Semi-Supervised Self-Organizing Maps with Time-Varying Structures for Clustering and Classification

Pedro H. M. Braga and Hansenclever F. Bassani
Centro de Inforatica (CIn)
Universidade Federal de Pernambuco (UFPE)
Recife, PE, Brazil
{phmb4, hfb}@cin.ufpe.br

***Abstract*—In recent years, the advances in technology have produced datasets of increasing size, not only regarding the number of samples but also the number of features. Unfortunately, creating a sufficiently large amount of adequately labeled data with enough examples for each class is not easy. Labeling is a challenging, expensive, and time-consuming task. It is usually done manually, which may contribute to the insertion of noise and errors in the data. Hence, it is of great importance to put forward intelligent models that can benefit from the distinct information that both labeled and unlabeled data can provide, since, for many applications, there is a plentiful amount of unlabeled data, but insufficient labeled ones. Semi-Supervised Learning (SSL) is employed to achieve this. It is halfway between supervised and unsupervised learning. In this sense, we highlight two very influential models: Self-Organizing Maps (SOM) and Learning Vector Quantization (LVQ). SOM is a biologically inspired neural model that uses unsupervised and incremental learning to produce prototypes of the input data, whereas the LVQ can be seen as its supervised counterpart. The unsupervised characteristic of SOM makes it unfeasible to execute SSL. In that way, the current work proposes new models that incorporate standard concepts from LVQ to the SOM algorithm to build semi-supervised approaches. Such proposals can dynamically switch between the two types of learning at training time, according to the availability of labels and automatically adjust themselves to the local variance observed in each cluster. The experimental results show that the proposed models can surpass the performance of other traditional methods not only in terms of classification but also regarding clustering quality. It also enhances the range of applications of SOM and LVQ-based models by combining them with Deep Learning in a synergic way to allow dealing with complex data structures, such as images and sound. Moreover, we explore forms of learning good representations of the input data, and manners to estimate the unsupervised error when no labels are provided. Our approaches demonstrated to be good at producing a meaningful topology and clustering prototypes that appropriately represent the data.**

***Index Terms*—self-organizing maps, semi-supervised learning, unsupervised learning, clustering, classification.**

- Student level: MSc
- Date of conclusion: February 26, 2019

The content of this paper is partially adapted from our previously published papers [1]–[4], as well as from the master's dissertation [5]. Moreover, the authors would like to thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), Brazil, for financing this research study with a scholarship, and the NVIDIA Corporation for the GPU Grant of a Titan V.

- Examining board members
  - Prof. Dr. Francisco Madeiro Bernardino Junior
    Universidade de Pernambuco (UPE)
  - Prof. Dr. Cleber Zanchettin
    Universidade Federal de Pernambuco (CIn-UFPE)
  - Prof. Dr. Hansenclever Bassani **[Advisor]**
    Universidade Federal de Pernambuco (CIn-UFPE)
- On-line version of the full dissertation:
  https://repositorio.ufpe.br/handle/123456789/33484
- Publications:
  - P. H. M. Braga and H. F. Bassani, "**A semi-supervised self-organizing map for clustering and classification**," in Proceedings of the International Joint Conference on Neural Networks (IJCNN). IEEE, 2018. Available at **https://ieeexplore.ieee.org/abstract/document/8489675**
  - P. H. M. Braga and H. F. Bassani, "**A Semi-Supervised Self-Organizing Map with Adaptive Local Thresholds**," in Proceedings of the International Joint Conference on Neural Networks (IJCNN). IEEE, 2019. Available at **https://ieeexplore.ieee.org/document/8851839**
  - P. H. M. Braga, H. R. Medeiros and H. F. Bassani, "**Backpropagating the unsupervised error of self-organizing maps to deep neural networks**," in LatinX in AI Research Workshop at NeurIPS 2019. Available at **www.researchgate.net/publication/340846247**
  - P. H. Braga, H. R. Medeiros and H. F. Bassani, "**Deep Categorization with Semi-Supervised Self-Organizing Maps**," in Proceedings of the International Joint Conference on Neural Networks (IJCNN). IEEE, 2020. [Accepted for Publication - arXiv Preprint]: **https://arxiv.org/abs/2006.13682**

I. Introduction

Over the last few years, the use of machine-learning technology has driven many aspects of modern society. Recent research on Artificial Neural Networks with supervised learning has shown great advances. It is the most common form

of machine learning [6]. A key to the success of supervised learning, especially deep supervised learning, is the availability of sufficiently large amounts of labeled training data. Unfortunately, creating such properly labeled data with enough examples for each class is not easy. As a result, the use of supervised learning methods became impractical in many applications, such as in the medical field, where it is extremely difficult and expensive to obtain balanced labeled data. In other areas, such as robotics, the dynamic imposed makes it impossible to have real-time labels. Also, in certain problems, new categories of elements may frequently arise, making it infeasible to create a comprehensive previously labeled training dataset.

On the other hand, due to the advances in technology that have produced datasets of increasing size, in terms of the number of samples and features, unlabeled data usually can be easily obtained. In this sense, unsupervised learning can be applied to perform clustering tasks. However, clustering is a more challenging problem, and the nature of the data can make it even more difficult. So, any kind of additional prior information with respect to the data can be useful to obtain a better performance. Therefore, it is of great importance to put forward methods that can combine both types of data in order to benefit from the information they can provide, each of them in their way [7]. To do so, Semi-Supervised Learning (SSL) is typically applied. It is halfway between supervised and unsupervised learning and can be used to both clustering and classification tasks [8].

In this regard, we highlight two very influential models proposed by Kohonen: Self-Organizing Map (SOM) [9] and Learning Vector Quantization (LVQ) [10]. SOM is a biologically inspired neural model that uses unsupervised and incremental learning to produce prototypes of the input data. It maps data from a higher-dimensional input space to a lower-dimensional output space, while preserving the similarities and the topological relations found between points in the input space. It can create abstractions and provide a simplified way of exhibiting information. The LVQ shares many similarities with SOM. Thus, it can be seen as its supervised counterpart. Therefore, these methods are good candidates for developing a hybrid approach that combines their concepts to allow SSL.

Various modifications of SOM and LVQ were proposed to improve their performance in more challenging datasets. For instance, high-dimensional data poses different challenges for clustering and classification tasks. In particular, traditional similarity measures may become meaningless due to the curse of dimensionality [11], in which objects may appear approximately equidistant from each other, which is aggravated by the presence of irrelevant dimensions. Hence, such problems require more sophisticated approaches. To do so, SOM and LVQ-based methods [12], [13] usually apply weights to the input dimensions, which has shown to provide outstanding performance. It involves not only the clustering itself but also the identification of the relevant subsets of the input dimensions, also known as Subspace Clustering [14].

Still, clustering high-dimensional raw data such as image and sound is a hard task, and the traditional prototype-based methods are not suitable for it. However, techniques based on Deep Learning (DL) have been very successful in yielding good high-level representations [15]. For instance, [16] showed that representations built by Convolutional Neural Networks (CNN) are better than the state-of-art handcrafted features, and [17] demonstrated that the representations learned by GoogleLeNet could be used for the task of Unsupervised Visual Object Recognition (UVOC), achieving about 75-90% of agreement with labels assigned by humans in an unseen dataset, when fed as input to a SOM-based clustering method. Moreover, in a purely unsupervised scenario, some of the most successful approaches for producing representations are Autoencoders (AE), Variational Autoencoders (VAE), and Generative Adversarial Networks (GAN). Therefore, those techniques can be applied in different ways, such as with self-labeling [18], or Deep Clustering (DC) [19].

Considering what has been set out, this research project proposes new Semi-Supervised models based on the concepts of both SOM and LVQ in order to improve the results obtained with traditional SSL methods in the literature. Moreover, we extend the application range of the models by scaling them to a variety of deep learning tasks, considering not only the SSL scenario but also the purely unsupervised perspective. This combination with DL is performed in a synergic way to allow dealing with complex data structures. We also explore forms of learning good representations of the input data, and ways to estimate the unsupervised error when no labels are provided. The results demonstrate our proposal to be good at surpassing its competitors in several scenarios, producing meaningful topologically ordered clustering prototypes that appropriately represent the data.

The rest of this article is organized as follows: Section II presents essential theoretical concepts and introduces related work. Section III describes in detail the first proposed model as well as its experiments and results. Section II presents an extension and novel approach to the first model that scales it to a broader range of tasks. Section V introduces a detailed explanation of the last proposed model, together with experiments, analysis, and results. Finally, Section VI concludes this work by analyzing the obtained results and indicating future directions and practical applications.

II. Background

A. Self-Organizing Maps

Self-Organizing Map (SOM), proposed by Kohonen [9], is an unsupervised neural network that maps data from a higher-dimensional input space to a lower-dimensional output space while preserving the similarities and the topological relations found between points in the input space. SOM is based on three essential processes: 1) Competition; 2) Cooperation; and 3) Adaptation, which leads to a *competitive learning*, where the neurons compete among themselves to be the most activated when an input pattern is presented. The competition results in a process that is called a *winner-takes-all* competition, which produces just one *winning neuron*. In a

SOM, the neurons are placed at the vertices of a *lattice* or grid that is commonly one or two-dimensional. Therefore, SOM is characterized by the formation of a topographic map of the input patterns, in which the spatial locations of the neurons (prototypes) in the grid are indicative of intrinsic statistical features contained in the input patterns [20]. These prototypes can be seen as abstractions of the data and a simplified way of exhibiting information.

An interesting fact about the development of SOM is its neurophysiological inspiration, in particular, in a distinct feature of the brain of primates: it is known that in certain cortical regions, neural activity is organized in such a way that topologically ordered computational maps represent different sensory inputs. [21]. It comes from both anatomical and physiological evidence of lateral interaction between cells: 1) in neural tissues, an activated neuron that triggers a pulse causes a short-range excitation of other neurons that ranges from 50 to 100 μm; 2) the propagation of the excitation to areas not related to the excitatory process is prevented by a penumbra of inhibitory activity around the exciting area; and 3) a weaker excitatory action surrounds the inhibitory penumbra and ranges up to several centimeters of radius. SOM captures the essential features of these maps and yet remains computationally tractable [21].

The basic structure of a SOM (Figure 1) consists of an input layer and an output layer. The input layer receives the sensory inputs from the environment and propagates them to the output layer. Let m denote the dimension of the input space and $\boldsymbol{x} = [x_1, x_2, ..., x_m]^T$ the input pattern vector. Let the synaptic weight vector of neuron j be denoted as $\boldsymbol{w}_j = \left[w_{j1}, w_{j2}, ..., w_{jm}\right]^T$, with $j = 1, 2, ..., l$, where l is the total number of neurons in the output map. The output layer computes the final map resulting from the self-organization process, and its topology usually is a two-dimensional *lattice*, where each node is connected with their immediate neighbors.

The Self-Organization in SOM occurs when an input pattern $\boldsymbol{x}$ is presented to the input layer, triggering the competition process in order to choose the winner node j. Then, the map adapts itself by moving the synaptic-weight vector $\boldsymbol{w}_j$ towards the input pattern. Moreover, it is crucial to the self-organization process that the nodes are not affected independently of each other, but as topologically related subsets. This is done in a cooperation step by also moving local neighbors of j to get closer to the input pattern, with a lower intensity.

B. Learning Vector Quantization

The Learning Vector Quantization (LVQ), also proposed by Kohonen [10], is a family of algorithms for statistical pattern classification that uses prototypes (codebook vectors) to represent class regions. These regions are defined by hyperplanes between prototypes, resulting in Voronoi partitions. While the basic SOM is unsupervised, the LVQ is characterized by supervised learning. Also, unlike in SOM, no neighborhoods around the winner are defined during the learning process, nor any spatial order of the prototypes is expected. Since LVQ was meant to be strictly for statistical classification and recognition, its only aim is to define class regions in the input space [10].

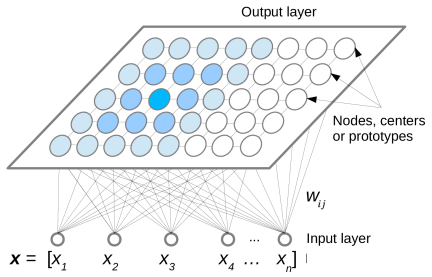


Fig. 1: The basic structure of a SOM. The units $\boldsymbol{x}$ are the input pattern. Each synaptic-weight vector $\boldsymbol{w}_{ij}$ represents a connection between the *i*-th node in the input layer and the *j*-th node in the output layer. Each node in the output layer is directly connected with its neighbors (Adapted from [22]).

Basically, the LVQ classification scheme is based on *winner-takes-all* strategy, as in SOM. LVQ iteratively tries to improve some initial set of P prototypes, which are characterized by $W = \left(\boldsymbol{w}_j, c_j\right), j \in 1, ..., P$, where $\boldsymbol{w}_j$ is m-dimensional, and $c_j \in \{1, ..., C\}$ its class label. Similarly to SOM, the winner prototype is selected as the closest one to the input pattern. The learning process aims to determine the weight vectors in a way that the training data are mapped to their corresponding class label region. Therefore, LVQ can be seen as the supervised counterpart of SOM.

C. Semi-Supervised Learning

In the past years, there has been a growing interest in a hybrid setting, referred to as Semi-Supervised Learning (SSL). SSL is a combination of supervised and unsupervised learning. The basic idea is to take advantage of both labeled and unlabeled data during the training, combining them to improve performance [7], [8], [23].

SSL can be further divided into semi-supervised classification and semi-supervised clustering [8]. Firstly, in the semi-supervised classification, the training set is given in two parts: $S = \{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathbb{R}^D, y_i \in Y, 1 \leq i \leq M\}$ and $U = \{\mathbf{u}_i \in \mathbb{R}^D | i = 1, \cdots, M\}$, where S and U are the labeled and unlabeled data, respectively. At first hand, it is possible to consider a traditional supervised scenario using just S. However, the unsupervised estimation of the probability function $p(\boldsymbol{x})$ of the input set can take advantage of both S and U to achieve better outcomes [8]. Many semi-supervised classification algorithms have been developed in the past decades, and, according to [23], they can be categorized as 1) Self-training, 2) SSL with generative models, 3) SSL with Graphs, 4) SSL with Committees, and 5) Semi-supervised Support Vector Machines (S^3VM), or Transductive SVM.

Secondly, in the semi-supervised clustering, the aim is to group the data in an unknown number of groups relying on some kind of similarity or distance measures in combination with objective functions. Clustering is a more challenging

problem, and the nature of the data can make it even more difficult. So, any kind of prior information with respect to the data could be integrated into the learning process to obtain a better performance. For instance, a subset of labeled data, or constraints on pairs of patterns in form of *must-link* and *cannot-link* [8], [23]. Prototype-based models (e.g., k-means, and SOM), Hidden Markov Random Fields (HMRFs), Expectation Maximization (EM), Label Propagation (LP), and Label Spreading (LS) are examples that have been successful applied in this area [8].

Moreover, it is also worth pointing out that the interest for SSL is growing in machine learning alongside with the DL context, as it is possible to see in [24]. Also, the term Deep Semi-Supervised Learning (DSSL) is often employed to express DL methods applicable to SSL.

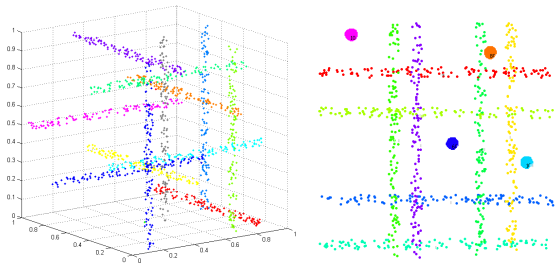
D. High-Dimensional Data and Subspace Clustering

High-dimensional data poses different challenges for clustering tasks. In particular, similarity measures used in traditional clustering techniques may become meaningless due to the curse of dimensionality [11]. The presence of irrelevance features strongly influences the appearance of clusters. Different subsets of features may be relevant for distinct clusters, and different correlations among the features may be relevant for different clusters. This phenomenon is called local feature relevance or local feature correlation [14].

A very common premise to reduce the infinite search space of all possible subspaces is to consider axis-parallel subspaces only. To illustrate this, Figure 2a displays a simulated dataset with three dimensions, in which there are 12 clusters. Note that, for each cluster, one of the three dimensions has the data points spreading among its whole domain. Thus, such a dimension is irrelevant for the clustering. Figure 2b is a 2D projection concerning only two dimensions. In this example, the data presents points of small variation (the dots), determining the clusters. In such a dataset, none of the three dimensions can be removed without losing relevant information for 8 out of 12 clusters. So, this task involves not only the clustering itself but also identifying relevant subsets in the input dimensions for each cluster [14]. One way to achieve this is by applying local relevances to the input dimensions.

E. Variants of SOM and LVQ

Various modifications of SOM and LVQ have been proposed to improve their performance in more challenging high-dimensional datasets. An important aspect to consider in the original SOM and in some of its variants is the fixed topology. It usually requires a deep understanding of the data, and may not adequately represent clusters that live in different subspaces [12]. This issue has been addressed by SOM-based models that present a time-varying structure, as in Local Adaptive Receptive Field Self-Organizing Map (LARFSOM) [25] and Local Adaptive Receptive Field Dimension Selective Self-organizing Map (LARFDSSOM) [26]. These maps learn the topology during the training and try to determine an optimum arrangement. This approach relies on an incremental and robust learning process, where not only the number of nodes but also the connections between them must be learned.



(a) Example of a 3D dataset (b) 2D Projection

Fig. 2: Subspace Clustering Problems (Adapted from [12])

Some influential examples can also be found in the supervised context. For instance, the Generalized Relevance Learning Vector Quantization (GRLVQ) is proposed to be used in high-dimensional real-world datasets as the weighting factors allow the model to approximately determine the intrinsic data dimensionality, i.e., the relevances identify the irrelevant dimensions. It is based on the LVQ, and discriminates the influence of different components of the input, increasing or decreasing its relevance at training time [13], so forth attenuating the curse of dimensionality.

F. Representation Learning and Self-Organizing Maps

Previous research has shown the potential that Deep Neural Networks (DNN) have in building representations that are useful not only for performing the task in which the network was trained for but also for correlated ones that take data from similar input distributions [16], [17].

Some of the most successful approaches for producing representations from unlabeled data are AE, VAE, and GAN. Those techniques can be applied in different ways, such as with DC [19]. Current DC approaches treat representation learning and clustering as a joint task and focus on learning representations that are clustering-friendly, i.e., that preserve the prior knowledge of cluster structure. [19] divides them into three different main strategies. First, the so-called Multi-Step Sequential Deep Clustering consists of two main steps: 1) learn a richer latent representation of the input data; 2) perform clustering on this latent representation. For instance, it can be distinguished by the use of transfer learning techniques [27], relying on the use of pre-trained models to create or extract representations that can be further fed to clustering models. Second, in Joint Deep Clustering, the step where the representation is learned is tightly coupled with the clustering. Hence, models are trained with a combined or joint loss function that favors learning a good representation while performing the clustering task itself. Third, the Closed-loop Multi-step Deep Clustering, which is similar to Multi-Step Sequential Deep Clustering. However, after pre-training, the steps alternate in an iterative loop, where the output of the

clustering method can be used to allow retraining or fine-tuning of the deep representation.

It has been shown in [15] that combining clustering algorithms working in the latent space of AE can obtain a good clustering performance. Moreover, [19] provided evidence that the most successful methods for clustering with DNN follow the same principle of using the representations learned by a DNN as input for a specific clustering method. However, to the best of our knowledge, only two other models combine AE or generative models with SOM to learn interpretable latent space representations. First, the Self Organizing Map Variational Autoencoder (SOM-VAE) [28] combines SOM, VAE, and a probabilistic model. Second, Deep Embedded Self-Organizing Map (DE-SOM) [29] combines an AE with traditional SOM from [9] with a Gaussian neighborhood function with exponential decay.

The concepts and ideas presented in this section arouse the interest in studying a hybrid environment where the advantages and concepts of each both SOM (unsupervised) and LVQ (supervised) could be explored in combination to perform SSL tasks. This results in proposed models that will be discussed further in Section III, Section IV and Section V.

III. Semi-Supervised Self-Organizing Map

Semi-Supervised Self-Organizing Map (SS-SOM)[1] is a semi-supervised hybrid LVQ-SOM, based on LARFDSSOM [26], with a time-varying structure [25] and two different ways of learning. More precisely, SS-SOM can learn in a supervised or unsupervised way. It switches between these two modes during the self-organization process according to the availability of a class label for each input pattern. To achieve this, we modified the LARFDSSOM to include concepts from the standard LVQ [10] when the class label of an input pattern is given. Moreover, as in LARFDSSOM, the nodes of SS-SOM can consider different relevances for the input dimensions and adapt its receptive field during the self-organization process.

The operation of SS-SOM consists of three phases: 1) organization; 2) convergence; and 3) clustering or classification. In the **organization**, there are two different forms to decide the winner node of a competition, which nodes need to be updated, and when a new node needs to be inserted. If the class label of the input pattern is provided, the supervised mode (Section III-B) is employed. Otherwise, the unsupervised mode (Section III-A) is used. As in LARFDSSOM, the cooperation is performed by adjusting the neighborhood around the winner node. However, the neighborhood of SS-SOM takes into account not only a similar subset of the input dimensions but also their class labels, connecting only nodes with the same class label or unlabeled nodes. Furthermore, after a period determined by a parameter (age_wins), the nodes that do not win for a minimum number of patterns are removed from the map, as in LARFDSSOM.

[1]Available at: https://github.com/phbraga/ss-som

The **convergence** starts after the organization. Here, the nodes are updated and removed when necessary, however, without inserting new nodes. After finishing the convergence phase, the map can cluster and classify input patterns. Depending on the amount and distribution of labeled input patterns presented to the network during the training, in the end, the map may have: 1) All the nodes labeled; 2) Some nodes labeled; and 3) No nodes labeled.

For the first case, the clustering and classification are straightforward: each test pattern is associated with the label of the winner node. For the second case, if the winner node has no class, the SS-SOM continues looking for another node with a defined class label, and an activation above the threshold a_t. For the last case, it is possible to identify the clusters of the input test patterns, but not their classes.

1) Structure of the Nodes: As in LARFDSSOM, each node *j* in SS-SOM represents a cluster and is associated with three *m*-dimensional vectors. They are: $\boldsymbol{c}_j = \{c_{ji}, i = 1, \cdots, m\}$, the center vector that represents the prototype of the cluster *j* in the input space; $\boldsymbol{\omega}_j = \{\omega_{ji}, i = 1, \cdots, m\}$, the relevance vector in which each component represents the estimated relevance, a weighting factor within [0, 1], that the node *j* applies for the *i*-th input dimension; and $\boldsymbol{\delta}_j = \{\delta_{ji}, i = 1, \cdots, m\}$, the distance vector that stores the moving average of the observed distance between the input patterns $\boldsymbol{x}$ and the center vector. Moreover, $\boldsymbol{\delta}$ is used exclusively to compute the relevance vector.

2) Activation of the Nodes: The activation function $ac(D_\omega(\boldsymbol{x}, \boldsymbol{c}_j), \boldsymbol{c}_j)$ (Eq. 1) of a node is calculated as a radial basis function of a weighted distance $D_\omega(\boldsymbol{x}, \boldsymbol{c}_j)$ that has its receptive field adjusted as a function of its relevance vector. The activation grows as the distance decreases and as the relevances increases.

$$ac(D_\omega(\boldsymbol{x}, \boldsymbol{c}_j), \boldsymbol{\omega_j}) = \frac{\sum_{i=1}^{m} \omega_{ji}}{\sum_{i=1}^{m} \omega_{ji} + D_\omega(\boldsymbol{x}, \boldsymbol{c}_j) + \epsilon}, \tag{1}$$

where ϵ is a small value to avoid division by zero and $D_\omega(\boldsymbol{x}, \boldsymbol{c}_j)$ is the weighted distance defined as per Eq. 2.

$$D_\omega(\boldsymbol{x}, \boldsymbol{c}_j) = \sqrt{\sum_{i=1}^{m} \omega_{ji}(x_i - w_{ji})^2}. \tag{2}$$

3) Node Update: In SS-SOM, in order to update the vectors associated with the nodes (the winner, the neighbors, or the winner of a wrong class), a fixed learning rate is used, depending on the undergoing procedure (supervised or unsupervised). So, given a learning rate, the node will be updated as follows:

$$\boldsymbol{c}_j(n+1) = \boldsymbol{c}_j(n) + e(\boldsymbol{x} - \boldsymbol{c}_j(n)), \tag{3}$$

where *e* is the learning rate.

To compute the relevance vectors, we estimate the average distance of each node to the input pattern that it clusters. As in LARFDSSOM, the distance vectors are updated through

a moving average of the observed distance between the input pattern and the current center vector.

$$\boldsymbol{\delta}_j(n+1) = (1 - e\beta)\boldsymbol{\delta}_j(n) + e\beta(|\boldsymbol{x} - \boldsymbol{c}_j(n)|), \tag{4}$$

where e is the learning rate, $\beta \in]0,1[$ controls the rate of change of the moving average, and the operator $|\cdot|$ denotes the absolute value.

After updating the distance vector, each component ω_{ji} of the relevance vector is calculated by an inverse logistic function of the distances δ_{ji} as follows in Eq. 5.

$$\omega_{ji} = \begin{cases} \frac{1}{1+\exp\left(\frac{\delta_{ji\text{mean}} - \delta_{ji}}{s(\delta_{ji\text{max}} - \delta_{ji\text{min}})}\right)} & \text{if } \delta_{ji\text{min}} \neq \delta_{ji\text{max}} \\ 1 & \text{otherwise,} \end{cases} \tag{5}$$

where $\delta_{ji\text{max}}$, $\delta_{ji\text{min}}$, $\delta_{ji\text{mean}}$ are the maximum, the minimum, and the mean of the components of the distance vector $\boldsymbol{\delta}_j$, respectively. The parameter $s > 0$ controls the slope of the logistic function [26].

4) Node Removal: The parameter *age_wins* controls the periodicity of nodes removal. Whenever *age_wins* is reached, any nodes which do not win at least the minimum percentage of the competitions *lp* $\times$ *age_wins* will be removed. To manage this, each node *j* in SS-SOM stores a variable $wins_j$ that represents the number of the node victories since the last reset. After each reset, the number of victories of the remaining nodes is set to zero.

5) Neighborhood Update: In SS-SOM, the neighborhood is formed by nodes with the same class or nearby unlabeled nodes that apply similar relevances for the input dimensions. So, a connection between two nodes means they cluster patterns with the same class or at least similar subspaces. Eq. 6 denotes this behavior.

$$\text{nodes } i \text{ and } j \text{ are } \begin{cases} \text{connected,} & \text{if (} class(i) = class(j) \text{ or} \\ & class(i) = noClass \text{ or} \\ & class(j) = noClass \text{)} \\ & \text{ and } \|\boldsymbol{\omega_i} - \boldsymbol{\omega_j}\| < minwd \\ \text{disconnected,} & \text{otherwise} \end{cases} \tag{6}$$

A. Unsupervised Mode

Given an unlabeled input pattern, SS-SOM looks for a winner node disregarding its class labels. Therefore, as in LARFDSSOM, the winner of a competition is the most activated node according to Eq. 1.

Furthermore, SS-SOM has an activation threshold a_t. According to this, if the activation of the winner is lower than a_t, a new node is inserted into the map at the position of the input pattern (i.e., the winner is not close enough). Otherwise, the winner and its neighbors are updated towards the input pattern (Section III-3). Thereby, as in LARFDSSOM, two fixed learning rates are considered : 1) $e_b \in]0,1[$ for the winner node; and 2) $e_n \in]0, e_b[$ for its neighbors.

B. Supervised Mode

In order to incorporate the supervised learning mode, each node in the map can be associated with a class label. Hence, when a labeled input pattern is given, we treat it properly. If it is the case, SS-SOM takes the labels into account when looking for a winner. If the most activated node s_1 has the same class of the input pattern or an undefined class, a very similar procedure to the unsupervised mode is executed. The difference is the fact that the class of s_1 is set as the same as the input pattern $\boldsymbol{x}$, as well as update its connections. Otherwise, if the winner node and the input pattern have different classes, SS-SOM continues trying to find another winner matching the following criteria: 1) has the same class of the input pattern or an undefined class, and 2) an activation higher than a_t. This procedure inhibits nodes of a different class.

If any node fulfills these conditions, a new winner s_2 is said to be found. Then, s_2 and its neighbors are updated. However, the fact that s_1 was considered a "wrong" winner indicates the need to push it away from this input pattern. Therefore, similarly as in the LVQ, we push s_1 away from the input pattern with a fixed learning rate of $-e_w$. Otherwise, if there is no new winner and the maximum number of nodes in the map has not been reached, a new node is inserted at the same position and with the same class of the input pattern $\boldsymbol{x}$.

C. Convergence

The organization phase does not guarantee that the remaining nodes in the map are well-positioned and connected as expected, according to the distribution, classes, and relations of the input data. Therefore, they may still need to be updated, for instance, to represent the input patterns previously clustered by removed nodes. In the convergence phase, the self-organization process continues, but the map is not allowed to create new nodes. It can be viewed as a reorganization process.

D. Clustering and Classification with SS-SOM

The center vectors $\boldsymbol{c}_j$ and the relevance vectors $\boldsymbol{\omega_j}$ of each node in the map can be used for clustering and classification of the test patterns. Although the most active node is used as the winner, if it has no defined class, SS-SOM continues trying to find another winner candidate with a defined class and an activation above a_t. If it exists, it is used to cluster and its label to classify the test pattern. Otherwise, the most active node is used only for clustering, and the model will not be able to classify it. All in all, if none of the nodes produces an activation equal to or above the threshold a_t for a particular pattern, it is considered as an outlier or noise.

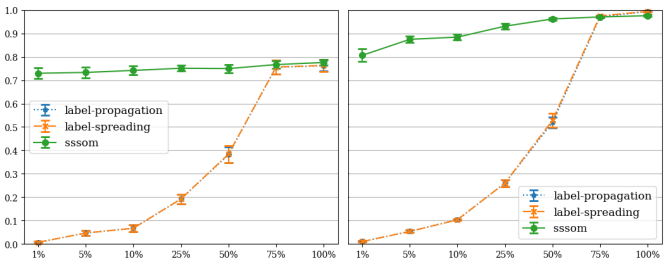
E. Experiments and Results

*1) **Classification Accuracy with Different Percentages of Labeled Data**:* In order to evaluate the classification capabilities of SS-SOM, it is compared with the following semi-supervised methods: Label Propagation [30] and Label Spreading [31]. In this sense, for studying the effects of the different levels of supervision, i. e., the percentage of labeled data. Seven real-world datasets from the OpenSubspace framework

[32] (Breast, Diabetes, Glass, Liver, Pendigits, Shape, and Vowel) were used with the following percentages of labels: 1%, 5%, 10%, 25%, 50%, 75% and 100%.

So, for all of the algorithms on each dataset, a 3-times 3-fold cross-validation was used. Each method was trained and tested 500 times for each fold with different parameter values sampled from the parameter ranges presented in [1], according to a Lating Hypercube Sampling (LHS) [33], while the best accuracy achieved by each method in each fold was recorded for each dataset. It comprises a total of 752,000 experiments. After that, the mean and the standard deviation of the best results for each dataset were calculated separately. For classification purposes, if available, the node class is used as the predicted class. Otherwise, it is straightforwardly considered as an error.

Figure 3 shows the obtained results for two datasets as a function of the percentage of labeled data. However, in all datasets, the performance of the SS-SOM is superior to the other semi-supervised methods concerning the supervision rate between 1% up to 75%, whereas with the highest percentage (100%) the difference is smaller, but it still outperforms them or obtains comparable results. These results show the robustness of SS-SOM in situations when only a small number of labeled data is available.



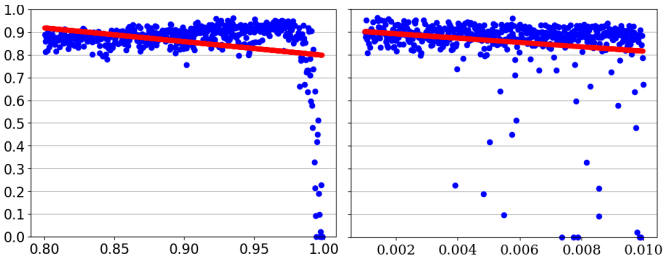
(a) Diabetes (b) Pendigits

Fig. 3: Best mean accuracy and standard deviation as function of the percentage of supervision on (a) Diabetes, and (b) Pendigits datasets for SS-SOM and its competitors.

*2) **Sensitivity Analysis and Sample Efficiency**:* A sensitivity analysis on the parameters of LARFDSSOM, presented in [26], indicates that the parameter a_t is primarily responsible for the variation observed in the results, followed by l_p. To get a deeper understanding of such analysis, we replicated it for SS-SOM. Note that SS-SOM works exactly as LARFDSSOM when there are no labels available. So, it is expected to reproduce the same behavior as its predecessor.

The charts displayed in Figure 4 were obtained from SS-SOM. It is clear that, as in LARFDSSOM, a_t presents the most significant impact on the results. Moreover, SS-SOM also has a low sample efficiency [34], since its framework ignores several samples in cases in which the map is full, and the winning node is not close enough from an input pattern.

*3) **Image Classification Performance**:* To demonstrate an extended capability and application range, the performance of SS-SOM was assessed in an image classification benchmark. More precisely, a pre-trained DenseNet-161 [35] network on ImageNet [36] was used to perform a transfer learning, extracting the features of Canadian Institute For Advanced Research (CIFAR-10) [37] to feed the SS-SOM. DenseNet was chosen due to its outstanding performance for object recognition tasks on CIFAR-10.



(a) at (b) lp

Fig. 4: Sensitivity analysis of the Accuracy obtained with SS-SOM as a function of its parameters, (a) a_t, and (b) lp, for Pendigits dataset using 50% of the available labels, and 1 fold randomly chosen from the 3-times 3-fold cross validation scheme. The red lines are the linear fits to the data.

TABLE I: Classification Accuracy on CIFAR-10 using 4000 and all labeled data.

Accuracy	4000	All
SS-SOM	**0.72**	**0.85**
Spike-and-Slab Sparse Coding	0.68	-
View-Invariant K-means	0.71	0.82

Particularly, the performance on the features extracted from CIFAR-10 dataset with DenseNet-161 [35] pre-trained model is measured. A sampling of 4000 balanced labeled data was made to avoid oversampling or undersampling of any class. After that, the SS-SOM was trained using the same ranges as those defined in [1]. However, only ten parameters set were sampled, and the best result was chosen to perform further comparisons with with Spike-and-Slab Sparse Coding (S3C) [38] and View-Invariant K-means (VIK) [39].

The S3C is an effective feature discovery algorithm for both supervised and semi-supervised learning with small amounts of labeled data based on inferences. The VIK is a modified simple K-means dictionary learning. It extends the concept of spatial pooling by drawing a strategy of directly modeling complex invariances of object features. Also, despite being remarkable in the literature, they both have an experimental methodology in which it was possible to compare without the need for implementation and rerunning. Table I presents the obtained results, and SS-SOM showed to be the best. This set of experiments clarified that SS-SOM can perform well with high-level features extracted from images.

IV. Deep Categorization with Batch SS-SOM

To extend the range of applications of SS-SOM, Batch Semi-Supervised Self-Organizing Map (Batch SS-SOM)[2] is

[2]Available at: https://github.com/phbraga/batch-sssom

introduced. Initially, to take advantage of Graphics Processing Units (GPU), to allow mini-batch training, and thus to be more integrated with other DL approaches that commonly use the same framework and structure, the implementation uses the PyTorch framework. Moreover, three important modifications to the baseline model in order to improve its performance under the new set of conditions are proposed.

First, when a mini-batch is given to the model, it is separated into two different mini-batches: 1) the unsupervised mini-batch; and 2) the supervised mini-batch. For the unsupervised case, the key-point modification is to compute an average vector $\overline{\boldsymbol{X}}_u$ of all unlabeled samples that each winner node j succeeded to be the most activated during the competition. After that, the process continues straightforwardly to the unsupervised procedure with all the average vectors and their representative winner nodes.

On the other hand, the supervised scenario results in three distinct situations, that must be handled differently after finding the winner node for each sample contained in the supervised mini-batch (likewise in SS-SOM): **A**: A node with an undefined class is the winner for a labeled sample; **B**: A node with a defined class is the winner for one or more samples of the same class; and **C**: A node with a defined class is the winner for one or more samples of different classes.

First, in **A**, the actions are to set the node class as the same as the input pattern and then update it towards such an input. Second, in **B**, it is necessary to compute the average vector $\overline{\boldsymbol{X}}_l$, where l is the unique related class label, considering all the samples that are under this situation. Following, the usual supervised update procedure of SS-SOM is called, where the class is the same for both node and average sample vector.

Finally, **C** is handled as follows: for all the classes contained in this subset of samples, every different class duplicates the original winner node j by preserving the centroid vector $\boldsymbol{c}_j$, the distance vector $\boldsymbol{\delta}_j$, and the relevance vector $\boldsymbol{\omega}_j$, but setting the class of the new duplicated node to be the same as the current treated class, as well as setting its number of victories to zero. After that, for each class l found in the subset, a vector $\overline{\boldsymbol{X}}_l$ is calculated, and the respective duplicated node is updated using both $\overline{\boldsymbol{X}}_l$ and l, suchlike in **B**, in which the winner node is updated using its corresponding vector and class.

Moreover, all the operations are performed in parallel on the GPU, which optimizes the computational cost and allows the model to be applied to more complex tasks, datasets, and architectures. Finally, in BATCH SS-SOM, it is possible to use the weighted distance (Eq. 2) as a loss function to estimate the error at training time. It can be further used to backpropagate errors to previous layers, when coupled to DL architectures.

A. Experiments and Results

To assess BATCH SS-SOM in a more challenging task, the following strategy was developed. First, a CNN model was trained with supervision from scratch, and then features were extracted. More specifically, we extracted the features before the classifier layer, using them as input to BATCH SS-SOM. Second, we defined several supervision rates, i.e., the percentage of available labels. It is worth mentioning that the sampling was not balanced. Also, this experiment indicates the effects of the number of labeled samples in the outcome results for MNIST, Fashion-MNIST, and SVHN. For this scenario, we started from MNIST and then expanded to the other datasets to guide a case study about the behavior of the model.

TABLE II: Accuracy obtained with BATCH SS-SOM on each dataset according to a percentage of labeled data.

%	MNIST	SVHN	Fashion-MNIST
1%	0.788	0.560	0.624
5%	0.9643	0.716	0.797
10%	0.974	0.713	0.798
25%	0.9793	0.777	0.834
50%	0.983	0.792	0.847
75%	0.9839	0.810	0.840
All	0.9836	0.826	0.846

The main idea is not to surpass any other model, but understand its behaviors when applied to more complex data structures or representations. Different CNN architectures were evaluated in order to achieve better results for each dataset in particular. Detailed description can be found in [4].

Table II illustrates the best results over 10 runs on each dataset with a batch size of 32. As expected, BATCH SS-SOM has increasing gains as the number of labeled samples grows, specifically for beginning percentages. Following through, at a certain point, around 5% of labeled data, the performance stabilizes. This behavior is observed across all the datasets, showing that the proposed method is a good approach to the problem at hand. Notice that transfer learning is a difficult task, and it is a challenge for a great variety of methods. Such performance defines a promising path through the use and application of SOM-based methods.

V. ADAPTIVE LOCAL THRESHOLDS SEMI-SUPERVISED SELF-ORGANIZING MAP

Adaptive Local Thresholds Semi-Supervised Self-Organizing Map (ALTSS-SOM)[3] is a SOM with Adaptive Local Thresholds [40] based on SS-SOM. Hence, being based on SS-SOM, ALTSS-SOM can also learn in a supervised or unsupervised way depending on the availability of labels, and maintains the general characteristics of its predecessors. However, it introduces new behaviors on both sides, supervised and unsupervised, to allow better usage, and consequently, a better understanding of the data statistics. By doing this, ALTSS-SOM aims at overcoming the problems presented by SS-SOM, such as the high sensitivity to the parameters, and the low sample efficiency.

Therefore, the parameterized activation threshold (a_t) used in both previous methods is replaced by an adaptive thresholding technique that takes into account the local variance to provide the model the ability to learn the receptive field of

[3]Available at: https://github.com/phbraga/alt-sssom

each node. The objective is to estimate optimal local regions in the space with respect to the distribution of the input patterns $\boldsymbol{x}$ for each node in the map. To do so, inspired by the Adam algorithm [41], a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement, ALTSS-SOM updates exponential moving averages of the distances between each input pattern $\boldsymbol{x}$ and the centroid of the nodes for each dimension. In SS-SOM and LARFDSSOM, this estimate was done by using not only β but also the learning rate e in equation (Eq. 4). However, ALTSS-SOM modified this approach to use solely the parameter $\beta \in [0, 1)$ for controlling the exponential decay rate of the moving averages.

The moving averages themselves are estimates of the first moment (the mean) of the distances between the input patterns and the centroids of the nodes. Because of that, such means can be used as estimates of the uncentered variance of the nodes in each dimension. However, these moving averages are initialized as vectors of zeros, leading to moment estimates that are biased towards zero, especially during the initial steps, and when the decay rate is low (close to 1) [41]. Still, according to [41], this initialization bias can be counteracted, resulting in a bias-corrected estimate $\hat{\boldsymbol{\delta}}_j$. During the learning process, this bias-corrected estimate $\hat{\boldsymbol{\delta}}_j$, together with the relevance vector $\boldsymbol{\omega}_j$ can be used as reject options [42], determining whether or not an input pattern is in the receptive field of a winner node.

The operation of ALTSS-SOM is similar to SS-SOM. However, before the update of a node, it is necessary to decide if it will affect the whole node structure or just the weighted averages and the relevance vectors. The neighborhood of ALTSS-SOM is also defined as the same as in SS-SOM.

A. Structure of the Nodes

In ALTSS-SOM, each node *j* in the map represents a cluster and is associated with four *m*-dimensional vectors: The first three vectors, $\boldsymbol{c}_j$, $\boldsymbol{\omega}_j$, and $\boldsymbol{\delta}_j$, are the same as defined in Section III-1. However, $\boldsymbol{\delta}_j$ in SS-SOM and LARFDSSOM is initialized as a vector of zeros. Because of that, it is biased towards zero, specifically at initial steps. Thus, it can be seen as the biased first moment estimate. To overcome this problem, ALTSS-SOM introduces a fourth vector, $\hat{\boldsymbol{\delta}}_j = \{\hat{\delta}_{ji}, i = 1, \cdots, m\}$, which is the bias-corrected first moment estimate.

B. Estimating bias-corrected moving averages

The distance vectors are updated through a moving average of the observed distance between the input pattern and the current center vector $\boldsymbol{c}_j$, as per Eq. 7:

$$\boldsymbol{\delta}_j(n+1) = \beta \boldsymbol{\delta}_j(n) + (1-\beta)(|\boldsymbol{x} - \boldsymbol{c}_j(n)|), \tag{7}$$

where $\beta \in \,]0, 1]$ is the parameter that controls the rate of change of the moving average, and $|\boldsymbol{x} - \boldsymbol{c}_j(n)|$ denotes the absolute value applied to the elements of the vectors.

In order to correct the bias towards zero of $\boldsymbol{\delta}_j$, ALTSS-SOM divides it by the term $\left(1 - \beta^{t_j}\right)$, as in Adam [41], where t_j indicates the current timestep of each node *j*. In sum, the bias-corrected moving averages vectors are updated at every node timestep according to the Eq. 8.

$$\hat{\boldsymbol{\delta}}_j(n+1) = \frac{\boldsymbol{\delta}_j(n)}{1 - \beta^{t_j}} \tag{8}$$

To obtain accurate information about the relevance of each dimension for a given node, an update of the relevance vectors must follow every moving averages update. It is calculated by an inverse logistic function of the bias-corrected estimated distances $\hat{\boldsymbol{\delta}}_{ji}$, as follows in Eq. 9.

$$\omega_{ji} = \begin{cases} \frac{1}{1 + \exp\left(\frac{\hat{\delta}_{j i\text{mean}} - \hat{\delta}_{ji}}{s(\hat{\delta}_{j i\text{max}} - \hat{\delta}_{j i\text{min}})}\right)} & \text{if } \hat{\delta}_{ji\text{min}} \neq \hat{\delta}_{ji\text{max}} \\ 1 & \text{otherwise,} \end{cases} \tag{9}$$

This function is pretty similar to Eq. 5, however, instead of using $\boldsymbol{\delta}_j$, the ALTSS-SOM replaces it by $\hat{\boldsymbol{\delta}}_j$ in order to get a more accurate and unbiased estimation.

C. Local Thresholds

The distance vectors $\hat{\boldsymbol{\delta}}$ represent the corrected moving average of the observed distances between the input patterns $\boldsymbol{x}$ and center vectors $\boldsymbol{c}$ for each node *j* in the map. As a result, they can be considered as the variances of the nodes.

In addition, the $\boldsymbol{\omega}$ vectors express how each of the dimensions is important for each node, which indicates its subspaces of the input dimensions. So, by combining $\boldsymbol{\omega}$ and $\hat{\boldsymbol{\delta}}$, it is possible to define a local region around each node center $\boldsymbol{c}_j$ to act like a reject option for some input patterns. If only the variances were used, some unimportant dimensions with a low variance could misguide the process when a similar input pattern $\boldsymbol{x}$ is outside the acceptance region of a node *j*, but only in dimensions that are not relevant to it. Therefore, a flexible variance is defined to act as a local threshold and rejection option to mitigate such problems:

$$\text{Var}(\hat{\boldsymbol{\delta}}_j, \boldsymbol{\omega}_j) = \frac{\hat{\boldsymbol{\delta}}_j}{\boldsymbol{\omega}_j} \tag{10}$$

When a dimension has a high relevance to the node, it will not impact its variance value. However, when a dimension has a small relevance, ALTSS-SOM will relax the constraints to allow a better definition of subspaces. Therefore, the general acceptance rule is defined by Eq. 11, where the idea is to approximate to an optimal rule.

$$A(\boldsymbol{x}, \boldsymbol{c}_j, \boldsymbol{v}_j) = \begin{cases} True, & \boldsymbol{x}_i \in \,]\boldsymbol{c}_{ji} \pm \boldsymbol{v}_{ji}[\,, \\ & \forall\ \mathbf{c}_{ji} \in \boldsymbol{c}_j,\ \boldsymbol{x}_i \in \boldsymbol{x}, \text{ and } \boldsymbol{v}_{ji} \in \boldsymbol{v}_j \\ False, & \text{otherwise,} \end{cases} \tag{11}$$

where $\boldsymbol{x}$ is the input pattern, $\boldsymbol{c}_j$ is the center vector, and $\boldsymbol{v}_j$ = $\text{Var}(\hat{\boldsymbol{\delta}}_j, \boldsymbol{\omega}_j)$ is the relaxed variance vector as per Eq. 10.

D. Unsupervised Mode

Given an unlabeled input pattern, the most activated node is considered as the winner, disregarding its class labels. In this sense, ALTSS-SOM verifies if the condition expressed by the Eq. 11 is satisfied. If so, the winner and its neighbors are updated towards the input pattern. Otherwise, a new node

is inserted into the map at the input pattern position. However, since s_1 is the original winner, it will improve its knowledge about the region where it is located by updating its moving averages and relevances, but not its center. This mechanism provides the nodes the ability to learn about the region they are inserted in. An additional case is handled when the map has reached the maximum number of nodes. In this situation, aiming at not losing the information that the input pattern can provide, as in previous models, thus, improving sample efficiency, ALTSS-SOM updates the moving average and the relevance vectors of the winning node.

E. Supervised Mode

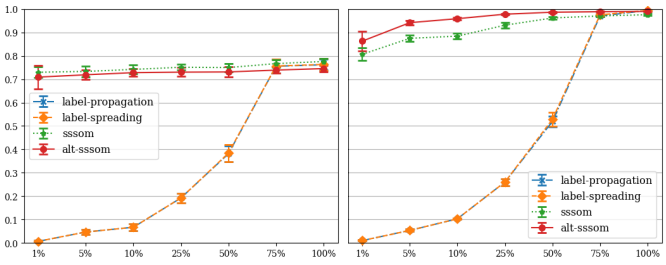
This procedure is similar to the one presented in Section III-B for SS-SOM. However, instead of using a_t as the threshold parameter that controls the activation, the acceptance criteria expressed by the Eq. 11 is employed.

If there are no new nodes to replace s_1 as a new winner, and the map is not full, s_1 node is duplicated, preserving the moving averages vectors, the centroid vector as well as the relevance vector. However, the class of this new duplicated node is set to the same as the input pattern. Moreover, whenever the map is full, the moving averages and relevance vector of the current defined winner continue to be updated in order to improve its knowledge about the surrounding area.

Moreover, both node removal and neighborhood update (include its definition) remain the same as in SS-SOM.

F. Experiments and Results

1) ***Classification Accuracy with Different Percentages of Labeled Data****:* In order to evaluate the classification rate of ALTSS-SOM, the experiments conducted in Section III-E1 were replicated, but adding the ALTSS-SOM in the comparison. The ranges used for ALTSS-SOM are defined in [3].



(a) Diabetes (b) Pendigits

Fig. 5: Best mean accuracy and standard deviation as function of the percentage of supervision on (a) Diabetes, and (b) Pendigits datasets for ALTSS-SOM, SS-SOM, Label Spreading and Label Propagation

Figure 5 shows the obtained results for two datasets. However, ALTSS-SOM improved the performance of SS-SOM overall, except for the Diabetes dataset (Figure 5a) where the results obtained were slightly lower, but yet comparable. The flexibility provided by the estimation of the receptive field of nodes and the improved sample efficiency allowed such results. Still, the standard deviation for all datasets in all supervision levels was also minimized, which indicates another positive aspect of the ALTSS-SOM: it is more robust to variations on both datasets and parameters. While the other semi-supervised learning methods surpassed SS-SOM in Pendigits and Vowel datasets, ALTSS-SOM achieved a consistent improvement by outperforming the results of both LP and LS, in all datasets, except for Vowel at 100%.

2) ***Clustering Performance****:* Aiming to assess the performance of ALTSS-SOM in a purely unsupervised clustering task due to the changes made in the original framework that it was inspired, it was compared with Densitive-based Optimal projective Clustering (DOC) [43] , PROjected CLUSTering algorithm (PROCLUS) [44], LARFDSSOM/SS-SOM and BATCH SS-SOM (with a batch size of 32). We refer to the LARFDSSOM and SS-SOM together due to their equivalence for clustering tasks solely. The first two methods are originally from the data mining area. This choice of comparison was defined in accordance with the analysis provided by [26]. The parameters used for ALTSS-SOM to execute clustering tasks are slightly different from the previous classification tasks. They are pointed in [3], whereas the ranges of the other methods were the same as those used in [26].

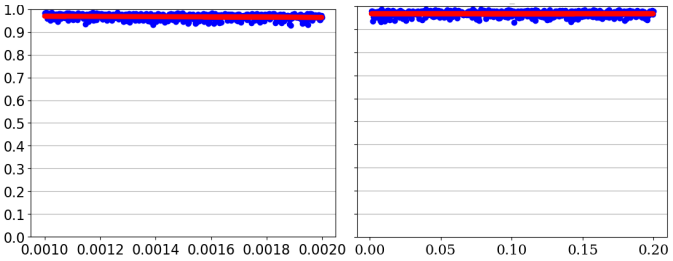
Table III shows the results of the Clustering Error (CE) obtained with the methods. None of the methods achieved the best result for all real-world datasets. ALTSS-SOM presented the best result for 6 out of 7 datasets, but achieved the same results of LARFDSSOM for Breast and Glass datasets. Moreover, PROCLUS presented good results when the parameter controlling the number of clusters was defined close to the correct value. Furthermore, the good results obtained by LARFDSSOM is directly related to a good choice of the parameters a_t and lp, which significantly impact the results. However, ALTSS-SOM achieves good results without needing an exact definition of the parameter values.

Furthermore, BATCH SS-SOM also showed to perform well. In Breast, it achieved the same value as other clustering methods. In Diabetes and Vowel, it was statistically equal to LARFDSSOM/SS-SOM. In the Glass, Liver, and Shape datasets, the batch size has a slightly negative influence on the outcome, which is an effect of the mean vector update rule.

3) ***Sensitivity Analysis****:* In previous methods, the most two critical parameters were the a_t and lp. a_t played a role of great importance due to its high impact on the results with just a small change on its values, i.e., a_t impacted the results exponentially. Because of that, a sensitivity analysis was also performed for ALTSS-SOM in order to elucidate the improvements. Figure 6 shows the scatter plots for lp and e_b. They were chosen due to their semantical importance. Notice that there are no significant trends to parameter values, and they do not damage the performance with slight changes in its values. This same behavior was observed for all the other datasets and parameters used in this set of experiments. Nevertheless, a thorough analysis is presented in [3].

TABLE III: CE Results for Real-World Datasets. Best results for each dataset are shown in bold

CE	Breast	Diabetes	Glass	Liver	Pendigits	Shape	Vowel	Avg	STD
DOC	**0.763** (1)	0.654	0.439	0.580	0.566	0.419	0.142	0.509	0.201
PROCLUS	0.702 (2)	0.647	0.528	0.565	0.615	0.706	0.253	0.574	0.156
LARFDSSOM/SS-SOM	**0.763** (1)	**0.727** (1)	**0.575** (1)	0.580 (2)	0.737 (2)	0.719 (2)	0.317 (1)	0.631	0.158
ALTSS-SOM	**0.763** (1)	0.697 (2)	**0.575** (1)	**0.603** (1)	**0.741** (1)	**0.738** (1)	**0.319** (1)	**0.633**	0.156
Batch SS-SOM	**0.763** (1)	0.723 (1)	0.537 (2)	0.580 (2)	0.735	0.693	0.301	0.619	0.151



(a) lp (b) e_b

Fig. 6: Sensitivity analysis of the Accuracy obtained with ALTSS-SOM as a function of its parameters, (a) lp, and (b) e_b, for Pendigits dataset using 50% of the available labels, and 1 fold randomly chosen from the 3-times 3-fold cross validation scheme. The red lines are the linear fits to the data.

*4) **Performance of SS-SOM and ALTSS-SOM in Fully Supervised Scenarios**:* To draw a better understanding of the behavior of both SS-SOM and ALTSS-SOM, they were evaluated in the scenario of a regular supervised learning task on the same UCI datasets from previous experiments (Breast, Diabetes, Glass, Liver, Pendigits, Shape, and Vowel). It is not expected they go well since they were not built for it. Still, they were compared with traditional supervised methods such as Multilayer Perceptron (MLP) [20], Support Vector Machines (SVM) [45], and Generalized Relevance Learning Vector Quantization [13]. The ranges of the parameters are given in [3]. Moreover, the semi-supervised methods used 100% of the labeled data to allow a fair comparison with supervised methods such as GRLVQ, MLP, and SVM.

The proposed methods showed results better than or at least close to the best. SS-SOM appears as the best overall among the semi-supervised methods, showing an average accuracy of 0.84. On considering all methods at 100% of supervision, the MLP is the best, achieving 0.851 of accuracy on average. However, in most of the datasets, there are no statistical differences between SS-SOM and MLP, as well as between ALTSS-SOM and MLP. [3] presents detailed information about particular results for each dataset.

VI. Conclusion

The lack of labels in the midst of the great volume of information that has been produced every day is still a problem for a significant number of machine learning models. This work was intended to take another step, attaining to build more sophisticated solutions.

SS-SOM and ALTSS-SOM were developed by combining the concepts of both SOM and LVQ to improve the results obtained with traditional SSL methods in the literature not only in terms of classification rate but also in clustering quality. The behavior of both models was shown to have led to significant improvements in classification results for small amounts of labeled data, establishing its position as a good option when dealing with such problems. It showed its robustness under this condition, being better than other semi-supervised models, achieving impressive results even with only 1% of labeled data, in comparison with other SSL methods.

Furthermore, to extend its application range, SS-SOM was tested with features extracted from images. This allowed the model to perform well, also with more complex data types. Nevertheless, Batch SS-SOM, a novel approach that allows a mini-batch training procedure for the traditional shallow architecture of a SOM, was proposed. The results were surprisingly good. It is true that a great range of techniques that may benefit the performance of Batch SS-SOM exist. However, this version established a baseline for future development.

Moreover, with ALTSS-SOM, a strategy was developed to estimate local rejection options as a function of both local variance and the relevance of input dimensions to make pattern rejection decisions. In addition to it, the model was able to reduce the dependency and the variability to the most important parameters. This parametric robustness can be considered as one of the most important contributions of this current work. In addition to it, the usage of a relaxed estimated variance allowed the method to explore the local information of the data clusters better. Also, it provided the ability to improve sample efficiency by not merely discarding data in certain cases but kept digging into its characteristics in order to establish a better estimation of their statistics. This can be summarized in the idea of developing models that can exploit to the utmost the information carried in the data, either for generating prototypes or for adjusting their receptive fields, but also to recognize and disregard outliers when necessary.

As a further matter, it is important mentioning some additional considerations. SS-SOM and ALTSS-SOM were put on trial in a scenario of full supervision by comparing their results with state-of-the-art supervised learning models. Even though not being built for that, both models presented good results, being better than or at least close to the best. Therefore, we consider the proposed models as useful tools for promoting the formation of meaningful representations of the data.

Finally, we have left for future work an adaptation of

the proposed models to act in a gradient-based approach, where the unsupervised error could be estimated and used in a jointly training process with AE or generative models in order to learn even better latent representations and compete with state-of-art models on a broader range of tasks. For instance, disentanglement and compositionality can be pointed out as interesting directions that could extend the application domains.

REFERENCES

[1] P. H. M. Braga and H. F. Bassani, "A semi-supervised self-organizing map for clustering and classification," in *2018 International Joint Conference on Neural Networks*. IEEE, 2018, pp. 1–8.

[2] P. H. Braga, H. R. Medeiros, and H. F. Bassani, "Backpropagating the unsupervised error of self-organizing maps to deep neural networks," in *Advances in Neural Information Processing Systems*, ser. LatinX in AI Research Workshop, 2019.

[3] P. H. M. Braga and H. F. Bassani, "A semi-supervised self-organizing map with adaptive local thresholds," in *2019 International Joint Conference on Neural Networks*. IEEE, 2019, pp. 1–8.

[4] P. H. M. Braga, H. R. Medeiros, and H. F. Bassani, "Deep categorization with semi-supervised self-organizing maps," in *2020 International Joint Conference on Neural Networks*. IEEE, 2020, pp. 1–8.

[5] P. H. M. Braga, "Semi-supervised self-organizing maps with time-varying structures for clustering and classification," Master's thesis, Universidade Federal de Pernambuco, 2019.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[7] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[8] F. Schwenker and E. Trentin, "Pattern classification and clustering: A review of partially supervised learning approaches," *Pattern Recognition Letters*, vol. 37, pp. 4–14, 2014.

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

[10] ——, "Learning vector quantization," in *Self-organizing maps*. Springer, 1995, pp. 175–189.

[11] M. Köppen, "The curse of dimensionality," in *5th Online World Conference on Soft Computing in Industrial Applications*, 2000, pp. 4–8.

[12] H. F. Bassani and A. F. Araújo, "Dimension selective self-organizing maps for clustering high dimensional data," in *The 2012 International Joint Conference on Neural Networks*. IEEE, 2012, pp. 1–8.

[13] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Networks*, vol. 15, no. 8-9, pp. 1059–1068, 2002.

[14] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 1, p. 1, 2009.

[15] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *preprint arXiv:1801.07648*, 2018.

[16] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017.

[17] H. R. Medeiros, F. D. de Oliveira, H. F. Bassani, and A. F. Araujo, "Dynamic topology and relevance learning som-based algorithm for image clustering tasks," *Computer Vision and Image Understanding*, vol. 179, pp. 19–30, 2019.

[18] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "Self-labelling via simultaneous clustering and representation learning," *preprint arXiv:1911.05371*, 2019.

[19] G. C. Nutakki, B. Abdollahi, W. Sun, and O. Nasraoui, "An introduction to deep clustering," in *Clustering Methods for Big Data Analytics*. Springer, 2019, pp. 73–89.

[20] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Prentice-Hall, 2009.

[21] R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh, *Computational maps in the visual cortex*. Springer Science & Business Media, 2006.

[22] H. F. Bassani, "Modelos neurais modulares para aquisição de linguagem natural," Ph.D. dissertation, Universidade Federal de Pernambuco, 2014.

[23] X. Zhu, "Semi-supervised learning literature survey," Department of Computer Science, University of Wisconsin-Madison, Tech. Rep., 2006.

[24] Z. Hailat, A. Komarichev, and X. Chen, "Deep semi-supervised learning," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Aug 2018, pp. 2154–2159.

[25] F. R. Araujo, H. F. Bassani, and A. F. Araujo, "Learning vector quantization with local adaptive weighting for relevance determination in genome-wide association studies," in *Proceedings of the 2013 International Joint Conference on Neural Networks*. IEEE, 2013, pp. 1–8.

[26] H. F. Bassani and A. F. Araujo, "Dimension selective self-organizing maps with time-varying structure for subspace and projected clustering," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 3, pp. 458–471, 2015.

[27] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *preprint arXiv:1804.09170*, 2018.

[28] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Som-vae: Interpretable discrete representation learning on time series," *preprint arXiv:1806.02199*, 2018.

[29] F. Forest, M. Lebbah, H. Azzag, and J. Lacaille, "Deep architectures for joint clustering and visualization with self-organizing maps," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2019, pp. 105–116.

[30] Z. Xiaojin and G. Zoubin, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Tech. Rep., 2002.

[31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, 2004, pp. 321–328.

[32] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1270–1281, 2009.

[33] J. C. Helton, F. Davis, and J. D. Johnson, "A comparison of uncertainty and sensitivity analysis results obtained with random and latin hypercube sampling," *Reliability Engineering & System Safety*, vol. 89, no. 3, pp. 305–330, 2005.

[34] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *preprint arXiv:1611.01224*, 2016.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 2261–2269.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[37] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[38] K. Y. Hui, "Direct modeling of complex invariances for visual object features," in *International Conference on Machine Learning*, 2013, pp. 352–360.

[39] I. Goodfellow, A. Courville, and Y. Bengio, "Large-scale feature learning with spike-and-slab sparse coding," *preprint arXiv:1206.6407*, 2012.

[40] X. Jiang and D. Mojon, "Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 131–137, 2003.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *preprint arXiv:1412.6980*, 2014.

[42] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory*, vol. 16, no. 1, pp. 41–46, 1970.

[43] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. Murali, "A monte carlo algorithm for fast projective clustering," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. ACM, 2002, pp. 418–427.

[44] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMoD Record*, vol. 28, no. 2. ACM, 1999, pp. 61–72.

[45] C. Cortes and V. Vapnik, "Support vector machine," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.