Self-Organizing Temporally Coded Representation Learning

Adrien Fois, Bernard Girau

Self-Organizing Temporally Coded Representation Learning

Adrien Fois[1,2] and Bernard Girau[1,2]

[1] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
[2] CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
{adrien.fois,bernard.girau}@loria.fr

Abstract. The self-organizing map (SOM) is an unsupervised learning algorithm that extracts representations from an input dataset and organizes them in a topographic manner. Nevertheless, the SOM is unable to handle event-based and asynchronous data such as spikes. This work introduces a spiking SOM that consists of a network of leaky integrate and fire neurons. Our spiking model differs from previous ones by demonstrating not only the ability to generate topographically ordered maps, but also the additional capability of vector quantization (VQ). Thus our model replicates for the first time the two key functions of SOM. To do so, we extend the VQ capabilities of a previous model by incorporating a novel neuromodulator, which enables the generation of ordered maps. We demontrate good performances on synthetic and real datasets.

Keywords: Spiking neural networks · self-organizing feature maps · temporal code · representation learning

1 Introduction

Topographically ordered maps are ubiquitous in the sensory cortex [8, 9] and are considered to be a fundamental organizational and computational principle. They are characterized by spatially close neurons sharing similar input representations, and they are created by projecting high-dimensional input data onto a low-dimensional surface (the cortex), minimizing the synaptic connections between neurons and enabling local calculations to be performed on nearby data points within the cortex [2]. This principle of local computation between neurons sharing similar representations can potentially be exploited by neuromorphic processors to increase their efficiency. These processors mimic the organization of the cortex by implementing hundreds of neurosynaptic cores, where memory (synapse) and computation (neurons) are co-localized [3]. By exploiting locality, the need for costly long-distance communications between cores can be reduced.

The canonical bio-inspired model for generating a topographically ordered map is Kohonen's SOM [6]. It consists of a vector quantization (VQ) module, which employs competitive learning to represent the current input, and a neighborhood function that enables cooperative learning among the winning neuron and its neighboring neurons. The combination of VQ and topographic

self-organization in SOM can have direct applications such as defining neural distances exploited to enable novelty detection by SOM [1].

Hardware implementations of SOM are able to preprocess and categorize the vast amount of digital data collected by embedded systems like IoT and edge computing. Implementing SOM on neuromorphic chips thus sounds promising for several embedded applications. These chips implement 3rd generation neurons [7] that communicate temporally through spikes. However, Kohonen's SOM is not suited for computing and learning with spikes, which are sparse event-based and asynchronous data. As a solution, several spiking models have been developed to replicate the functionalities of SOM [12, 11, 13, 5]. Although they exhibit some capacity to generate ordered maps, they fail to demonstrate the second crucial function of the SOM: vector quantization aiming for low reconstruction loss. Furthermore, apart from the work of [11], synaptic weights rather than delays are considered as learnable parameters to extract representations from temporal codes. In contrast to synaptic weights, delays intrinsically operate in the temporal domain. Hence, we want to use delays to store representations, as delays appear as a better candidate than weights to process and learn temporal codes.

Our work presents a novel model of spiking SOM, called Self-Organizing Temporally Coded Representation Learning (SO-TCRL). To the best of our knowledge, our model is the first to integrate the two key functionalities of Kohonen's SOM [6], namely, the ability to create topographically ordered maps and the capability of vector quantization (VQ). Our model is based on the VQ capability of [4], extended by our new neuromodulator to produce ordered maps. Neuromodulators in the brain act on sets of synapses to guide learning. Neuromodulators play a crucial role in shaping various essential properties of learning, including but not limited to the learning rate, as well as more intricate properties such as the temporal profile of STDP (Spike-Timing-Dependent Plasticity) [10]. Notably, neuromodulators can be found in neuromorphic processors, often referred to as eligibility traces [3].

Section 2 defines the main components of our model that implements the functionalities of the SOM algorithm in a network of spiking neurons, with a focus on our new neuromodulator. Several components of the model such as the temporal code and the STDP rules are based on the model of [4]. The experimental study is summarized in section 3, using synthetic and natural datasets.

2 Material and methods

This section depicts the different architectural and algorithmic components that make the neural model designed for learning representations and organizing them topographically. Our model uses a neuromodulator to regulate the learning rate of the STDP rules of [4], and its performance is assessed using both synthetic and real datasets. Emphasis is placed on the differences with the work presented by [4], with a focus on the new neuromodulator. For a more detailed analysis of the common algorithmic components of the two models, see [4].

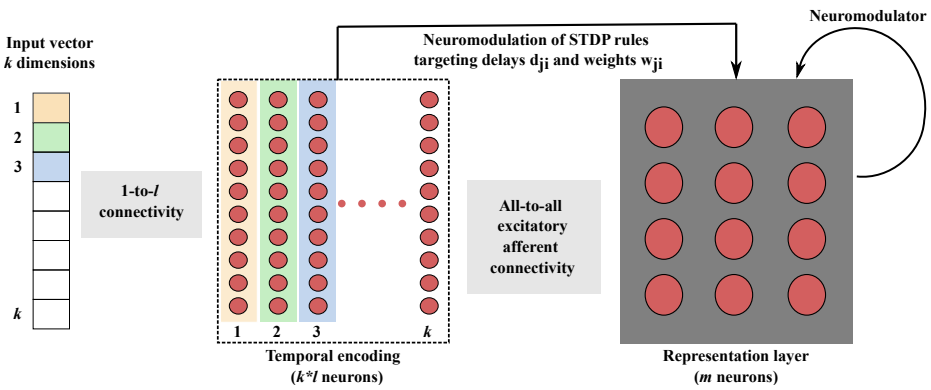

Fig. 1: Block diagram of the SNN architecture. Given a k-dimensional input vector, one dimension of the input vector is encoded by the relative firing latencies of l neurons. The sparse activity of the $k * l$ neurons is transmitted to the representation layer. Delays d_{ji} and synaptic weights w_{ji} (where i represents the index of a presynaptic neuron and j represents the index of a postsynaptic neuron) between these two layers are learned using two distinct STDP rules. Additionally, a novel neuromodulator operates based on the activity of the representation layer and modulates the learning rate of both STDP rules.

2.1 Architecture

The spiking neural network consists of two fully connected layers. The first layer encodes the input data into the relative latencies of sets of spiking neurons assigned to each input vector coordinate. The second layer extracts representations from the received spike patterns, while also possessing the ability to generate an ordered map thanks to the introduction of a neuromodulation.

These two layers are fully connected, as illustrated in Figure 1. Learning occurs between these two layers through the use of two STDP rules. One STDP rule adapts the delays d_{ji} to store representations in the temporal domain, while the other adapts the weights w_{ji} to filter the features based on their temporal variability. The learning rate of these STDP rules are modulated by our new neuromodulator.

2.2 Synapse and neuron model

Each synapse has access to one presynaptic trace $x_i(t)$ (with $i = 1, 2, \dots n$) and a postsynaptic trace $y_j(t)$ (with $j = 1, 2, \dots m$). The traces are governed by the following equations:

$$x_i(t) \leftarrow 1 \qquad \text{if } s_i(t) = 1, \qquad \tau_x \frac{dx_i(t)}{dt} = -x_i(t) \qquad \text{otherwise}$$

$$y_j(t) \leftarrow 1 \qquad \text{if } s_j(t) = 1, \qquad \tau_y \frac{dy_j(t)}{dt} = -y_j(t) \qquad \text{otherwise}$$

where $s(t)$ is an indicator function that returns 1 when a neuron emits a spike at time t, and 0 otherwise.

The neuron model is the Leaky Integrate-and-Fire (LIF). The potential $V_j(t)$ of neuron j is internally governed by a continuous evolution equation (1). Neurons of the encoding layer receive a continuous, time varying input $I_{\text{ext}}(t)$ similar to the first retinal coding stage producing analog voltages rather than discrete spikes (see 2.3). Conversely, neurons of the representation layer do not receive analog voltages from the encoding layer, i.e. $I_{\text{ext}}(t) = 0$, but are rather subject to instantaneous changes equal to the sum of the received presynaptic activities delayed by transmission delays d_{ji} and weighted by synaptic strengths w_{ji} (2). Firing is triggered when the potential reaches a threshold V_θ (3), and a potential reset is induced during a refractory period T_{refrac}:

$$\tau_m \frac{dV(t)}{dt} = -V(t) + I_{\text{ext}}(t) \tag{1}$$

$$V_j(t) \leftarrow V_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t - d_{ji}) \tag{2}$$

$$\text{if } V(t) \geq V_\theta, \text{ then } \begin{cases} s(t) = 1 \text{ (else } s(t) = 0) \\ V(u) = 0 \ \forall u \in]t, t + T_{\text{refrac}}] \end{cases} \tag{3}$$

2.3 Encoding input in spatio-temporal spike patterns

We use the same encoding procedure as presented in [4]. The input is a normalized k-dimensional vector of real numbers, with each dimension distributed across a population of $l = 10$ neurons. Each neuron within the population emits a single spike at a specific time, encoding the input value as a temporal code through a specific spatio-temporal pattern. Each neuron in the population has an associated gaussian receptive field in a circular space in range $[0, 1]$, with a center (or preferential value) μ_i and width σ. The centers are uniformly distributed between 0.05 and 0.95, while the width is constant at $\sigma = 0.6$ so that each gaussian covers the entire input interval. These preferential values used for the encoding process are then used again for decoding as illustrated in Figure 2.

2.4 From VQ to SOM : adding a new spatial neuromodulator

To address the limitation of [4], which lacks a mechanism for the self-organized generation of topographically ordered maps, we introduce a novel spatial neuromodulator that shapes the learning dynamics in the representation layer.

Each neuron j in the representation layer modulates its own learning rate in an event-driven and local manner based on its activity and the recent activity of the other neurons in the network. When neuron j fires a spike ($s_j(t) = 1$), it determines the Spiking Best Matching Unit (SBMU) as the index of the first neuron that fired for an input. To identify the SBMU, neuron j uses the postsynaptic

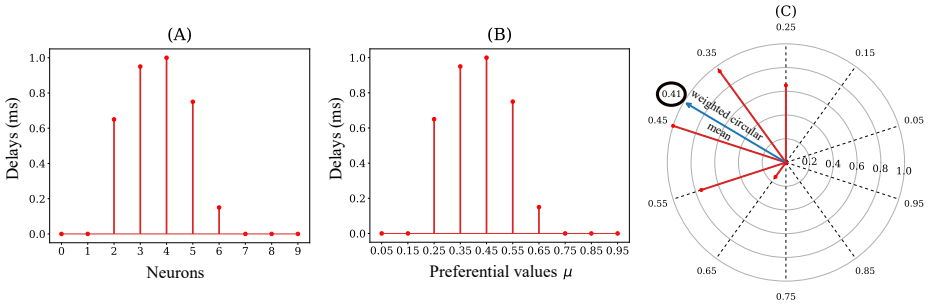

Fig. 2: Decoding process. **(A)** The relative spike timing relationships of the input spike patterns are stored in the delays. **(B)** Each presynaptic neuron has an associated preferential value μ. **(C)** The decoding process involves using a circular mean of the encoding neuron's preferential values, weighted by the delay values, to map the stored temporal representation back to the input space. In this example the value 0.41 was decoded from the delays.

traces $y_h(t)$ of all m neurons in the representation layer, where $h = 1, 2, ..., m$. The SBMU is found by identifying the postsynaptic trace with the lowest value within a time window of $3\tau_y$ after a postsynaptic spike, indicated by $y_h(t) > \epsilon$, with a threshold $\epsilon = 0.05 \approx e^{-3\tau_y/\tau_y}$.

$$\text{sbmu} = \underset{h=1,...,m}{\arg\min}\ y_h(t) \quad \text{if } y_h(t) > \epsilon \tag{4}$$

Next, the value of the spatial neuromodulator Θ_j of neuron j is determined by a Gaussian kernel that depends on the normalized Euclidean distance between the SBMU and neuron j in the map. The closer (farther) the neurons are in the map, the higher (lower) the value of the modulation. This allows two spatially close neurons to gradually learn to share similar representations in the input space. The spatial modulation can be interpreted as a static factor imposing a topological constraint on the map.

$$\Theta_j = \exp\left(-\frac{d(j, \text{sbmu})^2}{r^2}\right) \tag{5}$$

where hyperparameter r corresponds to the radius of the neighborhood centered on the position of the SBMU. A large (small) r implies a large (small) neighborhood radius.

2.5 Modulation of delay learning

After having introduced our neuromodulator, we now integrate it into the STDP rules of [4], starting with the STDP rule that targets the delays. This rule is based on two modules consisting of a vector quantization module and a regularization module. The vector quantization module is responsible for learning the

underlying structure of relative spike timings within the delays, with the goal of minimizing reconstruction loss. The regularization module, on the other hand, aims to promote small delay values in order to prevent the emergence of unnecessary large delays. By integrating the neuromodulator into the STDP rule that adapts the delays, we obtain the following rule with two adaptation cases:

$$\Delta d_{ji} = \begin{cases} \Theta_j \cdot \alpha^+ \Big(-\tau_x \ln\big(x_i(t)\big) - (d_{ji} + \lambda d_{ji}) \Big), & \text{if } s_j(t) = 1 \text{ and } x_i(t) > \epsilon \\ -\Theta_j \cdot \alpha^- \Big(-\tau_y \ln\big(y_j(t)\big) \Big), & \text{if } s_i(t - d_{ji}) = 1 \text{ and } y_j(t) > \epsilon \end{cases} \tag{6}$$

As the neuromodulator Θ_j falls in range $]0, 1]$, the learning rates in our STDP rule that adapts delays now vary between $]0, \alpha^+]$ and $[-\alpha^-, 0[$, respectively.

Note that the second adaptation case for a postsynaptic neuron j is triggered if this neuron j has previously emitted a postsynaptic spike in a time window set by $y_j(t) > \epsilon$ relative to the current time t. This implies that the use of the neuromodulator Θ_j is valid because Θ_j has previously been updated in a temporal proximity, at the time of the postsynaptic spike emission by neuron j.

2.6 Modulation of weights learning

The other STDP rule assigns relevance weights to the features by estimating the temporal variance of the features. A high (low) temporal variance induces a small (large) relevance weight. We apply the neuromodulator to all adaptative mechanisms in the network. Therefore, by integrating the neuromodulator into the STDP rule adapting the synaptic weights, we obtain the following rule with two adaptation cases:

$$\Delta w_{ji} = \begin{cases} \Theta_j \cdot \beta^+ \big(\exp\big(-\frac{v_{ji}}{\sigma^2}\big) - w_{ji} \big), & \text{if } s_j(t) = 1 \text{ and } x_i(t) > \epsilon \text{ and } e_{ji} \geq 0 \\ -\Theta_j \cdot \beta^- \big(1 - y_j(t)\big), & \text{if } s_i(t - d_{ji}) = 1 \text{ and } y_j(t) > \epsilon \end{cases} \tag{7}$$

Again, since the neuromodulator Θ_j fall in range $]0, 1]$, the learning rates in our STDP rule that adapts the weights now vary between $]0, \beta^+]$ and $[-\beta^-, 0[$, respectively. The local temporal error $e_{ji} = -\tau_x \ln\big(x_i(t)\big) - d_{ji}$ is already calculated and available in equation 6.

The use of the neuromodulator in the second adaptation case is valid for the same reasons as in the STDP rule that targets the delays.

When the constraints of the first adaptation case are satisfied, not only is the first adaptation case triggered, but also the online event-based estimation of the temporal variance v_{ji}. The adaptation rate of the exponentially moving variance thus also depends on the neuromodulator. A low value of Θ_j implies a low update of the variance v_{ji}, allowing for relative stability. For example, a

neuron spatially distant from the SBMU receives a low neuromodulation value, so that it remains locked onto the region of the input space that it clusters.

$$v_{ji} = (1 - \Theta_j \cdot \alpha^+ \cdot \gamma) \cdot (v_{ji} + \Theta_j \cdot \alpha^+ \cdot \gamma e_{ji}^2) \tag{8}$$

3 Experiments and results

In this section, we carry out a set of experiments to assess the expected features of a comprehensive SOM model, which include the ability to generate ordered maps and to perform vector quantization (VQ). In our experiments, we employed a toric topology for the map to suppress border effects.

Hyperparameters are fixed for all experiments except for the firing threshold V_θ of the neurons in the representation layer. We use a simple and effective method to automatically determine its value, as explained below.

We first evaluate the topographic ordering capability of our model using a synthetic dataset by measuring the preservation of distances between data points as distances between the positions of the neurons representing them. Next, we evaluate the model's ability to perform VQ and to preserve local neighborhood relationships between neurons in their code vectors, which promotes sharing of similar representations among neighboring neurons.

3.1 Parameters of the SNN

We propose a generic parameterization of the SO-TCRL model that can be applied to any input vector dimension k. The only dimension-dependent parameter is the neuron spiking threshold V_θ in the representation layer. Since each input dimension is encoded by a population of $l = 10$ neurons, the total number of emitted spikes is $k * l$. To ensure that a neuron fires in the representation layer, its spiking threshold V_θ must be set to $c * k * l$, where $c \in [0, 1]$ is a coefficient to be determined. We assume for convenience a linear relationship between $k * l$ and V_θ. Using an optimization method, we found as optimal value $c = 0.44$.

The parameters we use for the SO-TCRL model are provided in Table 1.

3.2 Metrics

We use three metrics to assess the quality of the generated map.

Root Mean Squared Error We use the Root Mean Squared (RMS) error to quantify the quality of the learned representations. It quantifies the difference between an input vector $\mathbf{a}_p$ and the associated code vector decoded from the synaptic delays of the SBMU $\hat{\mathbf{a}}_p$:

$$\text{RMS} = \frac{1}{P} \sum_{p=1}^{P} \sqrt{\frac{1}{k} \sum_{i=1}^{k} (a_{i,p} - \hat{a}_{i,p})^2} \tag{9}$$

Here, k is the input dimension and P is the number of input patterns, and $a_{i,p}$ is the ith coordinate of a_p.

Table 1: Parameters of SO-TCRL used in all simulations.

Neuronal parameters for ...							
... the encoding layer					... the representation layer		
V_θ	τ_m	T_{refrac}			V_θ	τ_m	T_{refrac}
0.5	10.0 ms	6 ms			0.44 kl	5.3 ms	6ms
Synaptic parameters					Neuromodulator parameter		
τ_x	τ_y	$d_{\min}$	$d_{\max}$				r
4 ms	3 ms	0 ms	10 ms				0.10
Parameters of the STDP rules							
λ	α^+	α^-	β^+	β^-	γ	σ	ϵ
0.58	0.07	0.042	0.18	0.036	0.24	10.0 ms	0.05

MDN Like Kohonen's SOM, the SO-TCRL model aims to maintain local neighborhood relations between neurons in their code vectors, creating a kind of local continuity in the map. We use the Mean Distance to Neurons (MDN) to quantify this property, which aggregates distances between a neuron's code vector and its neighboring neurons' code vectors. In our case each neuron has four neighbors. This measure is intrinsic to the map, meaning that it does not require an external reference. The MDN is formulated as follows:

$$\text{MDN} = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\begin{cases}||c_k - c_j||^2, & \text{if } \text{dist}(j,k) = 1\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

This formulation is based on code vectors c_j decoded from each neuron j with a weighted circular mean (see Fig 2).

EMDS The MDN is a local and intrinsic measure based on the code vectors of neurons, whereas the EMDS (expectation of multi-dimensional scaling measure) is a global and extrinsic measure based on the spatial coordinates of neurons on the map. While the MDN evaluates the preservation of neighborhood relations in the code vectors, the EMDS evaluates the ability to preserve distances between input data points on the map. The EMDS metric is based on a calculation of dissimilarity between the distance in input space for two data points and the spatial distance of the two neurons representing them. A value of 0 indicates perfect topological preservation. Distances in input space and on the map are normalized for comparison. The EMDS metric is given by:

$$\text{EMDS} = \frac{2}{P(P-1)}\sum_{p=1}^{P}\sum_{j<p}\Big(F(a_p,a_j) - G\big(M(a_p),M(a_j)\big)\Big)^2 \tag{11}$$

where P is the number of input vectors. F and G are similarity measures in input space and map space, respectively. $F(a_p, a_j)$ measures the similarity

between a pair of input vectors a_p and a_j using a normalized Euclidean distance. These two input vectors a_p and a_j are represented by two neurons on the map, whose spatial positions are given by $M(a_p)$ and $M(a_j)$. $G\big(M(a_p), M(a_j)\big)$ measures the similarity between these two positions $M(a_p)$ and $M(a_j)$ using a normalized Euclidean distance.

3.3 Numerical Tests

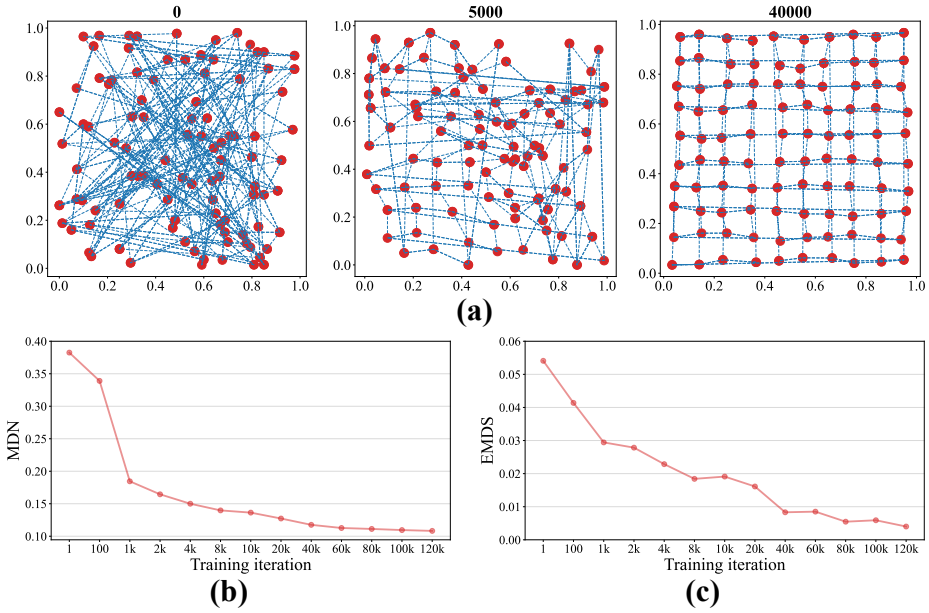

Fig. 3: Results for 2-D input. (a) Generation of an ordered map with a toroidal topology after 0, 5000, 40000 training iterations. The code vectors of the neurons are decoded and projected onto the 2-D input space and displayed as red dots. The connections of the neurons with their four neighboring neurons are depicted using blue dotted lines. (b)-(c) Model performance through training iterations in terms of (b) MDN and (c) EMDS.

2-D input First, we evaluate the ability of the model to generate topographically ordered maps using EMDS and MDN metrics. We perform a controlled experiment, based on the one described in [13], which aims to reach a theoretical global minimum of 0 for the EMDS value. We use a 10x10 two-dimensional map with uniformly spaced 10*10 points in the $[0, 1]^2$ interval as the dataset. This setup allows us to project any distance between two input points onto the spatial distance between two neurons.

We train the SO-TCRL with 120,000 randomly selected data points. The delays between the encoding layer and the representation layer are randomly

initialized within the range of $[0, 0.4]$ ms using a normal distribution centered on 0.2 ms with a standard deviation of 0.1 ms. The synaptic weights are initialized to their maximum values of 1. We evaluate the map quality obtained from 100 independent runs.

The EMDS and MDN measures consistently decrease with the number of training iterations. The EMDS (Fig. 3c) decreases until reaching a plateau after 80,000 iterations around a value of 0.005, indicating that the spatial distance between the two SBMUs selected for two data points converges to the distance between those two points in the input space. This shows that the map learns in a self-organized manner to preserve the distances between data points in the spatial distances of neurons. The mean EMDS after the training phase is 0.00521 ± 0.00663, equivalent to 0.00554 ± 0.00483 as reported by [13] (Tab. 2).

Table 2: Comparison of mean model performances.

Method	RMS Recon. Error	EMDS
Fois et al. [4]	0.06	-
Rumbell et al. [13]	-	0.00554
SO-TCRL	**0.05**	**0.00521**

At 80,000 iterations, the MDN (Fig. 3b) reaches a plateau at 0.11, revealing that neighboring neurons possess comparable representations. This metric can be related to the global input data distribution, where uniform spacing of 0.10 in each dimension specifies a theoretical ideal MDN value of 0.10 ± 0.00. The achieved MDN value is 0.11 ± 0.01, close to the ideal 0.10 ± 0.00.

Fig. 3a shows the successful unfolding of maps during learning iterations. The neurons' code vectors learned not only the uniform data distribution but also shared similar representations when located close to each other in a circular space. The map topology is toric, and encoding neurons have receptive fields located in a circular space, where extremal values are equivalent. The linear representation of Fig. 3a shows that code vectors close to the extremes of one dimension of the input space are also close in a circular space, resulting in connections between opposite sides of the input space.

Natural image dataset We now evaluate the SNN's ability to perform VQ and generate local continuity using non-uniform and higher-dimensional dataset.

For that purpose we use the natural image dataset used in [4]. Input normalization is restricted to $[0.05, 0.95]$ due to the projection of linear input data onto a circular space where extremal values 0 and 1 would become equivalent. Patches of 4x4 pixels extracted from 512x512 natural images are used as input vectors, with 60,000 patches provided during the training phase.

Both the RMS reconstruction error and MDN metric decrease as the SNN learns to compress input data distribution and reduce the distance between code

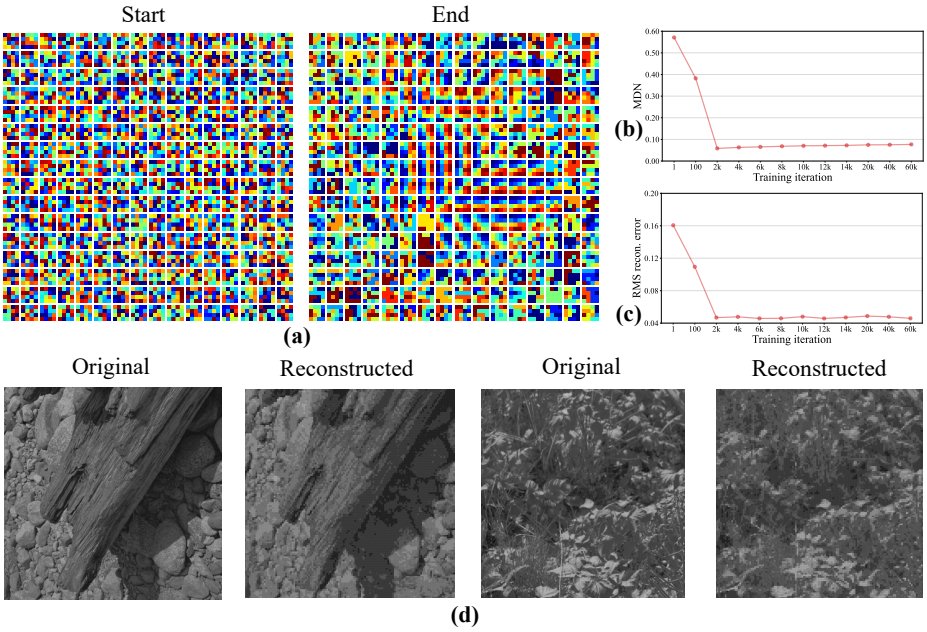

Fig. 4: Results for the natural image dataset with $m = 256$ neurons in the representation layer. (a) Code vectors at random initialization and after the training phase. The blue-red gradient represent minimum-maximum values. (b)-(c) Model performance through training iterations in terms of (b) MDN (ideal minimum is 0.10), and (c) RMS reconstruction error. (d) Natural images are presented as input to the network and reconstructed by the decoded code vectors.

vectors of neighboring neurons. Fig 4c and Fig 4b show the decrease in RMS reconstruction error and MDN respectively.

The code vectors in the representation layer have become selective to various visual orientations, see Fig. 4a. These orientations are arranged in an orderly manner on the map, with spatially close neurons sharing similar orientations.

Finally the reconstruction of natural images from the code vectors of the representation layer produces images of high quality, comparable to the original natural images, as shown in Fig. 4d. Experiments for other databases or network sizes have been carried out and show similar results.

4 Discussion

This paper presents SO-TCRL, a self-organizing model for representation learning based on temporally coded data handled by spiking neurons. To be best of our knowledge, SO-TCRL is the first complete model of spiking self-organizing map able to combine the two key functions of a self-organizing map, vector quantization, and the creation of topographically organized maps on generic datasets.

Furthermore, unlike in [13], our model does not need any manual adjustment of the maximum synaptic weight magnitude based on input data dimension.

Our main contribution is a novel spatial neuromodulator that enhances the VQ capacity of [4] by incorporating the ability to generate ordered maps. Unlike related works that use a lateral excitation-inhibition profile to influence spike timing, our neuromodulator regulates the learning rates of neurons to create orderly maps.

The SO-TCRL model was subjected to an experimental evaluation to test its key functions, and we found that it performed comparably to the state-of-the-art works by [4] and [13], combining the benefits of both models : a low reconstruction error and a low mapping error. As a future research direction, we now plan to investigate hierarchical architectures based on the SO-TCRL model for clustering hierarchical data.

Acknowledgments This work has been supported by ANR project SOMA ANR-17-CE24-0036.

References

1. Bernard, Y., Hueber, N., Girau, B.: Novelty Detection in Images Using Vector Quantization with Topological Learning. In: IEEE Int. Conf. on Electronics Circuits and Systems (ICECS) (2020)
2. Chklovskii, D.B., Koulakov, A.A.: Maps in the brain: What can we learn from them? Annual Review of Neuroscience (2004)
3. Davies, M., Srinivasa, N., Lin, T.H., et al.: Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro (2018)
4. Fois, A., Rostro-Gonzalez, H., Girau, B.: Unsupervised learning of visual representations using delay-weight spike-timing-dependent plasticity. In: 2022 International Joint Conference on Neural Networks (IJCNN) (2022)
5. Hazan, H., Saunders, D.J., Sanghavi, D.T., et al.: Lattice map spiking neural networks (LM-SNNs) for clustering and classifying image data. Annals of Mathematics and Artificial Intelligence (2020)
6. Kohonen, T.: Essentials of the self-organizing map. Neural Networks (2013)
7. Maass, W.: Networks of spiking neurons: The third generation of neural network models. Neural Networks pp. 1659–1671 (1997)
8. Moser, E.I., Roudi, Y., Witter, M.P., et al.: Grid cells and cortical representation. Nature Reviews Neuroscience (2014)
9. Ohki, K., Chung, S., Kara, P., et al.: Highly ordered arrangement of single neurons in orientation pinwheels. Nature (2006)
10. Pawlak, V., Wickens, J., Kirkwood, A., et al.: Timing is not Everything: Neuromodulation Opens the STDP Gate. Frontiers in Synaptic Neuroscience (2010)
11. Pham, D., Packianather, M., Charles, E.: A self-organising spiking neural network trained using delay adaptation. In: IEEE Int. Symposium (2007)
12. Ruf, B., Schmitt, M.: Self-organization of spiking neurons using action potential timing. IEEE Trans. Neural Networks (1998)
13. Rumbell, T., Denham, S.L., Wennekers, T.: A Spiking Self-Organizing Map Combining STDP, Oscillations, and Continuous Learning. IEEE Transactions on Neural Networks and Learning Systems (2014)