# FLORIDA INSTITUTE OF TECHNOLOGY

Department Of Computer Engineering & Science

CSE 5290 **Artificial Intelligence**

FALL - 2022


Under the guidance of
Dr. Debasis Mitra, Ph.D.

**Self Organizing Maps**

By

Austin Gay, Rishitha Mulagolla

# Table Of Contents

**Abstract**

The goal of this project was to create and train a Self Organizing Map to analyze the topological features of selected characters from an Indus Valley Civilization character set. To achieve this the code for a Self-Organizing Map was written using Python and various of its libraries to prepare the data and train the model. Initially, the SOM was trained using the MNIST dataset which achieved satisfactory results and acted as a proof of concept for training a SOM using the IVC character set. The SOM code was adjusted and the IVC character set was used to train the SOM. The results show that the SOM resulted in topological characteristics resembling the IVC characters, with similarly shaped characters falling close together or within one region of the Map.

**Hypothesis and Expected Results**

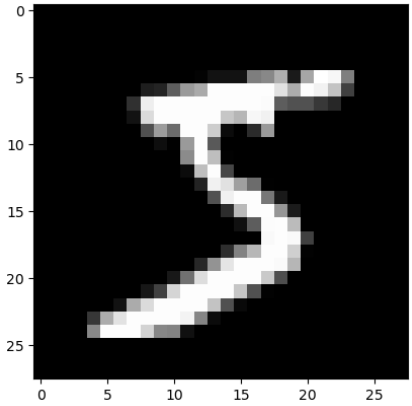
The self-organizing map that results from training it on the IVC dataset will produce a map that reflects the IVC characters within each of the nodes. Specific nodes will reflect certain characteristics of the input characters. For example, one node might have a combination of topological features from different IVC characters. Trained correctly, characters that share similar shapes or topological features will fall close together on the trained self-organized map.

**Basics of a Self Organizing Map**

A Self-Organizing Map is a type of neural network that produces a lower dimensional representation of the input data. The most important aspect of a Self-Organizing Map is its ability to retain the topological structure of the data. The training of a Self Organizing Map involved the following steps: Initialize Map, find Best-Matching Unit for input data, adjust weights in the Map. The most effective method of initializing a SOM is to randomly generate the initial data in order to obtain the greatest, more accurate, and consistent results. The Best-Matching Unit or BMU is found by calculating the Euclidean distance between the input data and each node in the SOM. The node with the lowest Euclidean distance is the BMU for the input. Finally, the BMU and its neighboring nodes are adjusted in the direction of the input data. As the algorithm runs the nodes will reflect the input data more accurately.

**MNIST Dataset**

The MNIST dataset is a commonly used dataset consisting of handwritten digits (0-9). The dataset contains 60,000 training images and 20,000 testing images. All images included in the dataset are provided with labels. In this project the labels are unnecessary as Self-Organizing Maps are a type of unsupervised training. The size of each image in the dataset is 28 by 28 pixels. The fact that every image provided in this dataset is the same size makes this dataset very easy to work with meaning that no preprocessing on the images needs to be done to use them.

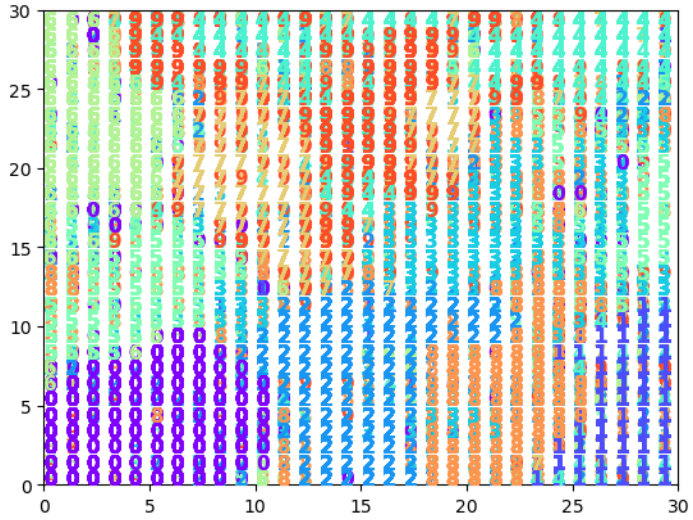


**Method using MNIST data**

The MNIST dataset contains images that are already preprocessed and are thus in an already sufficient state for processing. One of the conditions which the dataset already meets is that the pictures are all the same size. Three models of size (30, 30), (20, 20), and (10,10) are trained using all 60,000 images from the training data. The trained map is then displayed using a function to label and display the map. Using the test data, the best matching unit for the test data is found and the map is labeled with the label provided in the test data set.

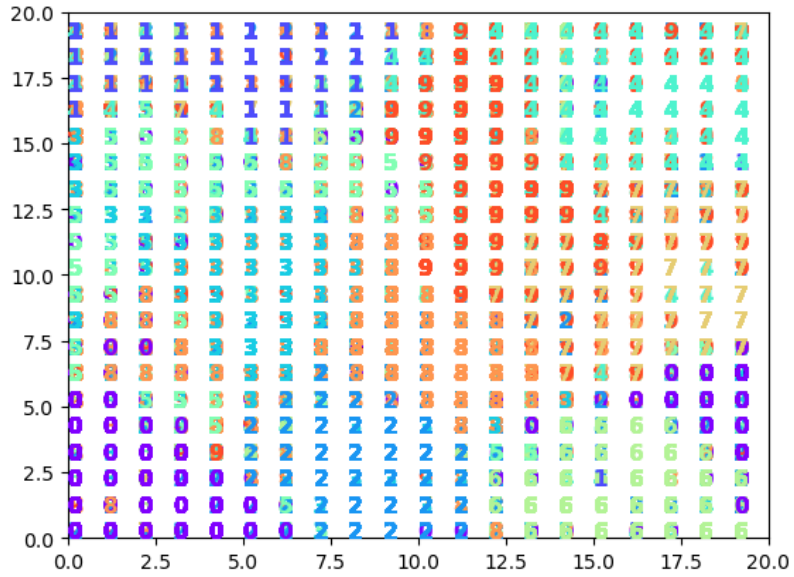
**Results using the MNIST dataset**

30 by 30 SOM Figure:



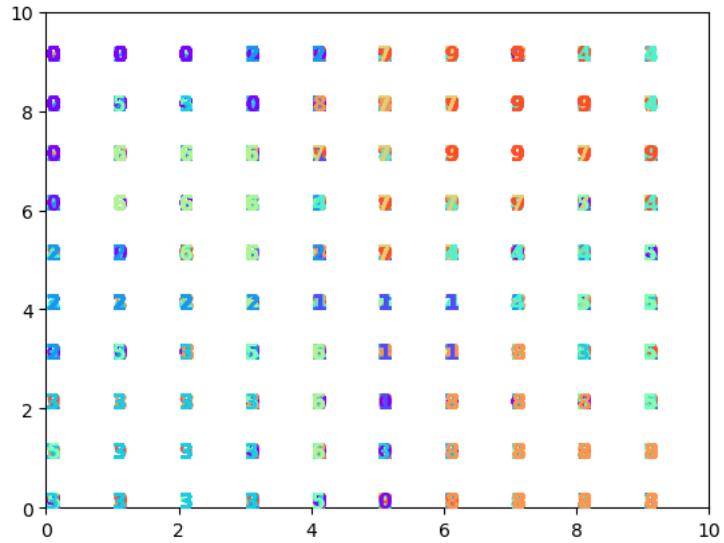

It can be observed that numerical values tend to be labeled on neurons within a close proximity. For example the vast majority of '0' labels are in the bottom left corner of the map. This demonstrates how self-organizing maps maintain the topological structure of the data when being trained. Additionally, looking into the map further, it can be seen that numbers with similar shapes fall within the same area of the map. It should even be noted that this relationship of similar shape can be proven by nodes that are assigned several different numeric labels. Overall the result of the 30 by 30 map produced shows that the training was successful in retaining the topological features of the data and visualizing the data set in reduced dimensions. 30 by 30 obviously gives the most nodes to train therefore it leads to more accuracy and in a way can be considered overtrained. Regularization is a technique in machine learning to regularize data or in other words, make the model more generic. One approach we might take to this is by allowing less nodes. Less nodes may give the effect of regularization and each node will have to take on a more generic set of characteristics of shape based on the numeric values of the MNIST data set.

20 by 20 SOM Figure:




The result of the 20 by 20 map produced very similar results to that of the 30 by 30 however at first glance the map seems cleaner and easier to read. This conclusion indicates that the 20 by 20 map has a more appropriate number of nodes for the number of classes contained in the MNIST data. It should be noted that the '5' class does not seem to have a clear cluster within the map. This could also be said of the '7' class as the nodes recognize several other numbers within the test set.

10 by 10 SOM Figure:

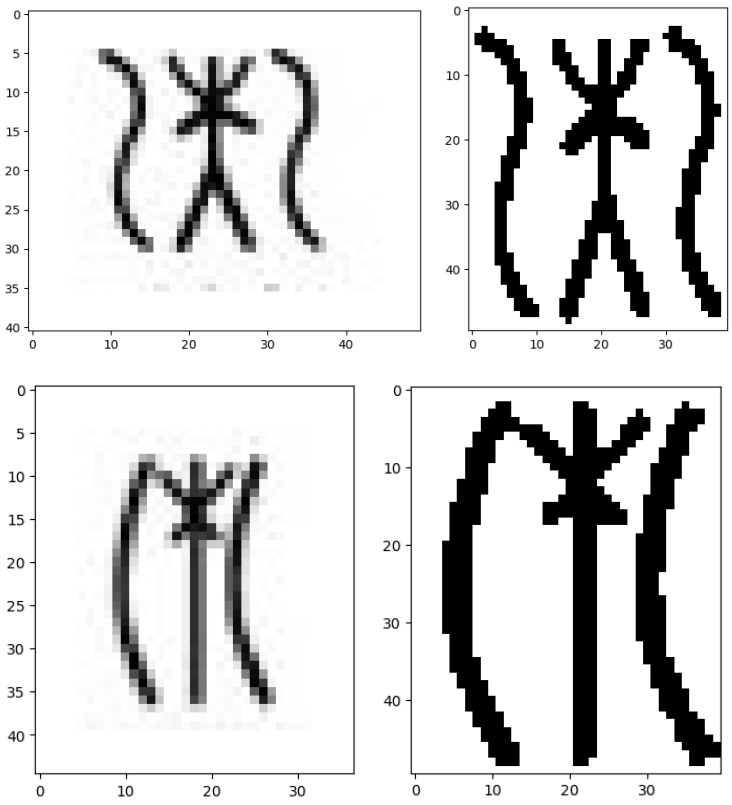



The final SOM produced using the MNIST training set was a 10 by 10 map. Ths map, like the others, maintains the topography of the data however, the clusters for each class are not as distinct. This may be due to the fact that 10 by 10 nodes is not sufficient for representing the MNIST dataset in 2 dimensions.

**IVC Characters dataset**

The IVC, Indus Valley Civilization, was a civilization located in the Northwestern regions of South Asia between 3300 BCE and 1300 BCE. Selected characters from this dataset are used to train the Self-Organizing Map. Exactly 100 characters were selected. The data was initially in the form of characters within a document.

To prepare the data, first each of the desired characters was screenshot and saved to a file. Next, the screenshots were preprocessed. The first step in preprocessing was to make all of the images the same size. This was achieved by first cropping the image so that the character took up the entire image and reduced the amount of blank space. After cropping the images, all of the images were resized to a specified image size using the Pillow resize function. In addition to resizing and cropping, the images were enhanced by running a smoothing filter, an edge enhancing filter, and binarizing (thresholding) the image. All of this preprocessing was to make the image more clear, clean, and less cluttered.



**Methods using IVC Character Set**

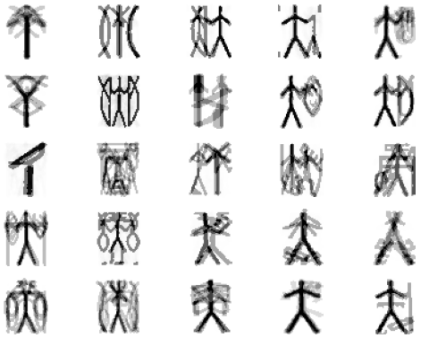
As mentioned before, the dataset comes in the form of screenshots, or a folder of images. Each image was imported into python and saved as a pandas dataframe. Essentially

each image was a n by m array of grayscale intensity values. The images were then preprocessed to produce consistency and clean the data. The data was then fed through the training function and a SOM was trained. The SOM was then printed by displaying each node as an image.

**Results using the IVC character set**

10 by 10 SOM Figure

The 10 by 10 SOM result as seen above has several nodes that contain random noise. This random noise is the initial state of the SOM and therefore we can conclude that during the training of the SOM those nodes were never chosen as the best matching unit. Due to this, it can be concluded that a SOM of size 10 by 10 has too many nodes for the IVC dataset. If we look into the results of the trained SOM, we can see that IVC characters with similar designs are portrayed in similar or by the same node in a region. The stick figure like IVC characters are on the left, while characters holding object's are on the top right. It can clearly be seen that the resulting image produced by certain nodes reflects or represents the design of multiple IVC characters. This can be more clearly seen on the SOM's that will be shown later.



Analyzing this last figure of a 5 by 5 SOM, we can clearly see that the primary shape of figures is very defined and has high intensity values. While features that complement the primary figure are much lighter in intensity and you can see features from multiple of the IVC characters in a single node. One such node, looking at the bottom left node, clearly has what looks like a stick body as the primary feature of the character and other complementary features in a lighter intensity. Each one of these nodes started as random noise and ended resembling the IVC characters to a high degree. Each node contains some characteristics of an IVC and thus the entire map contains features of every single IVC somewhere. This is what it means for the Self-Organizing Map to retain the features of the training data. This map has successfully represented the entire IVC training dataset and retrains its important topological features and characteristics.

**Conclusion:**

As per our given task, we have successfully trained a self-organizing map (SOM) using a few characters from the Indus valley civilization data set which was obtained as an extension of our previous submission SOM using MNIST data set. The trained SOM from the IVC dataset produced results in which each node reflected one or more different IVC characters. Each individual node would generally reflect the more common shapes in high intensity and the less common distinct features of each character in a lighter intensity. It can be concluded that the SOM's trained maintained the topological structure of the data and thus produced a lower dimensional representation of the IVC character dataset.

**References:**


- https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da
- https://stackabuse.com/self-organizing-maps-theory-and-implementation-in-python-with-numpy/
- https://en.wikipedia.org/wiki/MNIST_database
- https://iust-projects.ir/post/ann03/
- https://towardsdatascience.com/understanding-self-organising-map-neural-network-with-python-code-7a77f501e985
- https://en.wikipedia.org/wiki/Self-organizing_map
- https://www.geeksforgeeks.org/self-organising-maps-kohonen-maps/?ref=gcse
- https://www.geeksforgeeks.org/training-neural-networks-with-validation-using-pytorch/?ref=gcse