Problem Statement: Implement the Hopfield Neural Network algorithm.

Introduction: Hopfield Neural Network algorithm is a kind of unsupervised learning algorithm. It is said unsupervised because there is no parameter which monitors or supervises the learning process. Hopfield networks are consisted of single layer which consists of one or more fully connected neurons. It is mostly used for optimization and association tasks. In this report, the implementation of Hopfield network in image recovering is mentioned.

Dataset Description and Preprocessing: The data used for this task is two images of 250 * 250 pixels. One image is the original image and the later one is the corrupted version of the original one. The images were both loaded and converted into binary image for this task. Later the images were converted into numpy array. Then the algorithm was employed on them.

Fig: (1) Original binary image, (2) Corrupted binary image

Methodology: For this task first the images were converted into numpy arrays. Then each inputs were converted into either 1 or -1 i.e the bipolar inputs. The weight matrix was calculated for the original data. Later the Hopfield Neural Network algorithm was implemented. For breaking condition, comparing with the value from previous iteration was used. If the same value was generated consecutively for 20 times then the process was terminated and prediction was stored as output.

Code:

```
import numpy as np, random
from numpy import asarray
import cv2
from google.colab.patches import cv2_imshow

original_img = cv2.imread('/content/pic.jpg',2)
ret, bw_img1 = cv2.threshold(original_img,127,255,cv2.THRESH_BINARY)
cv2_imshow(bw_img1)

imgArray1 = asarray(bw_img1, dtype=np.uint8)
original_data = np.zeros(imgArray1.shape)
original_data[imgArray1 > 0] = 1
original_data[original_data == 0] = -1
original_data                    # original data for training

distorted_img = cv2.imread('/content/pic3.jpg',2)
ret, bw_img2 = cv2.threshold(distorted_img,127,255,cv2.THRESH_BINARY)
cv2_imshow(bw_img2)

imgArray2 = asarray(bw_img2, dtype=np.uint8)
distorted_data = np.zeros(imgArray2.shape)
distorted_data[imgArray2 > 0] = 1
distorted_data[distorted_data == 0] = -1
distorted_data                     # distorted data for testing

weight = np.dot(original_data.transpose(), original_data)
np.fill_diagonal(weight, 0)   #fills the diagonal positions of weight matrix with zeros
weight

def test(test_data,weight):

  y = test_data.copy()
  threshold = 20
  count = 0
```

```
    y_prev = np.zeros(y.shape)

    while 1:
      i = np.random.choice(range(len(y)))
      j = np.random.choice(range(len(y[0])))
      y[i,j] = test_data[i, j] + np.dot(y[i,:], weight[:, j])
      if y[i,j] > 0:
        y[i,j] = 1
      else:
        y[i,j] = -1
      if y[i,j] == y_prev[i, j]:
        count += 1
        #print("count: ",count)
        if count >= threshold:
          return y
          break
      else:
        count = 0
      y_prev[i,j] = y[i,j].copy()

prediction = test(distorted_data,weight)
print(prediction)

correct = 0
incorrect = 0
boolean = np.equal(original_data,prediction)
for i in range(len(original_data)):
  for j in range(len(original_data[0])):
    if boolean[i,j] == False:
      incorrect += 1
    else:
      correct += 1
accuracy = (correct/ (correct+incorrect)) * 100
print("Accuracy: {}%".format(np.round(accuracy, 2)))
```

recovered = np.zeros(imgArray2.shape, dtype=np.uint8)

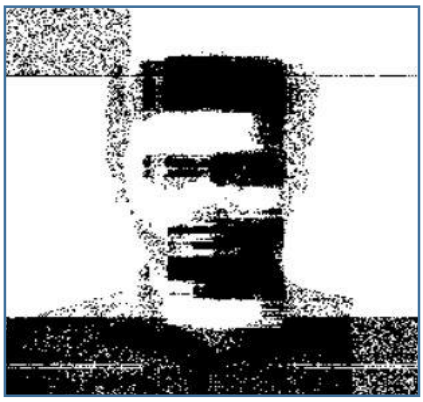
recovered[prediction == 1] = 255

recovered[prediction == -1] = 0

img = Image.fromarray(recovered)

display(img)

Result and performance analysis: An accuracy of 88.68% was achieved from this task. The outputs are demonstrated below.



```
[58] correct = 0
     incorrect = 0
     boolean = np.equal(original_data,prediction)
     for i in range(len(original_data)):
       for j in range(len(original_data[0])):
         if boolean[i,j] == False:
           incorrect += 1
         else:
           correct += 1
     accuracy = (correct/ (correct+incorrect)) * 100
     print("Accuracy: {}%".format(np.round(accuracy, 2)))

     Accuracy: 88.68%
```

Fig: (1) Reconstructed image, (2) Accuracy.

Conclusion: The reconstructed image quality can be enhanced i.e a better accuracy can be achieved tuning the hyperparameters.