ANN Lab 4

Hopfield Network

5.1 Convergence and attractors

Training patterns: **X_1: 0 0 1 0 1 0 0 1**

X_2: 0 0 0 0 0 1 0 0

X_3: 0 1 1 0 0 1 0 1

Input: All combinations of 8 bit binary string

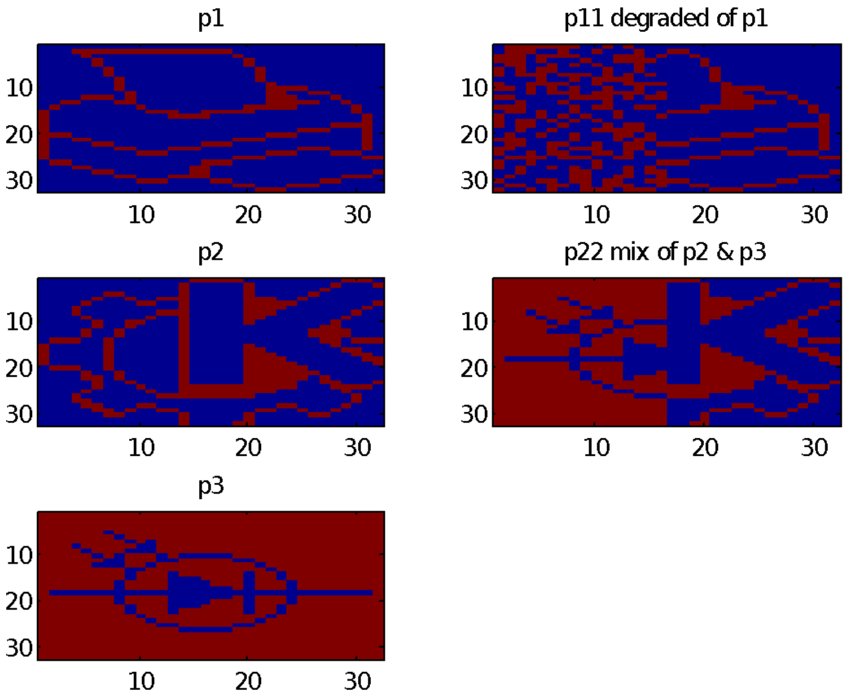
Results: Below are the attractors in this network (14 attractors)

0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**x2**
0	**0**	**1**	**0**	**0**	**1**	**0**	**1**	
0	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**x3**
1	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**inverted x3**
1	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**inverted x1**
1	**1**	**0**	**1**	**0**	**0**	**1**	**0**	
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
0	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**x1**
1	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**inverted x2**
1	**0**	**0**	**1**	**0**	**1**	**1**	**0**	
1	**1**	**0**	**1**	**1**	**0**	**1**	**0**	
0	**0**	**1**	**0**	**0**	**0**	**0**	**1**	
1	**0**	**1**	**1**	**1**	**0**	**1**	**1**	
1	**1**	**1**	**1**	**0**	**1**	**1**	**1**	

Max with 3 bit errors we can find attractors but with 4 bit errors (half of the bits) any attractor couldn’t be found and no convergence.

Also by changing X from [x1; x2; x3;] to [x1; x2; x3; x1; x1] we get 8 attractors (changing the input pattern, the number of attractors are changed).

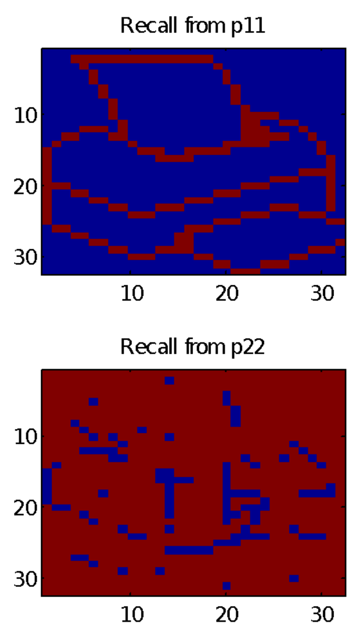
5.2 Sequential update



Synchronous update (Little network):

(In each iteration updating all units)

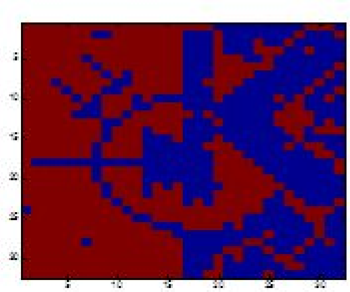
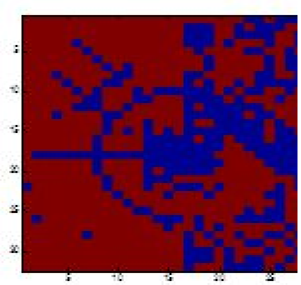
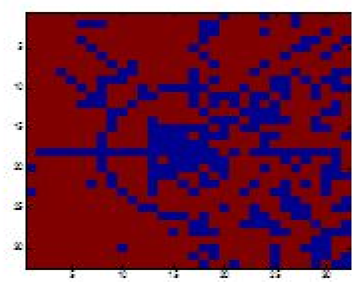
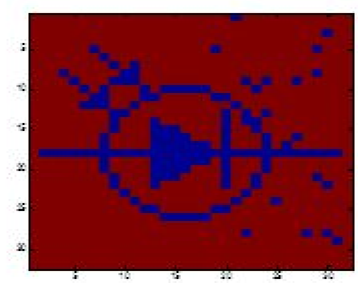
With 2 iterations p1 is recovered but in this method p22 is never completed and none of the p2 or p3 is not recovered.



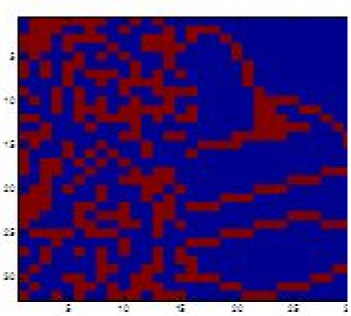
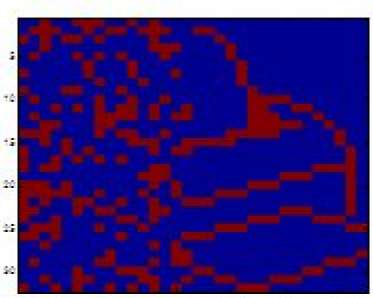
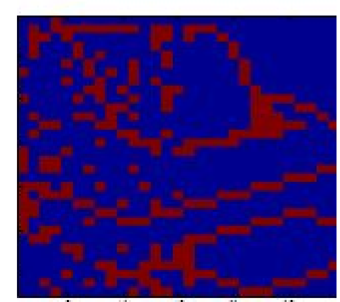
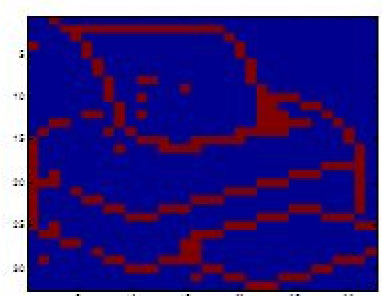
Asynchronous update: (Select one unit randomly each iteration)

This method needs more iteration to recover pictures.

Iteration=100 **Iteration=500** **Iteration=1000** **Iteration=3000**

Iteration=100 **Iteration=500** **Iteration=1000** **Iteration=3000**

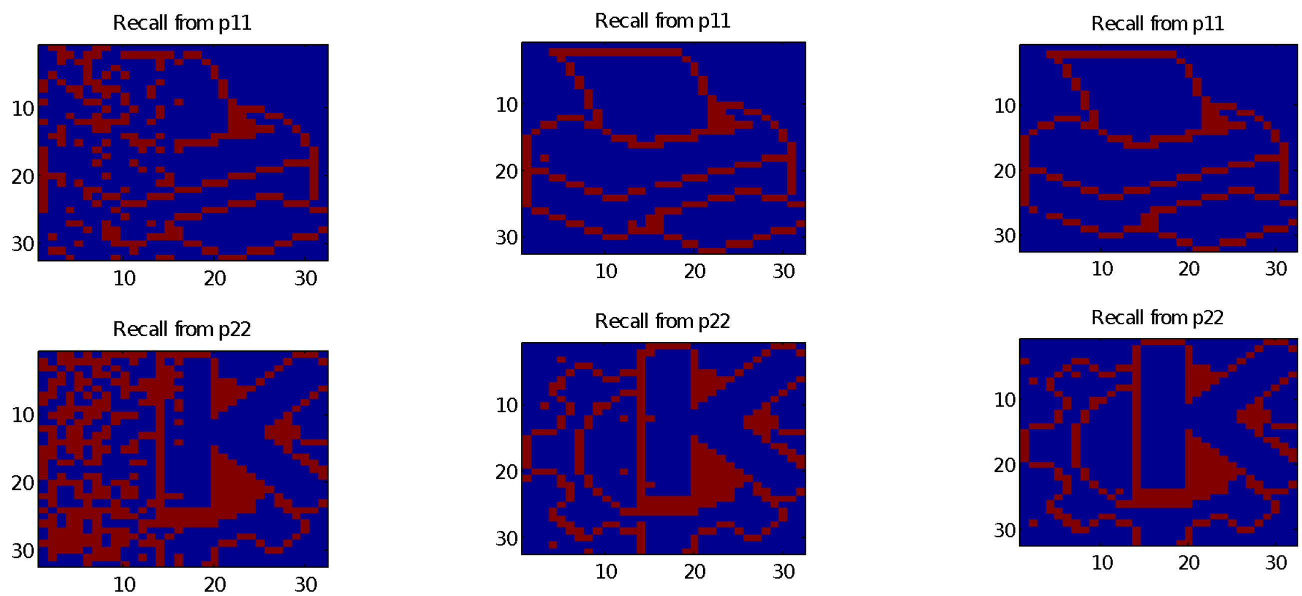





Asynchronous update (Sequential update):

(Each iteration updating 10% randomly selected units)

With 40 iterations the p1 and p2 are recovered. This method is very fast and need less iteration for recovering the pictures.

Iteration=10 Iteration=40 Iteration=50



5.3 Energy

• How do you express this calculation in Matlab? (Note: you do not need to use any loops!)

e = - diag(x * w * x')

• What is the energy at the different attractors?

Test on T5.1:	Test on picture:
e = -68 -68 -72	e = -1473936 (p1) -1398416 (p2) -1497344 (p3)

• What is the energy at the points of the distorted patterns?

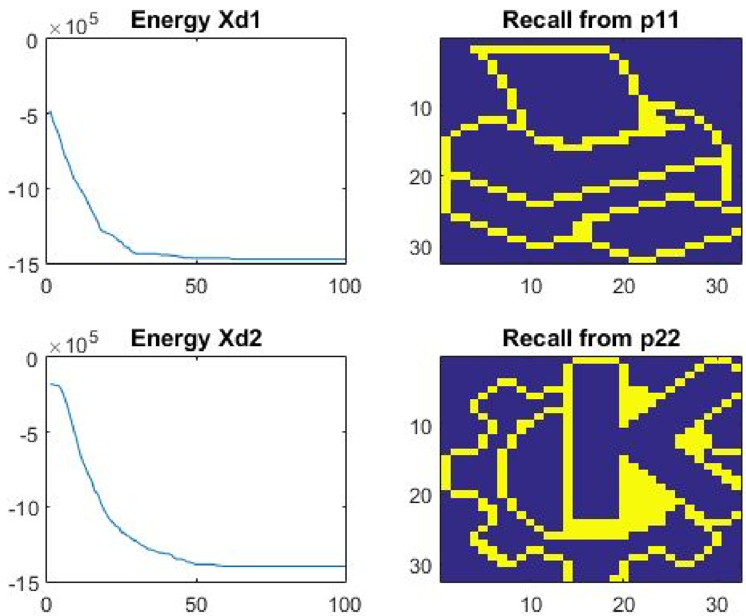
Test on T5.1:	Test on picture:
ed_ini = -40 -36 -24	ed_ini = -425964 (p11) -177664 (p12)

• Follow how the energy changes from iteration to iteration when you use the sequential update rule to approach an attractor.

Test on T5.1:

e_out = -40 -68 -68 -36 -56 -68 -24 -72 -72	When stuck in a not expected point: e_out = -40 -68 -68 -68 -36 -56 -68 -68 -24 -56 -68 -68 ……

Using sequential update: randomly update 10% units(100)

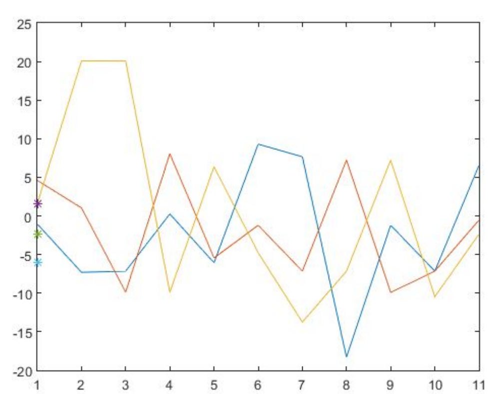
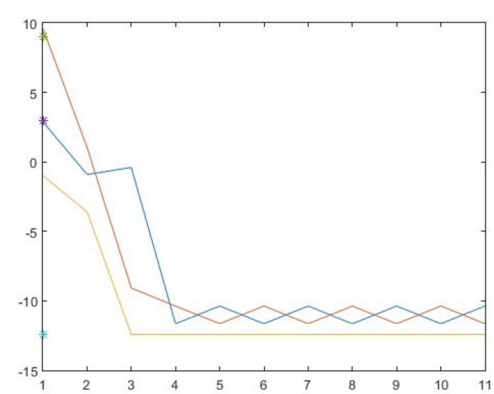


E1(end) = -1473936 = p1

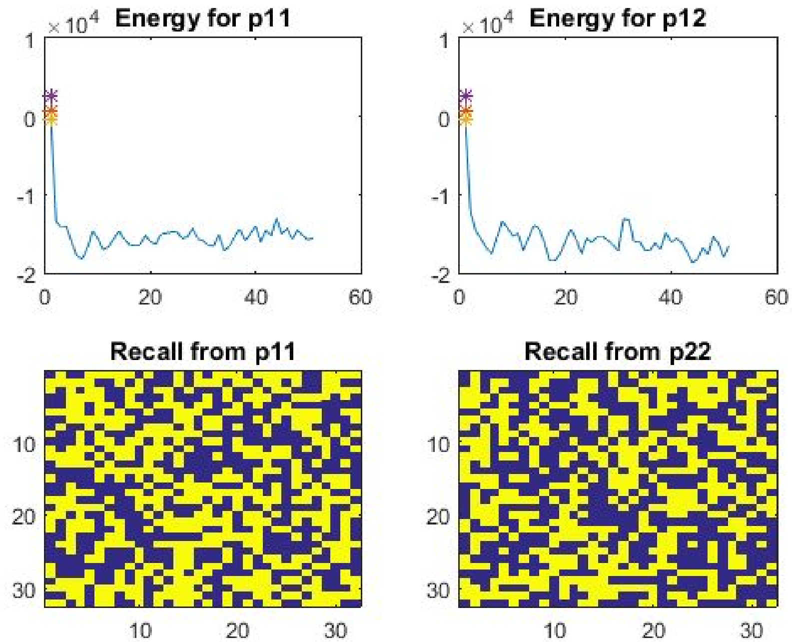
E2(end) = -1398416 = p2

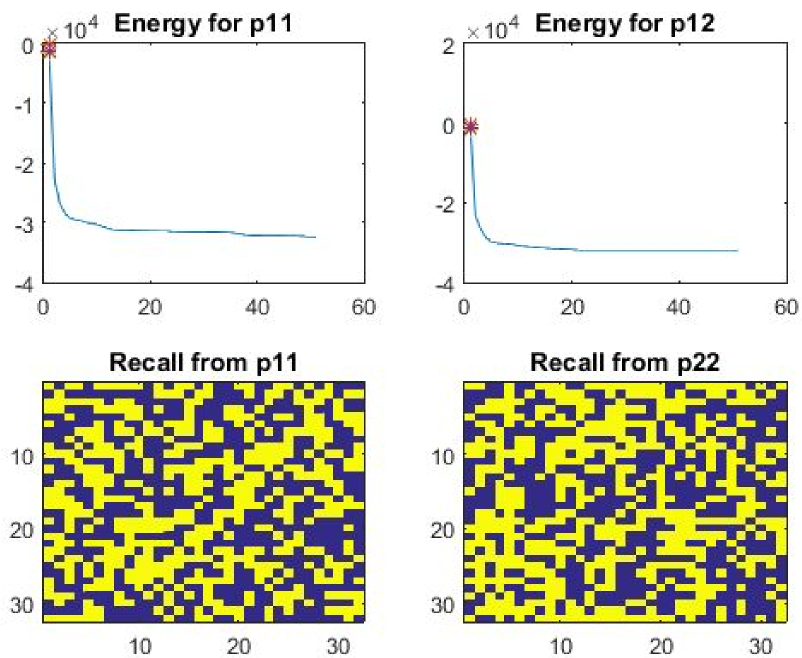
• Generate a weight matrix by setting the weights to normally distributed random numbers, and try iterating an arbitrary starting state. What happens?
• Make the weight matrix symmetric (e.g. by setting w=0.5*(w+w')). What happens now? Why?

Tset on T5.1

 asymmetric	 symmetric

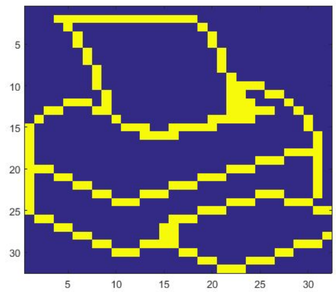
Test on picture:

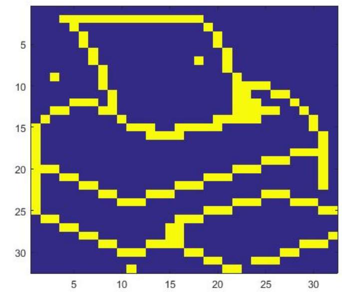




5.4 Distortion Resistance

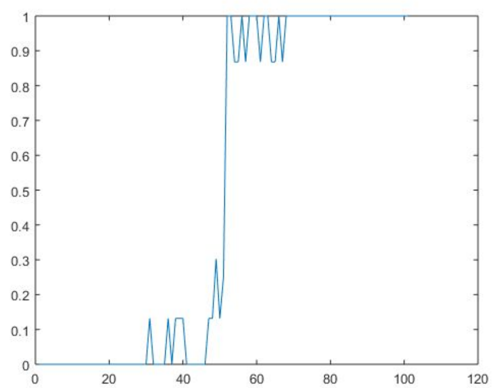
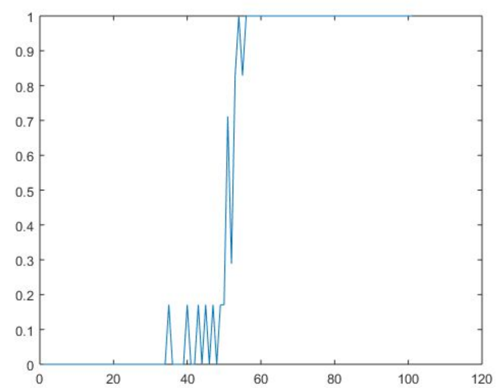
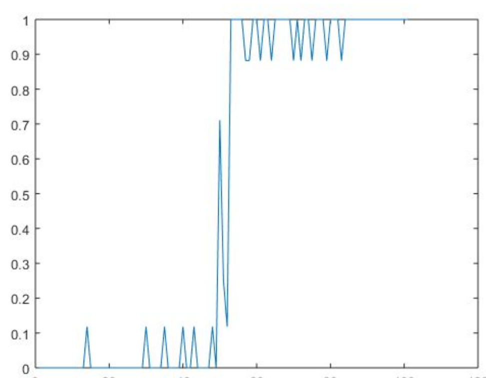
function of flip: flip(p1,5): randomly choose 5 pixels in the picture and turn it to the opposite value, i.e. turn 1 to 0 and 0 to 1.



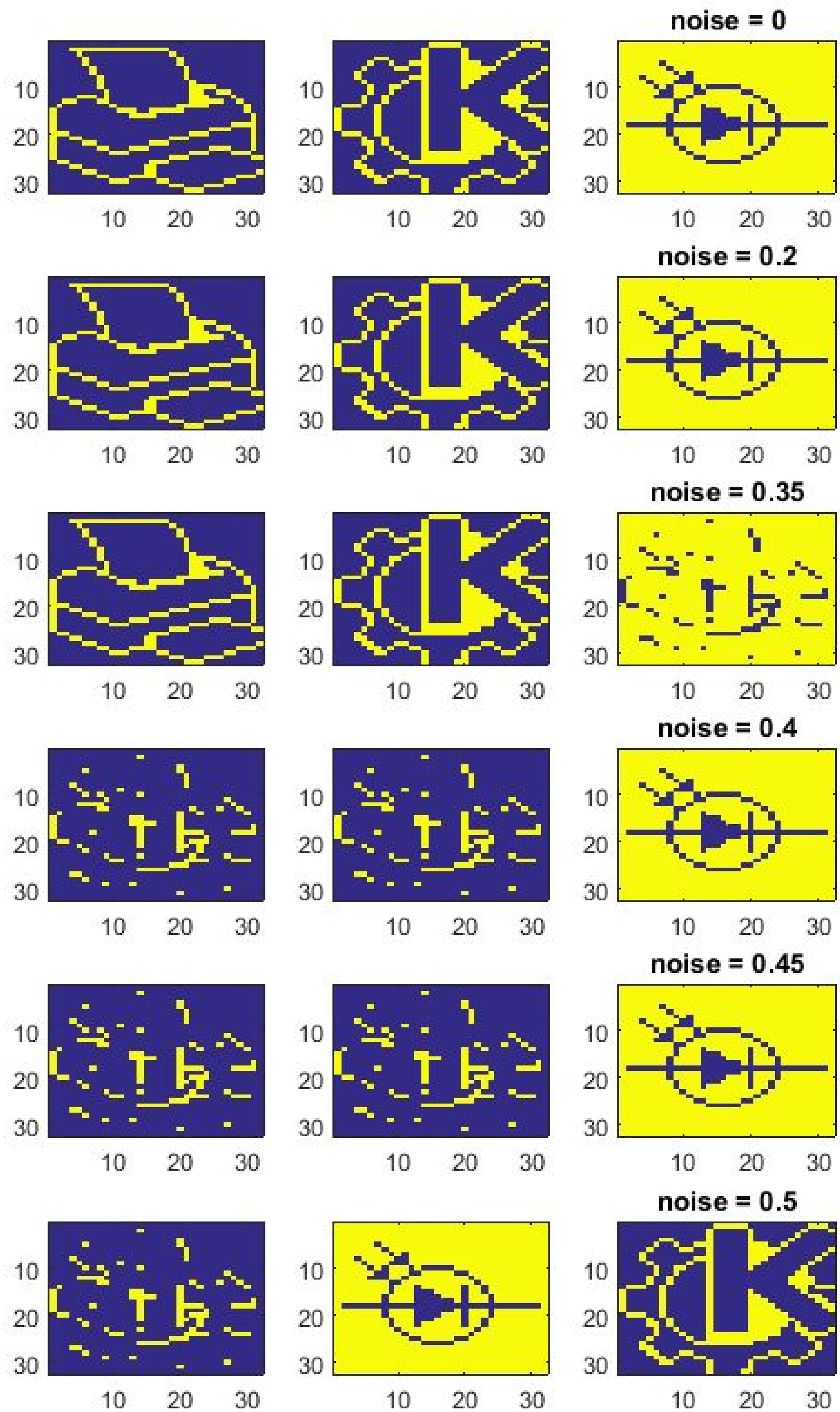


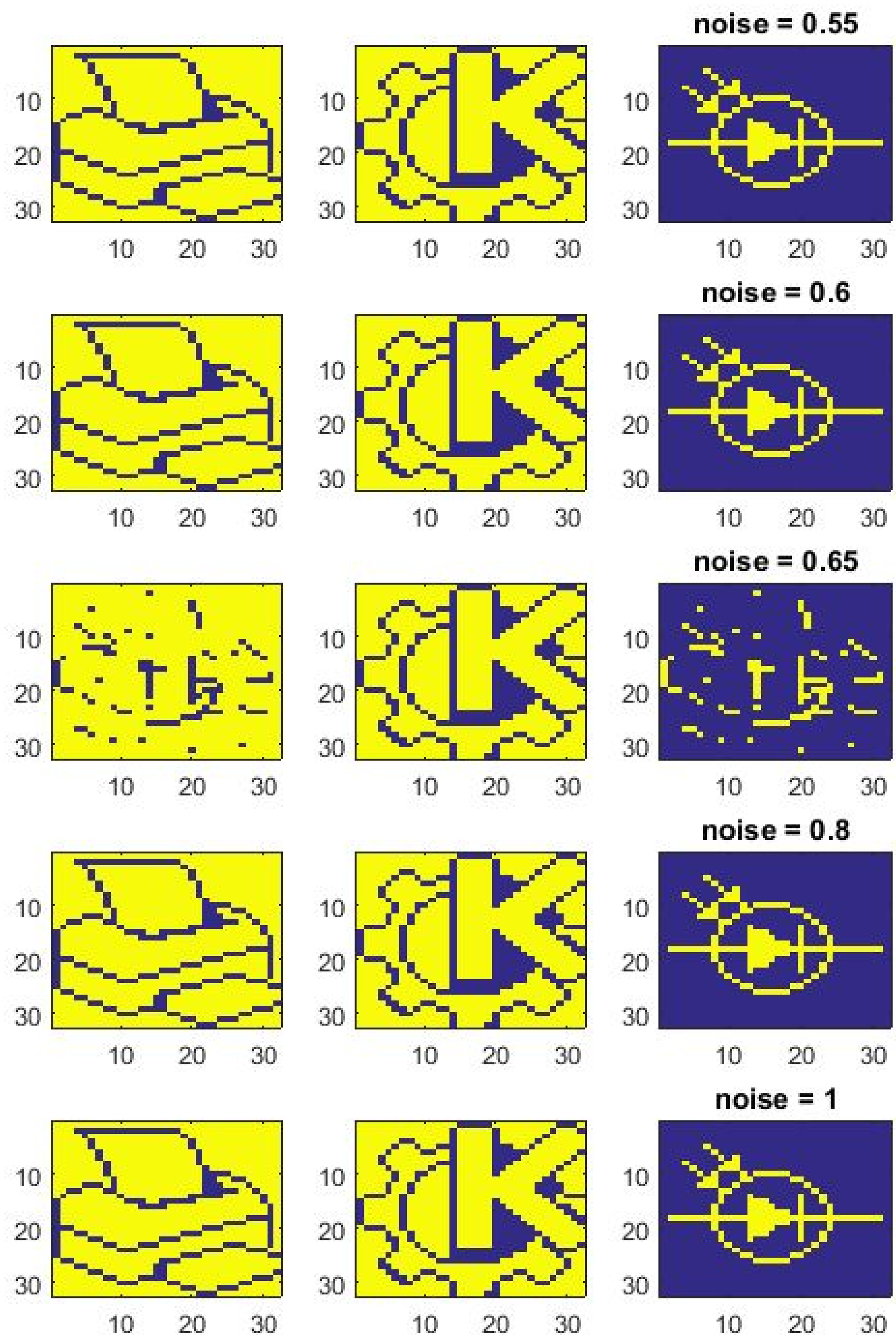
Noise test:

Trained with p1, p2, p3, and also noise is added to p1, p2 and p3

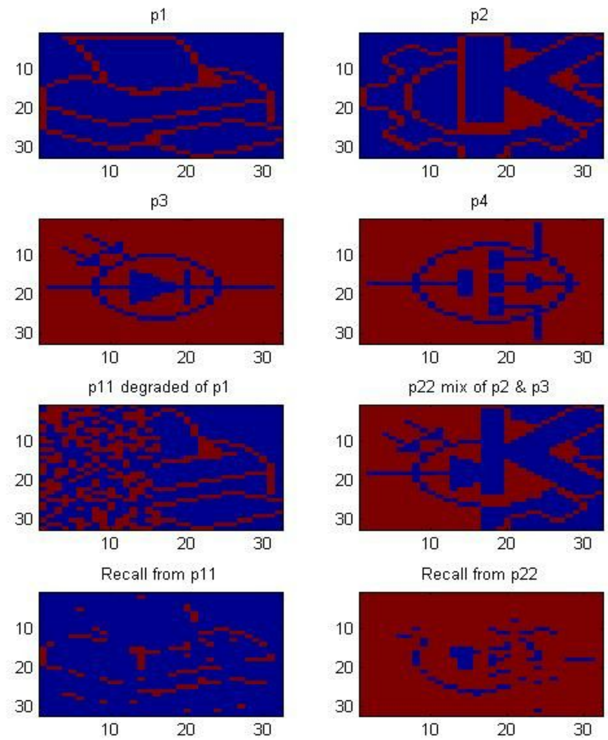
 picture 1	 picture 2
 picture 3	Conclusion: a good restoration when noise < 0.4, and when noise > 0.5, the picture tends to restore to an inverse of colour.

noise = [0, 0.2, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.8, 1];

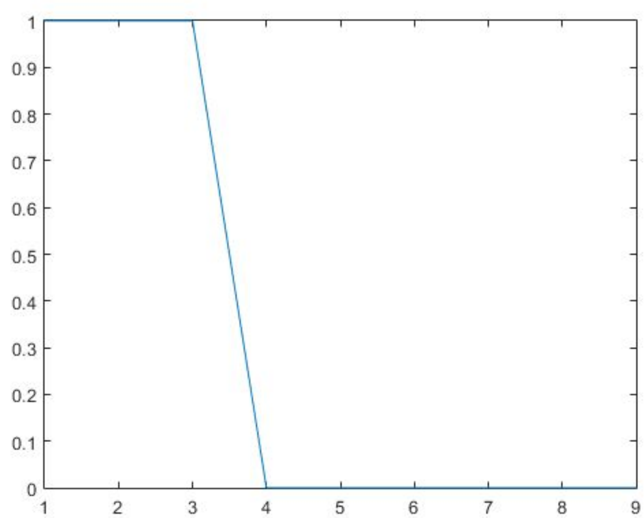


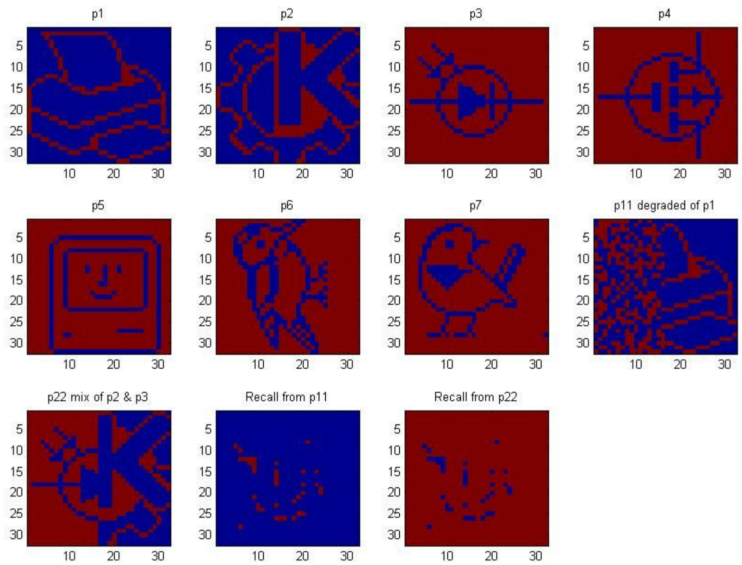

noise = 0.55
noise = 0.6
noise = 0.65
noise = 0.8
noise = 1

5.5 capacity

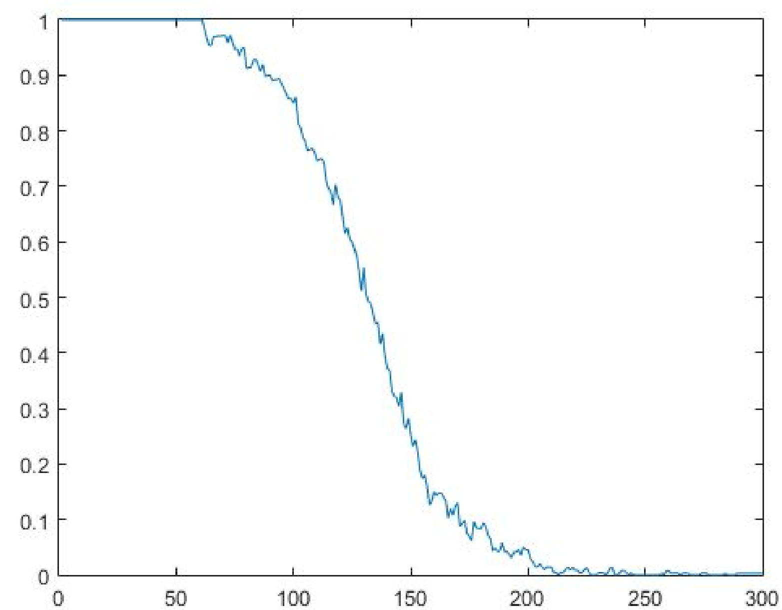


Result: by adding p4 into the weight matrix, no patterns could be safely stored after 10 iterations. The drop in performance is abrupt.



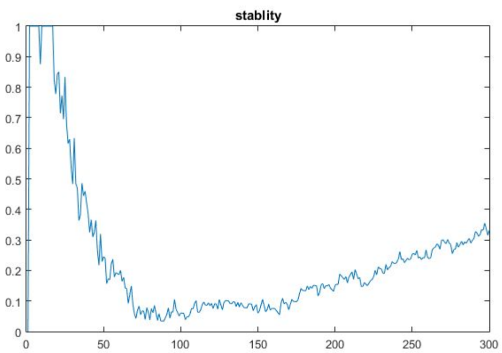
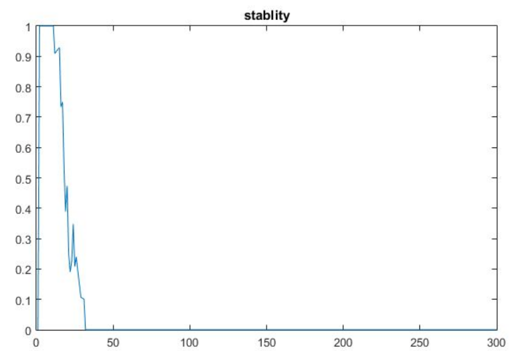


Result: by adding some random patterns the memory is partly stored, but not all.
Difference between random patterns and the pictures: random pictures have more entropy (more energy in general), hence increases the capacity.



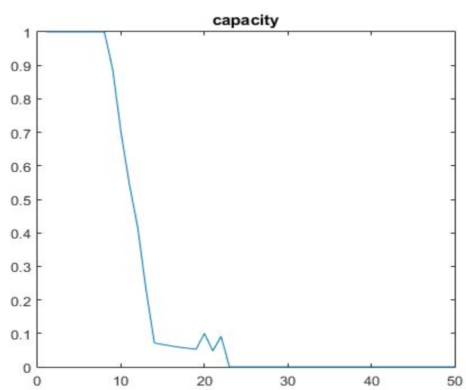
Increasing randomly created training patterns from 1 to 300, the capacity drop as the figure shows. 0.138*1024 = 141.312.

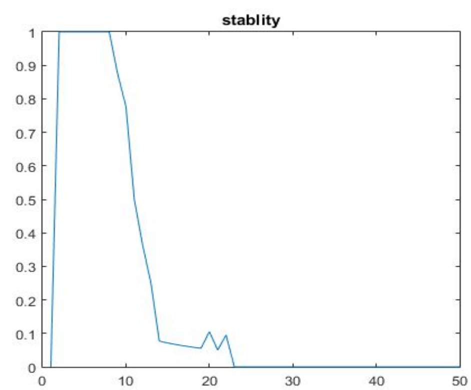
Stability:

 without setting w to diagnally zero	 set w diagnally to zero:

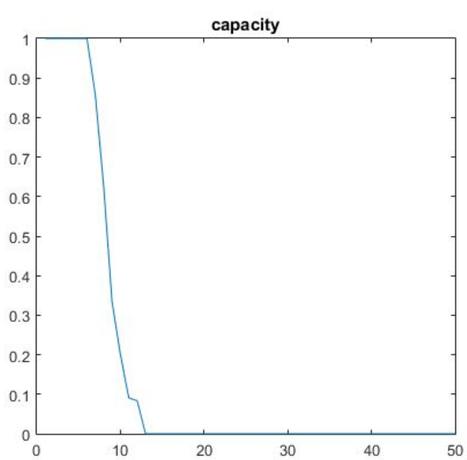
Bias:

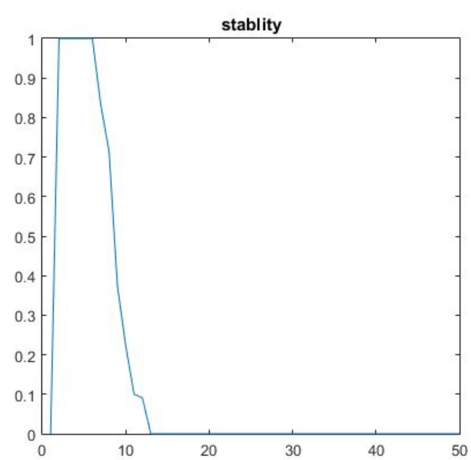
without setting w to diagnally zero





set w diagnally to zero:





5.6 sparse patterns

Set rho = 0.1 and theta = 0.1, the capacity is increased to around 50.
Set rho = 0.05 and theta = 0.1, the capacity is increased to around 110.