Self-Organizing Map for Handwritten Digits

Enrico Collini*
University of Florence
Master in Computer Engineering
encollini@gmail.com

Andrea Di Martino*
University of Florence
Master in Computer Engineering
adimartino95@gmail.com

***Abstract*— The self-organizing map (SOM) methodology does vector quantization and clustering on the dataset, and then project the obtained clusters to a lower dimensional space, such as a 2D map, by positioning similar clusters in locations that are spatially closer in the lower dimension space.**
This make the SOM methodology an effective tool for big dataset visualization. But SOM becomes an unattractive tool because of its time complexity. In this paper, we propose a different version of the SOM.
This kind of implementation is based on Python and on a Google library called TensorFlow. Our implementation has been used to cluster and project 25000 and 50000 image samples of handwritten digits from the MNIST dataset.
The main features we have elaborated are a log system to trace the development during training and testing, a termination mechanism, a 2d space heatmap of the SOM and a testing system with a graphical representation and tables to estimate the best dimensions for our scenario.

I. INTRODUCTION

This project is an assignment for the course Data Mining of the Professor Simone Marinai at the Master in Computer Engineering at the University of Florence.
The paper "Fast Emulation of Self-organizing Maps for Large Datasets" [1] inspired the development of a SOM for the classification of handwritten Digits.
The base implementation of Sachin Joglekar [2] and than modified by Wonik Jang [3] has been further developed in order to obtain a SOM with a termination training mechanism and the heatmap representation of the map.
The machine learning frameword used was Tensorflow.
The last part of this script was the study of which dimension of the self-organizing map was best suited for the application to the MNIST Dataset of handwritten digits.

II. SOM, TENSORFLOW, MNIST DATASET

A. Self-organizing Map

One of the enablers of Big Data is the intuitive presentation of information such as visualization. [4].
Visualization provides intuitive display of unstructured information. These types of data requiring visualization tools. One of these visualization tools is the Self-Organizing Map (SOM) [5]. SOM represents data using nodes as points in two-dimensional vector space.
These SOM nodes have weight vectors which are updated per iteration depending on the input vector from the data set. The initial value of the weight vector is random. Generally, the weight vectors are updated as follows.
$wi(t+1) = wi(t) + G(t)i(t)||x(t)wi(t)||$ where t represents the iteration number, we represent the weight vector of the ith node, x(t) is the input vector chosen randomly from the training set, i(t) is the learning rate of the adaptation process, G(t) is a window function which is typically a Gaussian window or a rectangular window, and $||x(t)wi(t)||$ is the Euclidean distance between x(t) and wi(t).

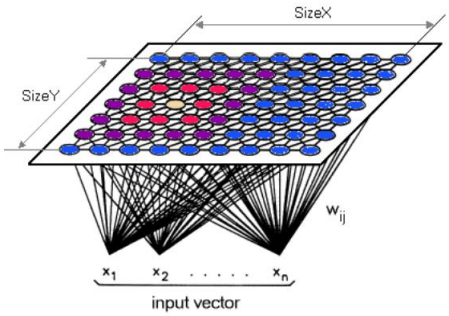

Fig. 1. SOM grid representation; photo by Sachin Joglekar [2]

A SOM falls under the rare domain of unsupervised learning in Neural Networks. Its essentially a grid of neurons, each denoting one cluster learned during training.
In a SOM, each neuron has a location, and neurons that lie close to each other represent clusters withsimilar properties. Each neuron has a weight vector, which is equal to the centroid of its particular cluster.

B. TensorFlow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.
Originally developed by researchers and engineers from the Google Brain team within Googles AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.[6]
Thanks to this library we can get better performance during the training.

C. Mnist Dataset

In this paper we propose the use of a large dataset of handwritten digits called MNIST.
The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used fortraining various

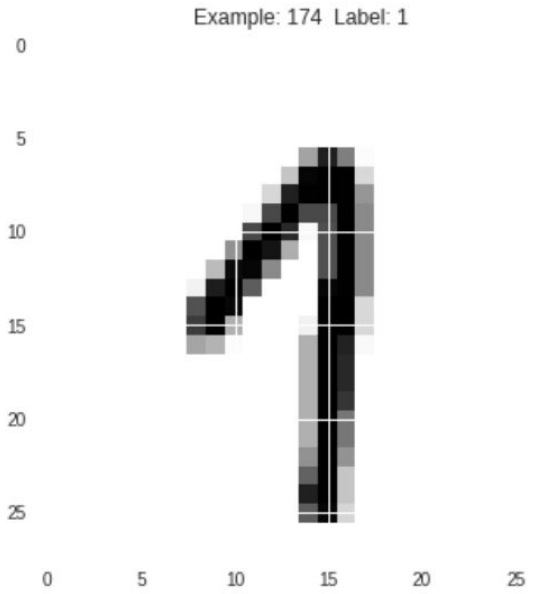

Fig. 2. example of an handwritten digit with label 1 of the MNIST Dataset

image processing systems.
The database is also widely used for training and testing in the field ofmachine learning. It was created by ”re-mixing” the samples from NIST’s original datasets. The creators felt that since NIST’s training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments.
Furthermore, the black and white images from NIST were normalizedto fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. [7]
In this paper we used a version of the MNIST that contain 55.000 images of handwritten digits.

III. DEVELOPMENT

A. LOG system

Starting from an existing TensorFlow implementation of the SOM [2] [3], in this paper, we try to show how the training was performed during the execution.
We have also tried to show all the most significant aspect of the computational phase of this paper.
To obtain this we have built a Log system, that easily show us the progress of training and testing. Generally, a LOG output is as in figure 3.

```
training started
Epoch number:  1
10 % training completed
20 % training completed
30 % training completed
40 % training completed
50 % training completed
60 % training completed
70 % training completed
80 % training completed
90 % training completed
100 % training completed
The norm of difference between two centroids is: 0.89341
.
.
.
.
Epoch number: n
10 % training completed
20 % training completed
30 % training completed
40 % training completed
50 % training completed
60 % training completed
70 % training completed
80 % training completed
90 % training completed
100 % training completed
Testing Started
Testing Completed
Centroid n°: 0 has: 180 images
.
.
.
.
Centroid n°: n has: 240 images
```

Fig. 3. Log system showing the percentage of training completed, the number of epochs and the norm of the difference between two centroids. It shows also when the testing is started and ended with the number of images per neuron

B. Centroids and Visualization

A centroid visualization from a SOM can be used to see the profiles of the values in the dataset that have been associated with a particular node.
The SOM centroid visualization shows the characteristics of the clusters, i.e., the representative profile (the centroid).
This provides important abstract information about how the gene expression data relates to the clustering provided by the SOM. It also shows the corresponding node’s reference vector, which allows comparison of the representative profile of the cluster with the node’s reference vector to determine how well (on average) the points associated with that node actually match that node’s characteristics.
In this way we can see which centroids have a certain label, and in the meantime notice the formation of classification areas. In fact, we will note that in a zone the numbers with label 0 are classified, in another those with label 1 and so on.
This is a kind of SOM visualization 4.

C. Termination Training Mechanism

The training phase is crucial, because there are two opposite situations that we must avoid: the Overfitting and Underfitting.
Overfitting is especially likely in cases where learning was performed too long or where training examples are rare, causing the learner to adjust to very specific random features of the training data, that have nocausal relationto thetarget function. In this process of overfitting, the performance on the training examples still increases while the performance on unseen data becomes worse. In the opposite way we speak about underfitting.
Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as

Fig. 4. SOM grid 10x10 trained with 20000 images. Through this we can see the centroids visualization, and understand the distribution of numbers concentrated in certain areas

it will have poor performance on the training data. To avoid these opposite situations, we propose a Termination-Training mechanism.

This mechanism is based on the norm of the difference between two centroids of two different epochs and the trend of this is shown in fig 5.

If the distance between the same centroids in five consecutive epochs is low, then the mechanism stops the training and lunch the testing phase. The threshold value was 0.8 initially, but it was then raised up to 5 for bigger dimension SOM as the 15x15 and 16x16

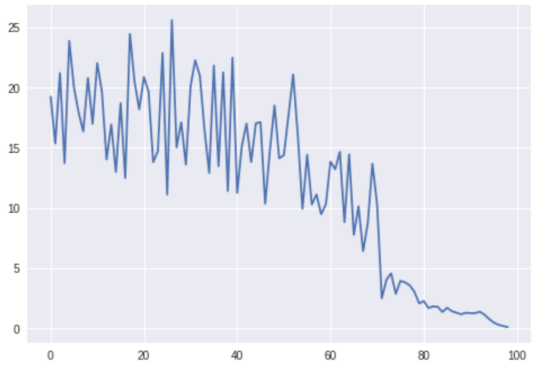

Fig. 5. Graph of a SOM 14x14 trained with 20000 images showing the trend of the norm of the difference for each iteration of the training mechanism.

D. Testing: accuracy assignment of neurons

During the execution of the algorithm, thanks to some checkpoints, we can see the accuracy assignment of neurons. We compute the accuracy percentage considering the max value of images classified in each neuron divided by the number of all the images in the specific neuron. Then we use a log system that shows us the results. The results obtained are also visible in the graph form in the fig 6 and inside the tables of the section ”Results”.

Fig. 6. Graph showing the classification accuracy of a SOM 14x14 trained with 20000 images and tested with other 20000 images

E. Heatmap Representation

Another graphic way to represent the SOM, is the Heatmap representation. We assign at two opposites colors, two opposites values.

In this case we assign Dark Blue to 100% and White to %.

In this way we can show how the value of accuracy is arranged between neurons fig 7.

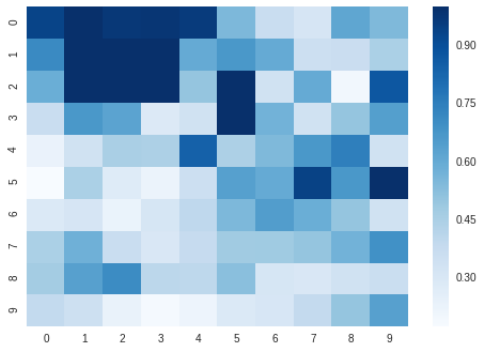

Fig. 7. Heatmap Representation of a Som 10x10 trained with 20000 images

IV. RESULTS

In this section are summarized the results we obtained modifying the dimensions of the SOM and the number both of training input size and testing input size in order to find the perfect balance between underfitting and overfitting as described in the section regarding the termination training mechanism.
In the tables are reported the outcomes of:

- SOM trained with 20000 images and tested with 5000
- SOM trained with 50000 images and tested with 5000
- SOM trained with 20000 images and tested with 2000

Exploring the results we notice that, fixed the training input size (20000), there is a transition between a case of underfitting (4x4) to a situation of overfitting (16x16). Between these two opposites there is the best SOM dimensions for the accuracy of the neurons.
With small dimensions there is the underfitting problem and this cause the SOM to misclassify the majority of the test images in neurons with accuracy lower than 50%.
Another important aspect to show is the time of training and testing that increases proportionally with the dimension of the SOM.

SOM dim	$A < 50$	$50 < A < 75$	$75 < A < 85$	$85 < A < 95$	$A > 95$
4x4	81.025%	11.25%	0%	7.725%	0%
8x8	54.92%	27.4%	11.32%	2.8%	3.56%
10x10	25.58%	39.37%	8.84%	19.8%	6.4%
11x11	17.12%	38.2%	7.34%	19.46%	17.88%
12x12	12.55%	24.04%	9.96%	18.18%	35.32%
13x13	4.34%	19.6%	11.46%	30.88%	33.72%
14x14	6.48%	8.78%	11.52%	21.26%	51.96%
15x15	12.38%	21.48%	12.22%	17.2%	36.72%
16x16	8.16%	19.82%	7.4%	22.02%	42.6%

TABLE I

PERCENTAGE OF IMAGES CLASSIFIED IN THE NEURON ACCURACY (A) CATEGORIES WITH TRAINING INPUT SIZE 20000 AND TESTING INPUT SIZE 5000

V. CONCLUSIONS

Exploring the results obtained our SOM implementation, when applied to the MNIST Dataset, works really well with the dimensions of 14x14 with 20000 training images and 16x16 with a training size of 50000.
The features implemented provide useful tools both from the representation of the SOM and the study of the progress when trained.
The training mechanism is probably the most influential implementation detail because it allows the SOM to be applied at different training dimensions not with a fixed number of training epochs, but according to the error of two different iteration, obtaining more specific results.
The LOG system has been fundamental for the development of this assignment specifying the details both during training and testing allowing us to better analyse the SOM.
The graphs has been a useful tool for the analysis of the trend during training but also for the testing fase.

VI. FUTURE DEVELOPMENTS

There are multiple possibilities of development.
Probably the most interesting thing to implement is a mechanism for improving the speed of the computation using the composition of other clustering algorithms.
And of course an important development is the application of the SOM implemented with other types of Datasets to study its performance.

REFERENCES

[1] Macario O. Cordel, Arnulfo P. Azcarraga, Fast Emulation of Self-organizing Maps for Large Datasets, Procedia Computer Science, Volume 52, 2015, Pages 381-388, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.05.002

[2] Sachin SOM.
[3] Wonik unsupervised SOM
[4] ITU Telecommunication Standardization Bureau. Big Data: Big today, normal tomorrow. Geneva: 2013 November
[5] Kohonen T. Self-Organizing Maps London: Springer 2001
[6] TensorFlow web page: https://www.tensorflow.org/
[7] Wikipedia about MNIST: Mnist

SOM dim	$A < 50$	$50 < A < 75$	$75 < A < 85$	$85 < A < 95$	$A > 95$
4x4	86.66%	6.92%	3.98%	0%	2.44%
8x8	72.56%	18.72%	2.3%	2.55%	3.92%
10x10	67.62%	25.88%	0.38%	3.2%	2.92%
11x11	75.56%	17.08%	0.38%	1.94%	5.04%
12x12	16.56%	29.98%	12.92%	11.24%	29.30%
13x13	4.34%	19.6%	11.46%	30.88%	33.72%
14x14	9.5%	18.8%	9.62%	19..94%	47.14%
15x15	3.36%	16.54%	7.14%	24.68%	48.28%
16x16	5.24%	12.1%	8.22%	16.16%	58.28%

TABLE II

PERCENTAGE OF IMAGES CLASSIFIED IN THE NEURON ACCURACY (A) CATEGORIES WITH TRAINING INPUT SIZE 50000 AND TESTING INPUT SIZE 5000

SOM dim	$A < 50$	$50 < A < 75$	$75 < A < 85$	$85 < A < 95$	$A > 95$
4x4	81.03%	11.25%	0.0%	7.72%	0.0%
8x8	59.34%	27.71%	0.31%	12.64%	0.0%
10x10	11.44%	28.47%	9.09%	19.52%	31.48%
11x11	7.025%	16.54%	9.61%	29.37%	37.44%
12x12	2.06%	12.80%	6.08%	39.93%	39.13%
13x13	3.53%	19.47%	10.21%	27.61%	39.18%
14x14	3.59%	11.65%	5.53%	29.83%	49.4%
15x15	9.53%	23.89%	10.76%	20.89%	34.93%
16x16	9.98%	21.77%	9.08%	24.46%	34.71%

TABLE III

PERCENTAGE OF IMAGES CLASSIFIED IN THE NEURON ACCURACY (A) CATEGORIES WITH TRAINING INPUT SIZE 20000 AND TESTING INPUT SIZE 20000