University of Groningen

Self-organizing neurons

Khacef, L.; Miramond, B.; Barrientos, D.; Upegui, A.

Published in:
 IJCNN 2019. International Joint Conference on Neural Networks

DOI:
[10.1109/IJCNN.2019.8852098](https://doi.org/10.1109/IJCNN.2019.8852098)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
 Publisher's PDF, also known as Version of record

Publication date:
 2019

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Khacef, L., Miramond, B., Barrientos, D., & Upegui, A. (2019). Self-organizing neurons: Toward brain-inspired unsupervised learning. In *IJCNN 2019. International Joint Conference on Neural Networks: Proceedings* [N-19097] IEEE. <https://doi.org/10.1109/IJCNN.2019.8852098>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Self-organizing neurons: toward brain-inspired unsupervised learning

Lyes Khacef¹, Benoît Miramond¹, Diego Barrientos², Andres Upegui²

¹Université Côte d'Azur, CNRS, LEAT, France

²InIT, hepia, University of Applied Sciences of Western Switzerland, Switzerland

¹firstname.lastname@univ-cotedazur.fr, ²firstname.lastname@hesge.ch

Abstract—During the last years, Deep Neural Networks have reached the highest performances in image classification. Nevertheless, such a success is mostly based on supervised and off-line learning: they require thus huge labeled datasets for learning, and once it is done, they cannot adapt to any change in the data from the environment. In the context of brain-inspired computing, we apply Kohonen-based Self-Organizing Maps for unsupervised learning without labels, and we explore original extensions such as the Dynamic SOM that enables continuous learning and the Pruning Cellular SOM that includes synaptic pruning in neuromorphic circuits. After presenting the three models and the experimental setup for MNIST classification, we compare different methods for automatic labeling based on very few labeled data (1% of the training dataset), and then we compare the performances of the three Kohonen-based Self-Organizing Maps with STDP-based Spiking Neural Networks in terms of accuracy, dynamicity and scalability.

Index Terms—brain-inspired computing, self-organizing maps, unsupervised learning, embedded image classification.

I. INTRODUCTION

With the recent developments of deep learning through Deep Neural Networks (DNNs) and the availability of machine learning frameworks (TensorFlow, Torch, McCafé, etc.), artificial intelligence is getting closer to our lives through a large panel of applications (speech and image recognition, autonomous driving, health assistance, etc.). Still, DNNs are limited in two aspects: on the one hand, DNNs software implementation on general purpose CPU and GPU remains too expensive in terms of computational resources and power consumption for embedded systems, and suffer from the Von Neumann bottleneck. Therefore, there is a growing interest toward dedicated hardware implementations with distributed neuromorphic architectures [1].

On the other hand, even though some recent works try to reconnect deep learning with neuroscience [2] [3], DNNs are moving away from the initial biological inspiration of Artificial Neural Networks (ANNs). Indeed, unlike the brain, DNNs are mostly based on supervised learning, which implies building huge labeled datasets for training, as the labels are needed to implement the gradient back-propagation algorithm that adjusts the neurons weights during training [4]. It suggests that we have to label each sample of each training database depending on the application, and that approach can obviously not be generalized to all types of data and applications. Moreover, it is unlikely that such an algorithm based on neuron-specific error signal would be implemented in the brain

[5]. Instead, biological evidences lead rather to unsupervised learning methods. Furthermore, DNNs are based on off-line learning: once the DNN is trained, there is no more learning during inference and thus no more adaptation to any change in the context or inputs from the dynamic environment.

We focus in this work on brain-inspired models of ANNs: the Self-Organizing Maps (SOMs). SOMs are inspired from the cortical plasticity, i.e. the self-organization of the cortical areas in the brain to specialize their learning in specific features. This mechanism was first modeled by Kohonen [6]. In the context of image classification with unsupervised learning, we propose a new approach where we apply the original Kohonen SOM (KSOM) [6], the Dynamic SOM (DSOM) [7] and the Pruning Cellular SOM (PCSOM) [8] to MNIST digits classification, without using labels for training. Afterwards, we propose three automatic labeling algorithms to assign the class of each neuron, trying to reach the best accuracy while minimizing the number of labeled images we need.

We first make the state of the art of image classification with unsupervised learning, then we describe the three self-organizing neural networks that we compare. Next, we define our general approach for training, labeling and testing our neural networks. Finally, we compare our results with the literature and assess the three models in terms of classification accuracy, learning dynamicity and hardware scalability.

II. STATE OF THE ART

A. Kohonen-based Self-Organizing Maps

Fig. 1. Self-Organizing Map topology

SOMs are brain-inspired unsupervised ANNs introduced by Kohonen [6]. Basically, for each input stimulus, the SOM neurons compete to "win", i.e. to be the Best Matching Unit (BMU) that is the neuron representing best the stimulus. Then, the winning neuron and its neighborhood neurons adapt their afferent synaptic weights in order to improve the representation of the input pattern as detailed in section III. The idea is that each neuron becomes a "prototype" that represents an average of similar-enough inputs.

In 2015, Liu et al. [9] proposed a Deep Self-Organizing Map (DeepSOM) for visual classification. The structure is a cascade of self-organizing layers made up of a number of SOMs such that each SOM behaves as a receptive field in a local region of the input. It is followed by sampling operators that organize the winning neurons of each SOM and form a new two-dimensional map, constructing more global information in higher layers as shown in Fig. 2. However, the whole DeepSOM is trained with supervised SOMs [10], which means that the labels of the training dataset are used as input data to explicitly influence the SOM learning. Their results show that DeepSOM outperforms the original supervised SOM by 07.17% on MNIST classification, but the comparison with supervised methods is out of the scope of this paper.

Fig. 2. Sampling layer creation in DeepSOM [11]

In 2017, Wickramasinghe et al. [11] took over the work of [9] and proposed a Parallelizable DeepSOM (ParalDeepSOM) that shows two major interests: first, the ParalDeepSOM uses SOMs of different sizes in the same layer, and thus has a better probability to learn features of different resolutions in the same layer. When the small map misses a certain feature, there is a high probability that the larger map can catch it. Second and more importantly, ParalDeepSOM is based on unsupervised learning, with no use of labels for training. It relies then on a subpart of the training database to label the neurons, i.e. assign each neuron the class it represents. The ParalDeepSOM accuracy on MNIST classification is 82.10%,

which is 01.64% better than the unsupervised version of DeepSOM [11]. However, this accuracy is achieved using a very large number of neurons, as shown in Tab. III.

B. STDP-based Spiking Neural Networks

Spike-Timing-Dependent Plasticity (STDP) is a brain-inspired unsupervised learning algorithm implemented in Spiking Neural Networks (SNNs). SNNs are more biologically plausible models for two reasons: first, they encode information in spikes, i.e. electrical impulses that correspond to a 1 bit digital representation. Second, the spiking neuron is the Leaky Integrate-and-Fire (LIF) which is a simplified model of bio-mimetic artificial neurons that try to mimic the behavior of biological neurons. The LIF computation is hence simpler than that of the formal neuron, and we have previously made a comparative study of the information coding impact of both spiking and formal models on neuromorphic architectures with off-line supervised learning [12]. It showed that the SNN has approximately 50% gain in the hardware implementation cost (resources and power). However, we did not explore the impact of unsupervised on-line learning with STDP that implies more complex computations, and thus a higher hardware cost.

Fig. 3. SNN with STDP architecture [13]

In 2015, Diehl et al. [13] proposed a SNN structure implementing STDP for unsupervised learning with lateral inhibition that generates competition amongst neurons, so that each neuron learns a different pattern. The SNN architecture is made of input, excitatory and inhibitory layers, as shown in Fig. 3. Each excitatory neuron receives an STDP-modifiable synapse from the input layer, and is connected to one neuron of the inhibitory layer which connects back to all the other excitatory neurons. Consequently, the first neuron that spikes inhibits all others. The SNN with STDP accuracy on MNIST classification using 100 neurons is 82.90%, which is approximately equal to [11].

In 2018, two extensions based on the work of [13] have been proposed. First, Saunders et al. [14] introduced the Convolutional SNN (CSNN) where inhibitory neurons are only connected to excitatory neurons in the other sub-populations

that share the same visual field. However, the accuracy results on MNIST classification do not reach those of [13] for the same number of neurons.

Second, Hazan et al. [15] extended the work of [14] by combining STDP with a characteristic of the SOM: the degree of competition imposed by the connections from the inhibitory layer is curved by increasing the level of inhibition with the distance between neurons. The Lattice Map SNN (LMSNN) achieved an accuracy of $85.71\% \pm 0.85\%$ on MNIST classification with 100 excitatory and inhibitory neurons, which is an improvement over the state of the art SNNs [15].

Nevertheless, the works of [13], [14] and [15] are limited in their labeling approach: here, we are dealing with semi-supervised classification, as we use unsupervised learning to adjust the neurons weights, followed by supervised labeling to assign each neuron the class it represents. Hence, we always need a labeled subset of the training dataset in order to label the neurons for evaluation purposes and inference. Still, we must not use the whole training dataset for labeling.

In the literature, the training is indeed performed in an unsupervised manner. However, the labeling is performed using one presentation of the whole training dataset [13] [14] [15]. It means that the DNNs initial limit of using labels is only shifted from the training part to the labeling part. In our work, we evaluated the minimal subset of labeled samples necessary to reach the best accuracy, as described in section V-C.

III. SELF-ORGANIZING NEURAL MODELS

Technically, neural self-organization is a vector quantization algorithm, as it models the probability density function of the training dataset into a set of prototype vectors that are represented by the neurons afferent weights [7]. It has been shown that SOMs have better performances in representing overlapping structures compared to classical clustering techniques such as partitive clustering or K-means [16]. SOMs already have a multitude of applications in speech recognition, robotics, process control and telecommunication [17].

We present in this section three models of SOMs: KSOM, DSOM and PCSOM, each using a two-dimensional grid of neurons. Each neuron has a respective two-dimensional position in the grid, and is connected to the input stimulus through afferent synapses that carry the weights where the learning occurs. The neuron weights are represented as an m -dimensional vector where m is defined by the dimensions of the input stimuli. Each neuron is then connected to its four neighbors from north, east, south and west through lateral synapses, as shown in Fig. 1.

A. Kohonen Self-Organizing Map (KSOM)

The original KSOM algorithm introduced by Kohonen [6] is the following:

Initialize the network as a two-dimensional array of neurons. Each neuron n is defined by a two-dimensional position p_n and a randomly initialized m -dimensional weight vector w_n .

for t from 0 to t_f **do**

for every new input vector v **do**

Compute the winner s such that w_s is the closest to v .

for every neuron n in the network **do**

Compute the neighborhood function $h_\sigma(t, n, s)$:

$$h_\sigma(t, n, s) = e^{-\frac{||p_n - p_s||^2}{2\sigma(t)^2}} \quad (1)$$

Update the weight w_n of the neuron n :

$$w_n = w_n + \epsilon(t)h_\sigma(t, n, s)(v - w_n) \quad (2)$$

end for

end for

Update the learning rate $\epsilon(t)$:

$$\epsilon(t) = \epsilon_i \left(\frac{\epsilon_f}{\epsilon_i} \right)^{t/t_f} \quad (3)$$

Update the width of the neighborhood $\sigma(t)$:

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i} \right)^{t/t_f} \quad (4)$$

end for

The KSOM has a decaying learning rate and neighborhood width, so that the learning stabilizes after a certain number of iterations. When $t = t_f$, the KSOM is almost unable to learn any change in the input stimuli, as $\epsilon_f \ll \epsilon_i$ and $\sigma_f \ll \sigma_i$. Therefore, the learning is stable but not dynamic. It can be considered as an off-line unsupervised learning algorithm.

B. Dynamic Self-Organizing Map (DSOM)

The DSOM introduced by Rougier et al. [7] is a variation of the KSOM algorithm where the time dependency of the learning rate and neighborhood function has been replaced by the distance between the winning neuron and the input stimulus, as shown in the following algorithm:

Initialize the network as a two-dimensional array of neurons. Each neuron n is defined by a two-dimensional position p_n and a randomly initialized m -dimensional weight vector w_n .

for every new input vector v **do**

Compute the winner s such that w_s is the closest to v .

for every neuron n in the network **do**

Compute the neighborhood function $h_\sigma(t, n, s)$:

$$h_\eta(n, s, v) = e^{-\frac{1}{\eta^2} \frac{||p_n - p_s||^2}{||v - w_s||^2}} \quad (5)$$

where η is the elasticity or plasticity parameter. If $v = w_s$, then $h_\eta(n, s, v) = 0$. Update the weight w_n of the neuron n :

$$w_n = w_n + \epsilon ||v - w_n|| h_\eta(n, s, v)(v - w_n) \quad (6)$$

end for

end for

In the DSOM algorithm, if a neuron is close enough to the stimulus, then this neuron is already representing well the stimulus, hence there is no need for any neuron to learn (the extreme case where $v = w_s$ and so $h_\eta(n, s, v) = 0$). In the

other case, if there is no neuron close enough to the stimulus, all neurons learn according to their weight distance to this new stimulus and their topological distance to the BMU.

This mechanism allows a dynamic learning to adapt to any change in the environment at any moment. The DSOM can therefore be used for on-line unsupervised learning. However, in contrast with SOM, the DSOM self-organizes into the support of the distribution of the input stimuli and does not try to match the density [7].

C. Pruning Cellular Self-Organizing Map (PCSOM)

The PCSOM introduced by Upegui et al. [8] is also abstracted from the time dependency of the KSOM, it is hence made for continuous on-line unsupervised learning. In addition, it models a specific mechanism of biological neurons: the synaptic pruning. Indeed, each neuron of the PCSOM has a number of associated lateral synaptic connections varying from 0 to 4 that define its lateral influence during training.

These lateral synapses can be seen as interconnection matrices that are initially interconnecting every neuron to its four physical neighbors. Afterwards, during the network lifetime where the network learns to represent the input stimuli, some of these synapses will be pruned in order to allow the prototype vectors to better fit their density function. After a certain number of iterations, the PCSOM begins to create clusters, i.e. group of topologically close neurons which represent variations of the same class. The idea is then to prune (remove) the synaptic connections between the frontiers neurons of different clusters and isolate them, so that one class does not affect the learning of the other and thus prevent the network to achieve better performances. The PCSOM algorithm is described as follows:

Initialize the network as a two-dimensional array of neurons. Each neuron n is defined by a two-dimensional position, a randomly initialized m -dimensional weight w_n vector and a set of synapses defining connections to other neurons with respect to the position.

for every new input vector v **do**

Compute the winner s such that w_s is the closest to v .
Update the weight of the winner w_s :

$$w_s = w_s + \alpha(v - w_s) \quad (7)$$

where α is the learning rate.

for every other neuron n in the network **do**

Update weights as follows:

$$w_n = w_n + \alpha(w_i - w_n)e^{(-\frac{1}{\eta} \frac{hops}{\|w_i - w_n\|})} \quad (8)$$

where w_n is the weight vector of the neuron to be updated, $hops$ is the number of propagation hops from the winner, w_i is the weight vector of the influential neuron, η is the elasticity of the network.

end for

for every synapse in the network **do**

Apply pruning following the probability:

$$P_{ij} = e^{(-\frac{1}{\omega} \frac{1}{\|w_i - w_j\| t_i t_j})} \quad (9)$$

where P_{ij} is the probability of pruning the synapse interconnecting n_i and n_j , ω is the pruning rate, and t_i is the time from the last winning of neuron n_i .

end for

end for

With respect to the neuron n to be updated, the influential neuron i is the connected neighbor neuron that is closest to the winner neuron s in terms of number of *hops*. In the case where two connected neighbor neurons have the same number of *hops*, the choice of the influential neuron is made randomly.

Thanks to the propagation of the neurons update through the neurons neighbors, the overall network weights are influenced by every new input vector depending on the network connectivity: if a neuron n is a topological neighbor of the influential neuron i , it is only updated if the synapse connecting n to i has not been pruned. The PCSOM is therefore modifying the connectivity of the neural network by pruning the useless synapses that connect two neurons whose activities are poorly correlated, as shown in Fig. 4. It allows the network to better fit the probability density function of the input stimuli. Preliminary results in [8] have shown that the proposed pruning mechanisms improve the network performance by reducing the Average Quantization Error (AQE) of the incoming two-dimensional stimuli, and thus allows better learning and representation of the information from the environment.

Fig. 4. (Left) Cellular architecture connections after training with $w = 3e - 07$. (Right) Weights and probability density after training with $w = 3e - 07$.

The PCSOM is therefore a dynamic and evolving (with respect to pruning) self-organizing neural network, that may reach better performances than DSOM for on-line unsupervised learning thanks to the pruning mechanism. Our results on image classification are shown in section V-D.

IV. METHOD

A. Training

In order to compare the accuracy performances of our three SOMs with each other and with respect to the state of the art, we used the MNIST dataset as reference. MNIST is a handwritten digits dataset of 60000 training samples and 10000 test samples of 784 pixels (28×28) [4]. The first step is hence to perform the training of each of the KSOM, DSOM and PCSOM over the MNIST training dataset for the same number of iterations, and then to perform the labeling and the test. Eventhough the DSOM and PCSOM enable on-line learning, we stop it during labeling and test to assess

the performances of our three SOMs with the same number of images for training. Once again, we do not use any label for training, even if they are actually available in the MNIST training dataset, as the goal of our work is to generalize our method for unlabeled datasets and data from the real-world environment where there is no label.

The next step was to determine the number of iterations that we needed for learning in order to achieve the best accuracy. In other words, we had to set a convergence criterion that would guide us to the number of iterations we needed, and we have chosen the AQE as done in [8]. It is calculated after each iteration on the same randomly chosen subset of 10% of the training dataset following Eq. 10. The results of AQE are shown in section V-B.

$$AQE = \frac{1}{K} \sum_{i=1}^K \min_{1 \leq n \leq N} \left(\sqrt{\sum_{p=0}^{783} (image_i[p] - neuron_n[p])^2} \right)^2 \quad (10)$$

Where K is the number of input vectors used as reference for computing the AQE (we use 10% of the training dataset, so $K = 6000$), N is the total number of neurons and p is the index of the pixel in the image and the corresponding synaptic weight in the neuron. We notice that we only consider the minimum euclidean distance between the image and the neurons, i.e. the distance between the image and the BMU.

B. Labeling

Algorithm 1: Proposed labeling algorithm

```

initialization: label_count = zeros[number_of_classes];
accumulator = zeros[number_of_neurons,
number_of_classes]
for every image in the labeling subset do
  for every neuron in the SOM grid do
    dist_matrix[neuron.index] =
      dist_method(neuron.weights, image.pixels)
  best_dist = best_dist_method(dist_matrix)
  label_count[image.label] += 1
  accumulator[neurons, image.label] +=
    norm_acc_method(dist_matrix, best_dist)
for  $i$  in range(number_of_classes) do
  accumulator[neurons,  $i$ ] /= label_count( $i$ )
for every neuron in the SOM grid do
  neuron_label[neuron.index] =
    find_label(accumulator[neuron.index])

```

After convergence of learning, or after a defined time in the environment where the neural network learns and adapts, we need to go through a very important yet often neglected step when dealing with semi-supervised classification: the neurons labeling. The labeling is the step between training and test (or inference) where we assign each neuron the class it represents

in the training dataset. In our case with MNIST, each neuron has to be assigned a digit label from 0 to 9.

For this work, we proposed a generic labeling algorithm in Algo. 1, and we tried three different methods that define the functions we used in the generic algorithm: *Activation*, *Distance* and *Gaussian*. For each of the three methods, we need a labeled subset as we need to know the class of each sample of it. We randomly took a labeled subset of the training dataset, and we tried to minimize its size while keeping the best test accuracy on MNIST classification. In fact, we are often confronted to classification tasks where we have huge unlabeled datasets for training, but very few labeled samples that we can use for labeling. Hence, we try to find the labeling method that requires the minimum number of labeled samples without reducing the best accuracy. Our method can then be generalized to larger datasets with very few labeled samples.

Our generic labeling algorithm can be summarized in four steps: first, we calculate the neurons distances to the input image. Second, we choose the winning neuron or BMU. Third, each neuron accumulates its normalized distance with respect to the BMU distance in the corresponding class accumulator, and we repeat the three steps for all the labeling subset images. Fourth and finally, we choose the label of each neuron depending on the class accumulator that has responded the best. The associated functions of the three methods are detailed in Tab. I, and their impact on accuracy with different labeling subset sizes is explained in section V-C.

The parameter σ that we use in the function *dist_method* of the *Gaussian* method is the width of the gaussian that weighs the relevance of each neuron to the input image with respect to its weights distance to the image. We calculate σ before the labeling following the Algo. 2.

Algorithm 2: Proposed σ computation algorithm for gaussian labeling method

```

initialize: dist_to_origin = zeros[number_of_neurons]
for every neuron in the SOM grid do
  dist_to_origin[neuron.index] =
     $\sqrt{\sum_{i=0}^{783} (neuron.weights[i])^2}$ 
 $\sigma = \text{standard\_deviation}(\text{dist\_to\_origin})$ 

```

C. Test

After training and labeling, the remaining step is the test for accuracy that we detail in Algo. 3. Here again, we tried three test methods for inference, each of them corresponding to one of the three labeling methods, as shown in Tab. I.

V. EXPERIMENTS AND RESULTS

A. Parameters exploration

We trained our SOMs on 10 epochs over the 60000 training images of MNIST. We calculated the AQE after each epoch to check the learning convergence, as shown in Fig. 5. We explored different hyper-parameters for training and summarized those with the best accuracies in Tab. II.

TABLE I
PROPOSED LABELING AND TEST METHODS

Function		Method		
		Activation	Distance	Gaussian
Labeling	$dist_method(w, p)$	$\sum_{i=0}^{783} (w[i] \times p[i])$	$\sqrt{\sum_{i=0}^{783} (w[i] - p[i])^2}$	$e^{-\frac{\sum_{i=0}^{783} (w[i] - p[i])^2}{\sigma}}$
	$best_dist_method(d_m)$	$max(d_m)$	$min(d_m)$	$max(d_m)$
	$norm_acc_method(d_m, b_d)$	d_m/b_d	$d_m - b_d$	d_m/b_d
	$find_label(acc[i])$	$argmax(acc[i])$	$argmin(acc[i])$	$argmax(acc[i])$
Test	$dist_method(w, p)$	$\sum_{i=0}^{783} (w[i] \times p[i])$	$\sqrt{\sum_{i=0}^{783} (w[i] - p[i])^2}$	$e^{-\frac{\sum_{i=0}^{783} (w[i] - p[i])^2}{\sigma}}$
	$best_neuron_method(d_m)$	$argmax(d_m)$	$argmin(d_m)$	$argmax(d_m)$

Algorithm 3: Proposed test algorithm

```

initialization: accuracy_count = 0; confusion_matrix =
zeros[number_of_classes, number_of_classes]
for every image in the test subset do
  for every neuron in the SOM grid do
    dist_matrix[neuron.index] =
       $dist\_method(neuron.weights, image.pixels)$ 
  best_neuron =  $best\_neuron\_method(dist\_matrix)$ 
  if best_neuron.label = image.label then
    accuracy_count += 1
  update(confusion_matrix)
accuracy = (accuracy_count ÷ 10000) × 100%

```

TABLE II
TRAINING HYPER-PARAMETERS

KSOM				DSOM		PCSOM		
ϵ_i	ϵ_f	σ_i	σ_f	ϵ	η	α	η	ω
1	0.01	5	0.01	0.005	0.1	0.01	0.1	0

B. Learning convergence

To assess the learning convergence, we calculated the AQE (Eq. 10) after each iteration on the same randomly chosen subset of 10% of the training dataset. The results in Fig. 5 (averaged over 10 runs of training) show that the three SOMs learnings converge with different speeds: the KSOM converges after approximately 7 iterations, while the DSOM and PCSOM converge after only 3 iterations (the two plots are almost superposed). This is due to the absence of temporal dependency in both DSOM and PCSOM. Nevertheless, the KSOM converges to a slightly better AQE value that is reflected by a better accuracy, as described in section V-D. The AQE, though, is not directly proportional to the accuracy, as two different representations may have the same AQE but different test (generalization) accuracies.

Fig. 5. AQE results

C. Labeling methods

After training the KSOM on 10 epochs over the MNIST training dataset, we performed the labeling on the same trained network using three methods as shown in Fig. 6 (averaged over 10 different subsets of same size each time).

On the one hand, the *Activation* method does not perform well because of the confusion it creates between two different samples during labeling: for example, the activity (multiplication and activation) between the prototype neuron 1 and the two digits 1 and 7 is nearly the same, because the horizontal bar of 7 that makes the difference is multiplied by zero. Hence, the *Activation* method can hardly recognize which of the digits 1 and 7 does the prototype neuron 1 represent. This confusion may also happen for other digits where one of them is "part of the other", like the digits 6 and 8. In contrary, the *Distance* method does not cancel this divergence. On the other hand, the *Gaussian* method leads to a better accuracy than the *Distance* method, because of the distance modulation mechanism that gives more importance to the neurons whose weights are closer to the input sample during labeling.

Fig. 6. Labeling methods comparison

In order to qualitatively assess our best labeling method with minimal labeled subset size (*Gaussian* method with 1% of labeled data), we displayed the KSOM trained weights in Fig. 9 and their corresponding labels in Fig. 10. We see that we would manually do the same labeling, except for one neuron which didn't converge well (neuron in the bottom right corner of the KSOM grid, labeled as 4). We could only hesitate between some labels whose representations are very close, like 4 and 9, and this is an interesting point that we discuss further in section V-D. Hence, we tried to manually label the KSOM, modifying some labels of the *Gaussian* method: the accuracy decreased by approximately 1%. It shows that the *Gaussian* method is even better than a qualitative manual labeling.

We want to point out that the labeling is necessary for classification tasks but not for all applications. Indeed, works have been done in [18] [19] to implement a form of hardware plasticity using SOMs. The system allows then to dynamically allocate computational resources in a dynamic and multimodal environment, and the clustering is performed without labeling based on the neurons weights distances.

D. Test accuracy

We computed the test accuracy after each training epoch, as shown in Fig. 7 (averaged on 10 runs of training, same as in Fig. 5). We see that the DSOM and PCSOM learn faster than the KSOM, but the KSOM reaches the highest accuracy of $87.36\% \pm 0.23\%$. Like for STDP-based learning [13], one good property of Kohonen-based learning is the absence of over-fitting even when performed for 100 epochs, i.e. 6 million samples (accuracy of approximately 87%). Indeed, unlike many ANNs which tend to over-fit the data [13], the SOMs learning is stable over time. We calculated the average and standard deviation of the test accuracy for each of the three models over 10 runs of training (10 epochs each, with random initialization and training dataset shuffle after each epoch). The results are summarized in Tab. III.

Fig. 7. Test accuracy

Fig. 11 shows the number of times each neuron has been the BMU during test. Since the number of digits per class in the MNIST test dataset is nearly equal, the neurons which have been BMUs the most are the neurons that are the fewest to represent a class, like the digit class 1. This is an interesting behavior which shows that the more diverse the class is, the more neurons it needs to represent it. The digit 1, in contrary, does not have a high diversity, it can thus be represented by fewer neurons which are then BMUs more often.

To understand the KSOM mistakes, we plotted the confusion matrix in Fig. 8 where the left column represents the image class and the top row represents the BMU label. We notice that the biggest values occur in the diagonal, with some peak values between the digits whose representations are close:

Fig. 8. KSOM confusion matrix

Fig. 9. KSOM neurons weights

Fig. 10. KSOM neurons labels

Fig. 11. KSOM BMU counter

16.75% of the digits 9 are classified as a 4, and 14.46% of the digits 4 are classified as 9. We find the same mistakes with a lower percentage between the digits 3, 5 and 8, because of their proximity in the 784-dimensional vector space.

VI. DISCUSSION

In terms of accuracy, the KSOM outperforms the DSOM and PCSOM. However, the DSOM and PCSOM enable on-line learning and can continuously learn from a dynamic environment. The PCSOM can also enable the lateral synaptic pruning, but this mechanism did not achieve a better accuracy on MNIST classification, and that's why we disabled it ($\omega = 0$ in Tab. II). Indeed, the hyper-parameter ω is very difficult to adjust so that we have the desired amount of pruning that separates the neurons clusters. Moreover, the neurons representing the same class are not always in the same topological neighborhood, as shown in Fig. 9. This is due to the close weights distance between neurons of different classes before convergence, and to the small number of neurons we are using. Therefore, the learning with pruning is inefficient for our network. Furthermore, in order for the PCSOM to be totally dynamic, it needs to implement in addition a form of sprouting so that two neurons that would have been disconnected could reconnect to each other if the input stimuli change accordingly.

The KSOM ($87.36\% \pm 00.23\%$) achieves a better accuracy than the state of the art LMSNN ($85.71\% \pm 00.85\%$) for the same number of neurons (100). However, the LMSNN, just like the DSOM and PCSOM, enables dynamic learning, and the three models reach approximately the same accuracy. From a hardware perspective, the SNNs computation is shown to be hardware-efficient [12], but the spike-conversion of the inputs is necessary before processing. The advantage of using the LMSNN, though, is the distributed computing of the STDP learning and LIF neurons that prevent from the Von-Neuman bottleneck of centralized implementations. However, even though the STDP algorithm itself is local, the excitatory / inhibitory mechanism that allows the STDP learning convergence implies an all-to-all connectivity that is not scalable, especially for embedded applications.

Thus, we have recently proposed the iterative grid formalism in [20] to distribute the SOMs computing while keeping a local connectivity to the four neighbor neurons. The distributed SOMs mechanism is divided in two steps: the first step is the election of the BMU with respect to the current stimulus. The idea is to propagate the neurons information through the neurons grid in a certain number of iterations until the global information, i.e. the BMU and the distance of each neuron to it, emerges from the local interactions of connected neurons. The iterative grid is based on a time to distance transformation, hence implying the use of a Manhattan distance in the neighborhood function despite of the Euclidean one often used in software or centralized hardware implementations. Then, the second step is the update of the afferent weights for all neurons with respect to the results of the first step.

Therefore, we transform the connectivity complexity into a time complexity in $O(2\sqrt{n})O_c$ with respect to the number of neurons n and the computing complexity of a neuron O_c regardless of the simulated model [20]. We have computed in software the behavior of the centralized models using a Manhattan distance in the neighborhood functions for the results to be comparable with the iterative grid. As expected, the obtained behavior is the same in both cases [20]. The iterative grid formalism can hence be used to distribute any SOM-like model in a scalable cellular architecture.

We intend to use the distributed cellular neuromorphic hardware as a self-organizing computing system for embedded applications [21] in the Self-Organizing Machine Architecture (SOMA) project. We are working on defining brain-inspired computational models that can self-organize at both computation and communication levels, and have the best performances in unsupervised learning. The models will be implemented in a completely decentralized manner and then prototyped onto Field Programmable Gate Array (FPGA) devices to validate their hardware cost (resources, power, latency) efficiency.

VII. CONCLUSION AND FURTHER WORKS

In the context of unsupervised learning for image classification, we have applied the KSOM, DSOM and PCSOM

TABLE III
MNIST CLASSIFICATION ACCURACY COMPARISON WITH UNSUPERVISED LEARNING

Learning	ANN Model	Number of neurons	Labeled images for labeling		Test accuracy (%)
			Number	% of the training dataset	
STDP	SNN [13]	100	60000	100	80.71 $\pm$ 01.66
	CSNN [14]	3200	60000	100	84.23 $\pm$ 00.98
	LMSNN [15]	100	60000	100	85.71 $\pm$ 00.85
Kohonen	DeepSOM [9]	38089	3000	5	81.46
	ParalDeepSOM [11]	59664	3000	5	82.10
	KSOM	100	600	1	87.36 $\pm$ 00.23
	DSOM	100	600	1	85.19 $\pm$ 00.54
	PCSOM	100	600	1	84.53 $\pm$ 00.95

for training without labels, and proposed three labeling and test methods. The best labeling method uses only 1% of labeled data and outperforms the qualitative manual labeling in terms of accuracy. We have then confronted the Kohonen-based SOMs with STDP-based SNNs on MNIST classification, and the KSOM achieves above state of the art accuracy (87.36% $\pm$ 00.23%) with the same number of neurons (100). This result has to be confirmed for larger networks.

Nevertheless, the KSOM is not dynamic due to the time-dependency of its training hyper-parameters. Hence, the DCOM and PCSOM as well as STDP-based SNNs remain attractive due to the continuous on-line learning and the closer relationship with the brain. Overall, the SOMs applied to the iterative grid offer a better compromise than SNNs with a better accuracy using very few labeled samples for labeling, as well as a scalable neuromorphic architecture.

Our future works will focus on adding a receptive field as input for feature extraction and dimensions reduction to the SOM classifier. This "deep" structure can indeed improve the classification accuracy. In addition, we will process multimodal inputs in the same SOM for two reasons: on the one hand, it can also improve the classification accuracy by combining the same information from different sensory modalities, like the image and the sound [22]. On the other hand, it will be an interesting application to implement the hardware plasticity that we target in the SOMA project.

ACKNOWLEDGMENT

This material is based upon work supported by the French research agency (ANR) and the Swiss National Science Foundation (SNSF) through SOMA project ANR-17-CE24-0036.

REFERENCES

- [1] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," *ArXiv e-prints*, 2017.
- [2] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an integration of deep learning and neuroscience," *Frontiers in Computational Neuroscience*, vol. 10, p. 94, 2016.
- [3] Y. Bengio, T. Mesnard, A. Fischer, S. Zhang, and Y. Wu, "Stdp as presynaptic activity times rate of change of postsynaptic activity," 2015.
- [4] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [5] R. C. O'Reilly and Y. Munakata, *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*, 1st ed. Cambridge, MA, USA: MIT Press, 2000.
- [6] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [7] N. Rougier and Y. Boniface, "Dynamic self-organising map," *Neurocomputing*, Elsevier, vol. 74, no. 11, pp. 1840–1847, May 2011.
- [8] A. Upegui, B. Girau, N. Rougier, F. Vannel, and B. Miramond, "Pruning self-organizing maps for cellular hardware architectures," in *12th NASA/ESA Conference on Adaptive Hardware and Systems*, 2018.
- [9] N. Liu, J. Wang, and Y. Gong, "Deep self-organizing map for visual classification," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.
- [10] W. Melssen, R. Wehrens, and L. Buydens, "Supervised kohonen networks for classification problems," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 2, pp. 99 – 113, 2006.
- [11] C. S. Wickramasinghe, K. Amarasinghe, and M. Manic, "Parallizable deep self-organizing maps for image classification," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.
- [12] L. Khacef, N. Abderrahmane, and B. Miramond, "Confronting machine-learning with neuroscience for neuromorphic architectures design," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018.
- [13] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, p. 99, 2015.
- [14] D. J. Saunders, H. T. Siegelmann, R. Kozma, and M. Ruzsinszky, "Stdp learning of image patches with convolutional spiking neural networks," in *2018 International Joint Conference on Neural Networks*, 2018.
- [15] H. Hazan, D. Saunders, D. T. Sanghavi, H. Siegelmann, and R. Kozma, "Unsupervised learning with self-organizing spiking neural networks," in *2018 International Joint Conference on Neural Networks*, 2018.
- [16] C. Budayan, I. Dikmen, and M. T. Birgonul, "Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy c-means method for strategic grouping," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11 772 – 11 781, 2009.
- [17] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1384, 1996.
- [18] L. Rodriguez, L. Fiack, and B. Miramond, "A neural model for hardware plasticity in artificial vision systems," in *Proceedings of the Conference on Design and Architectures for Signal and Image Processing*, 2013.
- [19] L. Rodriguez, B. Miramond, and B. Granado, "Toward a sparse self-organizing map for neuromorphic architectures," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 11, no. 4, 2015.
- [20] L. Rodriguez, L. Khacef, and B. Miramond, "A distributed cellular approach of large scale SOM models for hardware implementation," in *IEEE Image Processing and Signals*, Sophia-Antipolis, France, 2018.
- [21] L. Khacef, B. Girau, N. P. Rougier, A. Upegui, and B. Miramond, "Neuromorphic hardware as a self-organizing computing system," in *IJCNN 2018 Neuromorphic Hardware In Practice and Use workshop*, Rio de Janeiro, Brazil, 2018.
- [22] K. Man, A. Damasio, K. Meyer, and J. T. Kaplan, "Convergent and invariant object representations for sight, sound, and touch," *Human Brain Mapping*, vol. 36, no. 9, pp. 3629–3640, 2015.