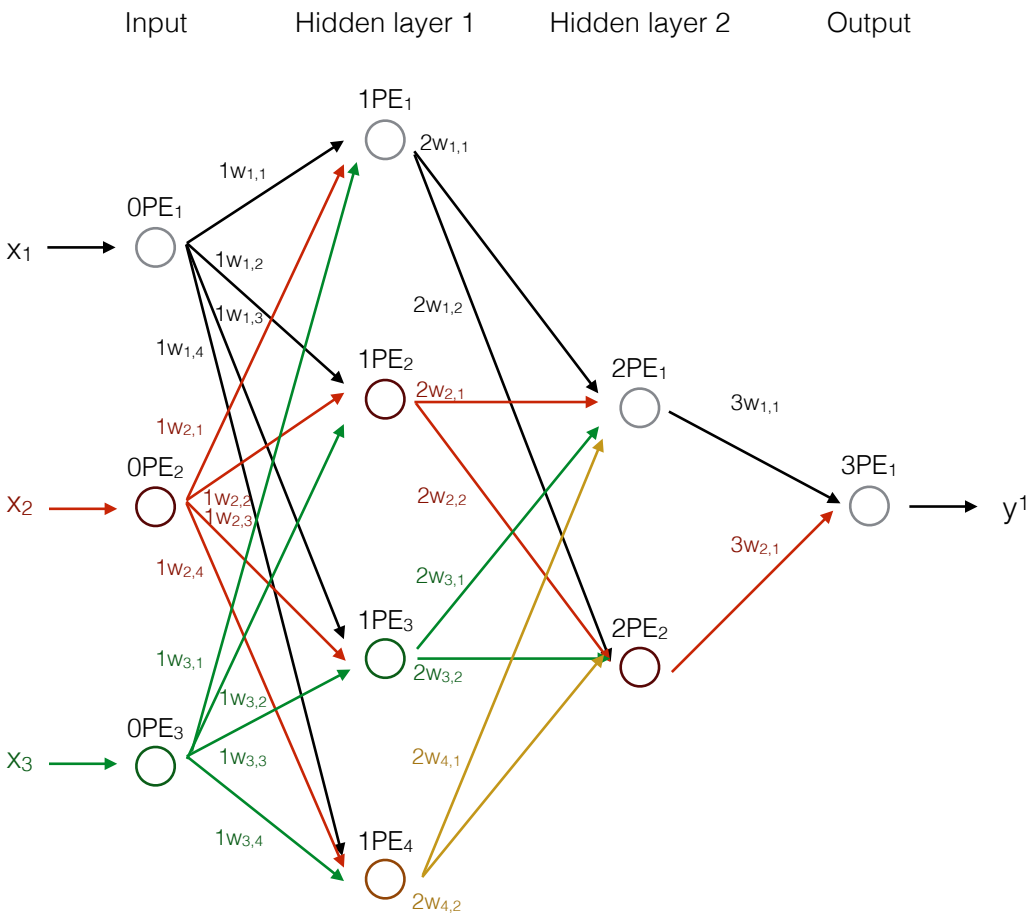
Zhuo Chen, Yidan Pan
Equally contributed

Homework 3

Total possible score: 60 points, 60 points = 100%

Problem 1.



Problem 2.
i)

Assume that $\mathbf{x}$ is an nx1 vector:
For the linear transfer in feed forward network $F : \mathbf{x} \to \mathbf{y}$, we can write it as:

$$F(x_i) = w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{in}x_n = \mathbf{w}_i x_i = y_i$$

where i=1,2,....n
Therefore:

$$\mathbf{y} = \{y_1, \ldots, y_n\} = \{\mathbf{w}_1 x_1, \ldots, \mathbf{w}_n x_n\} = \mathbf{W}\mathbf{x}$$

ii)

The structural complexity of linear transfer will not change during transfer and the errors will maintain: Assume the true input as $\mathbf{x}_i$(the ideal input)+$\mathbf{x}$(error in input), then:

$$\mathbf{y} = \{y_1, \ldots, y_n\} = \mathbf{W}(\mathbf{x} + \mathbf{e}) = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{e}$$

The result of transfer will always be the linear combination of input.

Problem 3.

a)

Show the original character images in matlab:

Then we modified the data by changing 0 to -1 and reshape the matrix. We got a 144×5 matrix, and it's used as X(input) and Y(desired output). Then we used function errcorr.m to train the network,and got the memory.

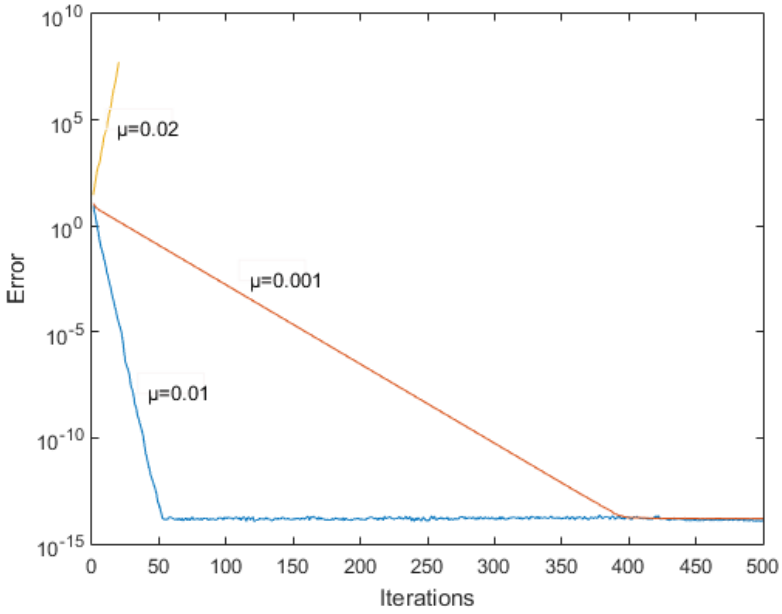


Figure 1: Changing of error along with iterations. Here the error is 2-norm(Y-MX)

From fig.1 we can see when $\mu = 0.02$ the function cannot converge. For μ=0.01 or 0.001 the function can converge, so we pick $\mu = 0.01$ and tol=10^{-5}, $maxn = 100$. Also we can see the error could be controled lower then 10^{-13}, which means the recall error of the training set is very low. Since the matrix can almost perfectly recall the training image, so we didn't show the difference map here.

b)

Flip the image, then recall it with the memory trained in (a). With different corruption fraction we have

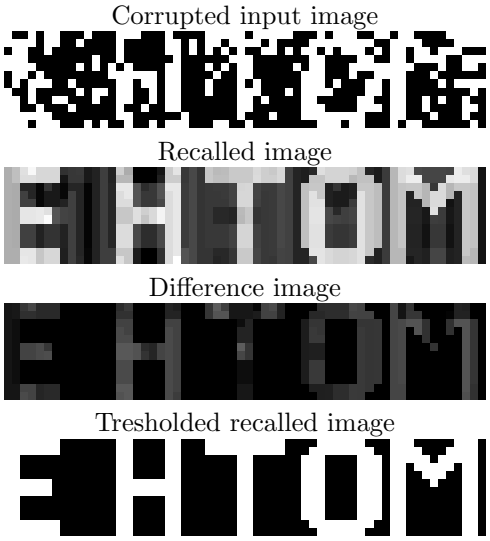


Figure 2: Recall with 20% corrupted data.

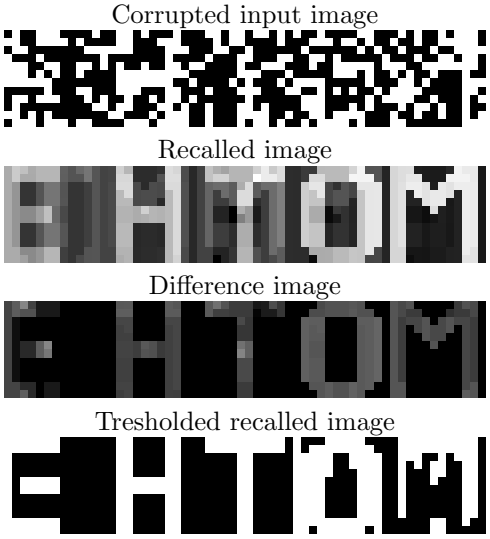


Figure 3: Recall with 30% corrupted data.

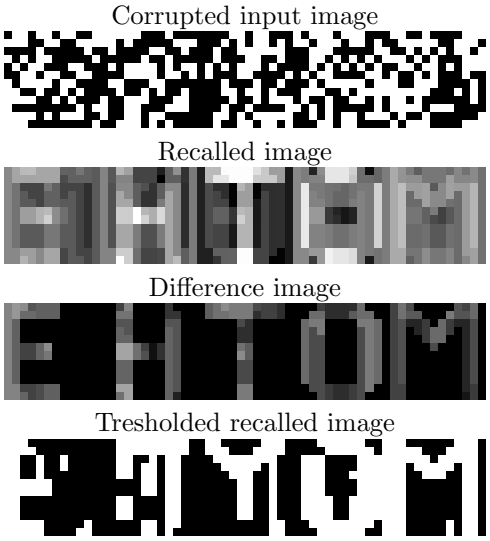


Figure 4: Recall with 40% corrupted data.

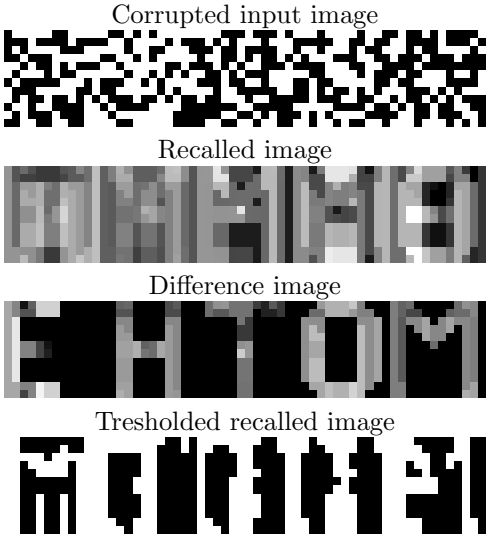


Figure 5: Recall with 50% corrupted data.

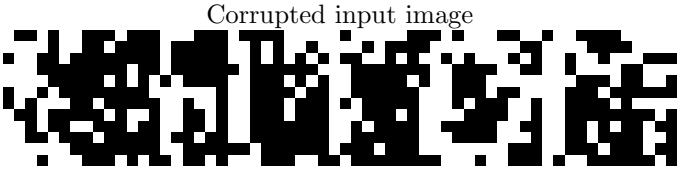
Corruption Fraction	0.2	0.3	0.4	0.5
Error(2-norm(Y-MX))	7.5	9.7	12.8	15.6
Fraction of mismatched pixels	0%	2.7%	12.4%	50.6%

* All the images is rescaled to [-1,1].
** Error is computed by avaraging several different runs.

c)

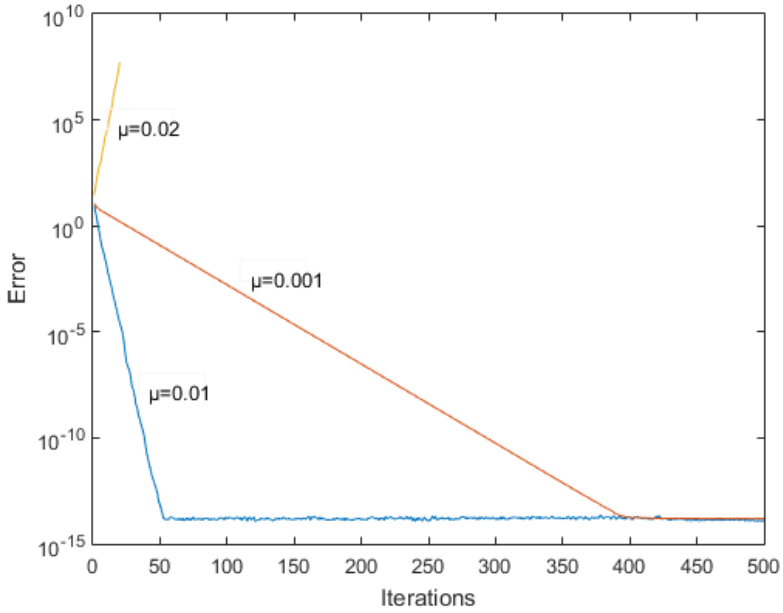
We used 20% corrupted data for training input and clean data for desired output. The training parameters are same as part (a).

Corrupted input image



Desired output image

The training curve is similar to part(a).



Then we test the recall accurancy of different input:

Difference fraction to training set	0.2	0.3	0.4	0.5	clean data*
Error(2-norm(Y-MX))	6.9	9.6	13.1	15.4	6.4
Fraction of mismatched pixels	0%	3.7%	11.9%	51.4%	0%

*Clean data also have 20% different pixels from training data.

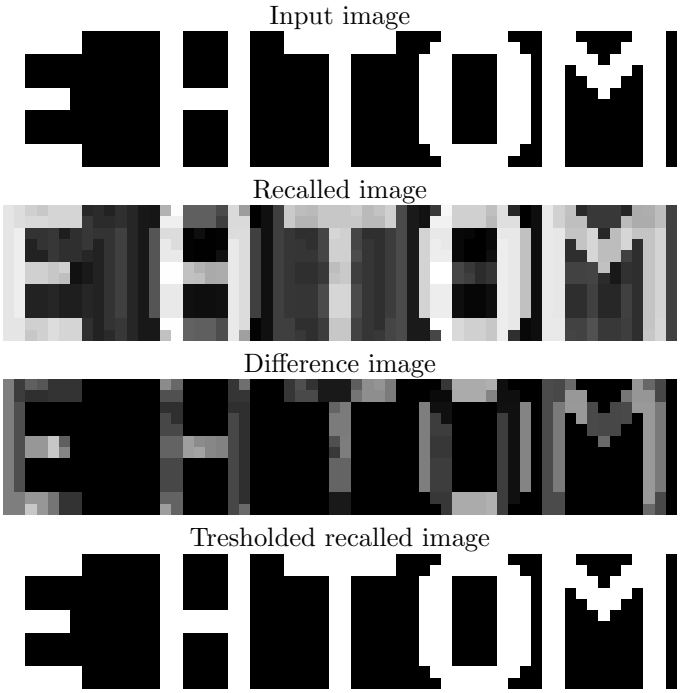


Figure 6: Recall with clean data.

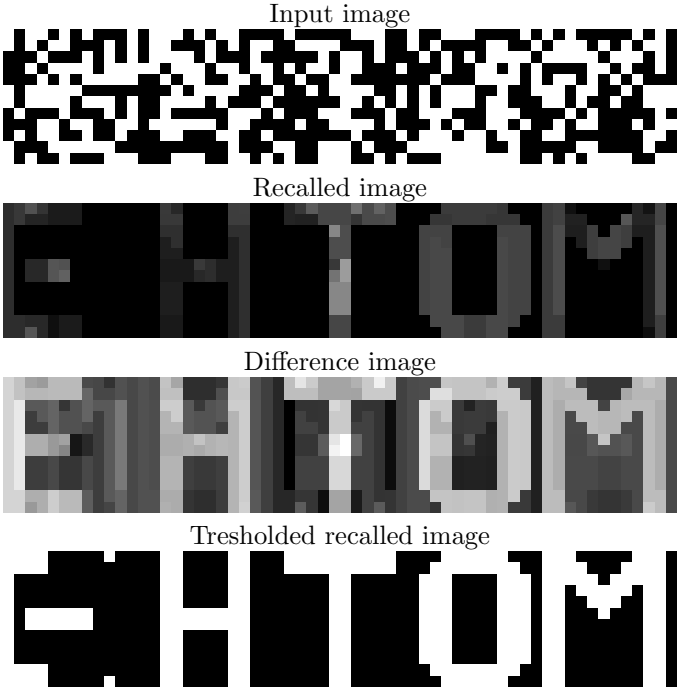


Figure 7: Recall with 30% different data.

From this experiment, we can see in this kind of memory, the recall accurancy only depends on the difference between the input and training set. It's independent of the similarty between the input and desired output.