University *of* Dayton

School of Engineering

Department of Electrical and Computer Engineering

ECE 477/595: Artificial Neural Networks

Project 6: Hopfield Neural Network

Instructor: Prof. K. Asari Vijayan

Student Name: Serge Alhalbi **Student ID:** 101682971

Date: 11/5/2022

Methodology

Loading Steps

- Using the “loadMNISTImages” and “loadMNISTLabels” functions, we load the both the training and testing images with their labels from the uploaded files respectively. Initially, both the training and testing datasets are made up of images with pixel values ranging from 0 to 255 (8-bit pixel). These pixel values are then normalized (divided by 255), allowing each pixel to have a value between 0 and 1.

- A subset is created from the original datasets using the “balance_MNIST_selection” function. The function is simple and clear. I have written another function that can be used to select images at random from each dataset. The user may specify the number of training and testing images any of these functions.

- If simple thresholding is being used, the data is binarized bipolarly taking either 1 or -1 as a pixel value. Otherwise, the values of the pixels would still range from 0 to 1.
  Bipolar images are created through simple thresholding at 0.5 the original data (output = 0 if input < 0.5 and output = 1 if input > 0.5).

- The user may use orthogonal patterns by using the output of the function “orthoimages”. These patterns have been built using the fact that two orthogonal patterns have a null dot product.

Hopfield Neural Network

1. Input each $X^s; s = 1,2, \dots, P.$ $(N \times 1)$
   Where X: $(N \times P)$

2. Initialize the weight matrix:
   $W = X \cdot X^T;\ (N \times N)$
   For stability concerns, the weight matrix should be symmetrical with zero entries for the diagonal. The first condition is already satisfied; however, each diagonal entry should be set to zero.

3. Loop over the patterns: $s = 1,2, \dots, P$

   a. Apply a distorted version of X^s:
      $Net = W \cdot X^s;\ (N \times 1)$

   b. Compute the output by thresholding:
      $$Y = \begin{cases} +1; & Net > 0 \\ X^s; & Net = 0 \\ -1; & Net < 0 \end{cases}$$
      Where Y: $(N \times 1)$

   c. Compute the energy function:
      $$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}\, x_j x_i = -\frac{1}{2} \cdot Net^T \cdot W \cdot Net$$
      Where E: (1×1)

   d. Replace the input with the output:
      $X(t+1) = Y(t)$

   e. Measure the change of the energy between the current time step and the previous one:
      $Change = E(t) - E(t-1)$

   f. Increase the iteration number:
      $Iteration = Iteration + 1$

   g. Repeat steps 3-a to 3-f until the $Change = 0$ or if $Iteration > 100$.

   h. Save the retrieved pattern as the final output and store it.

4. Compute the error matrix between each actual pattern and retrieved pattern, then plot it vs each pattern number:
   $Error_s = \frac{1}{N}\sum_{i=1}^{N} |Acutal\_X_i^s - Retrieved\_X_i^s|.$

Results

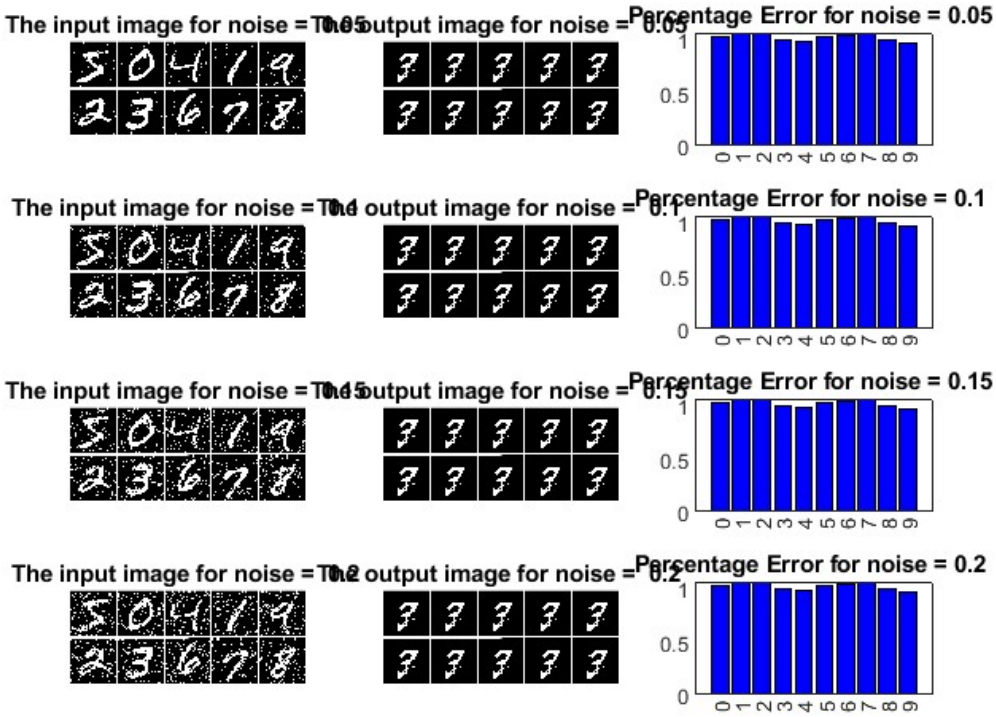

Figure 1: Pattern Association Using HNN - 10 Training Non-Orthogonal Patterns; Low Noise

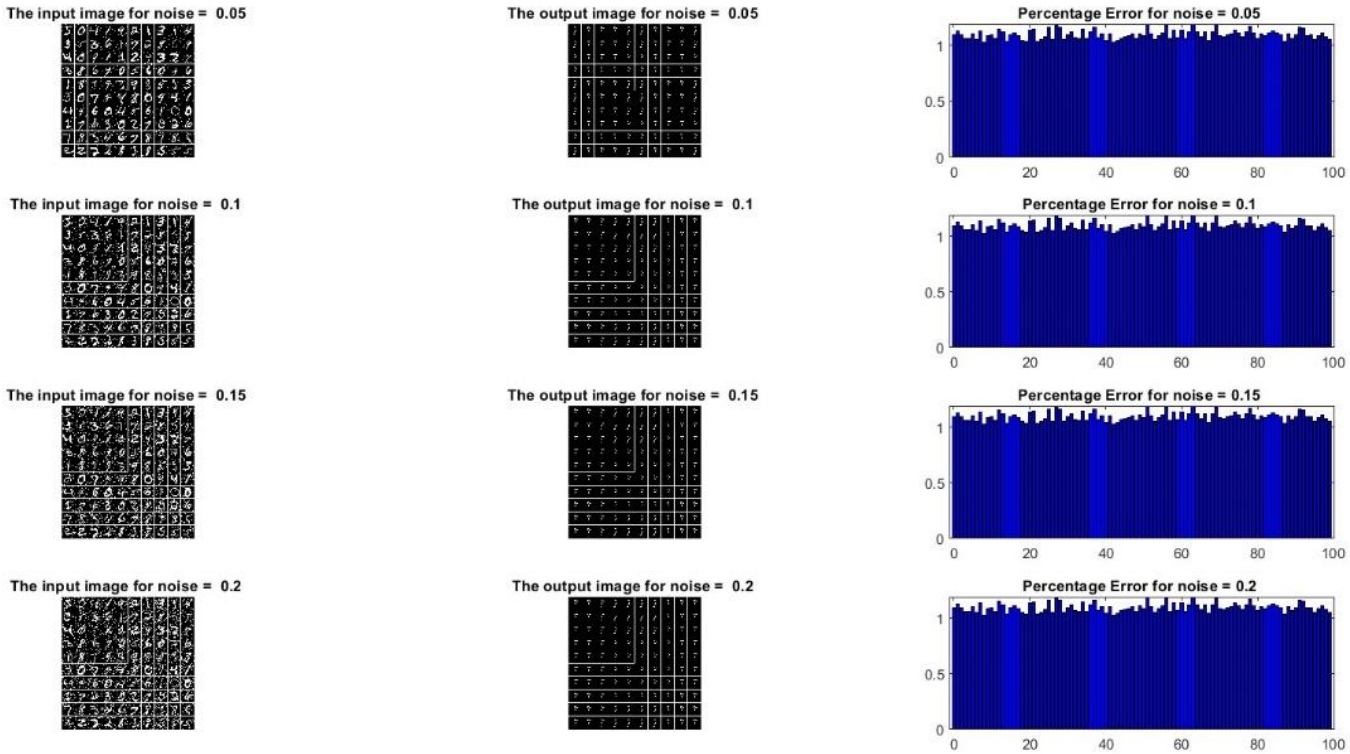

Figure 2: Pattern Association Using HNN - 100 Training Non-Orthogonal Patterns; Low Noise

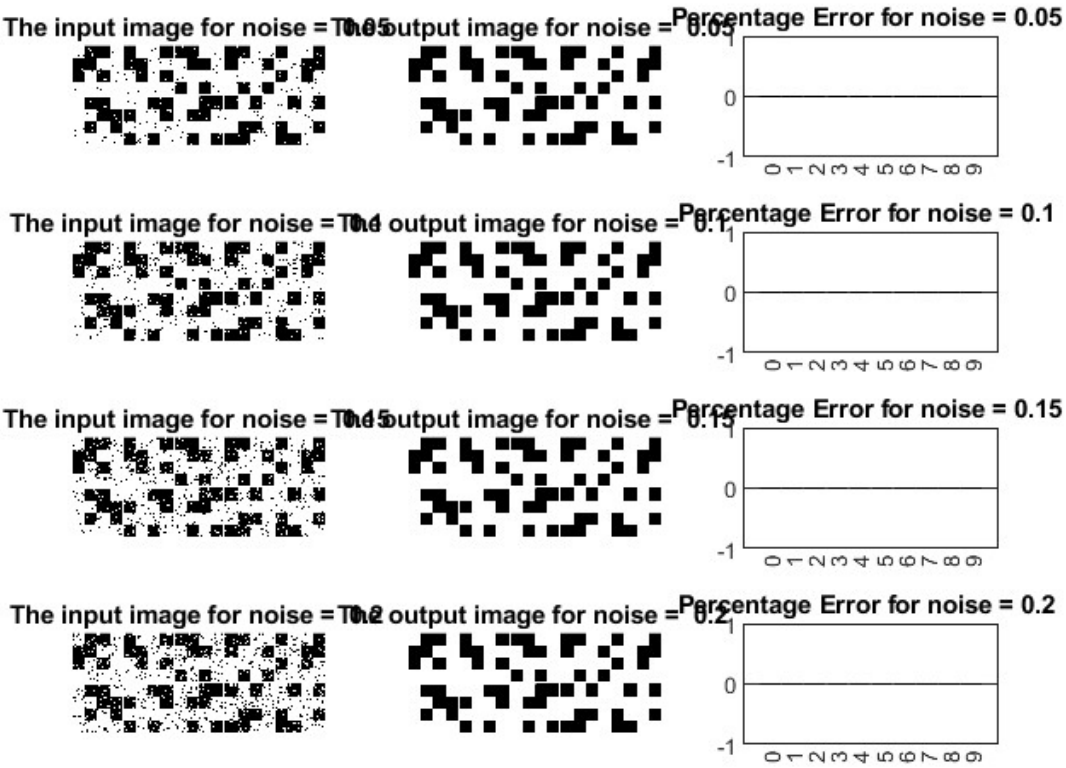

Figure 3: Pattern Association Using HNN - 10 Training Orthogonal Patterns; Low Noise

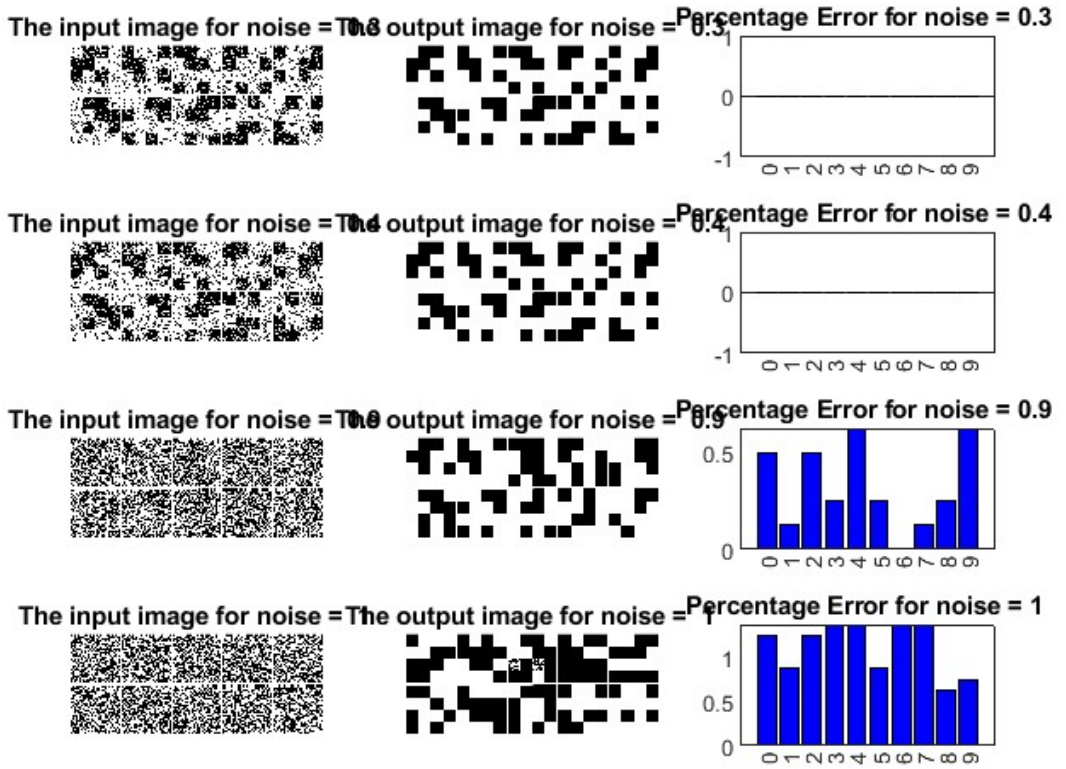

Figure 4: Pattern Association Using HNN - 10 Training Orthogonal Patterns; High Noise

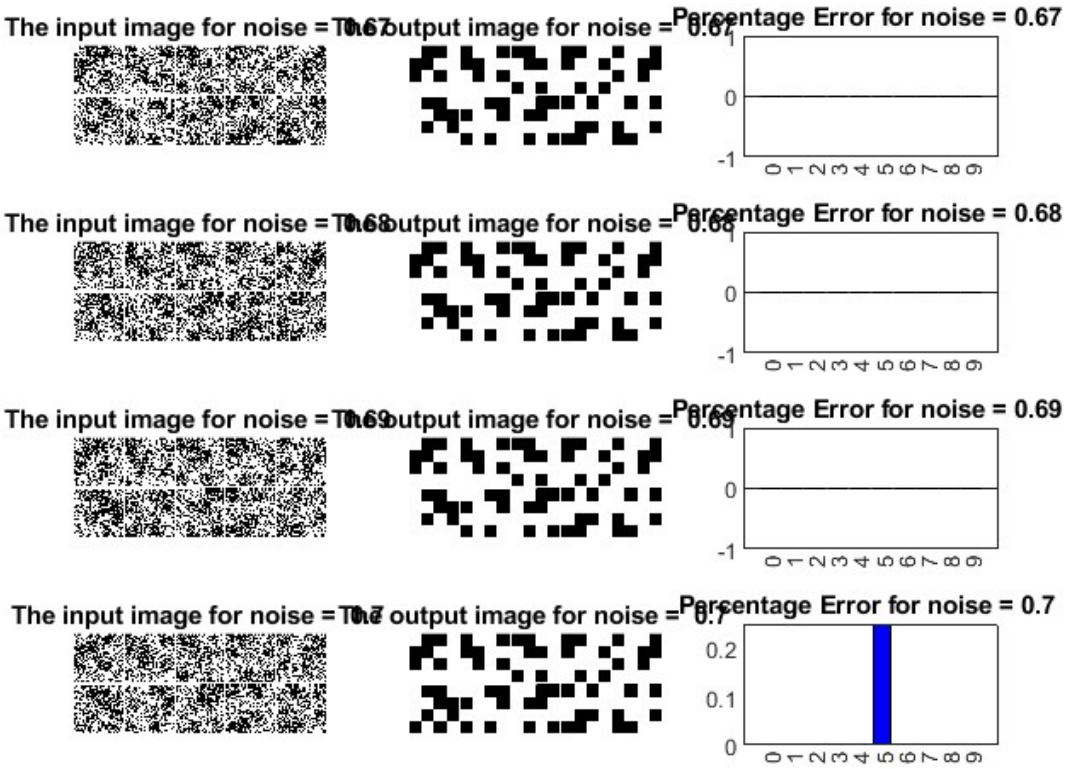

Figure 5: Pattern Association Using HNN - 10 Training Orthogonal Patterns; Critical Noise

Conclusion/Comments

The goal of this assignment was met, and a Hopfield Recurrent Neural Network based pattern association algorithm was developed. Some points may be discussed below:

- Even with low noise levels, the pattern association performed poorly in the case of non-orthogonal patterns. This is demonstrated by both the retrieved pattern images and the error bars. This was to be expected given that the handwritten digit patterns are not separable. The algorithm must have become stuck at a local minimum, and the correct patterns were not properly retrieved.

- The algorithm became stuck almost at the same local minima for all distorted and applied patterns. This is logical since the considered patterns are not separable and almost indistinguishable. As well, more patterns could not solve the problem as in the case of using one hundred patterns, the same outcome was obtained.

- The opposite is true when orthogonal patterns are used. For relatively low noise levels, all patterns were correctly retrieved, with no errors. This is one practical reason to believe that orthogonal patterns are required for the algorithm to function properly.

- As the noise level increases, the retrieved patterns become less like the actual ones. This is obviously logical because more noise will make it more difficult to retrieve the exact patterns.

- Even with high noise levels, some patterns could still be retrieved correctly while the human naked eye can’t even relate the distorted pattern to the actual one.

- The errors become different than zero when the noise intensity becomes more than or equal to 0.7.

- The algorithm is quick and simple to implement, with numerous applications.