SOM & Image Data Compression

Yashad Samant
Master's of Electrical & Computer Engineering
Colorado State University

Abstract

This paper references to the problem of sending a large image into a server without any quantization. We can only imagine the extent of the data if the image is not compressed and sent directly. Thus, to avoid such massive transfer of data, we have developed a vector quantization method using self-organizing maps. This paper also discusses various parameters used to improve the performance of the unsupervised learning method. We have also reconstructed the quantized image and observed the SNR_input and SNR_output to calculate the distortion rate and also the compression rate. It also discussed the relation between bits per pixel and SNR and resulting advantages and disadvantages of the method on the basis of the output, validation parameters etc.

I. Introduction

The Self-Organizing Map (SOM) consists of a regular, usually two-dimensional grid, onto which a distribution of input items is projected nonlinearly. The mapping tends to preserve the topologic-metric relations between input items. The projection is made by a matching process. With each grid unit, a generalized model is thought to be associated. For each input item, the closest model in some metric is identified. The collection of models is optimized to approximate all inputs. It may be well-known that SOMs can be constructed using any generalized distance function defined between the input items and that the updates can be made in batches.
Thus, using this general approach of SOM, we can implement vector quantization where a 4x4 block of 512x512 image can be mapped to a single element in vector codebook.

II. Data

DATASET: In this problem we were given three images each of size 512x512. We were asked to use 4x4 blocks of images as inputs. Thus, to achieve the same we used mat2cell function in MATLAB which converted the 512x512 matrix to 1x16384 which had 4x4 blocks in each of the cell. We flattened the cell to get the dimension of the input to be 16x16384. We have to be careful while pre-processing image, as images are not scale and transform invariant.
The sample images are given below:

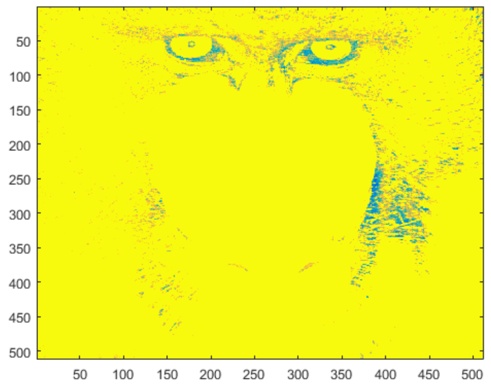

Fig 1: Baboon image

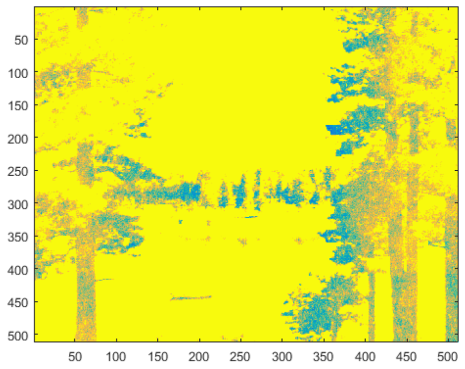

Fig 2: Boat

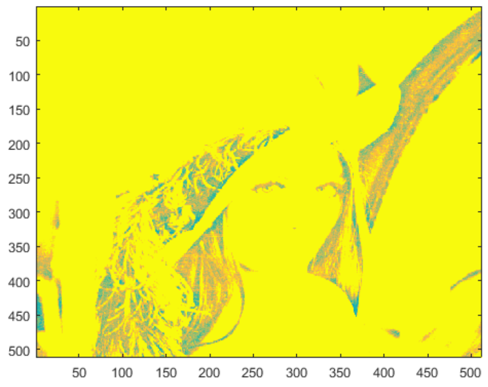

Fig 3: Lena

In this problem, we have considered Lena as the train image, baboon as the test image and boat is used for validation. We can shuffle without any constraints.

III. ALGORITHMS

Self-Organizing Maps:

Self-organizing maps (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduce the dimensions of data through the use of self-organizing neural networks. The problem that data visualization attempts to solve is that humans simply cannot visualize high dimensional data as is so techniques are created to help us understand this high dimensional data. Two other techniques of reducing the dimensions of data that has been presented in this course has been N-Land and Multi-dimensional Scaling. The way SOMs go about reducing dimensions is by producing a map of usually 1 or 2 dimensions which plot the similarities of the data by grouping similar data items together. So, SOMs accomplish two things, they reduce dimensions and display similarities.

Algorithm:

The way that SOMs go about organizing themselves is by competing for representation of the samples. Neurons are also allowed to change themselves by learning to become more like samples in hopes of winning the next competition. It is this selection and learning process that makes the weights organize themselves into a map representing similarities.
So, with these two components (the sample and weight vectors), how can one order the weight vectors in such a way that they will represent the similarities of the sample vectors? This is accomplished by using the very simple algorithm shown here.

- Initialize Map
- For t from 0 to 1
    - Randomly select a sample
    - Get best matching unit
    - Scale neighbors
    - Increase t a small amount
- End for

The first step in constructing a SOM is to initialize the weight vectors. From there you select a sample vector randomly and search the map of weight vectors to find which weight best represents that sample. Since each weight vector has a location, it also has neighboring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. In addition to this reward, the neighbors of that weight are also rewarded by being able to become more like the chosen sample vector. From this step we increase *t* some small amount because the number of neighbors and how much each weight can learn decreases over time. This whole process is then repeated a large number of times, usually more than 1000 times.
In the case of colors, the program would first select a color from the array of samples such as green, then search the weights for the location containing the greenest color. From there, the colors surrounding that weight are then made more green. Then another color is chosen, such as red, and the process continues.

IV. Experimental Results

a. SOM (16 neurons):

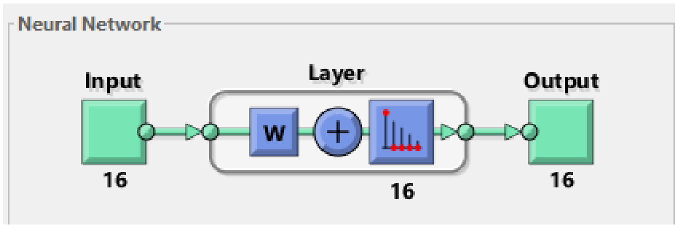

Fig 4: Structure (16)

The structure of the designed neural networks was as given above. When we implemented the result with 16 neurons, the time taken by the network to converge was 13 seconds. As discussed in the class, the network was run for 160 iterations which was 10 times the number of neurons. The topology and the hits by the networks can be given as follows:

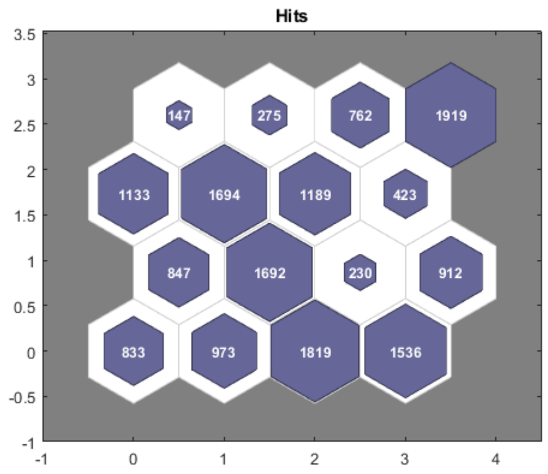

Fig 5: Topology Hits (16)

From the topology above, we can observe that the 16 neurons are divided into 4x4 array as given as input in MATLAB. We could also see the weights of each neuron. These weights play an important role in the reconstruction of the image as we multiply these weights with the code vector to reconstruct the original image.
We could see the distance between the neurons, where darkest have the highest distance. We could also observe the connection between neurons which is given by red line.

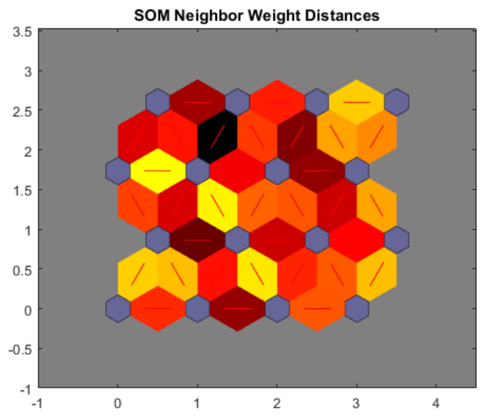

Fig 6: Neighbor distance (16)

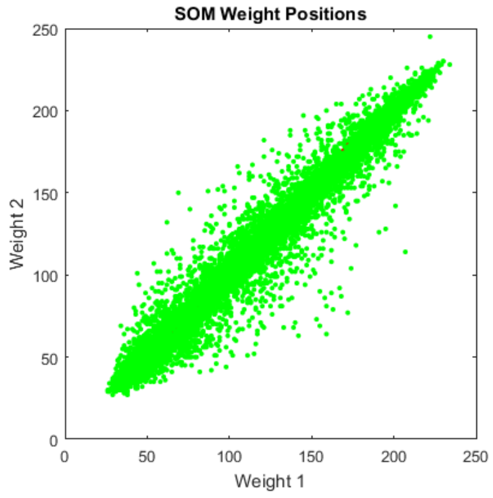

Fig 7: Weights (16)

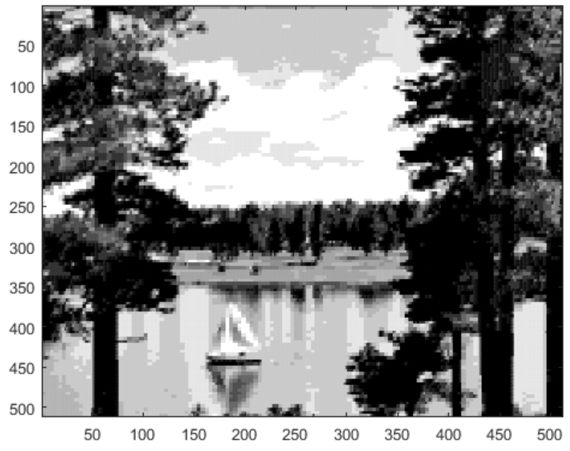

Fig 8: Reconstructed Image (16)

We can observe the original image and reconstructed image and visually how well it's reconstructed.

b. **SOM (32 neurons):**

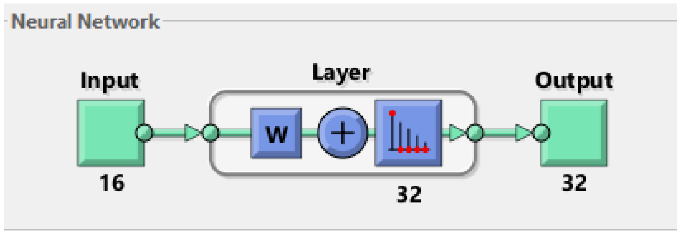

Fig 9: Structure (32)

The structure of the designed neural networks was as given above. When we implemented the result with 32 neurons, the time taken by the network to converge was 2 minutes 16 seconds. As discussed in the class, the network was run for 320 iterations which was 10 times the number of neurons. The topology and the hits by the networks can be given as follows:

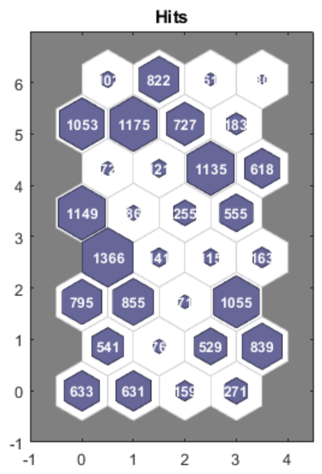

Fig 10: Topology Hits (32)

From the topology above, we can observe that the 32 neurons are divided into 4x8 array as given as input in MATLAB. We could also see the weights of each neuron. These weights play an important role in the reconstruction of the image as we multiply these weights with the code vector to reconstruct the original image.
We could see the distance between the neurons, where darkest have the highest distance. We could observe the connection between neurons by the red lines.

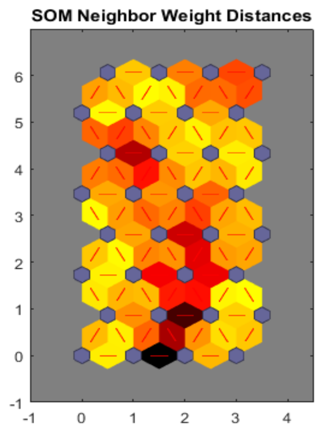

Fig 11: Neighbor distance (32)

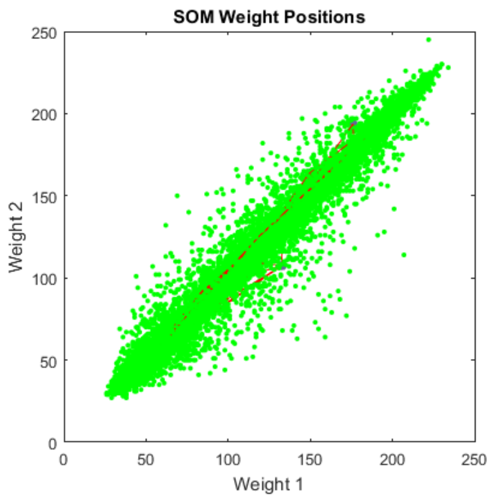

Fig 12: Weights (32)

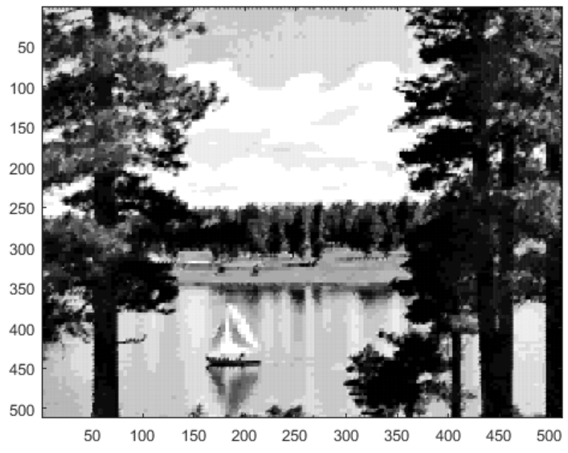

Fig 13: Reconstructed Image (32)

We can observe the original image and reconstructed image and visually how well it's reconstructed.

c. **SOM (64 neurons):**

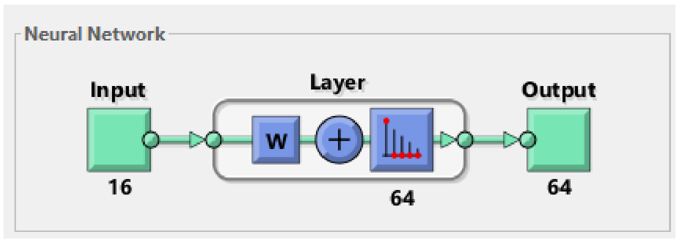

Fig 14: Structure (64)

The structure of the designed neural networks was as given above. When we implemented the result with 64 neurons, the time taken by the network to converge was 7 minutes 42 seconds. As discussed in the class, the network was run for 640 iterations which was 10 times the number of neurons. The topology and the hits by the networks can be given as follows:

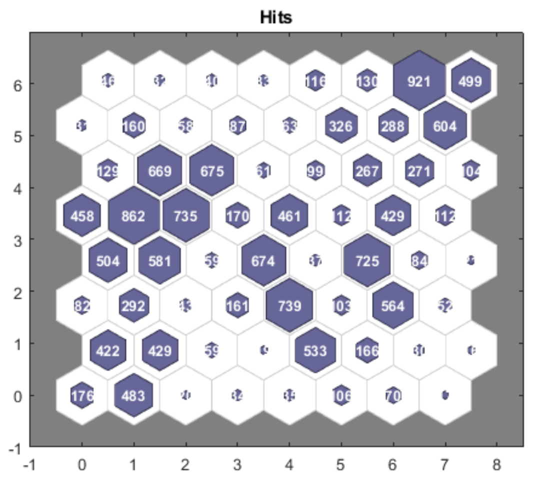

Fig 15: Topology Hits (64)

From the topology above, we can observe that the 64 neurons are divided into 8x8 array as given as input in MATLAB. We could also see the weights of each neuron. These weights play an important role in the reconstruction of the image as we multiply these weights with the code vector to reconstruct the original image.
We could see the distance between the neurons, where darkest have the highest distance. We could observe the connection between neurons by the red lines.

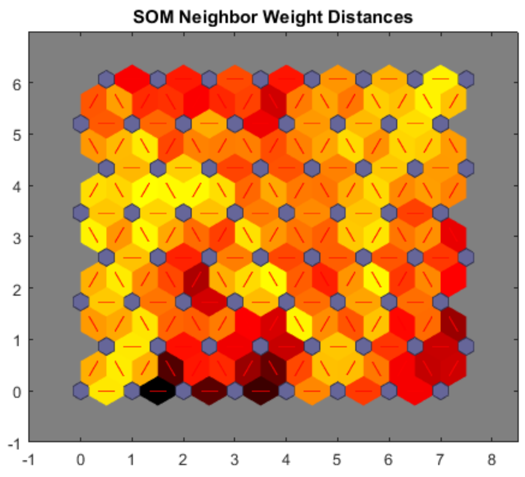

Fig 16: Neighbor distance (32)

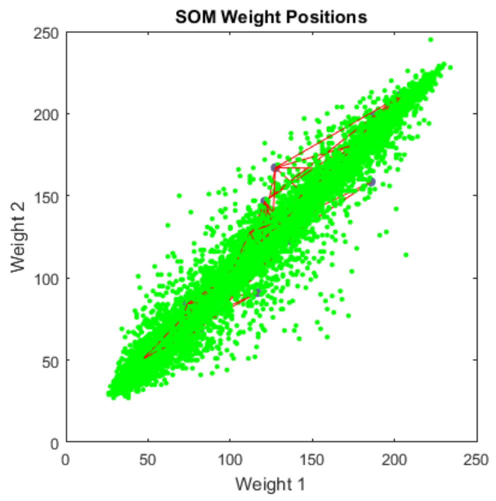

Fig 17: Weights (64)

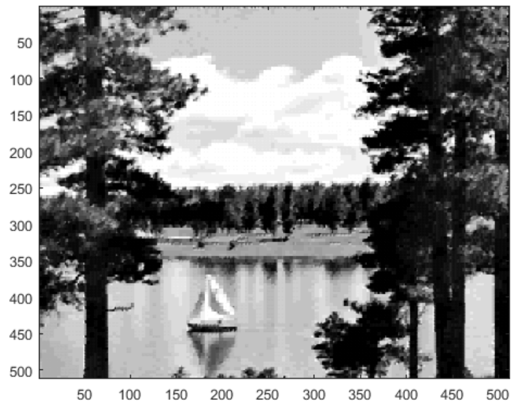

Fig 18: Reconstructed Image (64)

We can observe the original image and reconstructed image and visually how well it's reconstructed

d. **SOM (128 neurons):**

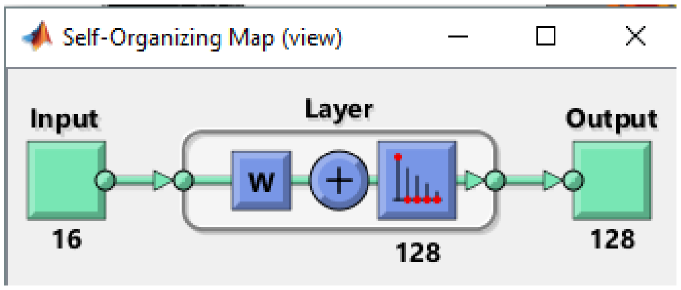

Fig 19: Structure (128)

The structure of the designed neural networks was as given above. When we implemented the result with 128 neurons, the time taken by the network to converge was 13 minutes 07 seconds. As discussed in the class, the network was run for 320 iterations which was 10 times the number of neurons. The topology and the hits by the networks can be given as follows:

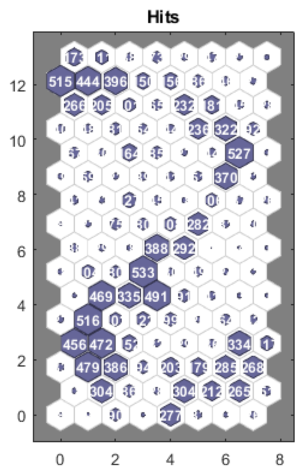

Fig 20: Topology Hits (128)

From the topology above, we can observe that the 128 neurons are divided into 8x16 array as given as input in MATLAB. We could also see the weights of each neuron. These weights play an important role in the reconstruction of the image as we multiply these weights with the code vector to reconstruct the original image.
We could see the distance between the neurons, where darkest have the highest distance. We could observe the connection between neurons by the red lines.

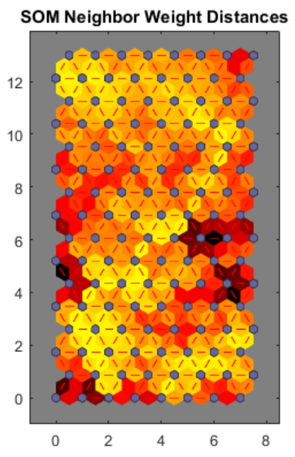

Fig 21: Neighbor distance (128)

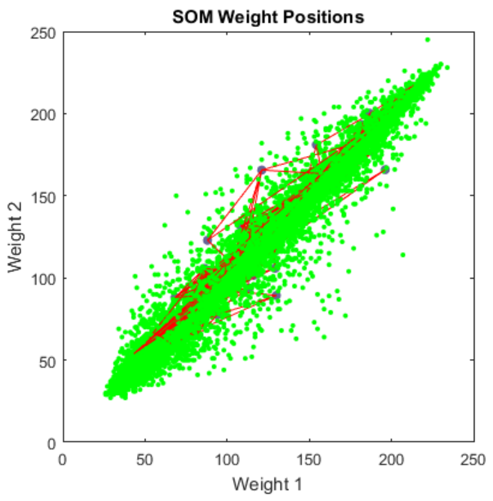

Fig 22: Weights (128)

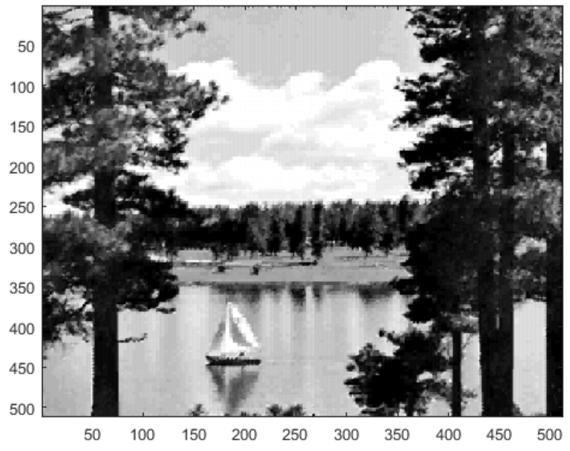

Fig 23: Reconstructed Image (128)

We can observe the original image and reconstructed image and visually how well it's reconstructed

e. **SOM (256 neurons):**

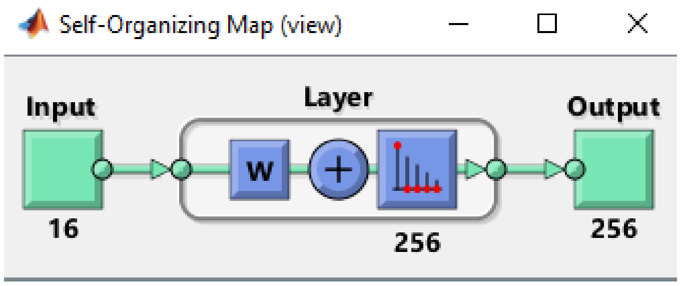

Fig 24: Fig 19: Structure (256)

The structure of the designed neural networks was as given above. When we implemented the result with 256 neurons, the time taken by the network to converge was 33 minutes 17 seconds. As discussed in the class, the network was run for 2560 iterations which was 10 times the number of neurons.

The topology and the hits by the networks can be given as follows:

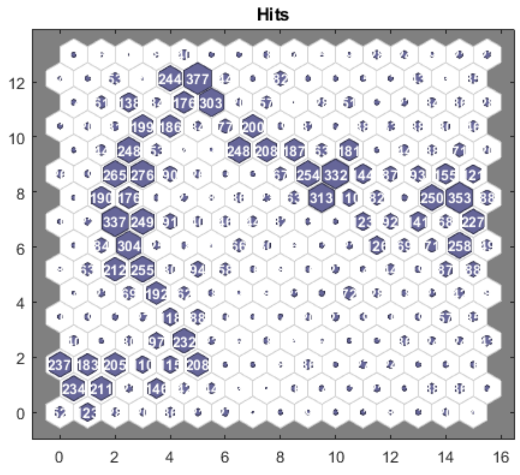

Fig 25: Topology Hits (256)

From the topology above, we can observe that the 256 neurons are divided into 16x16 array as given as input in MATLAB. We could also see the weights of each neuron. These weights play an important role in the reconstruction of the image as we multiply these weights with the code vector to reconstruct the original image.
We could see the distance between the neurons, where darkest have the highest distance. We could observe the connection between neurons by the red lines.

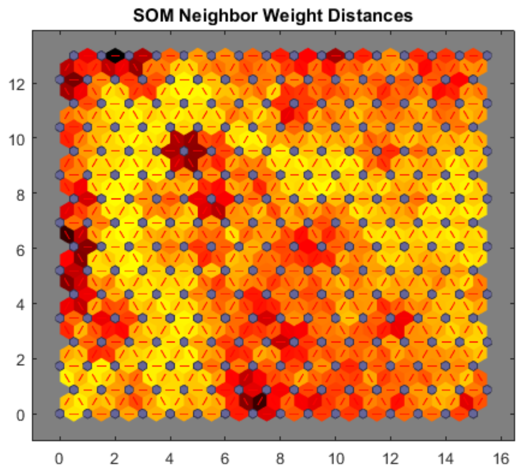

Fig 26: Neighbor distance (256)

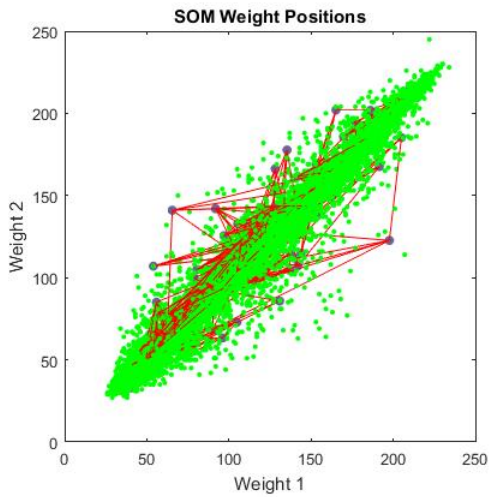

Fig 27: Weights (256)

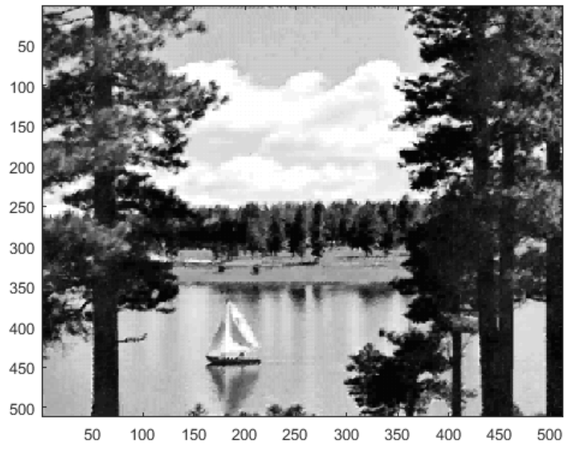

Fig 28: Reconstructed Image (256)

We can observe the original image and reconstructed image and visually how well it's reconstructed.

f. SNR vs BPP

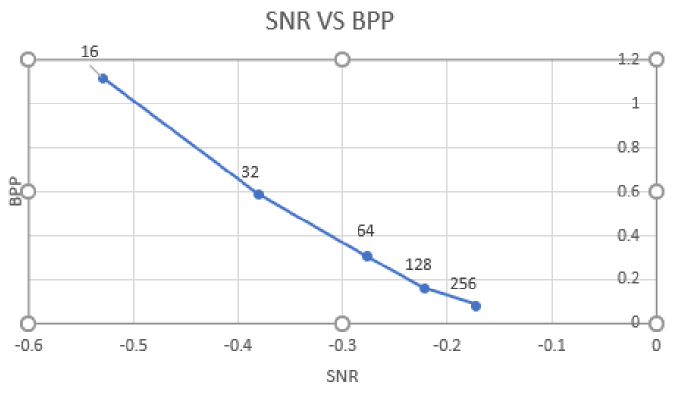

Fig 28: SNR vs BPP

We also calculated the plot between snr and bpp. We could see SNR and BPP are directly proportional. As the number of neurons increases the SNR decreases and so does the BPP. Initially, we could see the noise is more in the image but as we significantly increase the number of neuron, noise in the image decreases.

TABLE I

NEURONS	COMPRESSION RATE	DISTORTION RATE	BPP
16	32	-0.5311	1.125
32	16	-0.3809	0.5937
64	8	-0.2778	0.3125
128	4	-0.2223	0.1641
256	2	-0.1733	0.08594

We can see from the table the relation between compression and distortion rate. We can observe that as the compression rate decreases the distortion also decreases. That's the trade-off we have, highly the compressed image more noise in the reconstructed image.

g. General model:

We observed that the default parameters work best with the model.
Hexagon topology and Linkdist are the best parameters for this problem.
As we increase the number of iterations, we get lower and lower SNR, except at one point the training saturates and we get the same SNR irrespective of increasing the iterations.
As we observed earlier increasing the number of neurons increases the SNR and gives better visual image.
Reconstructed image with best parameters is as follows:

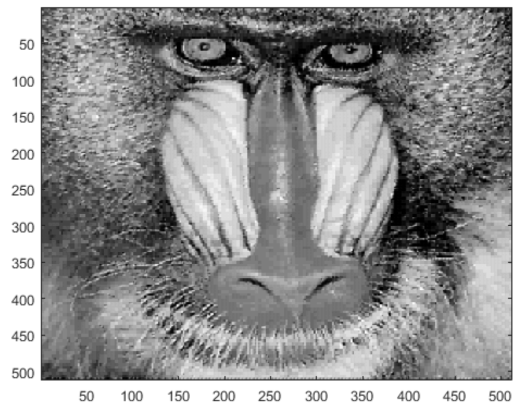

Fig 29: Validated image

h. Time Utilization:

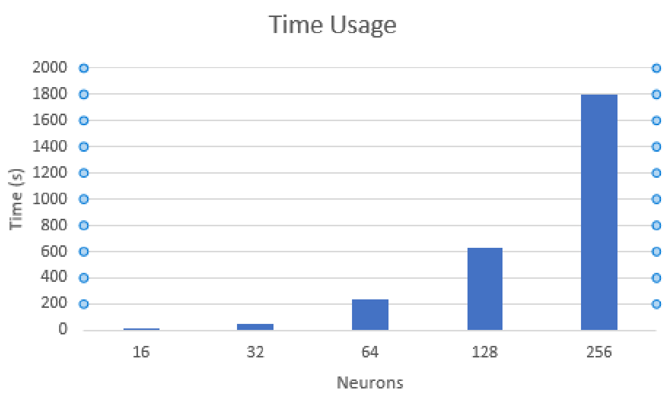

Fig 30: Time Utilization

From the plot above, we can observe that as the number of neurons increases, iterations increases and thus, there is almost a quadratic increase in time.

i. Complexity:

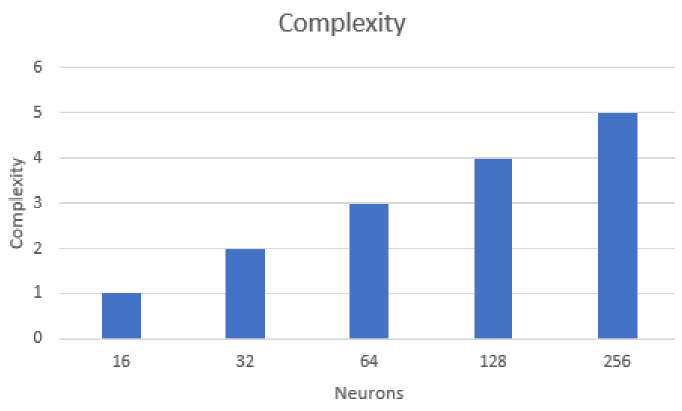

Fig 31: Complexity

Though, the formula for complexity is same for all the SOMs, but the complexity depends on the number of neurons as the number of multiplication increases. Thus, the higher the number of neurons greater the complexity.

j. Advantages & Disadvantages:

Advantages

1. Easy to understand.
2. No target required
3. Easily scalable

Disadvantages

1. Time Complexity
2. Convergence depends on data

V. CONCLUSION

Thus, we can conclude that we have successfully reconstructed the image and also plotted various relations between SNRs and BPPs and also observed the time utilized and complexity of the model. We can observe different topologies and how it changes with increasing neurons and how the reconstructed image gets better with increase in neurons. Though, we took SNR as a factor to see if the reconstructed images are better than the other images but we also can visually observe the change in reconstructed images with the increasing neurons. We also discussed and validated the best parameters.

VI. REFERENCES

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.1938&rep=rep1&type=pdf
2. http://www.philadelphia.edu.jo/academics/qhamarsheh/uploads/Lecture%2016_Self-organizing%20map%20using%20matlab.pdf
3. https://www.mathworks.com/help/nnet/ref/selforgmap.html
4. http://davis.wpi.edu/~matt/courses/soms/
5. https://en.wikipedia.org/wiki/Data_compression_ratio