Fast Self-Organizing Maps Training

Flavio Giobergia
flavio.giobergia@polito.it
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy

Elena Baralis
elena.baralis@polito.it
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy

***Abstract*—Self-organizing maps are an unsupervised machine learning technique that offers interpretable results by identifying topological properties in high-dimensional datasets and projecting them on a 2-dimensional grid. An important problem of self-organizing maps is the computational expensiveness of their training phase. In this paper, we propose a fast approach to train self-organizing maps. The approach consists of 2 steps. First, a small map identifies the most relevant areas from the entire high-dimensional input space. Then a larger map (initialized from the small one) is fine-tuned to further explore the local areas identified in the first step. The resulting map has performance (measured in terms of accuracy and quantization error) on par with self-organizing maps trained with the standard approach, but with a significantly reduced training time.**

***Index Terms*—self-organizing maps, unsupervised learning, fast training, clustering**

I. Introduction

Self-organizing maps (SOMs) are a type of artificial neural network used for unsupervised learning. They consist in a (typically) 2-dimensional grid of nodes, each with its own set of weights. During the training process, these weights are iteratively updated, becoming centroids for the clusters that emerge in the data. This process maintains the dataset's topological properties on the 2D grid. Centroids that are close in the input space will be close on the grid and further away from dissimilar ones.

The purpose of self-organizing maps is to model a possibly high-dimensional problem using a low-dimensional representation. This offers insights on the dataset properties and, by building a 2-dimensional grid, it may be adopted as a useful visualization technique. Additionally, the trained nodes can be used for quantizing the points in the dataset (e.g. for compression, or bucketing purposes).

In recent years, self-organizing maps have been used in a large number of works in a variety of fields (among the others, hydrology [1], physics [2], cytometry [3] and sociology [4]). In particular, all of these works leverage the self-organizing map's interpretability by extracting 2-dimensional visualizations and making use of the topological properties that self-organizing maps learn.

Self-organizing maps, however, have some limitations. A prominent one is their long training time. During training, each data point needs to be compared against each of the nodes of the grid. The size of this grid defines how granular the results will be. Hence, a larger self-organizing map typically offers better insights, but at the cost of a higher training time.

In this paper, we propose an approach for training self-organizing maps that relies on 2 steps. In the first step, a fraction of the available dataset is used to train a smaller SOM. The weights of the model learned this way are then used to initialize the weights of the larger, final SOM. In the second step, the larger SOM is fine-tuned using the remaining fraction of the dataset. This approach maintains performance comparable with standard self-organizing maps, but at a fraction of the training time. Thus, larger problems that would not have otherwise been tractable, become approachable. We analyze in-depth the performance of our approach, by considering several factors. It allows us to both define meaningful values for the required hyperparameters, and understand how the model works in specific situations (e.g. when data is scarce).

The rest of the paper is organized as follows. Section II introduces the related work. Section III first provides an overview of SOMs, then presents the proposed approach. Section IV provides a performance assessment in a variety of scenarios. Finally, Section V contains a discussion of the results achieved and makes some comparisons with other techniques, while Section VI concludes the paper with some final considerations.

II. Related work

Self-organizing maps were first introduced in [5]. Originally, weight initialization for the nodes was done randomly and resulted in already well-performing maps. Another well-established approach is to perform the initialization based on the adoption of the first principal components. This approach has been shown to have substantial practical advantages over alternative ones [6].

These approaches to initialization, though, introduce a significant computational overhead to the training of the self-organizing map. Further studies focused, instead, on fast initialization techniques that helped SOMs converge more quickly. Among these, [7] adopts the centroids extracted with the k-means algorithm as initial nodes for the map (with a heuristic for the placement of the nodes to preserve some topological order). Then, the weights are fine-tuned with a partial training of the SOM. Another approach, presented in [8], consists in the following three steps: (1) identify – within the dataset – those points that are furthest apart from one

another (thus building a "hyperbox" that wraps the points in their space) and use them as initial weights for the corners of the grid, (2) assign the weights of the edges of the grid as an interpolation of the corner weights and (3) fill the rest of the weights through similar interpolations. This approach, as the authors point out, requires $\mathcal{O}(M^2)$ comparisons (M being the number of points in the dataset) and is sensitive to outliers. To work around these problems, the proposed solution is once again based on using an initial k-means step to quantize the dataset. These works address the problem introduced in this paper. However, our approach does not rely on k-means nor any other algorithm other than self-organizing maps. The proposed approach will be shown to train significantly faster than k-means: in these terms, this makes the proposed approach a better candidate for larger datasets.

There have also been efforts toward building distributed versions of self-organizing maps to cut down the training time. One such work is presented in [9]. The SOM is divided into non-overlapping smaller maps that are distributed across different workers. For each point of the training set, each worker first identifies the local "winning" node (see Subsection III-A for more details), then all workers are synchronized to identify the global "winner". Finally, with this knowledge, each worker updates its map's weights. The union of all maps results in the final SOM. More recently, other distributed approaches to self-organizing maps have been proposed in [10]. They introduce a Map-Reduce approach where the *map* function is tasked with identifying the winning neuron for each point of the dataset (which is split across multiple workers) and emitting a *(winning node, input point)* key-value pair. Then the *reduce* function performs batch updates for each of the self-organizing map nodes. Both these distributed approaches leverage multiple workers to distribute the training of standard SOMs. Our approach is instead based on a single worker. However, our approach can be easily distributed across multiple workers using either [9] or [10]. This is possible because our algorithm is based on the training of standard SOMs.

III. Fast Self-Organizing Maps Training

This section will first offer a brief formal introduction to self-organizing maps, with considerations on their training complexity, followed by the presentation of the proposed 2-step approach to self-organizing maps training.

A. Self-Organizing Maps

A self-organizing map (or SOM) is comprised of a set of neurons, or nodes, displaced on a 2-dimensional grid (for simplicity, we assume the grid to be of $Q \times Q$ nodes). Let $X \in \mathbb{R}^{M \times N}$ be the available dataset, where each of the M rows represents a point in $\mathbb{R}^N$. Each node n_j of the SOM is associated with a weight vector $w_j \in \mathbb{R}^N$, typically randomly initialized. Additionally, at any step t of the training phase and for each node n_j a set of neighbors $N(n_j, t)$ can be defined (i.e. a set of nodes close to n_j on the grid). The time dependency is necessary to have the neighborhoods change throughout the training (e.g. by shrinking them during later steps). During the training phase, for each point $x \in X$, the closest node n_c according to a distance (e.g. the Euclidean one) is identified as:

$$n_c = \underset{n_j}{\operatorname{argmin}} \|w_j - x\| \tag{1}$$

n_c will also be referred to as the "winning" node. The weights of the nodes in the neighborhood of n_c, $\{w_j \mid n_j \in N(n_c, t)\}$, are updated to better resemble x:

$$w_j(t+1) = w_j(t) + \alpha(t)\beta_{jc}(t)(x - w_j(t)) \tag{2}$$

where $\alpha(t)$ is the learning rate and $\beta_{jc}(t)$ is a coefficient that dampens the learning as the neurons get further away from the winner. By training the neighboring nodes as well as the winning one, a topological order forms among neurons, where nodes that model similar points are close to one another. An example of a 16×16 SOM trained with the MNIST dataset (see Section IV for more details) is shown in Figure 1. Each cell of the grid represents one of the nodes trained by the self-organizing map. The 784 pixels that represent each node (28×28, the resolution of each MNIST digit) are the weights learned. The grid shows how weights evolve to resemble samples of the dataset, with similar shapes being close to one another. An interesting example of how "similar" nodes end up close to one another is the following. The first row of Figure 1 has nodes that resemble the digits "7" and "1". "1"s that present a slant are placed closer to the "7"s because of their greater similarity.

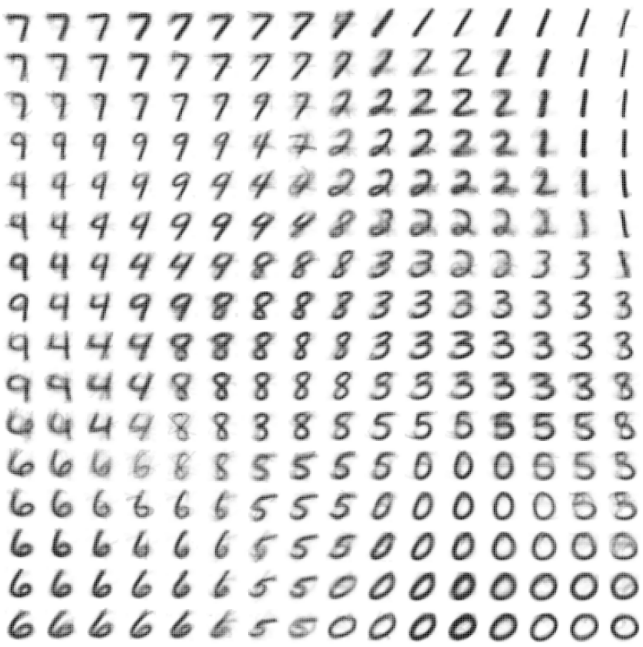
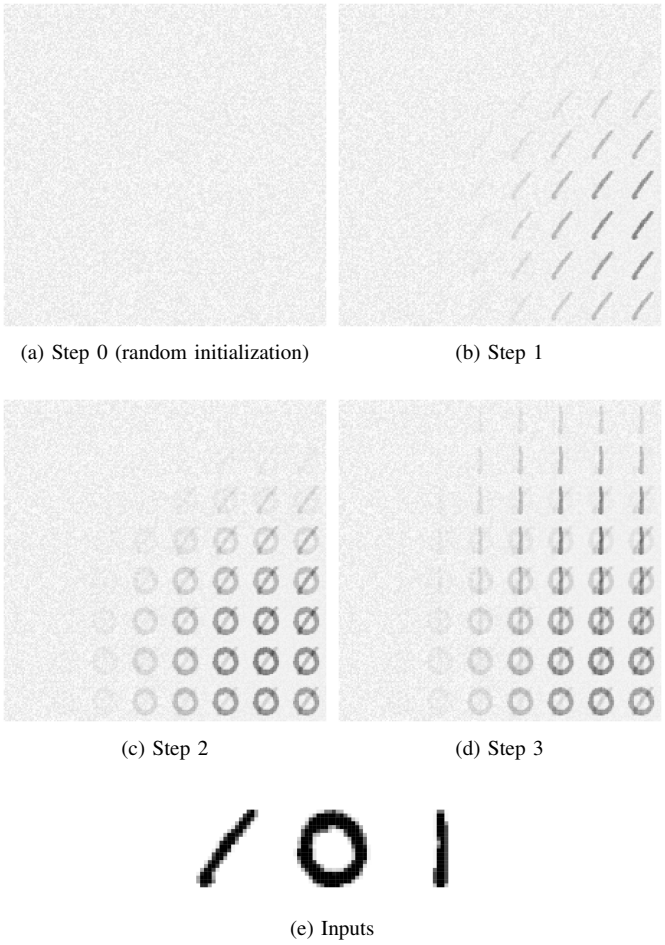


Fig. 1: 16×16 trained SOM



(a) Step 0 (random initialization)

(b) Step 1

(c) Step 2

(d) Step 3

(e) Inputs

Fig. 2: First three weight updates during the training of an 8×8 SOM

Figure 2 shows the first three training steps of a self-organizing map. Figure 2a shows the initial random weights. Figure 2b shows how the weights are updated after the left-most of the inputs in Figure 2e is used for the training. One of the random nodes has a slightly higher similarity to the input. The weights of this node are then updated (based on Equation 2) to better resemble the input, along with the weights of its neighbors. The same happens in Figures 2c and 2d for the second and third inputs.

During the training phase, for each of the points in the dataset, the distance from each node of the grid is computed. For a square $Q \times Q$ grid, MQ^2 operations are performed. This sets an upper bound to the maximum size of the SOM. Since having a larger number of nodes in a SOM helps better represent a dataset (and build more meaningful clusters), this limitation hinders the potential adoption of SOMs for processing very large datasets.

B. 2-Step Training Approach

To reduce the time complexity of SOM training, we propose to split the training phase in two steps. The first consists in training a smaller SOM, to produce a "coarse" grid with a low number of nodes that contains a coarse topology of the dataset. As a second step, this grid is extended to the final one with a larger number of nodes, to capture local topologies (e.g., sub-clusters that form within any of the coarse clusters). Hence, the initial effort of converging from the entire N-dimensional input space to its meaningful portions is done on a smaller number of nodes, thus requiring a significantly lower computational effort.

To achieve this model expansion for a $Q \times Q$ map, a single $P \times P$ (where $Q = nP, n \in \mathbb{N}$) SOM has been trained with a fraction η of the original dataset (Figure 3 shows one SOM with $P = 8$). Then, the trained nodes are replicated n^2 times in their locality, thus building a $Q \times Q$ SOM (as shown in Figure 4, where $n = 2$ and the replicated nodes are the ones from Figure 3). This SOM is then trained with the remaining $1-\eta$ portion of the dataset. This fine-tuning process is shown in Figure 5. With this 2-step training, only a fraction of the training requires working on Q^2 nodes.

The complexity of training a single $Q \times Q$ SOM, as already stated, is $\propto MQ^2$. Instead, building a single $P \times P$ map with a fraction η of the dataset is $\propto \eta MP^2$. If the small SOM is then replicated n^2 times to build a $Q \times Q$ SOM (with $Q = nP$) and the resulting grid is trained with the remainder of the dataset (i.e. $1 - \eta$), the required time is $\propto (1-\eta)Mn^2P^2$, resulting in a total training time $\propto (\eta(1-n^2) + n^2)MP^2$. In terms of complexity, both approaches are $\mathcal{O}(MP^2)$, but the proposed approach reduces the training time by a factor ρ, defined as:

$$\rho(n, \eta) = \frac{n^2}{\eta + n^2(1-\eta)} \tag{3}$$

Thus, it is evident that $\eta = 0$ is a special case in which the entire dataset is used to train the large SOM and none of it is used to train the small one. In this case $\rho(n, 0) = 1$, which is equivalent to training a SOM with the standard approach. The opposite edge case, $\eta = 1$, is the one where the entire dataset is used to train the smaller SOM, resulting in a time gain of n^2. In this case, no fine-tuning is done on the large SOM. In terms of performance (including training time), the resulting SOM is identical to a $P \times P$ one.

Additionally, for a fixed Q, a larger value of n implies a smaller P, thus leading to a faster training of the smaller SOM. The following limit holds:

$$\lim_{n \to +\infty} \rho(n, \eta) = \frac{1}{1-\eta} \tag{4}$$

This introduces a bound on the training time reduction that can be achieved with the proposed approach. This bound corresponds to the time gain that can be obtained by only training the larger SOM on the smaller fraction of data (i.e., a scenario in which the training time of the smaller SOM is negligible).

IV. Experimental results

In this section, standard SOMs (which will be considered as being the baseline) are compared to self-organizing maps

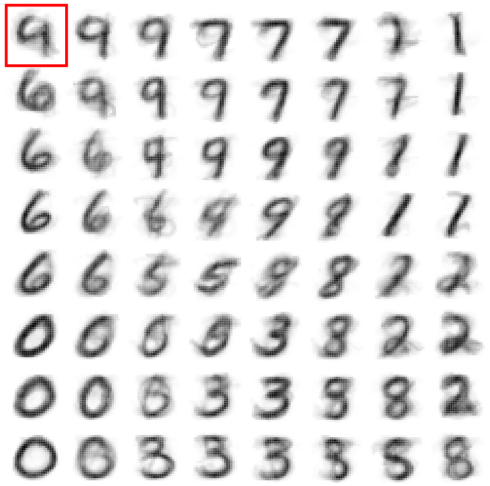


Fig. 3: An 8×8 SOM.

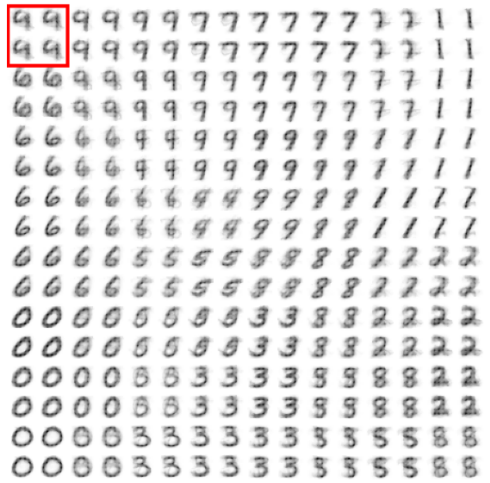


Fig. 4: A 16×16 SOM with replicated nodes from Figure 3. In red are the 4 nodes replicated from the single node highlighted in red in Figure 3

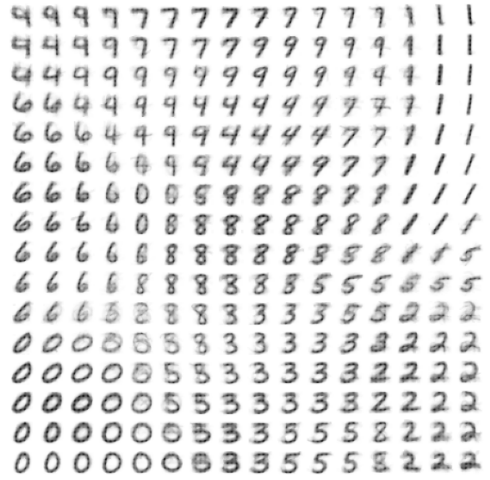


Fig. 5: A 16×16 SOM after the fine-tuning of the SOM from Figure 4

trained with the proposed 2-step approach. The comparisons are based on the following metrics:

- *Training time*: while Equation 3 already defines the theoretical time improvement, the actual training time is experimentally measured to verify its accordance with the expected result.
- *Quantization error*: SOMs can be used to quantize a set of points with the trained nodes. Given a dataset X, the quantization error q_e is defined as:

$$q_e = \frac{1}{|X|} \sum_{x \in X} \|x - w_c\| \tag{5}$$

  This is an indication of how well the map can represent a given dataset with a limited number of "buckets".
- *Accuracy*: given a labelled dataset, the trained SOM can be used as a classifier. After training, each (labelled) input in the training set is assigned to a node (the winning one for the given point). Each node is then assigned a class label based on the labels of the points that have been assigned to it (with majority voting). Next, new points can be assigned the label of the node that is closest to them. For this classifier, its accuracy (i.e., the fraction of correctly classified elements) can be computed.

Both quantization error and accuracy have been computed on a separate test set.

For each experiment, three different SOMs have been built: a baseline (i.e. the "standard" SOM) and two SOMs built with the proposed technique, respectively with $n = 2$ and $n = 7$.

The experiments have been performed on a machine running Ubuntu 16.04, equipped with an Intel Xeon X5650 (6 cores, 12 threads) @ 2.66 GHz and 32 GB of memory. The source code has been developed using Python 3.5, with the MiniSom library for the implementation of self-organizing maps and scikit-learn [11] for the comparisons with k-means (an implementation of the optimized algorithm presented in [12] has been used, unless otherwise stated). The main dataset used is MNIST [13], which is comprised of 28×28 hand-written digits from 0 to 9, divided into a training set of 60,000 entries and a test set of 10,000.

The presented experiments will cover various aspects of the proposed approach. First, analyses of the performance as η and Q vary are presented in Subsections IV-A and IV-B. Then, Subsection IV-C explores scenarios with smaller datasets. The baseline and the proposed approach are then compared to another clustering technique, k-means, in Subsection IV-D. Datasets other than MNIST are used in Subsection IV-E as a validation of the proposed approach and finally, in Subsection IV-F, the algorithms are run on larger datasets to assess their scalability.

A. *Effect of varying* η

η is the fraction of dataset used to build the small SOM. Hence, $1 - \eta$ is used to train the larger SOM.

Two special cases are given:

- $\eta = 0$. In this case, the entire dataset is used to build the large SOM. The training time is expected to be the same as the one for the baseline case.
- $\eta = 1$. In this case, the entire dataset is used to build the small SOM. The training time should be reduced by a factor of n^2 (as only the small SOM is trained). The performance in terms of accuracy and quantization error are expected to be the most degraded, since this corresponds to reducing the SOM size by a factor of n^2 (the final SOM will still have the expected number of nodes, but they will be repeated in groups of n^2).

The performance for $0 < \eta < 1$ are expected to be in between these two edge cases. Indeed, Figures 6, 7 and 8 show this behavior for 28×28 SOMs (this size has been chosen as it presents satisfactory results when trained with the baseline approach, as well as being suitable for $n \in \{2, 7\}$).

In terms of training time, the results behave as expected. The case $n = 7$ is faster than $n = 2$. The ratio of the time curves is also in accordance with Equation 3, when computing $\rho(7, \eta)/\rho(2, \eta)$ (i.e., the expected ratio of the training times for the two values of n, as a function of η).

In terms of accuracy and quantization error on the test set, a significant degradation occurs for $\eta > 0.8$. This happens because of the insufficient data for the training of the larger SOM. For this reason, the study of the performance as Q changes have been performed for $\eta = 0.8$ (in this case, based on Equation 4, the maximum time gain possible should be 5x).

B. *Effect of varying Q*

Varying Q alters the shape of the final SOM. It drives the time complexity with a squared factor, making the problem intractable for large values.

In this analysis Q ranges between 5 and 60. The selection of this range is based on a heuristics adopted in literature, recommending a number of nodes $\approx 5\sqrt{M}$ (M being the cardinality of the dataset) [14]. For MNIST, it corresponds, approximately, to a square map with $Q = 35$. Considering that, as mentioned in [15], a "trial-and-error" approach – if feasible – may identify the best size for the SOM, the range $5 \div 60$ has been used for more exhaustive results.

The most significant result is related to training time, as shown in Figure 9. The reduction in time with the proposed technique is in accordance with the one expected from Equation 3. In particular, $\rho(2, 0.8) = 2.5$, and the experimental training time gain is approximately 3x, while $\rho(7, 0.8) \approx 4.6$, the same as the rounded experimental result.

The second significant result involves quantization error (Figure 10) and accuracy (Figure 11). As expected, as Q grows, the quantization error reduces and the accuracy increases. It is particularly interesting, though, that the degradation of the performance for the proposed SOMs is particularly limited, if not negligible in some cases (on average, the degradation is smaller than 1% of the baseline). Hence, this effect, combined with the lowered training time, yields similar performance with a much lower computational effort or, with the same effort, it allows achieving better performance, by training a larger SOM.

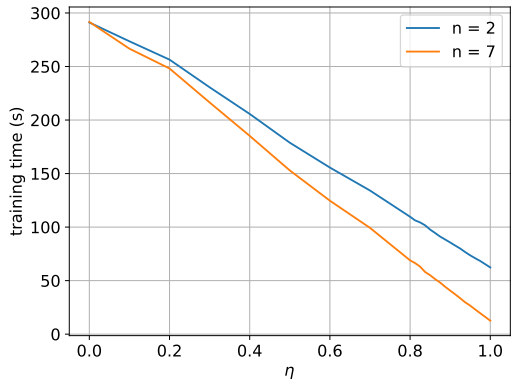


Fig. 6: η vs training time

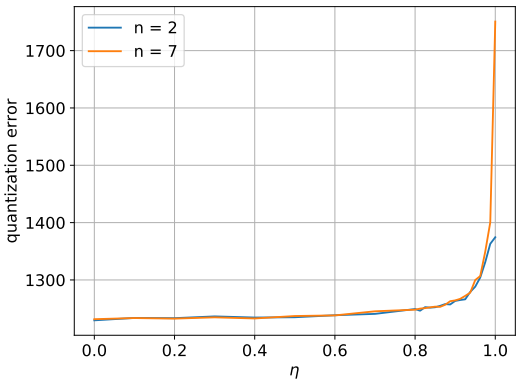


Fig. 7: η vs quantization error

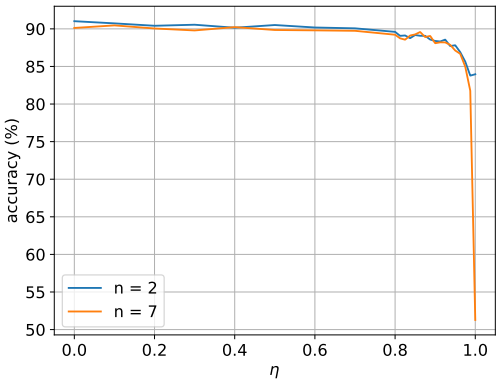


Fig. 8: η vs accuracy

C. *Behavior with low-cardinality datasets*

Intuitively, the 2-step approach leverages a large portion of data to "identify" the regions of the high-dimensional space where the data is distributed, displacing the centroids of the small SOM there. Then, the centroids are replicated and fine-tuned with the remainder of the data.

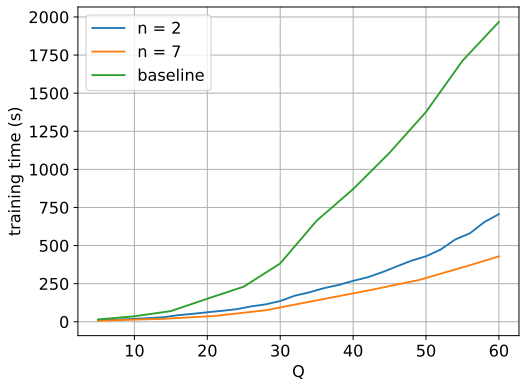


Fig. 9: Q vs training time

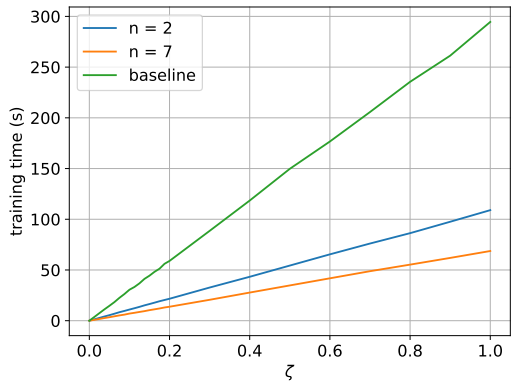


Fig. 12: ζ vs training time

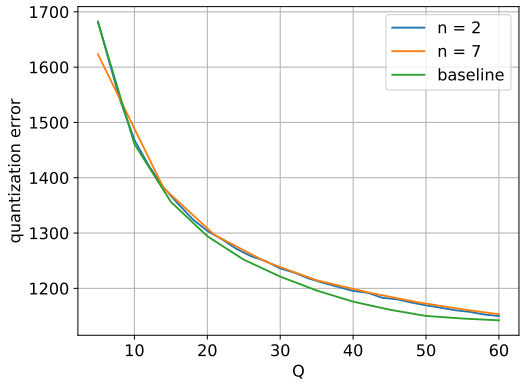


Fig. 10: Q vs quantization error

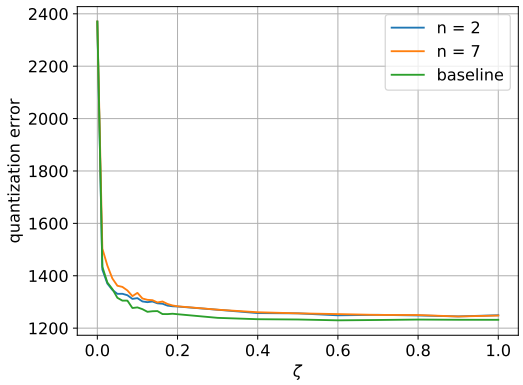


Fig. 13: ζ vs quantization error

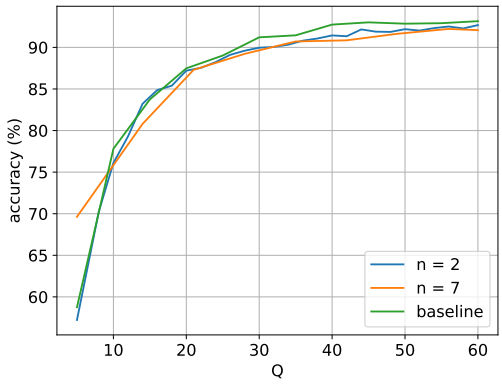


Fig. 11: Q vs accuracy

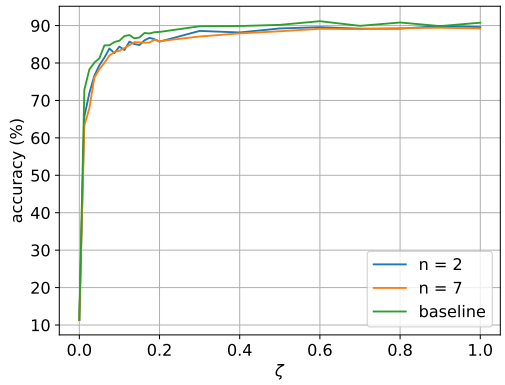


Fig. 14: ζ vs accuracy

As such, an important scenario to be studied is the one where only limited data is available. To further explore this case, a new kind of experiment has been devised. Here, Q and η have been fixed (to 28 and 0.8 respectively). Then, subsets of the entire dataset have been used to (1) train a standard SOM and (2) apply the proposed 2-step approach. These subsets used are fractions of the original dataset X, ranging from 0% (i.e. an "empty" training set) to 100% (i.e. the entire dataset) of X. Let ζ be the used fraction of X. Then, the proposed approach exploits a fraction $\zeta\eta$ of the dataset for the training of the small SOM, and $\zeta(1-\eta)$ for the large SOM.

The expected result in terms of training time is the same one as before, where the training of the 2-step approach is significantly faster than the baseline approach. In this case, since n and η are fixed, the training time is only linearly dependent on ζ. Indeed, Figure 12 shows this linear dependence.

In terms of accuracy and quantization error, the performance of the baseline and of the proposed approach are expected to

be similar as ζ varies. A divergence between the curves would imply that the 2-step model only provides satisfactory results because the fraction $1-\eta$ of data used for the final fine-tuning is still sufficient to learn a model from scratch (thus making the proposed approach no better than a random initialization). As shown in Figures 13 and 14, though, there only is a slight, constant performance degradation between the two models. This is evidence of the fact that there is indeed an advantage in using the 2-step approach that is not only due to the dataset size.

D. Comparison with k-means

To understand how the proposed SOM approach fares in absolute terms, a performance comparison with other state-of-the-art unsupervised techniques is needed. Of the many unsupervised algorithms, k-means [16] is one of the most relevant for this comparison, because of the many traits it shares with self-organizing maps. Indeed, both algorithms rely on the definition of the number of clusters to be identified (k for k-means, Q^2 for SOMs) and both introduce an iterative approach where centroids are identified by repeated adjustments.

Self-organizing maps have an advantage over other clustering algorithms: The output of SOMs is a 2-dimensional grid in which nodes are distributed based on the similarity among one another. This is not the case with k-means, which instead provides points in a (possibly) high-dimensional space, with no relationships between one another. The output of self-organizing maps proves particularly useful when used in combination with visualization techniques for interpreting the results (instead of reducing the dimensionality of the results through other typically computationally expensive algorithms, such as PCA or t-SNE), or as a pre-processing step where the points are distributed into buckets that have a concept of neighborhood and of distance (e.g. Manhattan) in a low-dimensional space.

Comparisons between various unsupervised algorithms have already been done extensively in literature. One study in particular [17] is focused on comparing self-organizing maps and k-means. The results presented in that work highlight how, on the datasets used for the study, k-means has a slight performance boost when compared to SOMs. As such, we expect this to be the case for the proposed 2-step SOM.

The performance of the three algorithms (k-means, standard and 2-step SOM) are compared in terms of training time, quantization error and accuracy. As expected, k-means performs better than either SOMs in terms of quantization error and accuracy, as shown in Figures 16 and 17. In particular, k-means is approximately a constant 2-3% more accurate than SOMs for larger values of Q, completely in accordance with the results presented in [17].

In terms of training time, instead, two important clarifications needs to be made.

- Most implementations of k-means are executed multiple times, each with a different initialization. This is done to help avoid local minima solutions. To present a fair comparison, the k-means results presented are all based on a single initialization. The results in terms of accuracy and quantization error do not vary significantly when using multiple initializations, but the training time for P initializations is clearly P times that of a single initialization
- The adopted k-means implementation parallelizes part of the workload, by distributing it across 12 concurrent workers. The adopted SOM library, on the other hand, runs a single worker. Running k-means without paral-

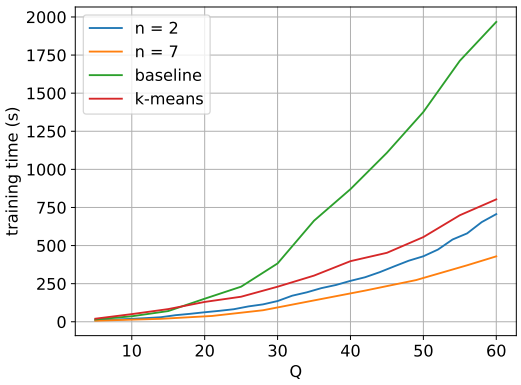


Fig. 15: Training time for k-means and SOMs

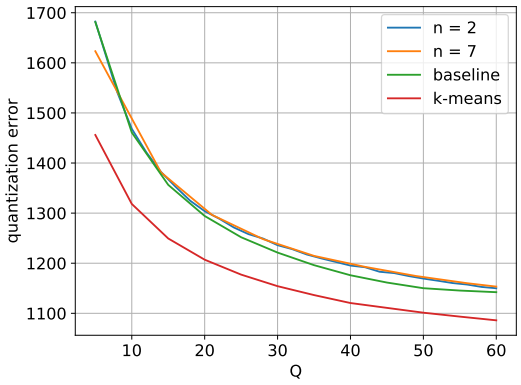


Fig. 16: Quantization error for k-means and SOMs

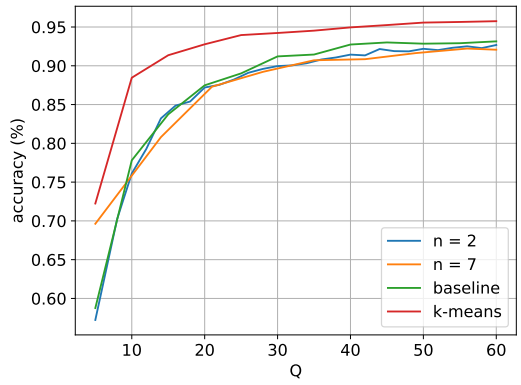


Fig. 17: Accuracy for k-means and SOMs

lelization leads to heavily slower training times when compared to self-organizing maps (so much so that a comparison between the them would be meaningless). Hence, only the parallel training times are reported for k-means.

Figure 15 shows the training time for the different algorithms. It is particularly interesting that, with a standard implementation, self-organizing maps are slower than the parallelized version of k-means. This, paired with the slight underperformance in terms of other metrics, made SOMs not competitive with other unsupervised algorithms.

The proposed 2-step approach, on the other hand, is faster than k-means. Additionally, as already mentioned, SOMs introduce a level of interpretability that is missing from k-means and other unsupervised techniques. While this interpretability comes with a slight degradation in performance, this trade-off is still useful for those domains where data mining techniques are used in human-in-the-loop scenarios.

E. Additional datasets

To validate the proposed technique, the 2-step training approach has been also applied to the following datasets (in addition to MNIST):

- *Fashion-MNIST* [18]. A dataset of fashion items (e.g. shirts, trousers, bags) that maintains the same image size (28×28 grayscale), the same train-test splits (60,000 training points, 10,000 test points) and the same number of classes (ten) as MNIST. This is intended as a more challenging alternative to MNIST
- *CIFAR-10* [19]. A collection of 32×32 color images divided into 10 categories, with a training set of 50,000 samples and a test set of 10,000
- *Character Trajectories* [20]. A UCI dataset that contains a collection of multiple, labelled samples of pen tip trajectories recorded while writing individual characters. For each character, three time series are collected: the pen tip velocity for the x and y axes, and the pressure applied to the pen tip. The dataset is available in a normalized form, but with various time series lengths (depending on how long it took to draw the character). For uniformity, all signals have been upsampled to match the longest time series available (205 samples). The training/test split ratio has been set to 6/1 (i.e. the same used for MNIST). The dataset contains the data for 2,858 hand-written characters, each belonging to one of 20 classes. Given the low number of rows for this datasets, all algorithms have iterated 10 times over the training set

The results obtained and the SOM configurations are displayed in Table I. The results obtained on these additional datasets are consistent with the ones presented for MNIST. A significant (up to 8x for $(\eta, n) = (0.9, 7)$) training time speed-up occurs for the 2-step approach, at the cost of a slight degradation in performance. In terms of accuracy, this degradation is $\approx 1\%$ for all cases, with the exception of Character Trajectories. For that dataset, the performance degradation is slightly more impactful – 2 to 4% less than the baseline case. Considering that the test set for this dataset is comprised of only 409 samples, every misprediction has a more significant impact on the performance degradation compared to the other datasets. Similar considerations can be made in terms of quantization error.

F. Algorithm scalability

To assess the time gain of the proposed approach on larger scales, two synthetic dataset have been used. These have been generated using scikit-learn's `make_blobs` functionality, which generates clusters based on a Gaussian distribution of values. Dataset 500k includes 500,000 entries, each in a 500-dimensional space, with a total of 50 separate clusters. Dataset 1M, instead, contains 1,000,000 entries in a 1,000-dimensional space, with 100 clusters.

The training times have been measured for $Q \in \{50, 100\}$ for the baseline approach, for k-means and for the 2-step approach with $(\eta, n) = (0.8, 5)$. Considering that all models successfully identified the clusters in the datasets (i.e. all accuracies are 1), only the results in terms of training time are proposed in Table II.

All 2-step versions have a training speed-up in accordance with $\rho(5, 0.8) \approx 4.3$ (on average, the experimental factor is of 4.5, with a standard deviation of 0.1). For both datasets, scikit-learn's version of k-means that implements [12] runs out of memory before completion. Hence, scikit-learn's version based on [16] has been used instead. This introduces an impactful increase in training time. This makes k-means even slower than standard self-organizing maps: for Dataset 1M and $Q = 100$, the execution of k-means was stopped after 3 days, while the baseline SOM finishes in less than 35 hours and the 2-step approach in less than 8. This makes SOMs (and particularly with the proposed approach) an attractive alternative for very large datasets.

V. Discussion

Differently from other pre-training methods presented in the related work, the 2-step approach is based on the training of two standard self-organizing maps, without other time-consuming operations involved. This means that existing works on the parallelized training of SOMs (e.g. [9]) can be applied to the training of the two SOMs of our approach for a further training time reduction. This reduction is proportional to the number of parallel workers. In [9], assuming that the training of a $Q \times Q$ SOM is distributed onto m^2 parallel workers, the training time for the entire SOM will be $\propto MQ^2/m^2$, since m^2 smaller SOMs are being trained simultaneously. This reduces the training time of a large SOM down to the time needed to train a SOM that is $1/m^2$ of the original one.

The only additional time the 2-step approach (with $n = m$) requires, compared to the distributed approach, is the time it takes to train a large SOM with a small (20%, for $\eta = 0.8$) fraction of the dataset. Hence, the 2-step approach, which runs on a single worker, may reach comparable training times

Dataset	Method	Q	η	n	Training time (s)	Quantization error	Accuracy (%)
MNIST	baseline	28	-	-	323.13	1232.71	90.29
	k-means	28	-	-	202.92	1162.21	94.17
	2-step	28	0.7	2	136.16	1242.02	89.71
	2-step	28	0.8	2	116.49	1247.04	88.96
	2-step	28	0.9	2	87.95	1262.59	89.02
	2-step	28	0.7	7	103.79	1243.75	89.57
	2-step	28	0.8	7	73.58	1245.86	89.39
	2-step	28	0.9	7	41.38	1263.18	89.07
Fashion-MNIST	baseline	28	-	-	312.01	1031.78	77.93
	k-means	28	-	-	178.32	976.97	80.21
	2-step	28	0.7	2	139.05	1037.45	77.75
	2-step	28	0.8	2	111.13	1042.90	77.51
	2-step	28	0.9	2	87.79	1053.60	77.20
	2-step	28	0.7	7	102.89	1039.47	77.50
	2-step	28	0.8	7	72.62	1044.39	77.98
	2-step	28	0.9	7	41.71	1053.46	77.32
CIFAR-10	baseline	28	-	-	1464.72	2394.83	32.88
	k-means	28	-	-	568.33	2335.88	35.16
	2-step	28	0.7	2	640.77	2402.47	32.20
	2-step	28	0.8	2	516.65	2407.27	32.12
	2-step	28	0.9	2	373.68	2418.99	31.56
	2-step	28	0.7	7	474.76	2403.36	32.12
	2-step	28	0.8	7	322.18	2406.28	32.19
	2-step	28	0.9	7	171.98	2421.23	31.94
Character Trajectories	baseline	28	-	-	92.45	5.51	94.87
	k-means	28	-	-	38.20	5.45	97.56
	2-step	28	0.7	2	42.81	5.93	92.67
	2-step	28	0.8	2	35.44	6.25	91.20
	2-step	28	0.9	2	28.28	6.68	90.71
	2-step	28	0.7	7	30.84	5.94	92.67
	2-step	28	0.8	7	22.36	6.27	92.42
	2-step	28	0.9	7	13.10	6.73	90.22

TABLE I: Performance on multiple datasets

Dataset	Method	Q	η	n	Training time (hh:mm:ss)
Dataset 500k	baseline	50	-	-	01:50:58
	k-means	50	-	-	04:25:36
	2-step	50	0.8	5	**00:25:14**
	baseline	100	-	-	08:47:22
	k-means	100	-	-	16:52:35
	2-step	100	0.8	5	**01:54:45**
Dataset 1M	baseline	50	-	-	07:30:49
	k-means	50	-	-	23:54:43
	2-step	50	0.8	5	**01:40:46**
	baseline	100	-	-	34:27:02
	k-means	100	-	-	Stopped after 72h
	2-step	100	0.8	5	**07:26:50**

TABLE II: Performance on Dataset 500k and Dataset 1M

with respect to the distributed approach in [9] requiring m^2 workers.

Finally, we observe that works such as [7] and [8] both rely on an initial k-means for the initialization of the weights in the SOM. Based on the time-wise comparison presented in Subsection IV-D, the 2-step approach has been shown to be faster than k-means. As such, using k-means as a pre-processing step, only to perform further fine-tuning on the obtained result is bound to have a training time higher than that of 2-step self-organizing maps.

VI. Conclusions

In this paper, we presented a 2-step training approach for self-organizing maps. With this approach, a large portion of the training set is used to initialize a small self-organizing map, while the rest of the dataset is used to fine-tune a larger SOM obtained from replicating the small SOM's nodes.

It has been shown that this approach has slightly worse performance (in terms of accuracy and quantization error) when compared to a standard self-organizing map. Despite that, the training time of the proposed approach is significantly (approximately 2x to 8x, based on the experimental results) faster than the standard (baseline) approach. Larger datasets have been adopted to demonstrate how the 2-step training can make larger problems tractable.

All self-organizing maps perform slightly worse than k-means in terms of accuracy and quantization error. However, the 2-step approach has a significantly lower training time than k-means, making it a valid choice for larger problems. Additionally, the lower performance w.r.t. k-means is compensated by a higher degree of interpretability. To obtain similarly interpretable results with other clustering algorithms requires the application of additional techniques, with heavy computational overheads. Hence, self-organizing maps can be considered a well-performing, lighter and more interpretable alternative to other unsupervised techniques.

VII. Acknowledgements

This work has been partially supported by the Smart-Data@PoliTO center on Big Data and Data Science.

References

[1] I.-T. Chen, L.-C. Chang, and F.-J. Chang, "Exploring the spatio-temporal interrelation between groundwater and surface water by using the self-organizing maps," *Journal of Hydrology*, vol. 556, pp. 131–142, 2018.

[2] R. Buchs, C. Davis, D. Gruen, J. DeRose, A. Alarcon, G. Bernstein, C. Sánchez, J. Myles, A. Roodman, S. Allen, *et al.*, "Phenotypic redshifts with self-organizing maps: A novel method to characterize redshift distributions of source galaxies for weak lensing," *arXiv preprint arXiv:1901.05005*, 2019.

[3] F. L. Kriegel, R. Köhler, J. Bayat-Sarmadi, S. Bayerl, A. E. Hauser, R. Niesner, A. Luch, and Z. Cseresnyes, "Cell shape characterization and classification with discrete fourier transforms and self-organizing maps," *Cytometry Part A*, vol. 93, no. 3, pp. 323–333, 2018.

[4] E. Oliver, I. Vallés-erez, R.-M. Baños, A. Cebolla, C. Botella, and E. Soria-Olivas, "Visual data mining with self-organizing maps for "self-monitoring" data analysis," *Sociological Methods & Research*, vol. 47, no. 3, pp. 492–506, 2018.

[5] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[6] A. Ciampi and Y. Lechevallier, "Clustering large, multi-level data sets: an approach based on kohonen self organizing maps," in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 353–358, Springer, 2000.

[7] M.-C. Su and H.-T. Chang, "Fast self-organizing feature map algorithm," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 721–733, 2000.

[8] M.-C. Su, T.-K. Liu, and H.-T. Chang, "Improving the self-organizing feature map algorithm using an efficient initialization scheme," , vol. 5, no. 1, pp. 35–48, 2002.

[9] N. Bandeira, V. Lobo, and F. Moura-Pires, "Training a self-organizing map distributed on a pvm network," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, vol. 1, pp. 457–461, IEEE, 1998.

[10] D. Garabato, C. Dafonte, M. Manteiga, D. Fustes, M. A. Álvarez, and B. Arcay, "A distributed learning algorithm for self-organizing maps intended for outlier analysis in the gaia–esa mission," in *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*, Atlantis Press, 2015.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 147–153, 2003.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Som toolbox for matlab 5," *Helsinki University of Technology, Finland*, vol. 109, 2000.

[15] T. Kohonen, "Matlab implementations and applications of the self-organizing map," *Unigrafia Oy, Helsinki, Finland*, 2014.

[16] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[17] U. A. Kumar and Y. Dhamija, "Comparative analysis of som neural network with k-means clustering algorithm," in *2010 IEEE International Conference on Management of Innovation & Technology*, pp. 55–59, IEEE, 2010.

[18] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[19] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[20] D. Dua and C. Graff, "UCI machine learning repository," 2017.