CS F433: Computational Neuroscience

SECOND SEMESTER 2024-25

ASSIGNMENT REPORT:
Hopfield Networks

SUBMITTED BY

NAME	ID NUMBER
Akhil Mohammad	2022B3A7PS0360H
Manasvi Chervela	2022A7PS2015H

Table of Contents

Introduction

Implementation details:

Functions for reading and writing .pbm files were made and converted into numpy arrays for ease. Four images were taken as more than four images failed to retain any memory. Synchronous and asynchronous update functions were written with synchronous update function having a maximum limit of 100 iterations and ending when it converges.

A small if statement checking consecutive states is used to implement it. Asynchronous update goes on until 1000 iterations and it chooses a neuron at random proceeding to update it. There are times when one or two neurons are apart from the original image and don't get updated due to randomness. State at each iteration is printed in synchronous update and state at every 100 iterations is printed in asynchronous update.

For task 2, the first training image is corrupted once and then cropped another time which is then put through sync and async updates. The states for all the four cases were noted.

For task 3, each p value was taken for each training image. Sync updates were applied to all of them 20 times and results were noted. A total of 20 x 4 = 80 tests were performed for each p value.

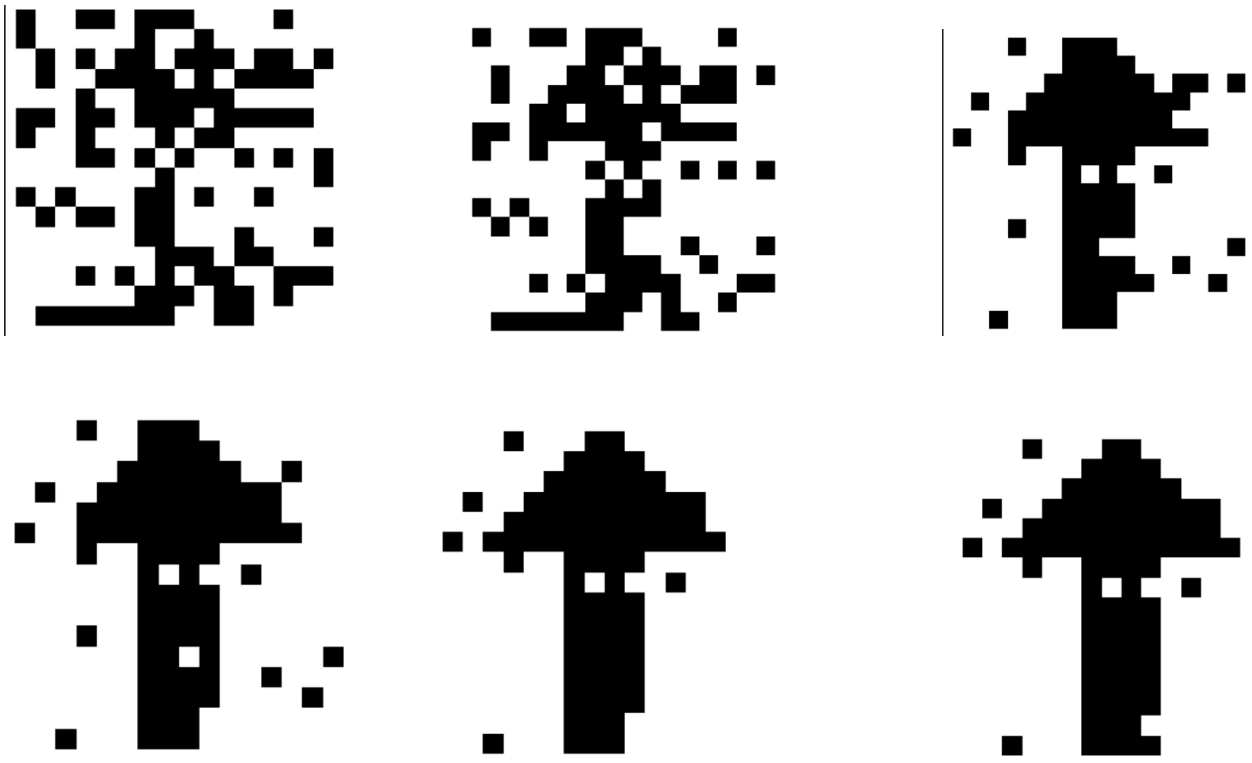
For each p value, the number of successful convergences were calculated. Later on, for each successful convergence, the average number of iterations was calculated and plotted. The last three values of probability had no successful convergences and hence remain undefined.

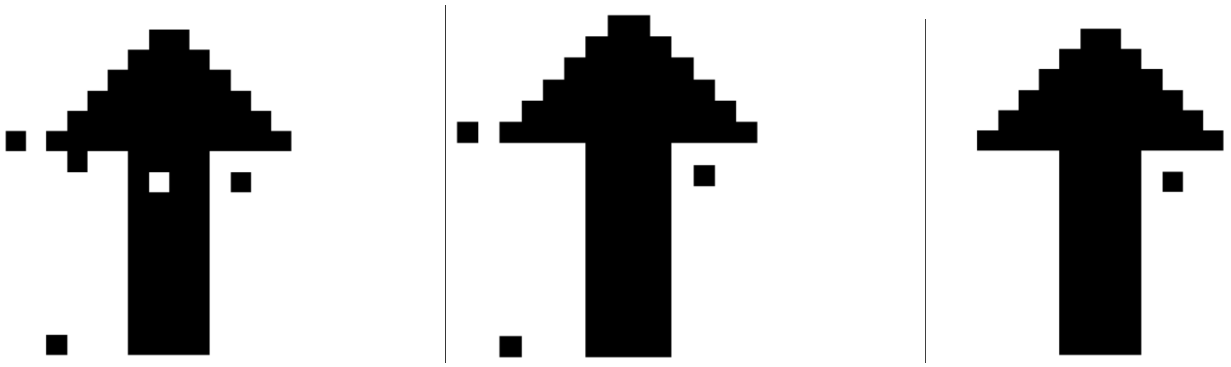
Task 2: Testing on Noise

This test involves two methods of corrupting the given memory: flipping each pixel with a probability of 0.3 and bounding the image in a box with converting all the pixels out of the box to either black or white. This was tested on both synchronous and asynchronous updates. The results are as follows:

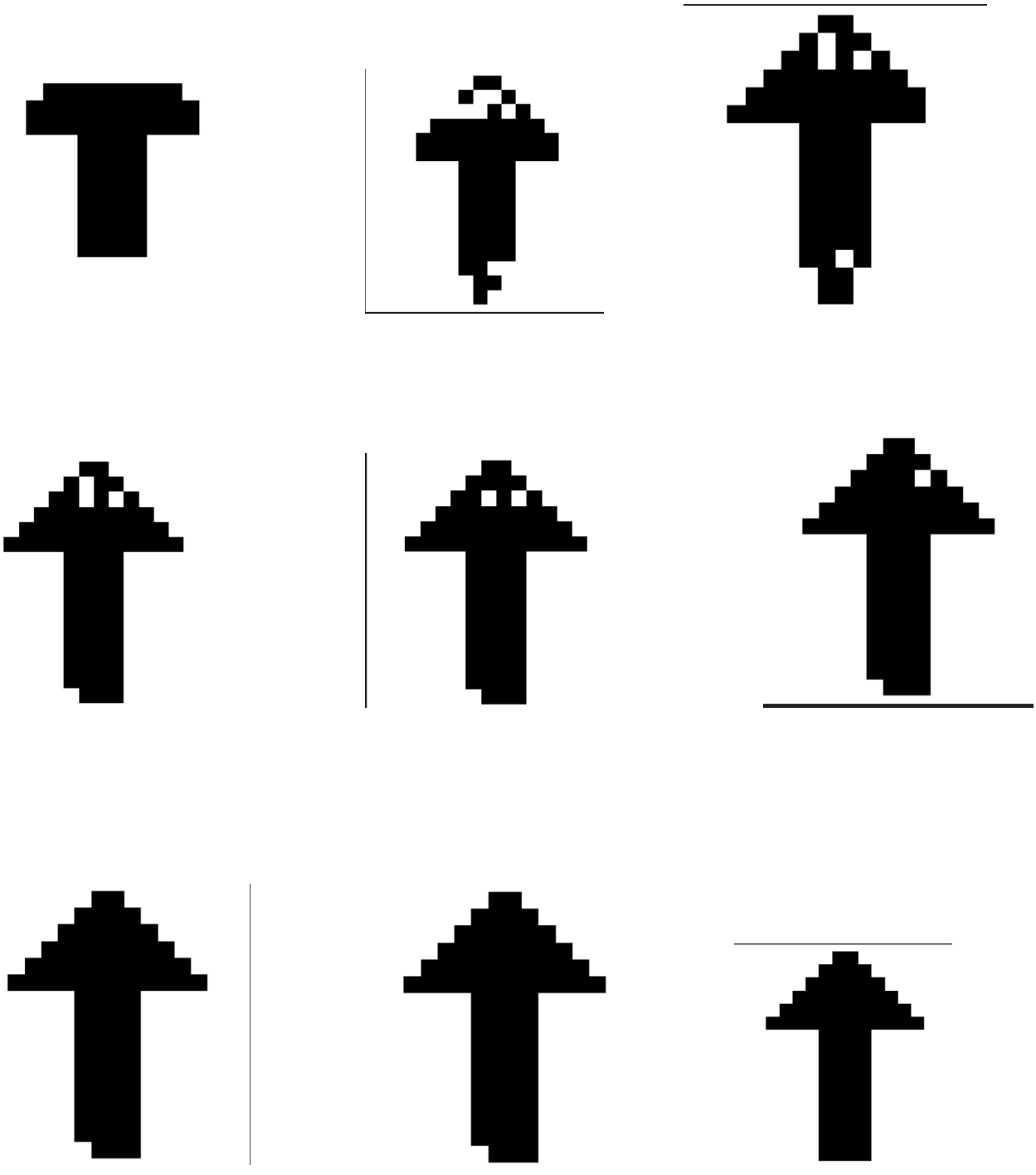
Asynchronous Updates

1. Flipping the Pixels:





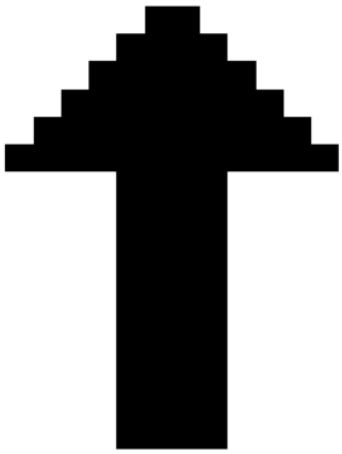
2. Bounding within a box



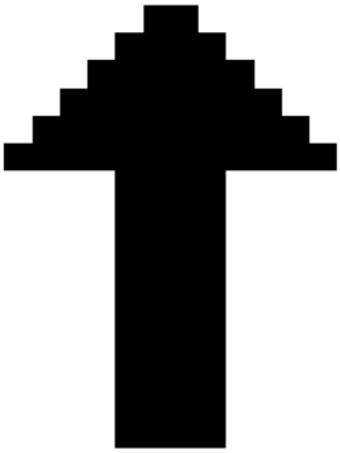
Synchronous Updates

The synchronous updates for both the corrupted files converged in a single step as follows:

1. Flipping the Pixels

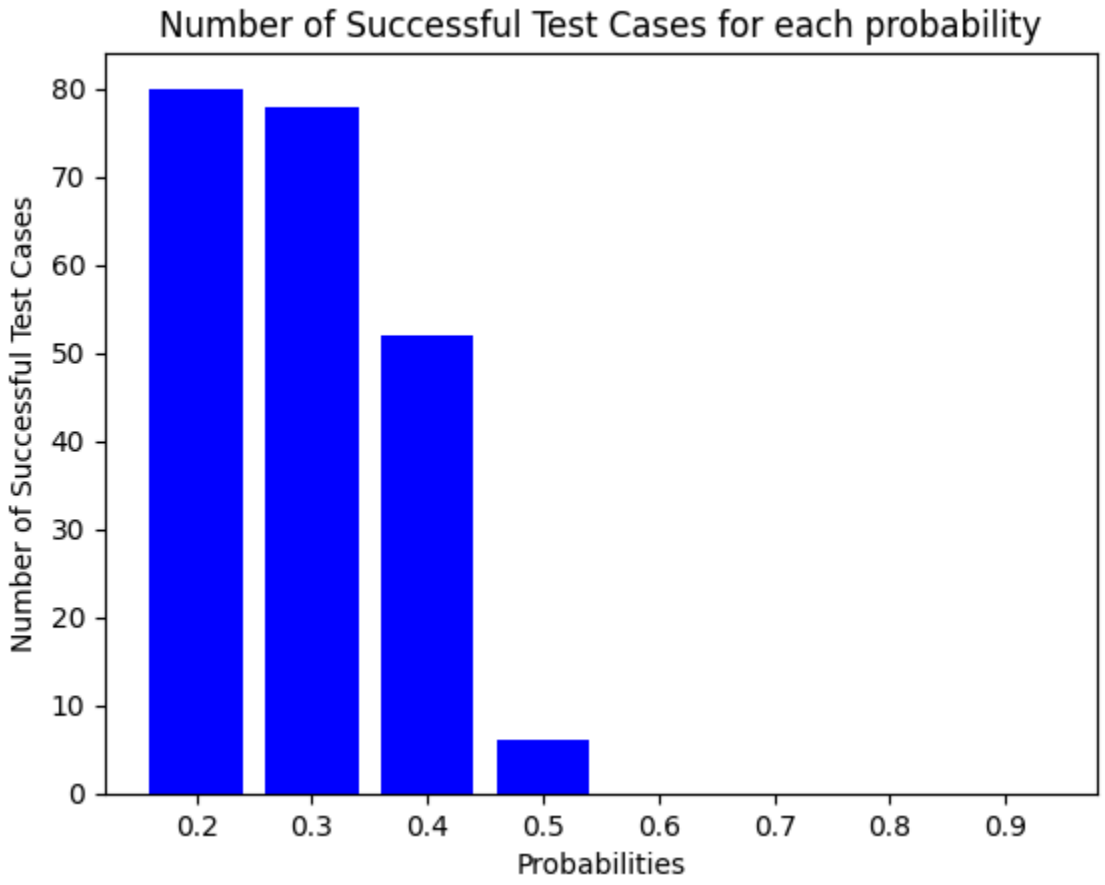


2. Bounding within a box



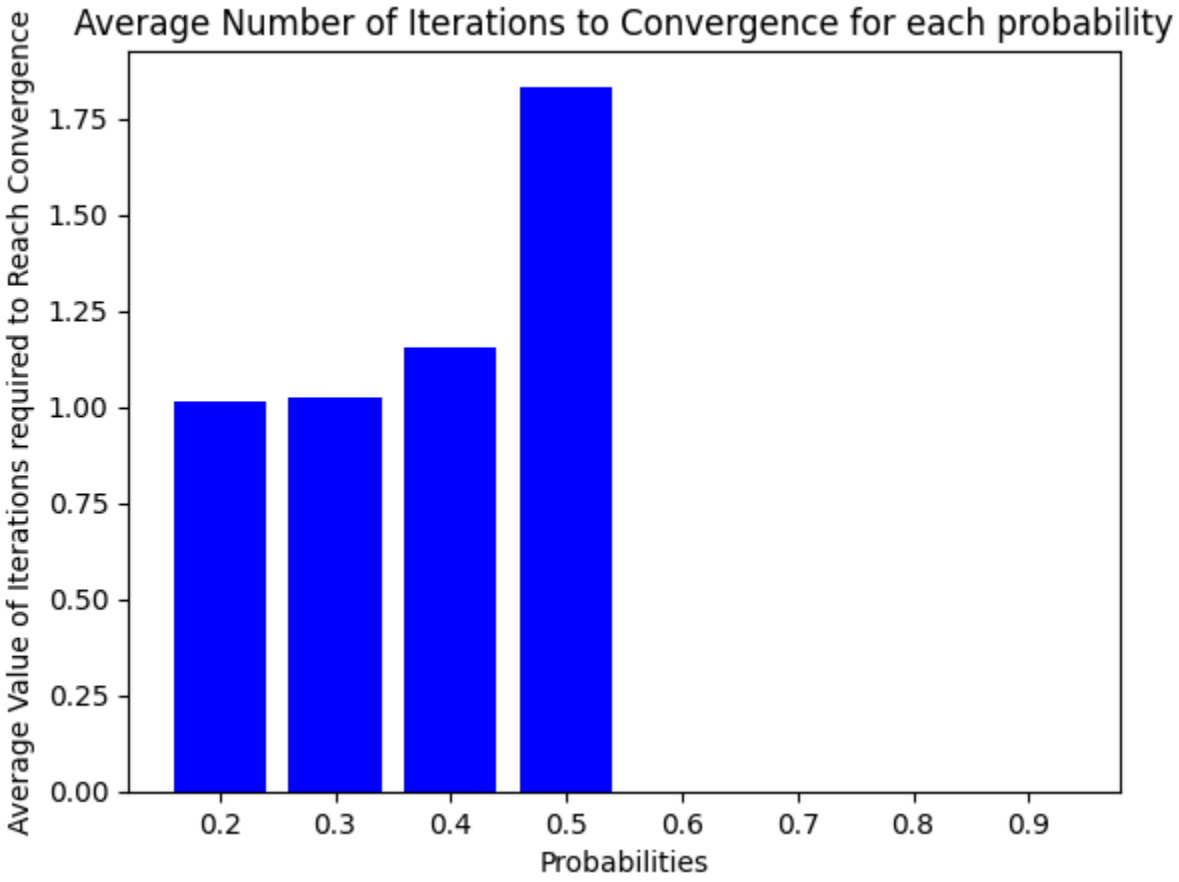
Task 3a

Each p value was taken for each training image, ie, 8 x 4 = 32. Sync updates were applied to all of them 20 times and results were noted. A total of 20 x 4 = 80 tests were performed for each p value. For each p value, the number of successful convergences were calculated. A total of 8 x 4 x 20 = 640 are taken.



Task 3b

For each successful convergence, the average number of iterations was calculated and plotted against the p value. The last three values of probability had no successful convergences and hence remain undefined.



Inferences

In Task 3a, we see that the number of successful convergences reduces as we increase the p value for distortion, which can be observed as the image gets more noisier.

In Task 3b, we see that the average number of iterations for a successful convergence increases as we increase the p value, which intuitively is inferred as we know the more distorted the image the higher the number of iterations it would require to converge.

Conclusion

We have successfully implemented a Hopfield network using the Hebbian rule over 4 images and observed how the levels of distortion by switching a few pixels at certain probabilities and cropping out the image by binding it within a box affect the parameters and results of the update functions.

We had our ups and downs but had an overall positive experience. My main takeaway from this assignment is the fun we had along the way. We feel more confident in our understanding of the Hopfield network and gained immense knowledge as to working with python and numpy.

Appendix

(is attached below)