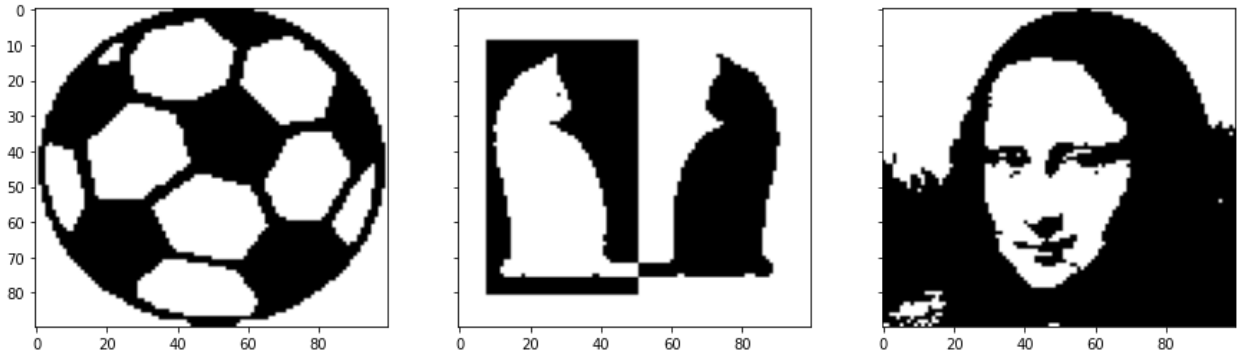
Report

Chella Thiyagarajan N me17b179

Part 1:

a) **Visualization**



b) **Develop a code for Hopfield network with N=9000 neurons which are fully connected.**

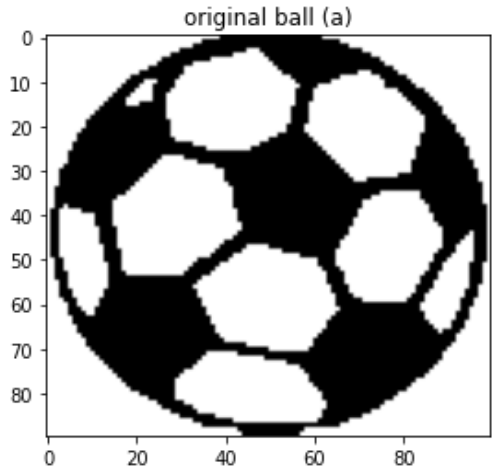
```
class hopfield_network:
    def __init__(self, S, neurons=90*100, dt=0.01, lamda=10):
        self.pattern = S.flatten()
        #print(self.pattern)
        self.weights = np.outer(self.pattern, self.pattern)/neurons
        #print(self.weights.shape)
        self.neurons = np.zeros(neurons)
        self.dt = dt
        self.lamda = lamda
        self.rmse = float('inf')

    def set_trigger_cue(self, cue):
        self.neurons = cue.flatten()

    def take_a_step(self):
        #print(self.weights.shape, self.neurons.T.shape)
        du = ((-1*self.neurons) + self.weights@(np.tanh(self.lamda*self.neurons).T)) * self.dt
        u = self.neurons + du
```

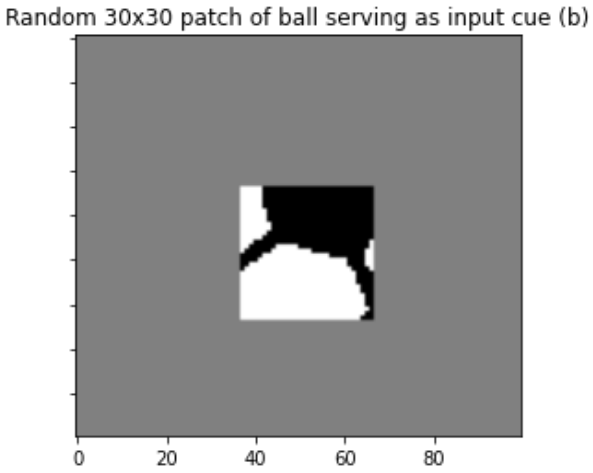
```
    self.neurons = u

def cal_rmse(self):
    self.rmse = np.sqrt(((self.neurons - self.pattern) ** 2).mean())
```

Part 2:

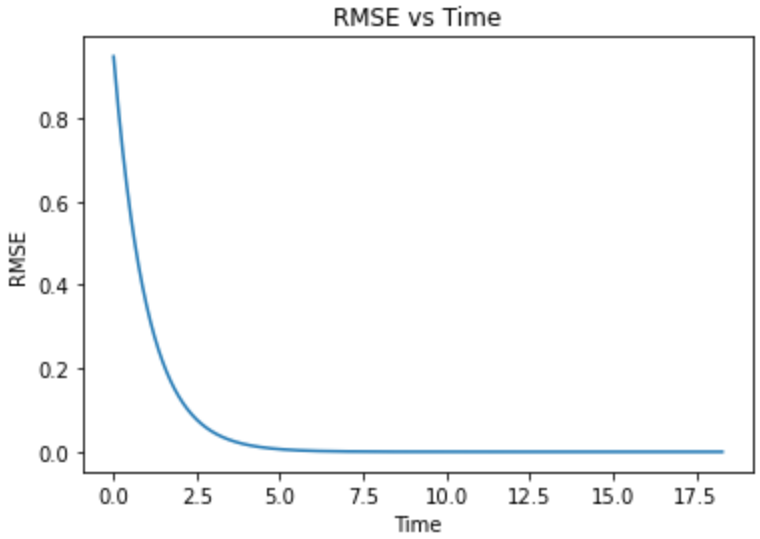
I have Chosen: Lambda = 1000 and dt = 0.01

For sub-section a) and b)





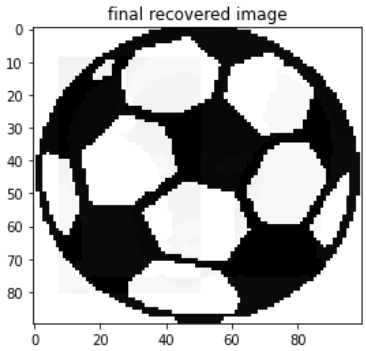
c) Plot the Root Mean Squared (RMS) error with time:

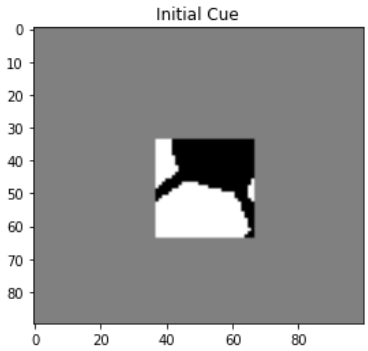


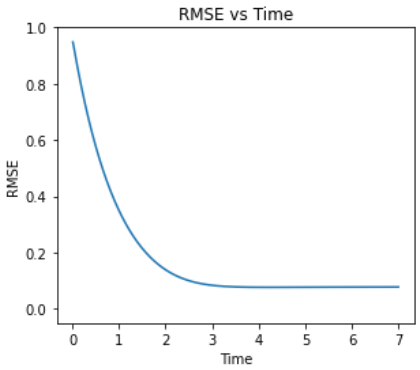
Part 3:

Image: ball

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

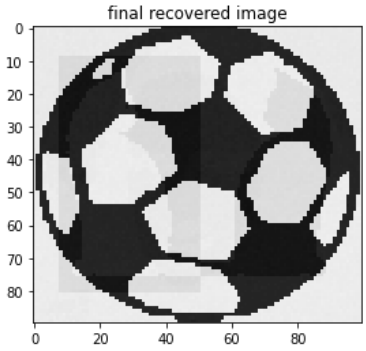


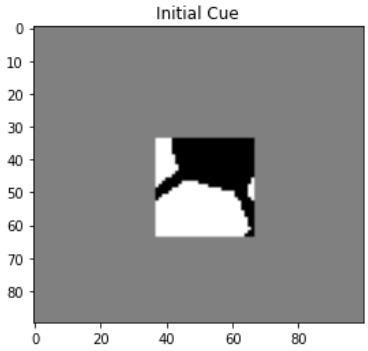


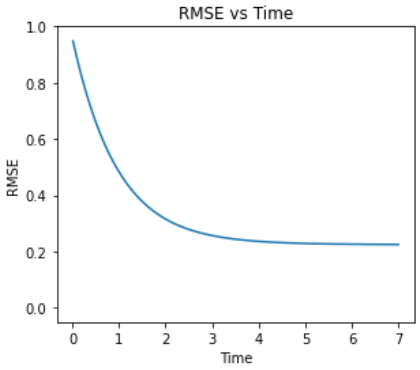


25% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

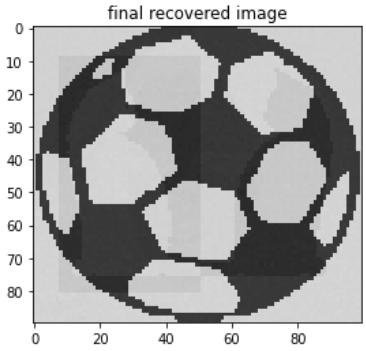


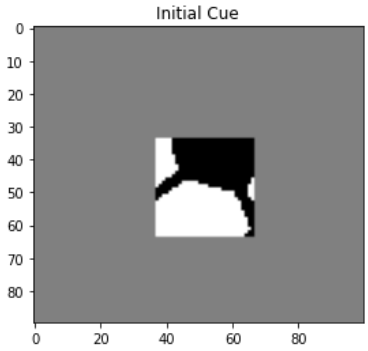


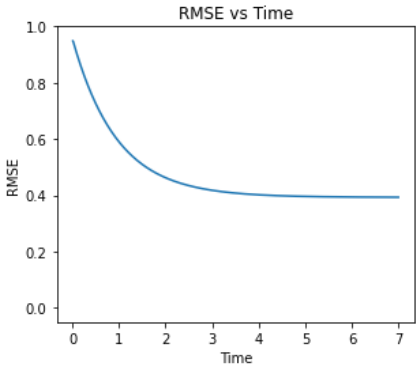


50% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

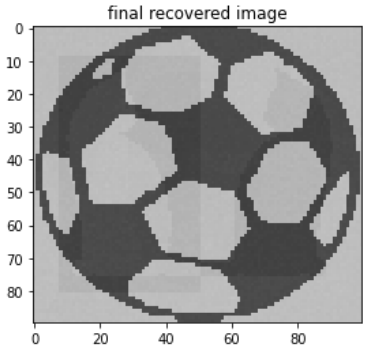


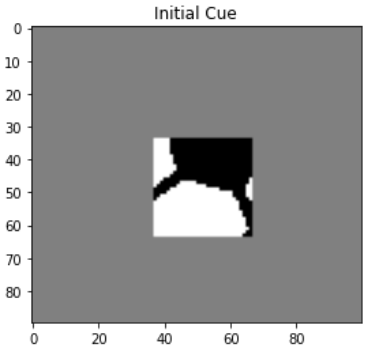




80% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :





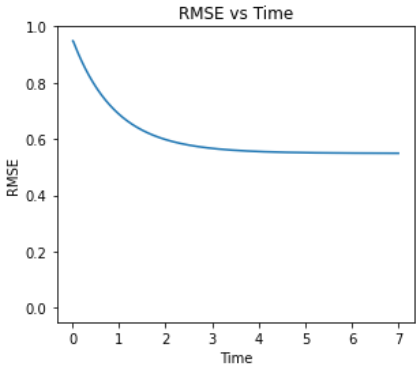
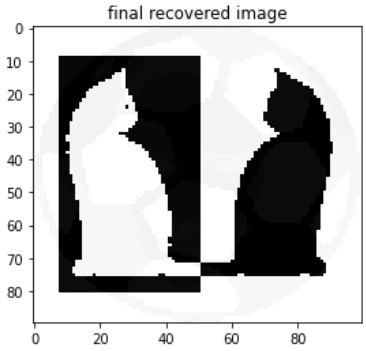
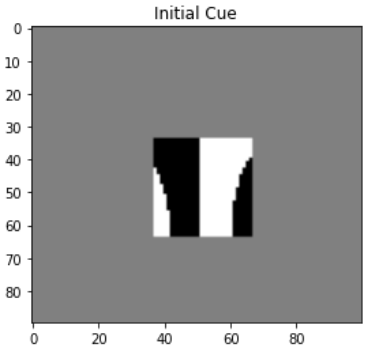
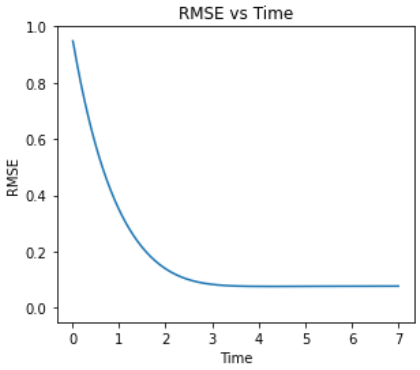


Image: cat

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

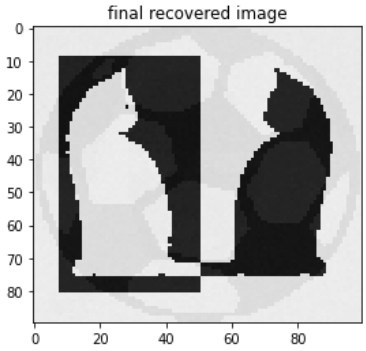


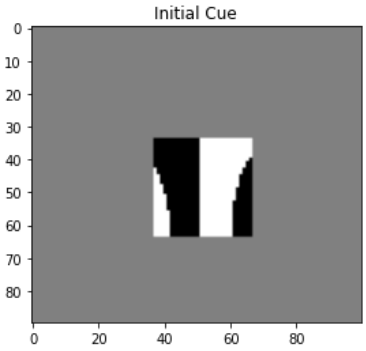


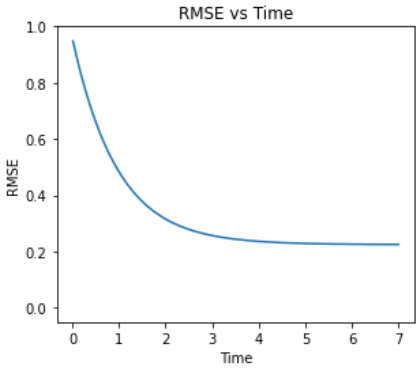


25% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

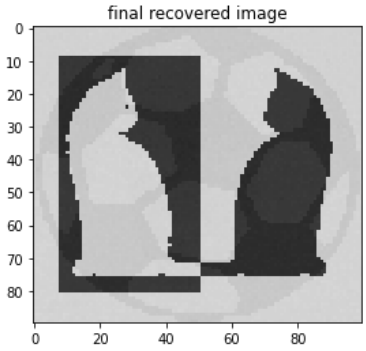


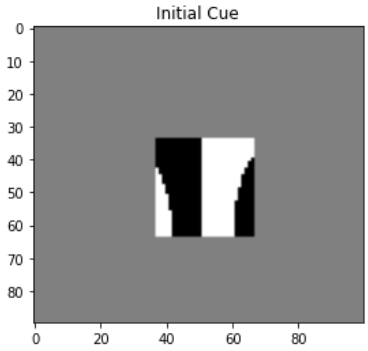


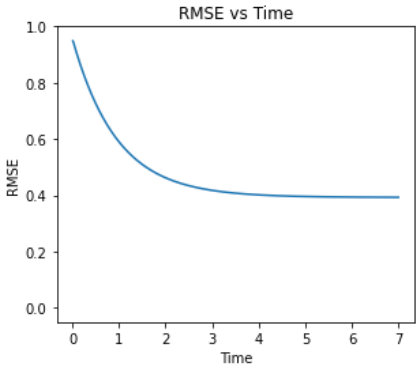


50% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

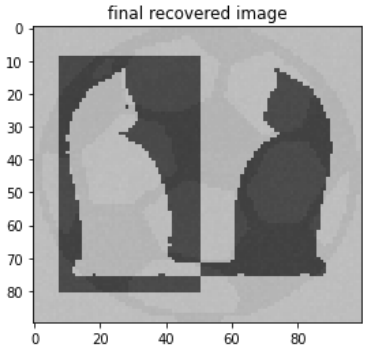


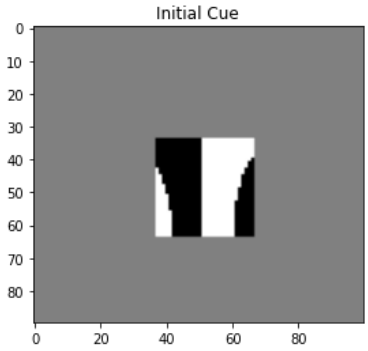




80% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :





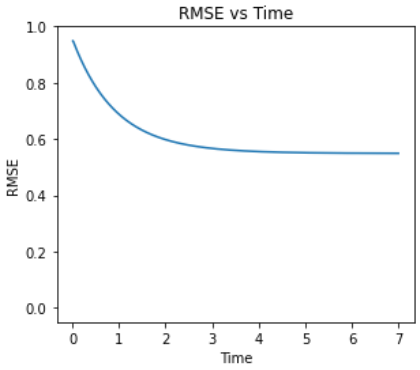
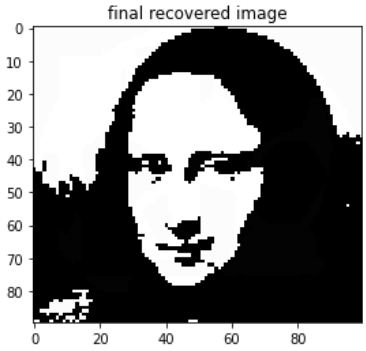
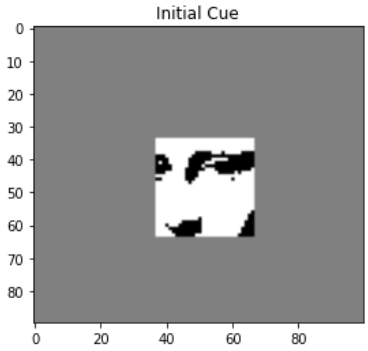
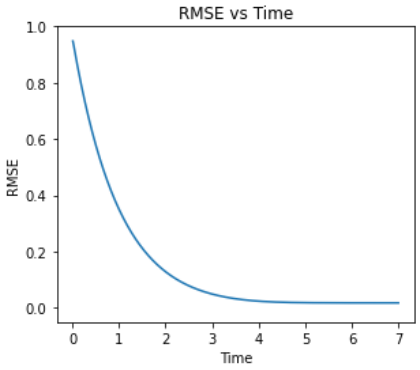


Image: mona

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

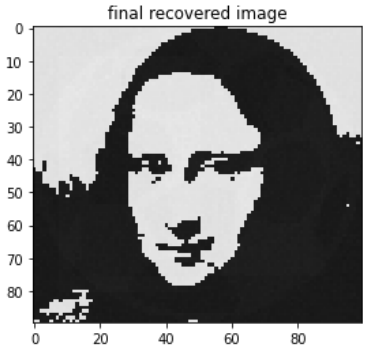


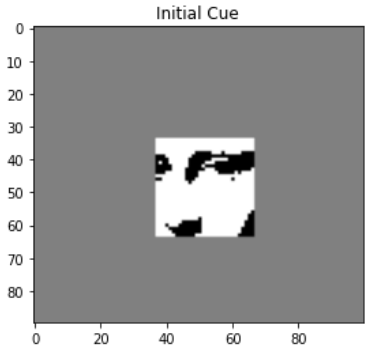


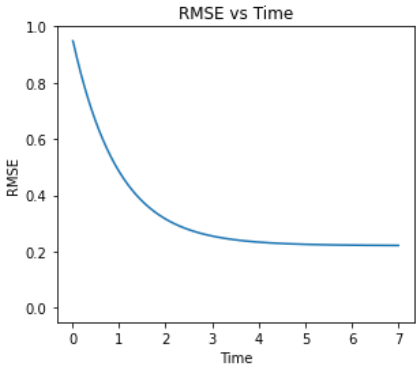


25% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

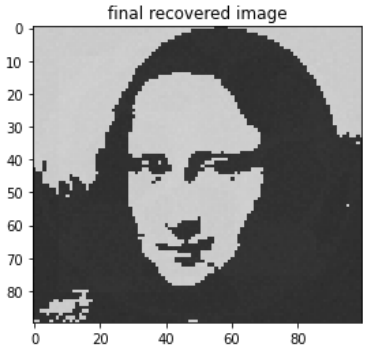


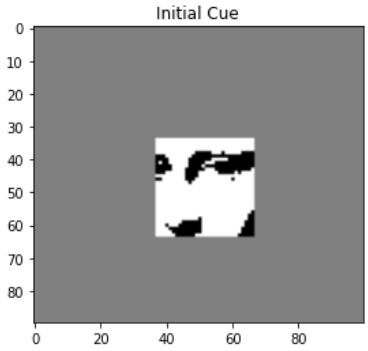


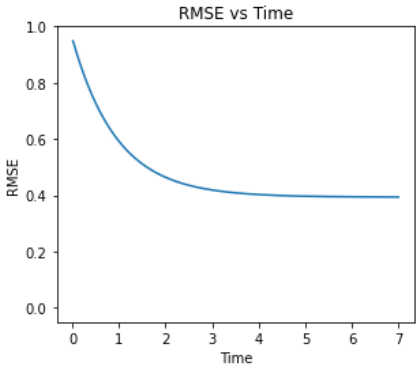


50% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :

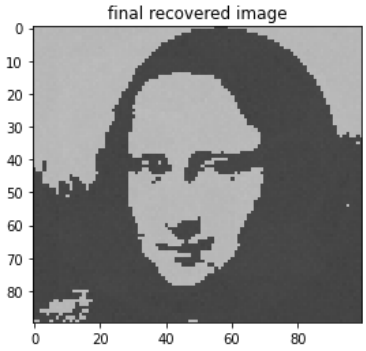


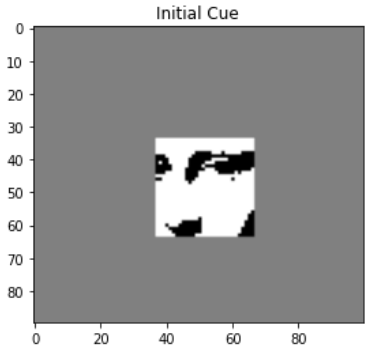


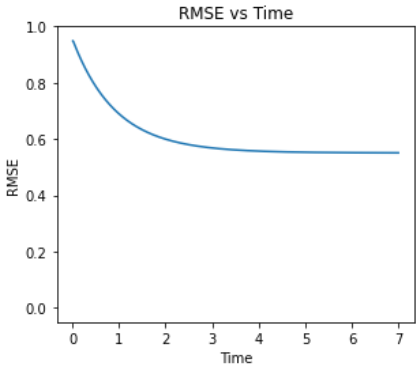


80% of the weights are made zero:

Retrieved Image, Cue Image, and Root Mean Squared (RMS) error vs time plot :







Observation:

We can observe very gradual degradation, although very robust as we increase the percentage of contribution of neurons being erased. Initially, there is a linear decrease in RMSE whereas after a certain point it exponentially decreases and quickly saturates. We can conclude that our network performs well even after a certain percentage of neurons are degraded but the Images appear darker and our RMSE curve saturates at a higher error value. As the damage increases our network exhibits large errors that cannot converge or recovered.

Code and Video Files:

1. BT6270-Assignment3.ipynb is an ipython notebook which contains all the necessary python codes for part 1, 2, and 3.
2. BT6270-Assignment3.ipynb notebook also contains all the necessary videos of iterations in a proper manner, easy to understand.
3. You can also find the videos in this drive link.
4. Videos are also attached in this zip folder.