See how well the NHN did in remembering the patterns. Stop when it gets it right or when you decide it's gone on long enough.

This Hopfield update rule helps the NHN remember and recognize patterns, which is really useful for tasks needing memory and pattern skills. There are also other methods like the Hebb Algorithm and Training Algorithm for Discrete Hopfield Neural Network that people use.

# References

1] Zheqi Yu , Amir M. Abdulghani, Adnan Zahid , Hadi Heidari, Muhammad Ali Imran , James Watt, Qammer H. Abbasi on “An Overview of Neuromorphic Computing for Artificial Intelligence Enabled Hardware-Based Hopfield Neural Network ” IEEE VOLUME 8, no.: 67097, April 21, 2020.

2]https://en.wikipedia.org/wiki/Hopfield_network#:~:text=Updates%20in%20the%20Hopfield%20network,updated%20at%20the%20same%20time.

3]https://www.geeksforgeeks.org/hopfield-neural-network/

4]https://www.javatpoint.com/artificial-neural-network-hopfield-network#:~:text=If%20Wij%20%3E%200%2C%20the,its%20value%20Xj%20%3D%20%2D1