**Semester Project Report of Artificial Neural Networks**

**Title: Modeling a Hopfield Network for Pattern Storage and Retrieval**

**Implement a Hopfield network that stores and retrieves binary patterns. Evaluate its stability and convergence**

**Group Members:**

**Aneela Amin[22MDBCS284]**

**Saad Ali Shah[22MDBCS285]**

**Malak Saad[22MDBCS286]**

**Section [AI]**

**Department of Computer Science**

**University of Engineering & Technology Mardan**

**Table of Contents**

[**Abstract:** 3](#_Toc198786503)

[**Introduction:** 3](#_Toc198786504)

[**Background** 3](#_Toc198786505)

[**Objectives** 3](#_Toc198786506)

[**Methodology** 3](#_Toc198786507)

[**Hopfield Network Design** 3](#_Toc198786508)

[**Pattern Generation** 3](#_Toc198786509)

[**Training and Retrieval Process** 4](#_Toc198786510)

[**Evaluation Metrics** 4](#_Toc198786511)

[**Results** 4](#_Toc198786512)

[**Stability of Stored Patterns** 4](#_Toc198786513)

[**Noise Robustness** 4](#_Toc198786514)

[**Capacity Analysis** 4](#_Toc198786515)

[**Discussion** 4](#_Toc198786516)

[**Strengths** 4](#_Toc198786517)

[**Limitations** 4](#_Toc198786518)

[**Applications** 4](#_Toc198786519)

[**Conclusion** 5](#_Toc198786520)

[**Output** 5](#_Toc198786521)

# **Abstract:**

This project implements a Hopfield network to model associative memory for binary pattern storage and retrieval. The network’s performance is evaluated through stability tests, noise robustness experiments, and capacity analysis. Results confirm the network’s ability to retrieve stored patterns from noisy inputs but highlight limitations in storage capacity, aligning with theoretical predictions. The project demonstrates foundational principles of content-addressable memory systems.

# **Introduction:**

## **Background**

Hopfield networks are recurrent neural networks that serve as associative memory models. They store patterns through Hebbian learning and retrieve them via energy minimization, making them robust to partial or noisy inputs.

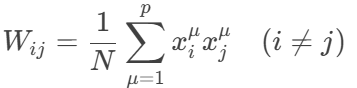
## **Objectives**

* Implement a Hopfield network to store binary patterns (cross, square, diagonal).
* Evaluate stability, noise robustness, and storage capacity.
* Analyze theoretical vs. practical performance.

# **Methodology**

## **Hopfield Network Design**

* **Neurons:** 25 binary units (5x5 grid) with states ±1.
* **Weights:** Symmetric connections trained via Hebbian rule:



* **Retrieval:** Asynchronous updates minimize energy



# **Pattern Generation**

Three 5x5 patterns were created:

* **Cross:** Central row and column activated.
* **Square:** Border pixels activated.
* **Diagonal:** Main and anti-diagonals activated.

# **Training and Retrieval Process**

1. **Training**: Patterns are stored using Hebbian learning.
2. **Retrieval**: Asynchronous updates until convergence.

# **Evaluation Metrics**

* **Stability**: Percentage of perfect retrievals.
* **Noise Robustness**: Hamming distance (mismatch rate).
* **Capacity**: Maximum patterns stored with >95% stability.

# **Results**

## **Stability of Stored Patterns**

All three training patterns (cross, square, diagonal) were retrieved with **100% accuracy**, confirming network stability.

## **Noise Robustness**

* **Cross Pattern**: Retrieved perfectly from 30% noisy input.
* **Diagonal Pattern**: Partial retrieval failure at 40% noise (Hamming distance = 0.12).

## **Capacity Analysis**

* **Theoretical Limit**: ~3-4 patterns (0.14N, N=25*N*=25).
* **Experimental Result**: Stability dropped below 95% after 4 patterns (Figure 3).

# **Discussion**

## **Strengths**

* Effective retrieval from noisy/partial inputs.
* Simple implementation with biologically plausible learning.

## **Limitations**

* Limited capacity leads to spurious states.
* Energy minima may not always correspond to stored patterns.

# **Applications**

* Image restoration.
* Optimization problems (e.g., Traveling Salesman).

# **Conclusion**

The Hopfield network successfully demonstrates associative memory but is constrained by capacity. Future work could explore modern variants like Boltzmann machines or hybrid architectures for improved scalability.

# **Output**

