```python
import numpy as np

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

    def train(self, patterns):
        for p in patterns:
            p = np.array(p)
            p = p.reshape(self.size, 1)
            self.weights += np.dot(p, p.T)
        np.fill_diagonal(self.weights, 0)  # No self-connections
        self.weights /= len(patterns)      # Normalize (optional)

    def recall(self, pattern, steps=5):
        p = np.array(pattern)
        for _ in range(steps):
            for i in range(self.size):
                raw = np.dot(self.weights[i], p)
                p[i] = 1 if raw >= 0 else -1
        return p

# Convert binary {0, 1} to bipolar {-1, +1}
def to_bipolar(vector):
    return [1 if x == 1 else -1 for x in vector]

# Example: 4 binary patterns of length 6
stored_patterns = [
    [1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1],
    [1, 1, 0, 0, 1, 1],
    [0, 0, 1, 1, 0, 0]
]


# Convert to bipolar format
bipolar_patterns = [to_bipolar(p) for p in stored_patterns]

# Create and train Hopfield network
hn = HopfieldNetwork(size=6)
hn.train(bipolar_patterns)

# Try recalling each pattern
print("Recalled Patterns:")
for idx, pattern in enumerate(bipolar_patterns):
    recalled = hn.recall(pattern.copy())
    print(f"Pattern {idx + 1}: {recalled}")
```

```
Recalled Patterns:
Pattern 1: [ 1 -1  1 -1  1 -1]
```

```
Pattern 2: [-1  1 -1  1 -1  1]
Pattern 3: [ 1  1 -1 -1  1  1]
Pattern 4: [-1 -1  1  1 -1 -1]
```