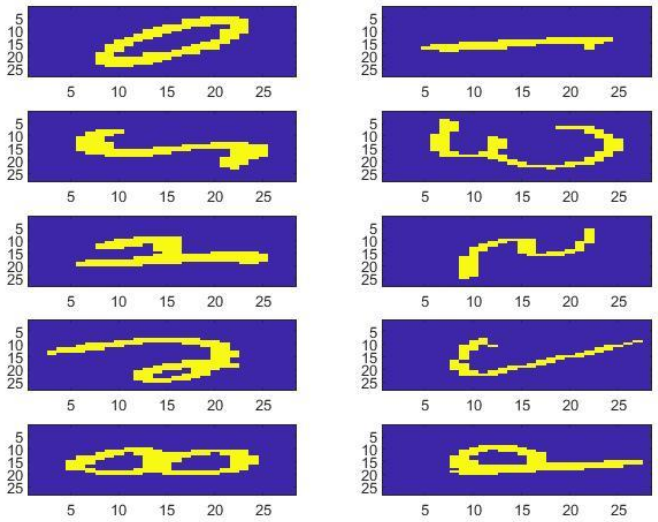
Part A

1a.

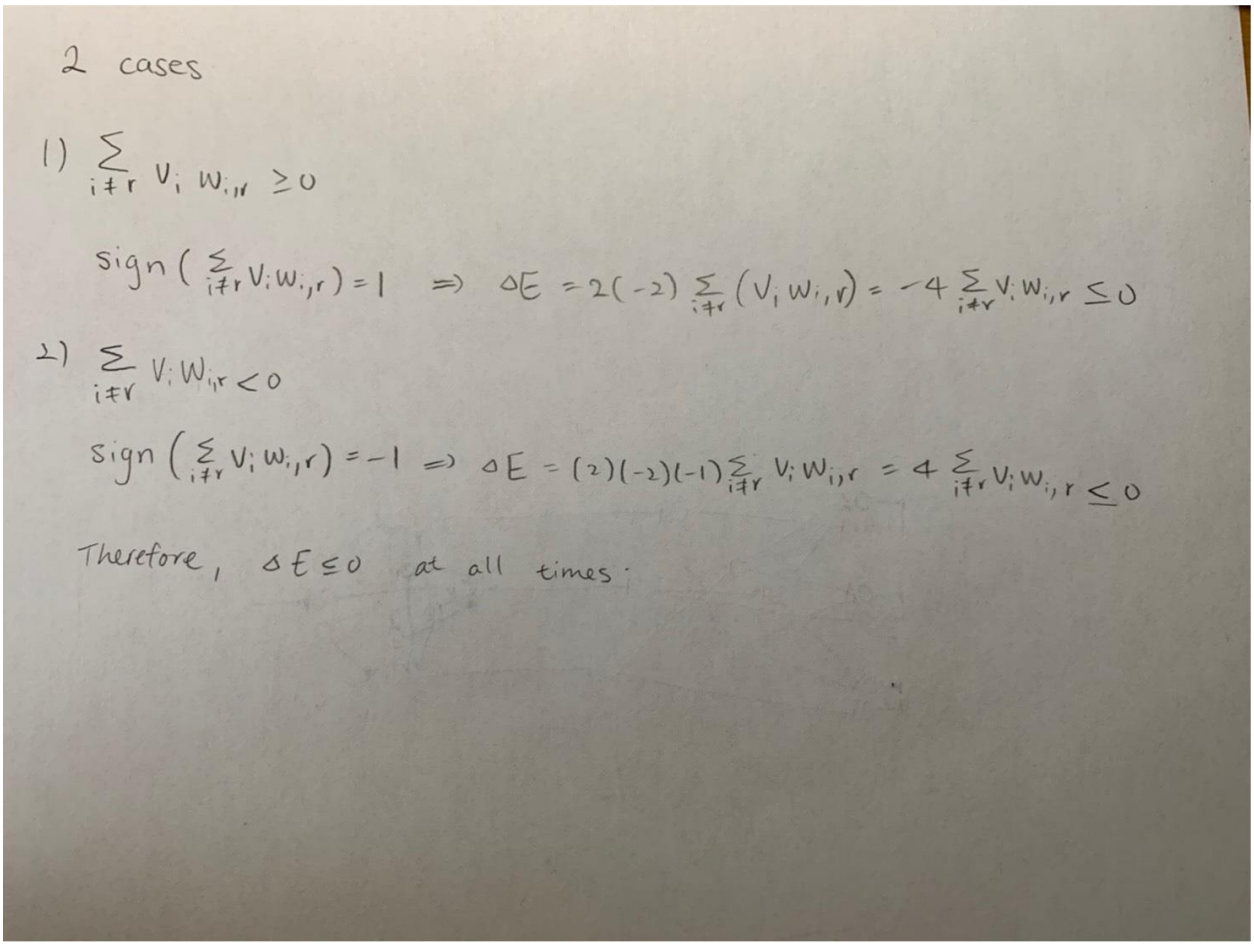


1b. The outer product rule is multiplying each element of the weight with each element of the signals. It makes sense to encode image this way to associate the weight and the signal matrix so that the weight matrix can potentially be used to reconstruct images or retrieve memories.

```
for i=1:n
    s = data(i, :);
    w = w + s' * s;
end

w = w/n;
```

2a.

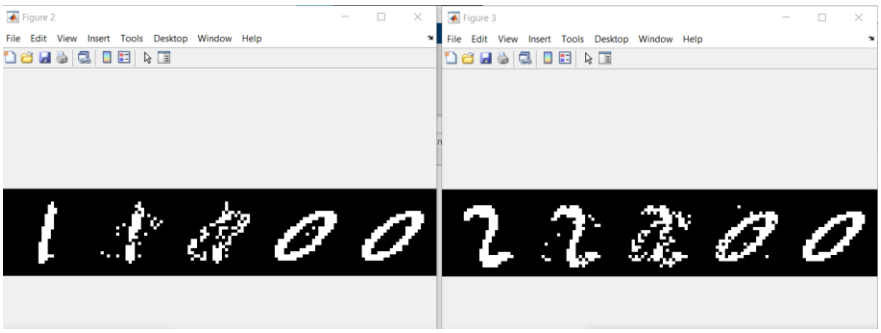


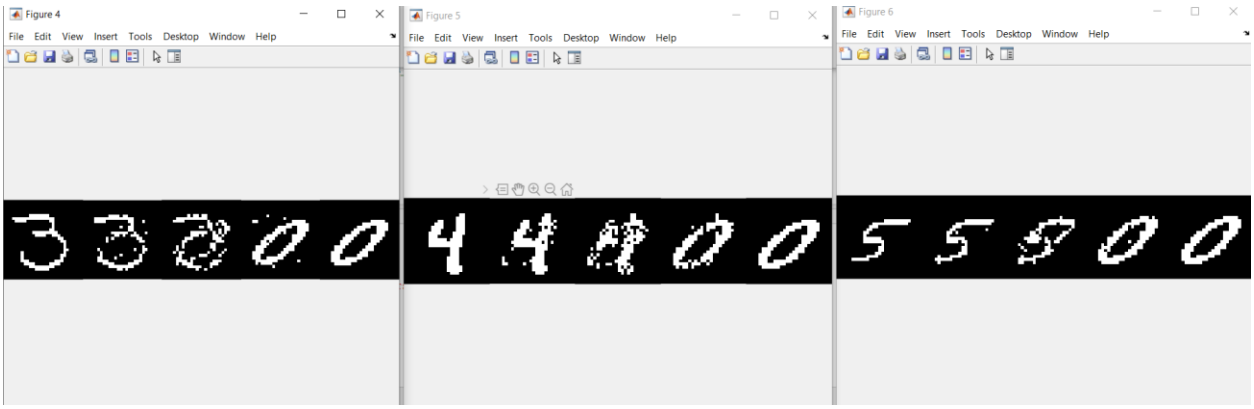
The network will always settle down to global minimum since it keeps decreasing.

```
data(d,node) = sign(sum(data(d, :) * w(:, node)));
```

2b.

3a. We retrieve the digits regardless of the initial response configuration. It always goes to the same attractors. The distribution of final configurations seems to be the same across different initial configurations as seen in the visualization below. The network is trained with digit 0, and the final configuration always end up a 0.





All the initial conditions settled to the same final configuration given the digits. Retrieval of memory looks to be equally successful for all digits. I did not observe a visible difference in success across the digits in all 50 retrievals. This makes sense that the weight matrix is able to associate and form connections of signals for the network, there shouldn't be a big performance difference given similar starting conditions. The rest of the repetitions are in the code folder.

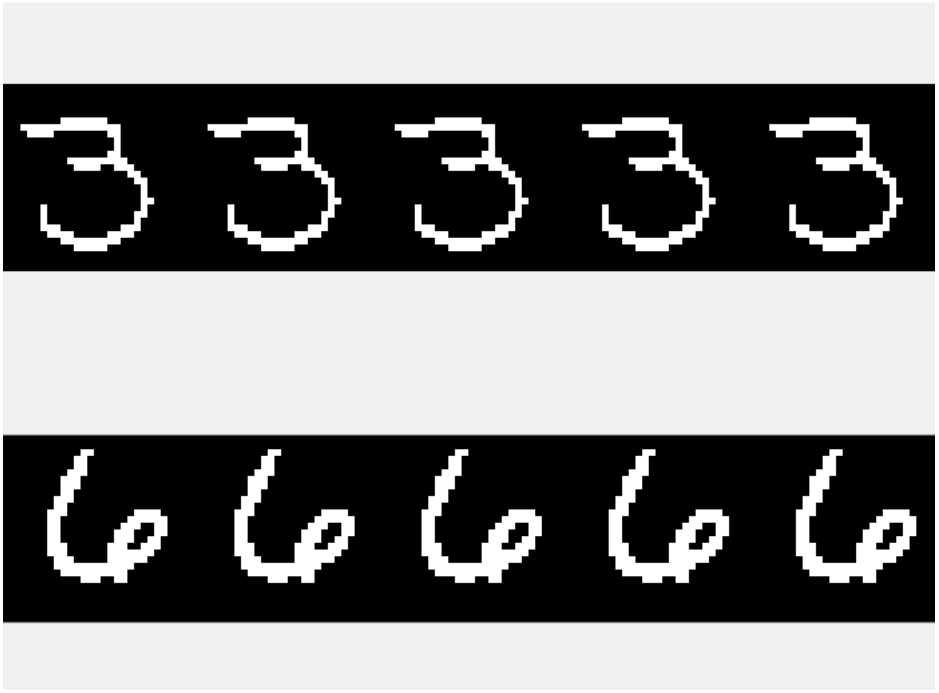
3b.

Random initialization:

After testing, the network only seems to be able to encode and retrieve 2 patterns.

Self-encoding:

After testing, the network only seems to be able to encode and retrieve 2 patterns as well.

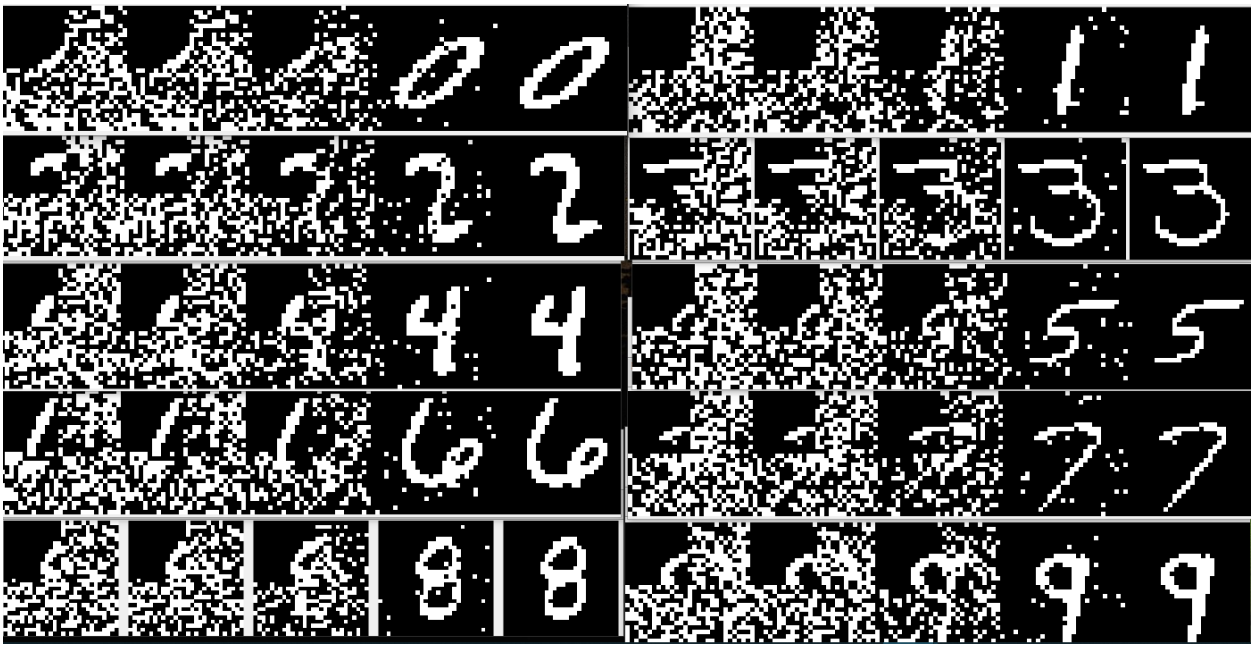


Noise corruption level: 10%, 25%, 50%

For each noise level, the network can remove the noise for each digit and there isn't an observable difference between noise levels.

The number of patterns we can reliably retrieve is 2 across all different trials and conditions. This is much lower than the theoretical possible capacity of 5%-13% of the nodes.

4.



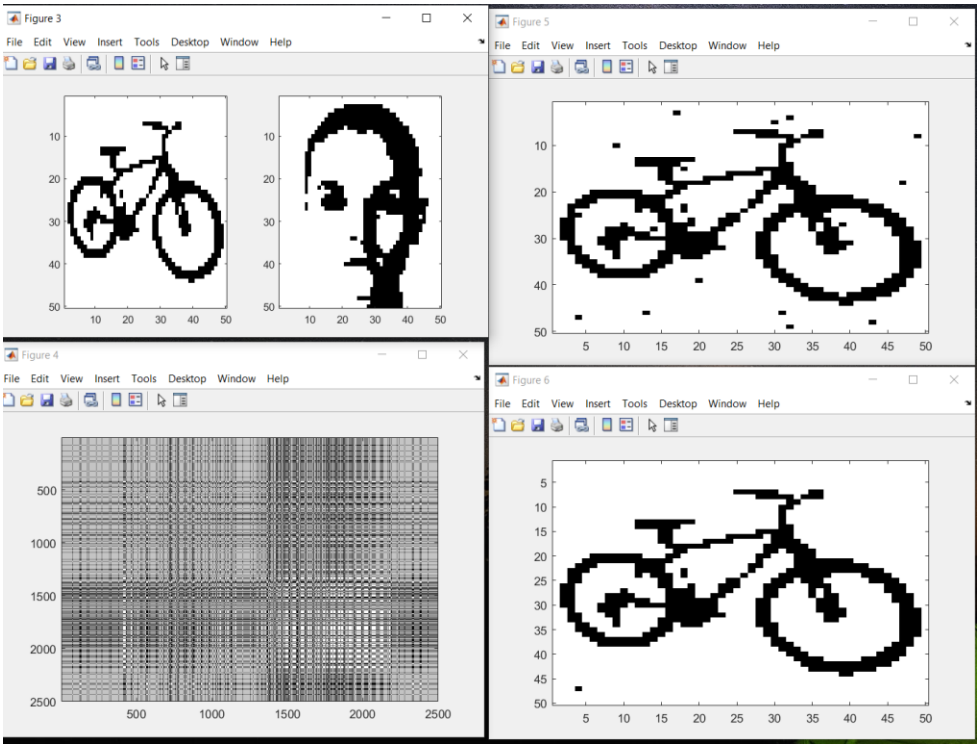
The network did a good job at reconstructing the original image from the partial image. Even with most of the original image occluded, the network can reliably retrieve the memory.

Part B

1.

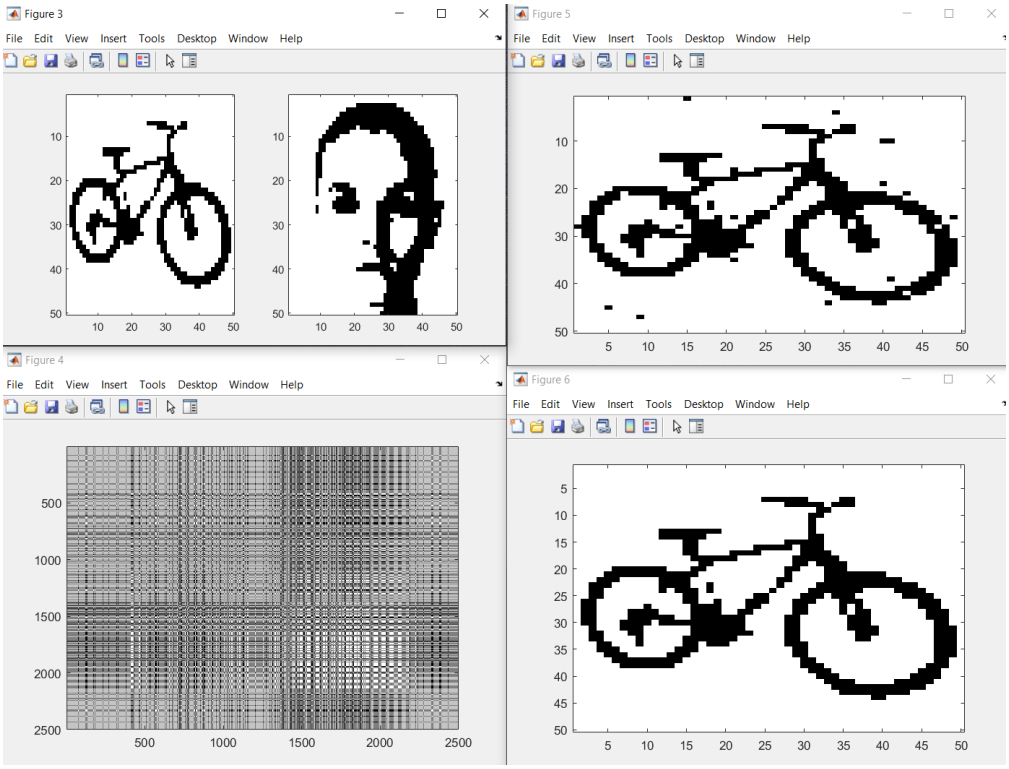
Random:

```
  %  STUDENT TASK:  update each random selected i-th node using the Hopfield dynamics.
tmp(i, :) = sign(Ui);
```

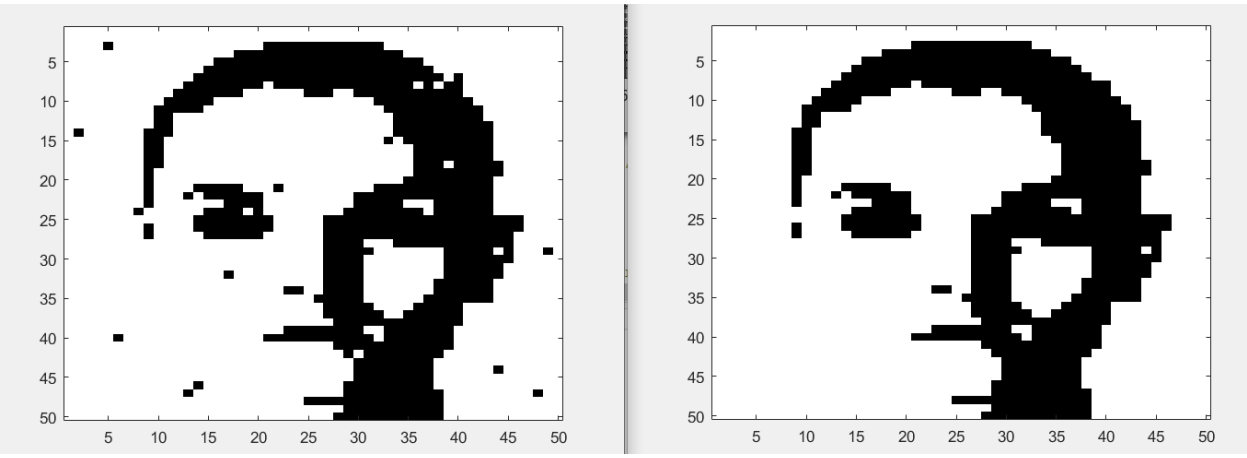


All:

```
% STUDENT TASK  compute U, i.e. values of all the nodes U in the network in one simultaneously.
U = T * tmp;
```



When asking the network to encode image 1, the network can reliably retrieve the original image. The two sets of images are as following: original images (upper left), corrupted image (upper right), weight visualization (bottom left), and reconstructed image (bottom right).



The network can also retrieve the second image encoded, however, cannot go any higher than two images in our testing.

2.

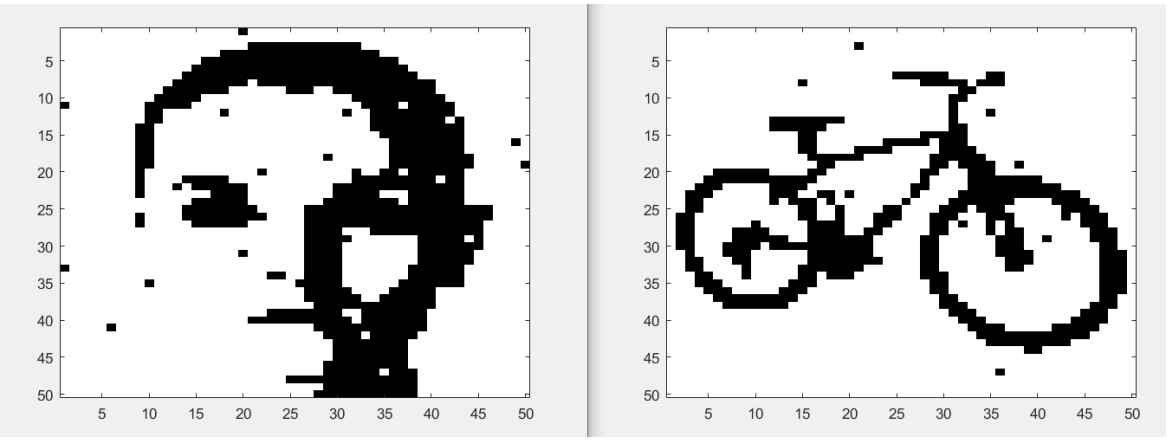
The first implementation performs better with a better retrieved memory. The number of patterns each network can learn is 1 in our testing, which is demonstrated below that when asked to retrieve the second image (face), the network retrieves the first anyways (bike).

The first implements: **Oja rule:** $\Delta w_{ij} = r_i r_j - (r_i)^2 w_{ij}$

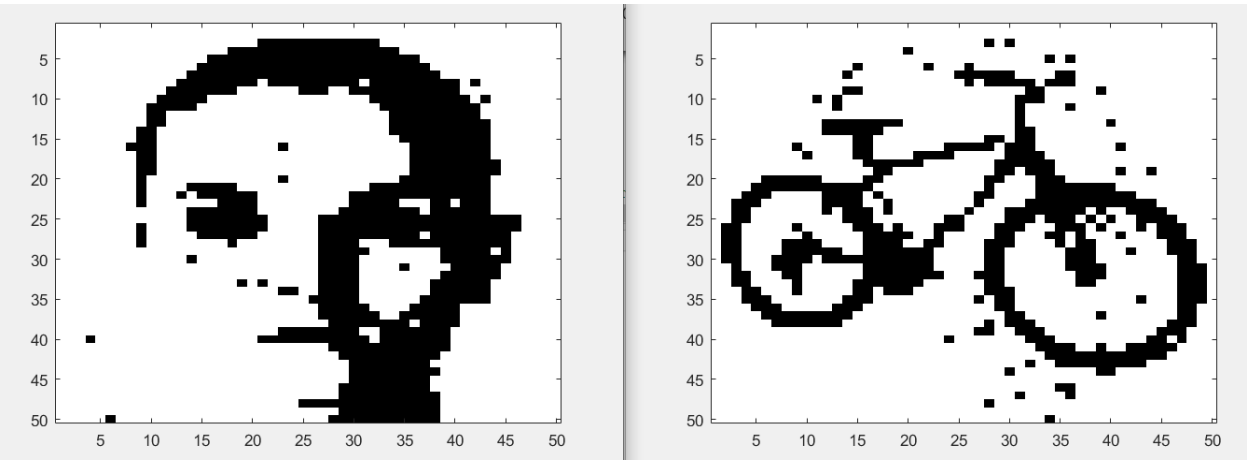
The second implements: $$\Delta \mathbf{w} = \eta\, y\, (\mathbf{x} - y\, \mathbf{w})$$

It seems that the second implements it more literally, giving the same form. The first implementation is less literal.

First



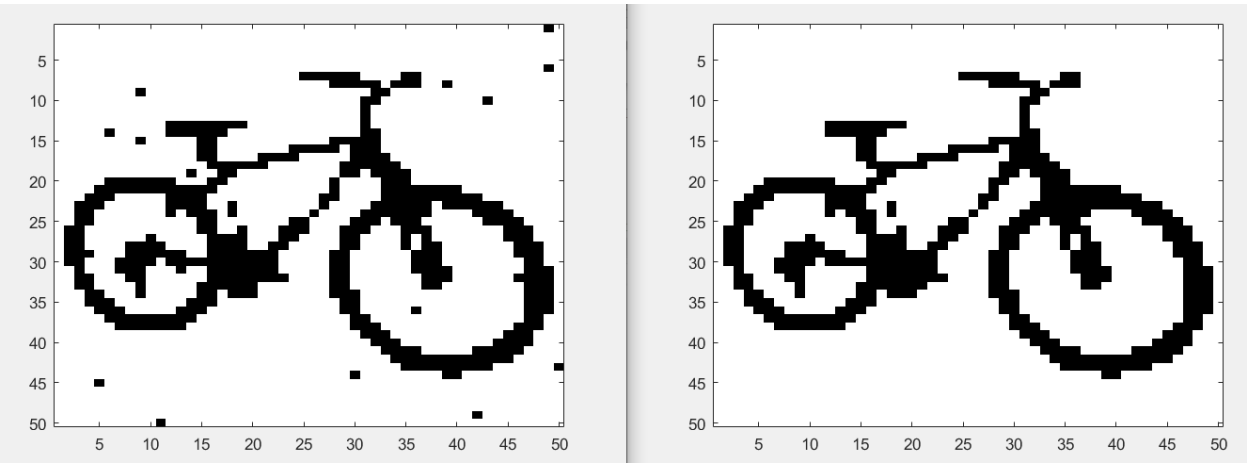
Second



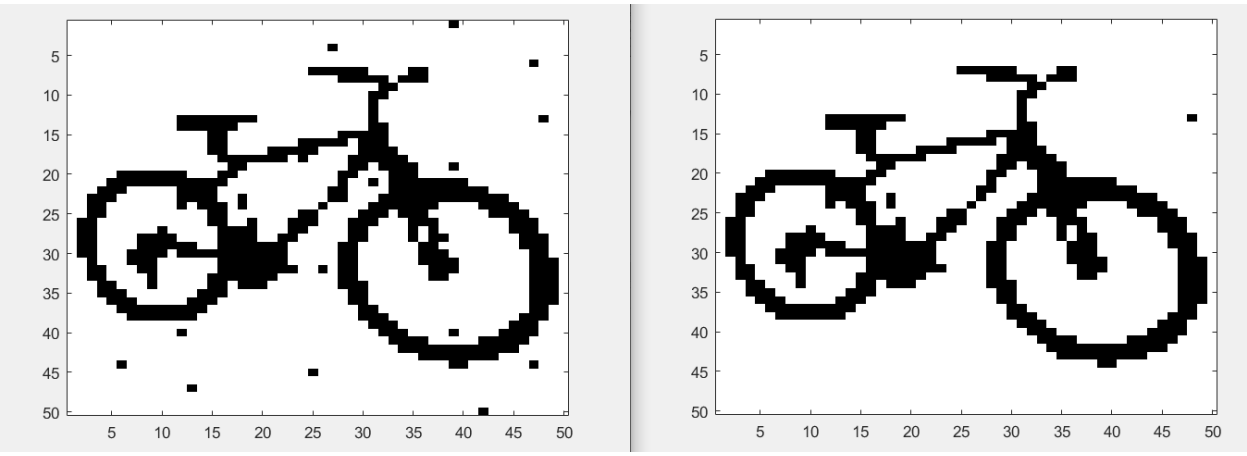
3.

I chose the first rule. Learning rate does not seem to have a huge effect as seen in the figures below. The network is able to retrieve the image correctly with noise almost entirely removed regardless of the rate of learning.

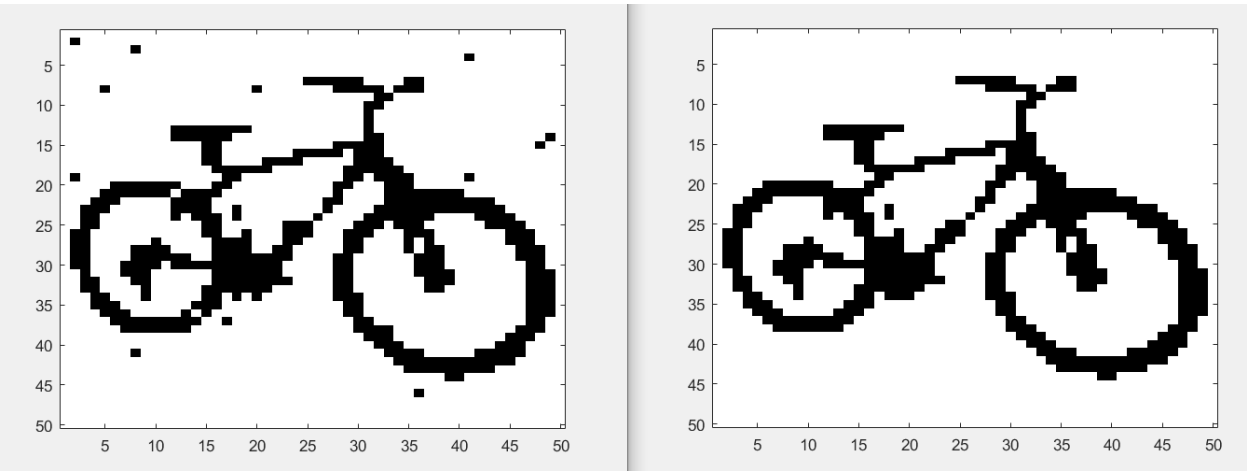
0.3



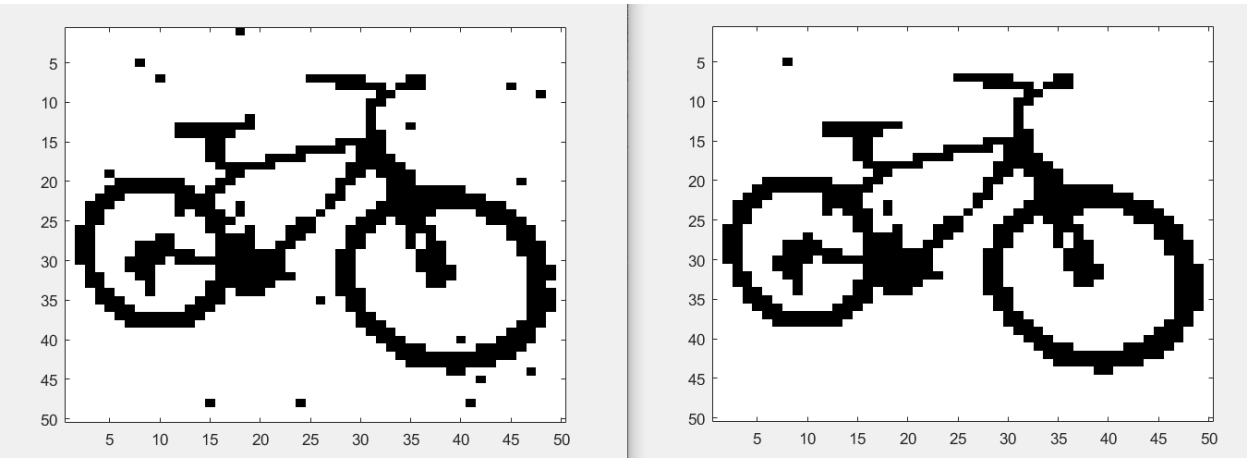
0.5


5 10 15 20 25 30 35 40 45 50

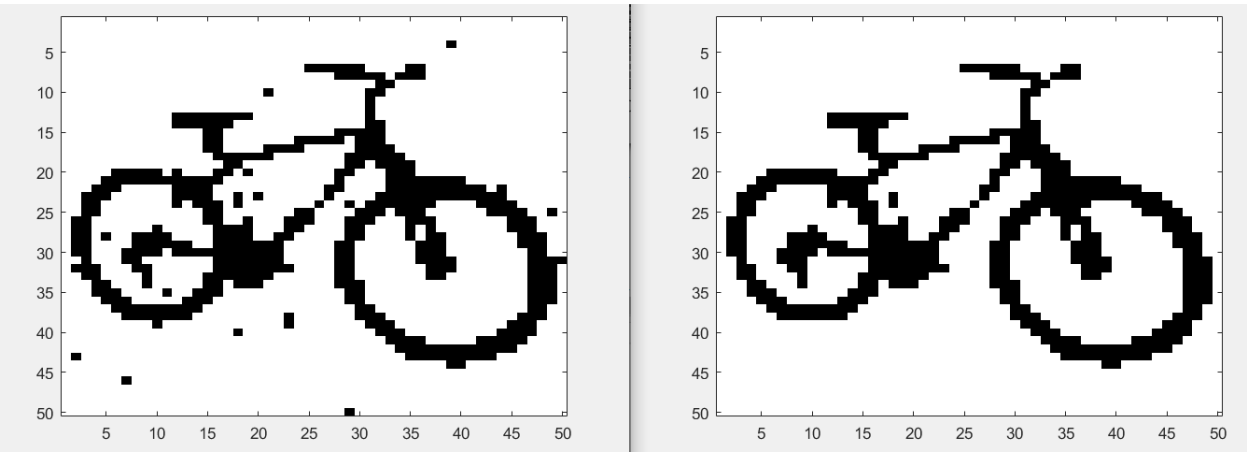
0.7


5 10 15 20 25 30 35 40 45 50

0.9

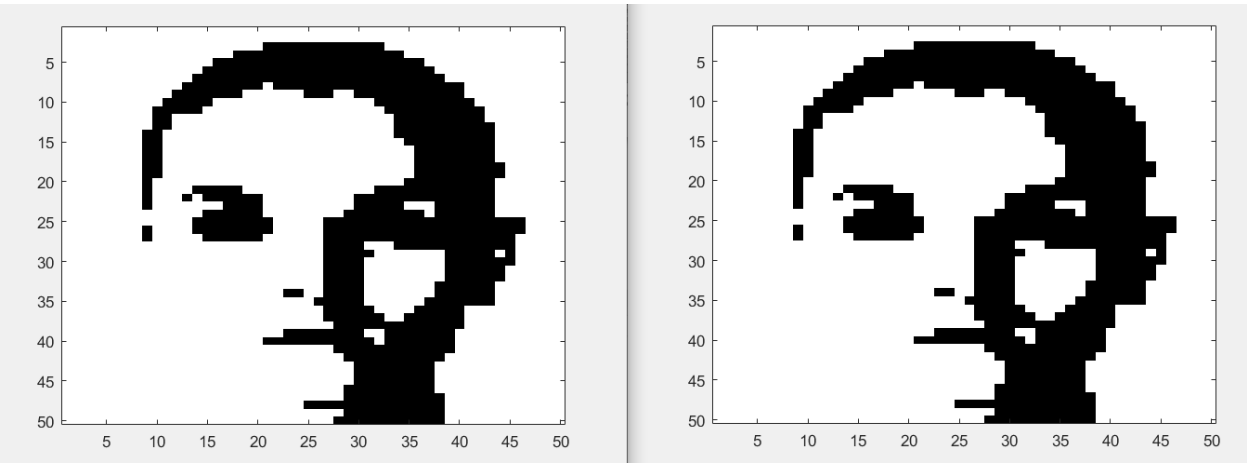

5 10 15 20 25 30 35 40 45 50

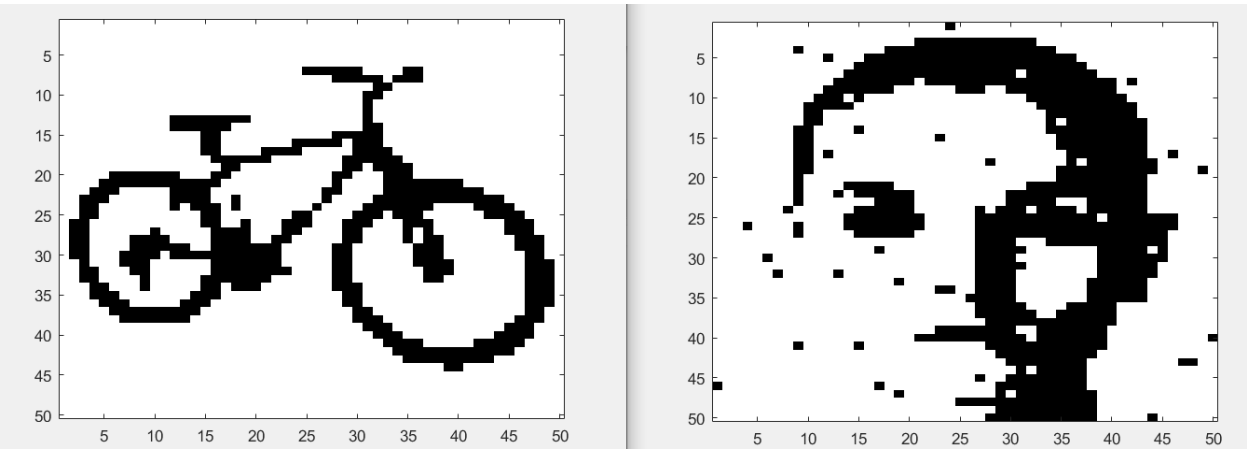
1.1



4.

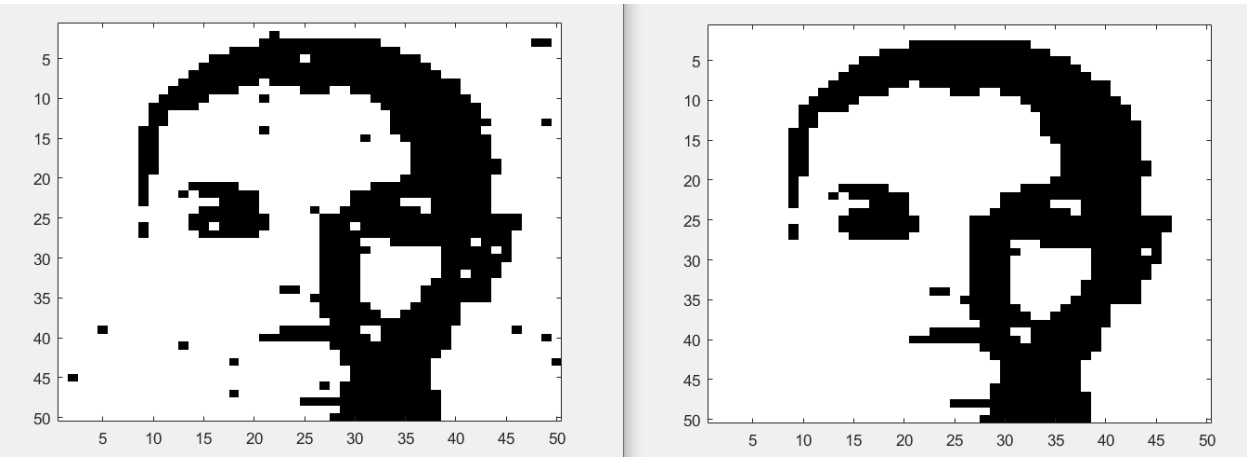
The first rule of oja rule is used. I can only retrieve one image as seen in below, when asked to retrieve the second image, the first is retrieved with some more noise. This number is fairly reliable since we have seen a similar pattern in previous parts. However, just from the work done in this section, it is not the most reliable because only several patterns were tested and I was not able to test every single possible pattern.





5.

Corrupt



At 25% corruption, the network is able to retrieve the image well. However, we see the same pattern where only the first image is retrieved.

Similarly, the network can retrieve the image well at around 50% occluded. However, it can only retrieve one image.