```
Name: Aarya Admane 22630

import numpy as np

class HopfieldNetwork:
    def __init__(self, size):
        self.size = size
        self.weights = np.zeros((size, size))

    def train(self, patterns):
        for pattern in patterns:
            pattern = np.array(pattern).reshape(-1, 1)  # Ensure
column vector
            self.weights += np.outer(pattern, pattern)
        np.fill_diagonal(self.weights, 0)  # No self-connections

    def recall(self, pattern, steps=5):
        pattern = np.array(pattern)
        for _ in range(steps):
            pattern = np.sign(self.weights @ pattern)
            pattern[pattern == 0] = 1  # Convert 0s to 1s for
stability
        return pattern

if __name__ == "__main__":
    # Define 4 stored patterns (bipolar: 1 or -1)
    stored_patterns = [
        [1, -1, 1, -1],
        [-1, 1, -1, 1],
        [1, 1, -1, -1],
        [-1, -1, 1, 1]
    ]

    size = len(stored_patterns[0])  # Size of the pattern vectors
    hopfield = HopfieldNetwork(size)
    hopfield.train(stored_patterns)

    # Testing pattern with noise
    test_pattern = [1, -1, 1, 1]  # Slightly different from first
pattern
    recalled_pattern = hopfield.recall(test_pattern)

    print(f"Original test pattern: {test_pattern}")
    print(f"Recalled pattern: {recalled_pattern.tolist()}")

Original test pattern: [1, -1, 1, 1]
Recalled pattern: [-1.0, -1.0, 1.0, -1.0]
```