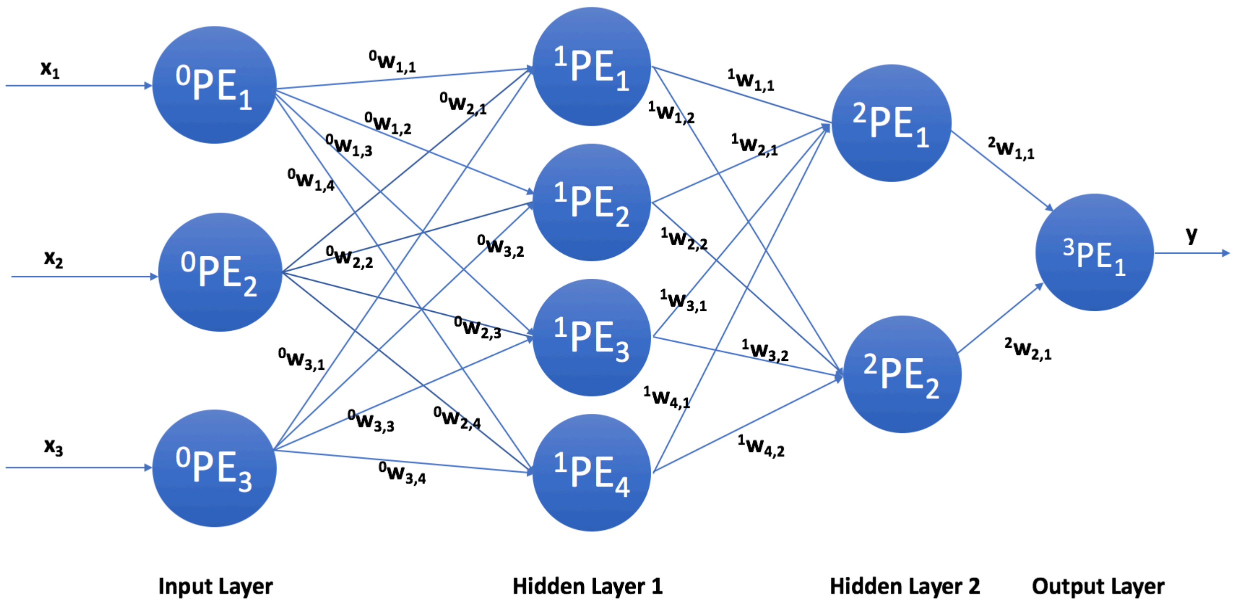
NML 502: Neural Machine Learning I

Homework III

Prashant Kalvapalle, Ragib Mostofa

Problem I:

The architectural scheme for the neural network described in the problem definition has been presented below.



Problem II (a):

As per the problem definition, the transfer function to be used is linear, i.e., $\mathbf{f(x) = x}$ (we know this from Problem 2(i) of Homework II) or $\mathbf{f(x) = mx}$, but recall that the slope can be absorbed into the inputs or the weights. The computations of the Processing Elements (PEs) are basically the edge-weighted sums of the input vector components. In addition, since the transfer function is linear or $\mathbf{f(x) = x}$, this weighted sum happens to be the final output for a particular PE.

Now, consider a multilayer, fully connected neural network with 1 input layer, **n** hidden layers and 1 output layer, or in other words, with n layers of weight computations. Suppose that the number of PEs in the $\mathbf{i}^{\text{th}}$ layer is $\mathbf{L_i}$ and that the input vector **x** has **p** components while the output vector **y** has **q** components. Moreover assume that $^k\mathbf{w_{i,j}}$ is the weight of the edge connecting node (or PE) **j** from the $\mathbf{k}^{\text{th}}$ hidden layer to node **i** in the next layer.

Hence, the first processing element in the first hidden layer would calculate $^0\mathbf{w_{1,1}x_1 + \ldots + {}^0w_{1,p}x_p}$, (the superscript, 0 refers to the input layer) the second processing element would calculate $^0\mathbf{w_{2,1}x_1 + \ldots + {}^0w_{2,p}x_p}$ and so on until the $\mathbf{L_1}^{\text{th}}$ processing element calculates $^0\mathbf{w_{L\_1,1}x_1 + \ldots + {}^0w_{L\_1,p}x_p}$. Encoding this in a matrix, we get, $\mathbf{x^1 = W_1x}$, where $\mathbf{x^1}$ is the output of the first hidden layer with the weighted sums described above as components and,

$$\boldsymbol{W_1} = \begin{bmatrix} w^0_{1,1} & \cdots & w^0_{1,p} \\ \vdots & \ddots & \vdots \\ w^0_{L_1,1} & \cdots & w^0_{L_1,p} \end{bmatrix}$$

Notice that since we are working with linear transfer functions or **f(x) = x**, the $\mathbf{x}^1$ computed above is the final output of the first hidden layer.

Likewise, we can calculate $\mathbf{x}^2$ for the second hidden layer by first constructing the matrix $\mathbf{W}_2$ from the weights of the edges leading to the hidden layer as was done with the first hidden layer above. Repeating this for all n hidden layers and the, we should be able to express the output vector $\mathbf{y} \in \mathbf{R}^q$, in terms of the input vector $\mathbf{x} \in \mathbf{R}^p$ and the weights of all n hidden layers appropriately arranged as follows:

$$y = \begin{bmatrix} w^n_{1,1} & \cdots & w^n_{1,L_n} \\ \vdots & \ddots & \vdots \\ w^n_{L_{n+1},1} & \cdots & w^n_{L_{n+1},L_n} \end{bmatrix} \begin{bmatrix} w^{n-1}_{1,1} & \cdots & w^{n-1}_{1,L_{n-1}} \\ \vdots & \ddots & \vdots \\ w^{n-1}_{L_n,1} & \cdots & w^{n-1}_{L_n,L_{n-1}} \end{bmatrix} \cdots \begin{bmatrix} w^1_{1,1} & \cdots & w^1_{1,L_1} \\ \vdots & \ddots & \vdots \\ w^1_{L_2,1} & \cdots & w^1_{L_2,L_1} \end{bmatrix} \begin{bmatrix} w^0_{1,1} & \cdots & w^0_{1,p} \\ \vdots & \ddots & \vdots \\ w^0_{L_1,1} & \cdots & w^0_{L_1,p} \end{bmatrix} x$$

$$y = W_{n+1} W_n \ldots W_2 W_1 x$$

$$y = Wx$$

where $\mathbf{W} = \mathbf{W}_{n+1}\mathbf{W}_n \ldots \mathbf{W}_2\mathbf{W}_1$ and each $\mathbf{W}_i$ is constructed from the weights of the edges leading to the **i**th hidden layer as shown above.

Note: Due to formatting issues and constraints, $^k w_{i,j}$ is represented as $w^k_{i,j}$ in the equations above.

Problem II (b):

In order for a neural network to approximate or learn a function/mapping, it must have the correct number of layers and the right transfer function. Having more layers means nesting these functions inside each other and this certainly allows the network to represent more and more complex functions. Hence, this proves an apparent law - the simpler the transfer function, the more structurally complex (i.e. more hidden layers, meaning more processing elements) the artificial neural network is required to be for the same data or mapping.

Problem III (a):

The following network and learning parameters were employed for each of the image matrices in this exercise.

Learning rate (μ)	0.1
Maximum training iterations (n)	100,000
Error threshold for stopping (tol)	0.4

Justification for Learning Parameters:

- **Learning rate (μ):** While the learning rate is chosen to be as high as possible for the interests of speed, care must be taken so that it is not set so high that the algorithm shifts away from the desired equilibrium points. Since $\mu = 0.1$ fits these criteria perfectly, this value of the learning rate is used.

- **Maximum training iterations (n):** We choose a value as high as 100,000 for the maximum number of training iterations in order to ensure that for most training runs, the algorithm reaches the desired accuracy or falls below the input error threshold first before the maximum iteration is reached.
- **Error threshold (tol):** We wish the error threshold to be realistically small, given the restrictions of the memory algorithm, which is why tol = 0.4 is used.

Stopping Criteria:
Training is halted in one of two cases: when the maximum number of learning iterations have been exceeded (i.e. when $n > 100{,}000$ for this case) or when the error threshold for stopping has been reached (when $tol < 0.4$ in this case), depending on which occurs first.

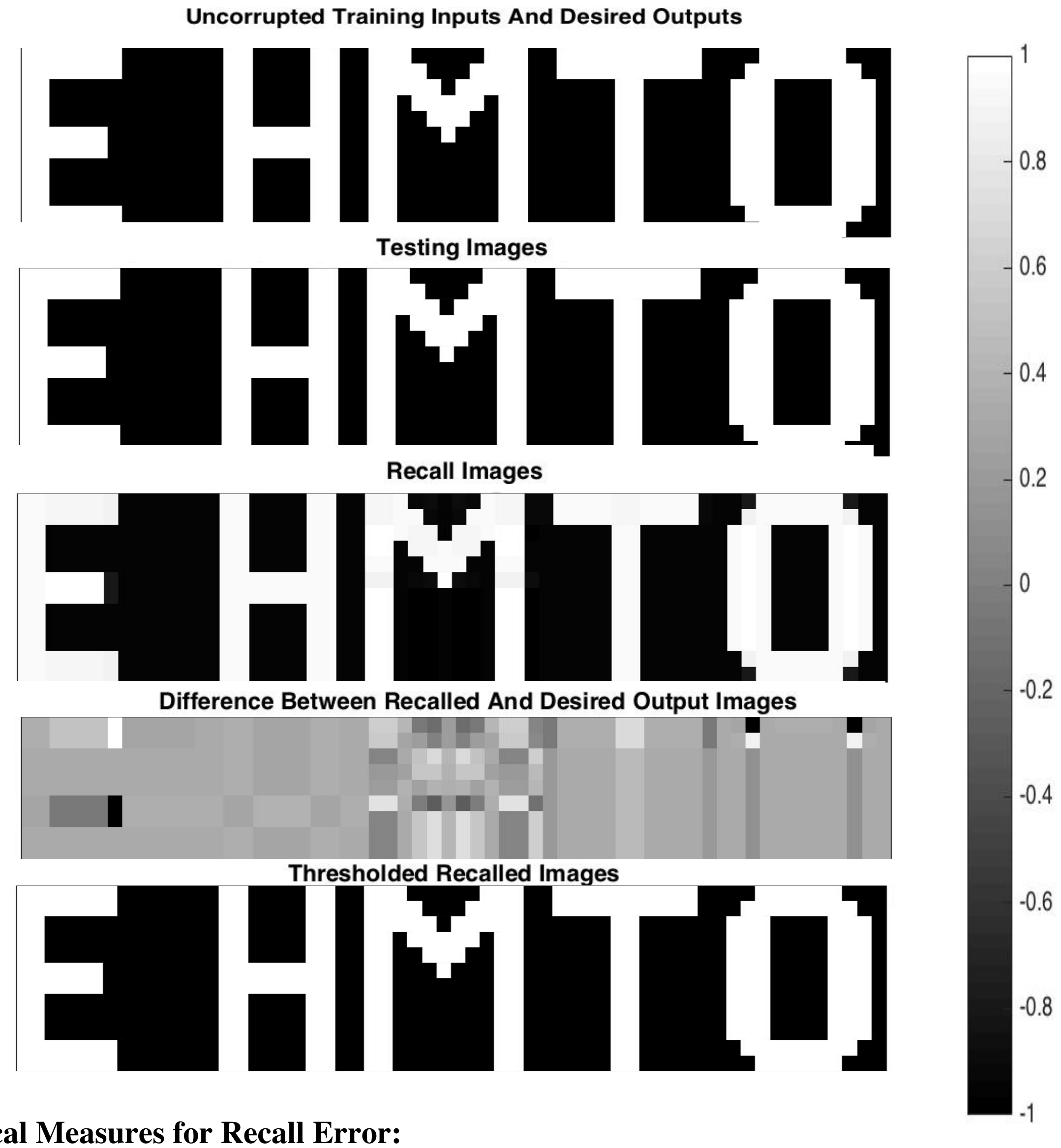
Thresholded Images:
Images are thresholded (as shown on the 5th row below), by setting pixels to 1 if the recalled value was positive and -1 if the recalled value was negative. Pixels with the 0 value were left unchanged.

Number of Training Iterations Required:
Despite the fact that this measure varies for different runs on the same input character/image matrix (due to the random permutations of the input row vectors), it has been reported below so that a general idea can be obtained for the number of training iterations required for each of the characters below.

- E: 36
- H: 24
- M: 48
- T: 24
- 0: 36

(The 5 resulting images are shown on the next page)

The 5 resulting images are below. The 5 images in each group are placed in a row with no gap between them.



Numerical Measures for Recall Error:

- Average absolute difference (average of absolute pixel values in the Difference images):
    - E: 0.0108
    - H: 0.0037
    - M: 0.0279
    - T: 0.0081
    - 0: 0.0099
    - Total: 0.0604
    - Simple Average: 0.0121
- Percentage of mismatched pixels in the Thresholded Recalled images:
    - E: 0.00
    - H: 0.00
    - M: 0.00
    - T: 0.00
    - 0: 0.00

- Total: 0.00
- Simple Average: 0.00

Brief Interpretation of Results:
The results were extremely positive for this experiment, mainly due to the fact that the training and testing input images had no noise in them at all. Moreover, the learning parameters were also maxed to obtain very favorable results. Although, the average percentage of mismatch as reported above is 0.00 % it could be misleading to state that the memory algorithm performs with a 100 % accuracy since we have not yet tested it against noisy data.
Note: The thresholded image acts similar to a Heaviside transfer function here.

<u>Problem III (b):</u>

The same network and learning parameters were employed for each of the image matrices in this exercise as compared to the last one, so that the results may be compared validly.

Learning rate (μ)	0.1
Maximum training iterations (n)	100,000
Error threshold for stopping (tol)	0.4

Justification for Learning Parameters:
The justification for choosing the values of the learning parameters are the same as above.

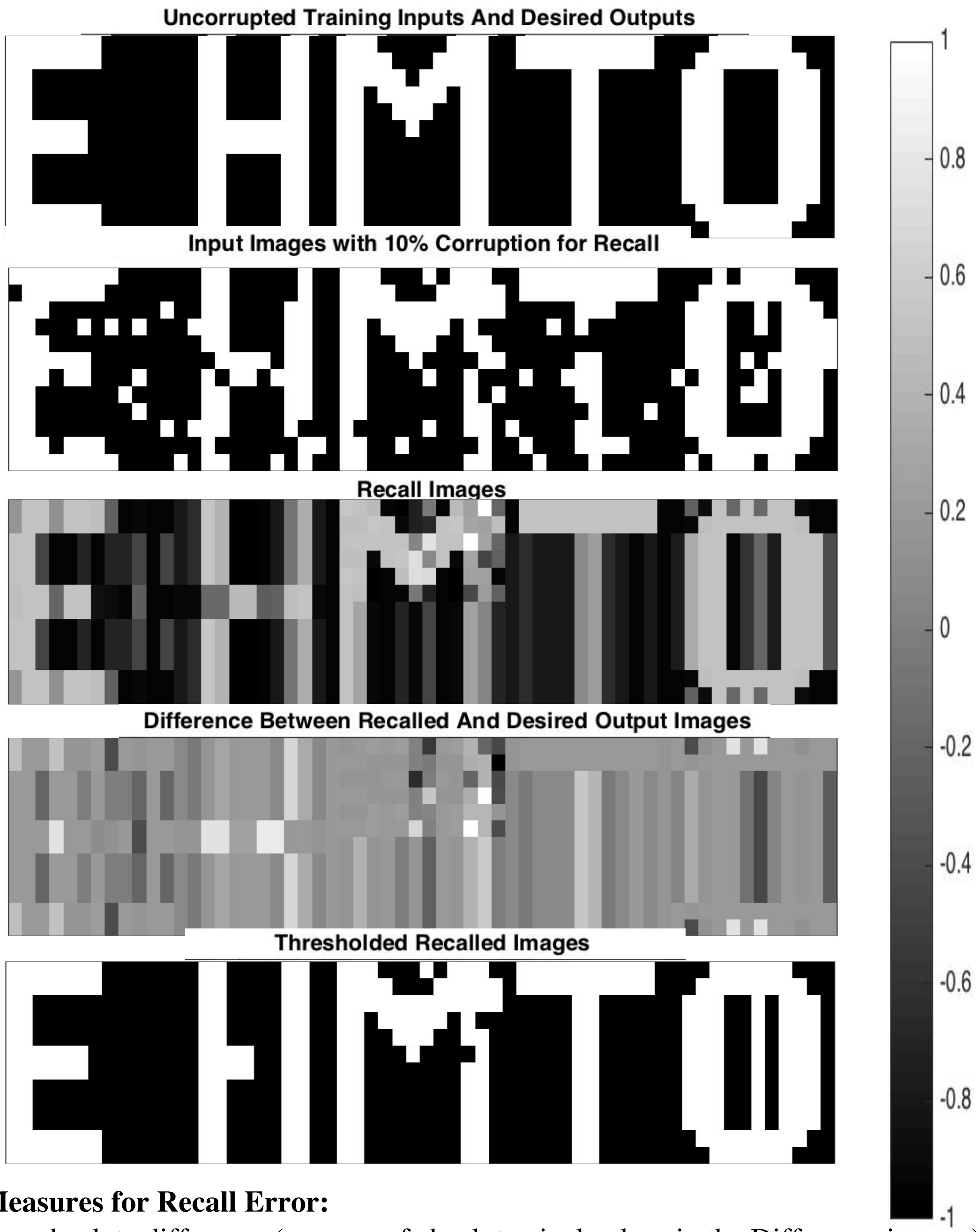
Stopping Criteria:
Training is halted in one of two cases: when the maximum number of learning iterations have been exceeded or when the error threshold for stopping has been depending on which occurs first.

Thresholded Images:
Images are thresholded (as shown on the 5th row below), by setting pixels to 1 if the recalled value was positive and -1 if the recalled value was negative. Pixels with the 0 value were left unchanged.

Keeping the parameters above, constant, we train and test the memory algorithm on the provided image matrices using different levels of corruption.

(The 5 resulting images are shown on the next page)

Corruption Level: 10 %



Numerical Measures for Recall Error:

- Average absolute difference (average of absolute pixel values in the Difference images):
    - E: 0.2128
    - H: 0.2202
    - M: 0.2550
    - T: 0.2095
    - 0: 0.2119
    - Total: 1.1094
    - Simple Average: 0.2219
- Percentage of mismatched pixels in the Thresholded Recalled images:
    - E: 0.00
    - H: 2.78
    - M: 4.17

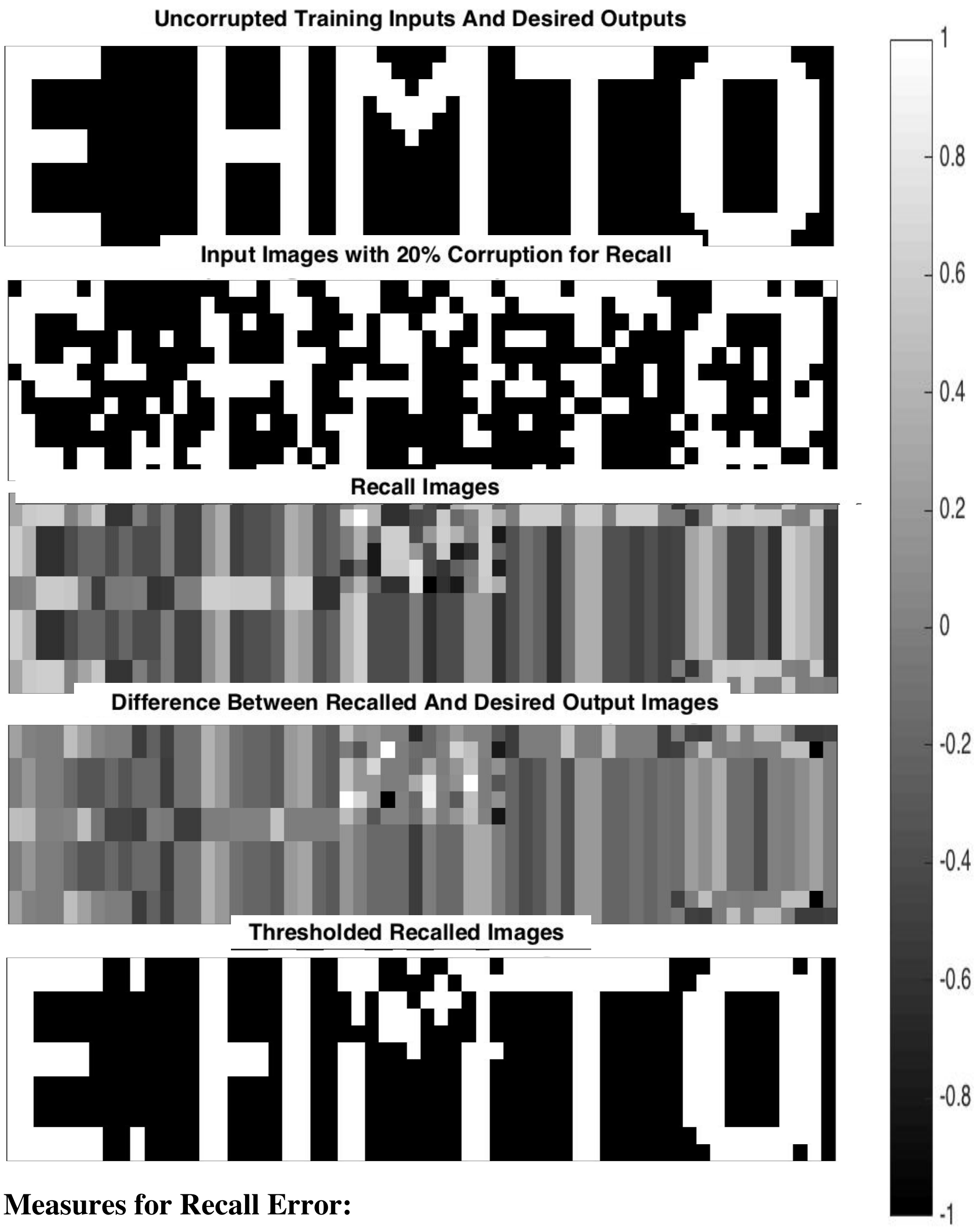
- T: 0.00
- 0: 5.56
- Total: 12.51
- Simple Average: 2.50

Brief Interpretation of Results with 10 % Corruption:
With 10 % corruption, error starts creeping into the recalled images as demonstrated by the figures above. The simple average for the average absolute difference has risen by 0.1998 which is significant considering that only 10% noise has been introduced. The percentage of mismatched pixels however shows a much more positive picture since there was only a 2.5 % rise in error. However, this could be due to the fact that the percentage of mismatched error is based on the thresholded recalled images which provide better results/output in general than the normal recalled images.

(The next corruption level is investigated on the next page)

Corruption Level: 20 %

The five images are shown below.



Numerical Measures for Recall Error:

- Average absolute difference (average of absolute pixel values in the Difference images):
    - E: 0.4074
    - H: 0.4027
    - M: 0.4572
    - T: 0.4031
    - 0: 0.4020
    - Total: 2.072
    - Simple Average: 0.4145
- Percentage of mismatched pixels in the Thresholded Recalled images:
    - E: 2.78

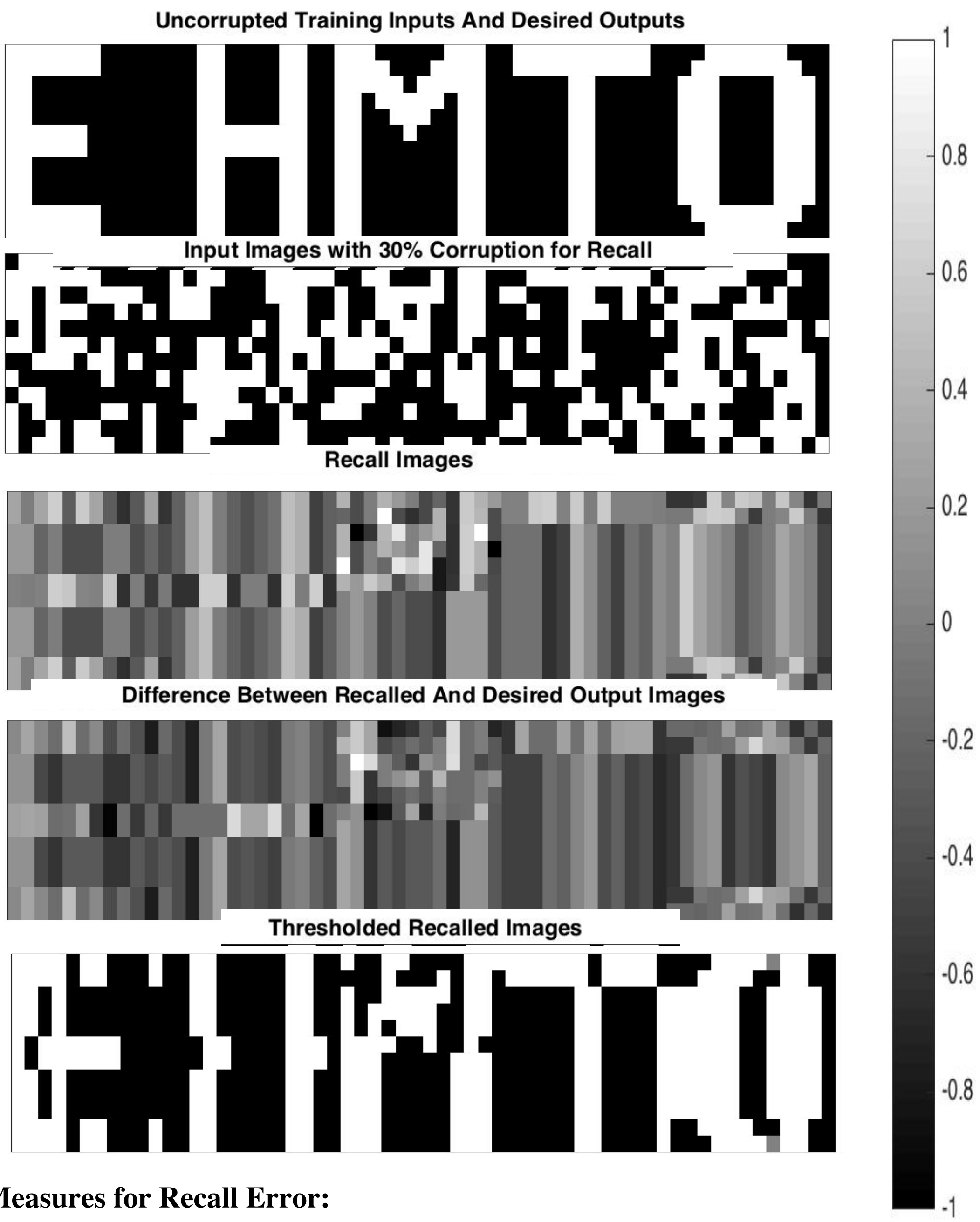
- H: 1.39
- M: 11.11
- T: 2.78
- 0: 2.78
- Total: 20.84
- Simple Average: 4.17

Brief Interpretation of Results with 20 % Corruption:
With 20 % corruption, the error has doubled as compared to results with only 10 % corruption. The simple average for the average absolute difference has more than doubled to 0.4145 while the average for the percentage of mismatched pixels has risen to less than twice the error seen last time. The recalled images have become progressively less identifiable with the actual images. However, the thresholded recalled images perform better than other recalled images just like last time.

(The next corruption level is investigated on the next page)

Corruption Level: 30 %

The five images are shown below.



Numerical Measures for Recall Error:

- Average absolute difference (average of absolute pixel values in the Difference images):
  - E: 0.6122
  - H: 0.6111
  - M: 0.6759
  - T: 0.6128
  - 0: 0.6161
  - Total: 3.1281
  - Simple Average: 0.6256
- Percentage of mismatched pixels in the Thresholded Recalled images:
  - E: 13.89

- H: 20.83
- M: 25.00
- T: 9.72
- 0: 22.22
- Total: 91.66
- Simple Average: 18.33

Brief Interpretation of Results with 20 % Corruption:
With 30 % corruption, once again, the error has increased by about 0.20 for the average absolute difference error measure, which is a linear growth. However, the percentage of mismatched pixels error measure has exponentially grown since last time with 20 % corruption.

Overall, the exponential increase in error makes it harder for the human eye to distinguish between recalled images tested using further levels of corruption. The higher levels of corruption can be investigated if we were to begin by training the memory on tighter leaerning parameter values, i.e. if the error tolerance was set to a value lower than the 0.4 that we have been using.

<u>Problem III (c):</u>

The same network and learning parameters were employed for each of the image matrices in this exercise as compared to the last one, so that the results may be compared validly.

Learning rate (μ)	0.1
Maximum training iterations (n)	100,000
Error threshold for stopping (tol)	0.4

Justification for Learning Parameters:
The justification for choosing the values of the learning parameters are the same as above.

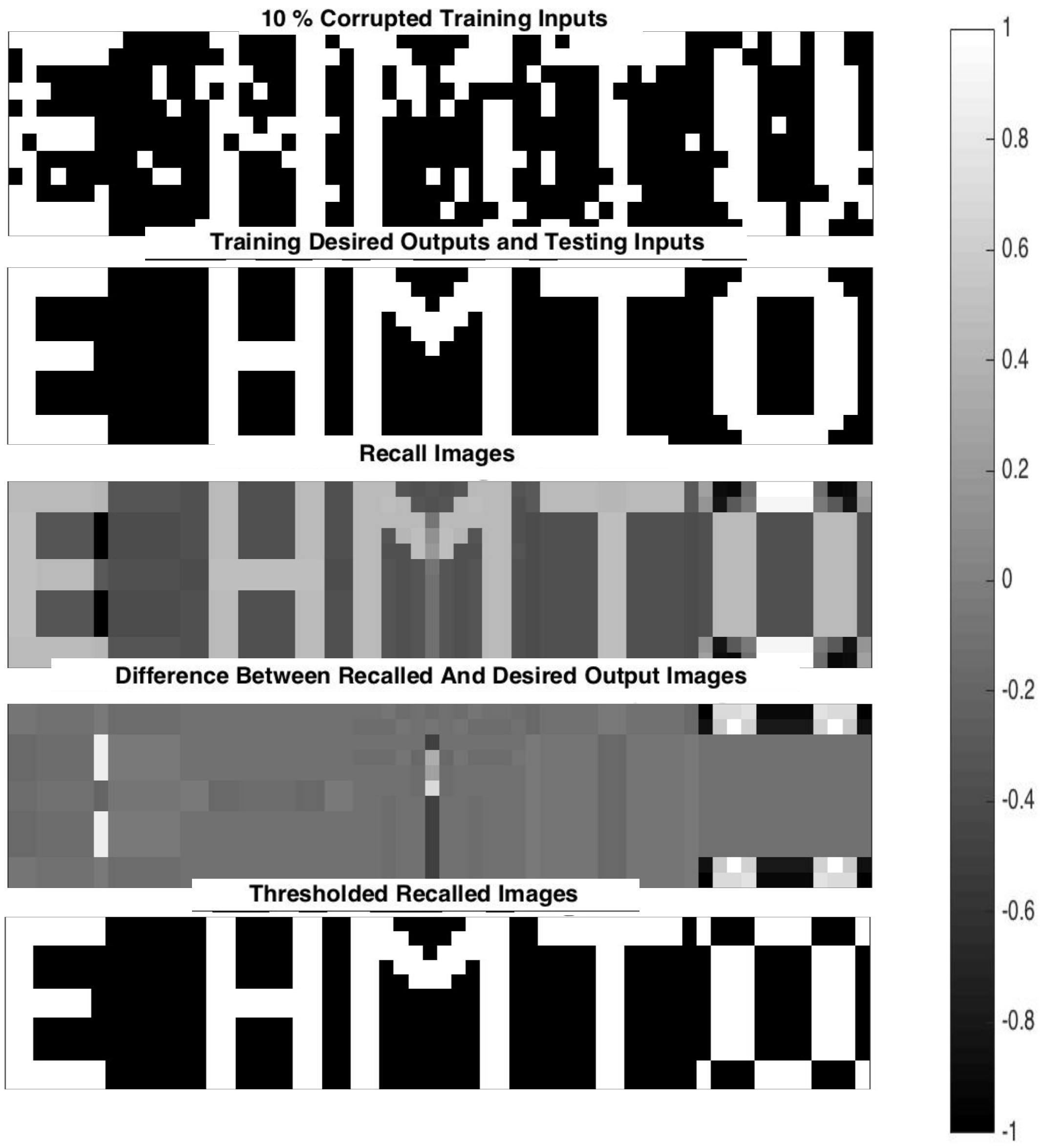
Stopping Criteria:
Training is halted in one of two cases: when the maximum number of learning iterations have been exceeded or when the error threshold for stopping has been depending on which occurs first.

Thresholded Images:
Images are thresholded (as shown on the 5th row below), by setting pixels to 1 if the recalled value was positive and -1 if the recalled value was negative. Pixels with the 0 value were left unchanged.

We use 10% corruption level images to train a new memory using the learning parameter as stated above.

Number of Training Iterations Required:
- E: 1188
- H: 480
- M: 252
- T: 528
- 0: 100,000

The five images are shown below.



Numerical Measures for Recall Error:

- Average absolute difference (average of absolute pixel values in the Difference images):
  - E: 0.1184
  - H: 0.0220
  - M: 0.0717
  - T: 0.0628
  - 0: 0.4559
  - Total: 0.7308
  - Simple Average: 0.1462
- Percentage of mismatched pixels in the Thresholded Recalled images:
  - E: 0.00
  - H: 0.00
  - M: 0.69
  - T: 0.00
  - 0: 13.89

- Total: 14.58
- Simple Average: 2.92

Brief Interpretation of Results:
The number of iterations required to finish training the memory was significantly greater for this exercise than for part (a). While in part (a), all the letters were learnt before the maximum iteration count was reached, one of the letters, namely 0, exceeded this count for part (c). This is due to the fact that the actual images were used to train the memory matrices in part (a) but images with 10 % corruption levels were used for part (c). Moreover, the average absolute difference of the pixel values was much higher in part (b) for 10 % corruption than for this part of the problem.