A Spiking Neural Architecture for Vector Quantization and Clustering

Adrien Fois, Bernard Girau

A Spiking Neural Architecture for Vector Quantization and Clustering

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. Although a couple of spiking neural network (SNN) architectures have been developed to perform vector quantization, good performances remains hard to attain. Moreover these architectures make use of rate codes that require an unplausible high number of spikes and consequently a high energetical cost. This paper presents for the first time a SNN architecture that uses temporal codes, more precisely first-spike latency code, while performing competitively with respect to the state-of-the-art visual coding methods. We developed a novel spike-timing-dependent plasticity (STDP) rule able to efficiently learn first-spike latency codes. This event-based rule is integrated in a two-layer SNN architecture of leaky integrate-and-fire (LIF) neurons. The first layer encodes a real-valued input vector in a spatio-temporal spike pattern, thus producing a temporal code. The second layer implements a distance-dependent lateral interaction profile making competitive and cooperative processes able to operate. The STDP rule operates between those two layers so as to learn the inputs by adapting the synaptic weights. State-of-the art performances are demonstrated on the MNIST and natural image datasets.

Keywords: Neural network models · Self-organizing map · Vector quantization · Temporal coding · Representation learning

1 Introduction

Spiking Neural Networks (SNNs) have gained an increasing attention in the recent years and have been used to perform supervised and unsupervised learning [17, 4] tasks. From a neuromorphic perspective, SNNs offer the advantages of energetic and communication efficiency [15]. Indeed, they don't need to produce and send real-valued outputs at each iteration. Instead they sparsely emit binary events, the so-called spikes. This event-driven message passing scheme greatly alleviates the communication channels as compared to the clock-driven communication of traditional neural network implementations. Further energetic and communication improvements can be obtained by making the memory storage (the synapses) locally accessible to the computational unit (the neurons). This hardware architectural scheme can then be fully exploited at the algorithmic level

by the use of local learning rules such as STDP (spike timing dependent plasticity). Local learning rules are leveraging information available at the presynaptic terminal (or synaptic input) and the postsynaptic cell membrane (or synapse output). This paper follows this algorithmic line of work. We propose a SNN architecture that incorporates a novel STDP rule operating locally in space and time to perform vector quantization and clustering.

The paper is organized as follows. In section 2, we rapidly situate this work with respect to vector quantization, self-organizing maps and spiking neural networks. The architecture and learning method of our model is described in section 3, and the experimental results are presented in section 4.

2 Background

Vector quantization (VQ, see [19]) consists in approximating the probability density of a possibly highly multi-dimensional input space with a finite set of prototype vectors, often called codewords, the set of codewords being called the codebook. Many VQ algorithms exist such as k-means [10], self-organizing maps (SOM) [8], neural gas (NG) [12], growing neural gas (GNG) [5] or growing when required (GWR) [11]. The well-known k-means method provides a codebook without any internal structure, meaning that no additional knowledge is provided with respect to the similarities between codewords. Algorithms such as NG, GNG or GWR create a codebook as a network of codewords and connections between codewords. These connections stand for significant similarities between codewords, thus inducing the codebook structure that derives from the learned data. On the contrary, SOM are based on a static underlying topology and a fixed number of codewords (neuron weight vectors). Each codeword has an associated position in the underlying map that is usually a 2D lattice, and the learning process captures the topographic relationships between the inputs: similar input samples are represented by the weights of spatially close neurons in the map.

All above mentioned neural models for VQ use analog neurons. More biologically inspired computational paradigms recently attract more and more attention. The so-called spiking neurons mimic the neurons in the brain that communicate thanks to action potentials. Their success is tightly linked to the recent trend towards neuromorphic processors (such as IBM TrueNorth or Intel Loihi) or sensors (such as DVS event cameras) that make use of spikes. Our work aims at defining an efficient spiking model for VQ inspired by SOM. Other works already tackles the problem of vector quantization in SNN with local rules [18, 3, 7]. However, they make use of rate coding. Rate coding is energetically expensive as it encodes a continuous variable in the spike rate of a neuron. The higher the value the higher the rate. Furthermore, rate-coding is not an efficient coding scheme for non-stationary data, that are the norm rather than the exception in real-world applications. Indeed, the firing rate is defined as a limit that theoretically involves an infinite number of spikes [2]. Getting a reliable firing rate estimate then requires to integrate enough spikes either with a large temporal window or with a high number of spiking neurons [4]. Consequently temporal

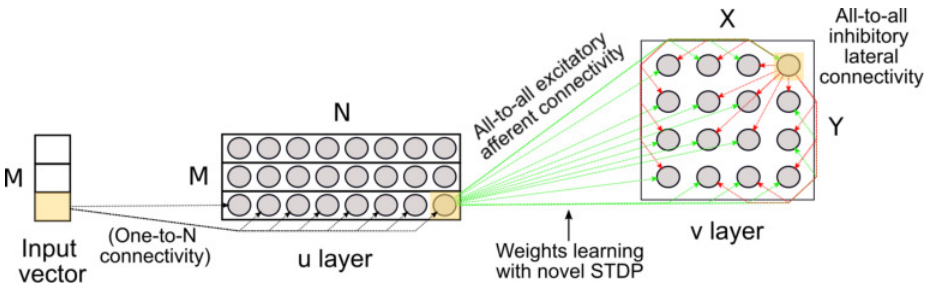

Fig. 1. Spiking neural network architecture.

codes are often preferable due to their biological plausibility and energetical efficiency. Contrary to rate codes, temporal codes leverage the precise spike times to encode continuous variables. Among temporal codes we can cite first-spike latency code that uses the precise spike time and rank-order-code, which can be considered as a special case as it only leverages the rank of the spikes.

Our model performs vector quantization on the basis of a temporal coding of data. It also addresses other common limitations of SNN architecture such as the lack of scalability: changing input data dimensionality often requires an additional fine tuning of parameters. To sum up, we propose here a new SNN model for fast and efficient vector quantization and clustering, that differs from previous works by making use of a new STDP rule for temporal codes, more precisely first-spike latency code, and by being easily scalable to N input data dimensions, without additional parametric fine-tuning.

3 Methods

Our SNN architecture is composed of two layers of LIF neurons (leaky integrate and fire). The first layer u encodes a vector of continuous variables into spike times, thus producing a temporal code. This layer is fully connected to the second layer v that we call the spiking SOM. Layer v performs vector quantization in a way that is conceptually similar to the initial SOM algorithm [8]. The first neuron that spikes with respect to an input vector is considered to be the best matching unit (BMU), i.e. the neuron that is best tuned to the current input vector. This spike triggers the learning of the input vector by its afferent synaptic weights with our new STDP rule. The spike produced by the BMU also influences the other neurons in layer v via a distance-dependent lateral interaction profile that mimics the usual neighbourhood kernel in the usual 2D underlying topology of a Kohonen SOM. Spatially close neurons will tend to spike in close temporal proximity and thus learn the input vector, while more distant neurons will not spike and therefore not learn it. By iterating this process, the map gets gradually topologically ordered, i.e. vectors that are close in the input space will be represented by spatially close neurons in layer v.

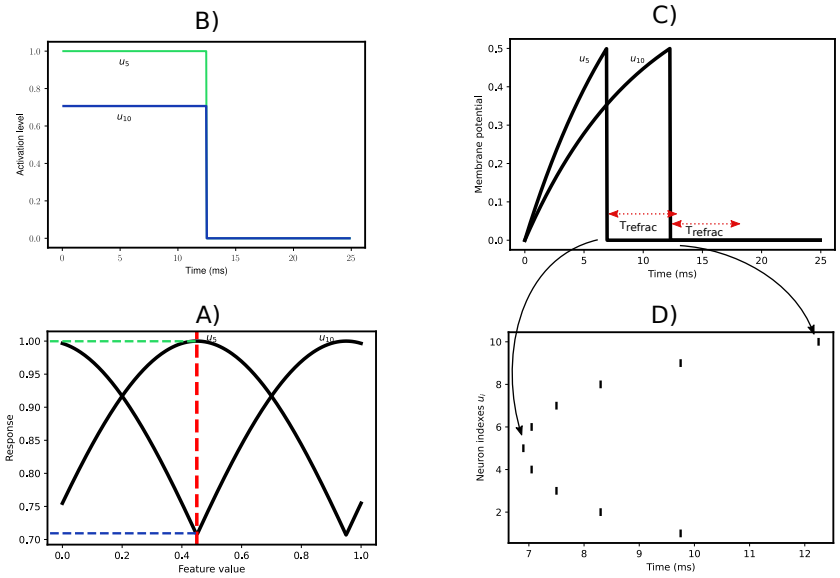

Fig. 2. A feature value of 0.45 is encoded by a spatio-temporal spike pattern. A) A bank of 10 neurons represent this feature. Each neuron u_i of that bank has a gaussian tuning curve centered aroung a preferential value. u_5 and u_{10} have 0.45 and 0.95 as preferential values, respectively. The tuning curves of u_5 and u_{10} and their resulting activation levels for an input value of 0.45 are plotted. B) These activations levels are constant for an interval of time of 12.5 ms and are followed by a quiescent period of the same duration. C) The activation levels are integrated by the membrane potentials of u_5 and u_{10}. The membrane potential of u_5 grows faster than for u_{10} as the input is higher. Consequently u_5 reaches its threshold faster, is resetted and enters in a refractory period. D) Each neuron u_i emits a spike when it crosses it's threshold, thereby producing a spatio-temporal spike pattern at a mesoscopic level.

The architecture was simulated using the BRIAN2 simulator. The following sections more precisely describe the different components of our model.

3.1 Neuron and Synapse Model

The network only contains leaky integrate-and-fire (LIF) neurons. This one-variable model captures the basic behavior of a biological neuron while maintaining a low computational cost and analytical tractability. The dynamic of a LIF neuron is given by

$$\begin{aligned} \tau_m \frac{dV}{dt} &= -V(t) + E(t) \\ V(t) &\leftarrow V_{\text{reset}}, \text{ if } V(t) \geq \theta \end{aligned} \tag{1}$$

where τ_m is the membrane time constant, $V(t)$ is the membrane potential, and $E(t)$ is the input to the neural membrane at time t. When the neuron's membrane potential $V(t)$ crosses its membrane threshold θ, the neuron instantaneously emits a spike and the membrane potential is reset to V_{reset}. The neuron then enters an absolute refractory period T_{refrac} during which it cannot emit a new spike as the membrane potential stops being integrated.

The synaptic transmission is modeled with an instantenous jump of the post-synaptic potential (PSP) by the synaptic weight when a spike is emitted by a presynaptic neuron, followed by an exponential decay. This exponential decay can be expressed by a linear differential equation. Rather than modeling the synaptic transmission with one differential equation per synapse, only one differential equation is needed for the whole set of synapses, thanks to the superposition principle of linear systems. This reduces the computational load significantly.

$$\begin{aligned} E_j(t) &\leftarrow E_j(t) + \sum_{i=1}^{I} s_i(t) w_{ij}(t), && \text{if a presynaptic neuron } i \text{ spikes} \\ \tau_f \frac{dE_j}{dt} &= -E_j(t), && \text{otherwise} \end{aligned} \tag{2}$$

where i and j index the pre- and post-synaptic neurons, respectively. $E_j(t)$ is the input potential of postsynaptic neuron j. $s_i(t)$ is an indicator function returning the value 1 when a presynaptic neuron i spikes at time t, 0 otherwise. $w_{ij}(t)$ is the synaptic weight between pre-synaptic neuron i and post-synaptic neuron j, and τ_f is the fall time constant.

3.2 Input Encoding

Continuous input variables need to be encoded into spike times in layer u. This can be achieved by distributing a variable x over a bank of N neurons (see Fig. 1), where each of the $i \in 1, ..., N$ neuron of that bank has a receptive field size σ and is tuned to a preferential value μ_i in a periodic space. In our experiments $N = 10$ and all neurons used an identical $\sigma = 0.6$ while the prefential values μ_i were equally spaced between 0.05 and 0.95. A wrapped gaussian activation function $G(\sigma, \mu_i, x)$ is then used to compute the input potential of each neuron. The resulting periodic tuning curves cover the entire interval of variation of the normalized input variable. The neuron whose preferential value is the closest to the current input value will get the highest input, and will thus spike first. The other neurons of the bank then spike in an order that is determined by the distance between the input value and their preferential values (see Fig. 2). In this way a continuous variable is encoded in a spatio-temporal spike pattern. This coding scheme can be easily generalized for an input vector of M dimensions, where each dimension is independently encoded by a corresponding bank of N neurons in layer u, as inspired by [1].

3.3 Learning Rule

STDP rules that are not weight-dependent result in the steady-state in a bimodal distribution, where the synaptic weights are either fully potentiated, or fully depressed, as shown by Rossum [14]. With our rule, the learned distribution is unimodal and thus the interval of variation of the synaptic weights is fully leveraged to learn the inputs. This makes possible an easy subsequent decoding step.

To learn the inputs encoded in first-spike latency codes, we developed an event-based plasticity rule derived from a vector-quantization criterion. It takes the form of a novel STDP rule operating 1) locally in space with a single pair of pre and postsynaptic neuron i and j, and 2) locally in time within a temporal learning window. STDP rules are characterized by a weight change Δw_{ij} that is a function of the time difference $\Delta_t = t_{post} - t_{pre}$ between the post and presynaptic spikes, respectively. The magnitude of the weight change Δw_{ij} usually depends on the term $e^{-\Delta_t/\tau}$. Two different time constants τ_+ and τ_- account for $\Delta_t > 0$ and $\Delta_t \leq 0$, respectively and determine the width of the temporal learning window. To keep computation efficient and local, we implemented an online version of STDP. Each synapse has two local state variables x_{ij} and y_{ij}. These variables serves as memories of recent pre and postsynaptic activities, respectively. When a pre (post) synaptic spike is emitted, $x_{ij}(t)$ ($y_{ij}(t)$) is set to 1 and then decays exponentially with a time constant τ_+ (τ_-) to 0. In this way we reproduce the behavior of $e^{-\Delta_t/\tau}$.

$$\begin{aligned} x_{ij}(t) &\leftarrow 1, && \text{if neuron } i \text{ spike} \\ \tau_+ \frac{dx_{ij}}{dt} &= -x_{ij}(t), && \text{otherwise} \\ y_{ij}(t) &\leftarrow 1, && \text{if neuron } j \text{ spike} \\ \tau_- \frac{dy_{ij}}{dt} &= -y_{ij}(t), && \text{otherwise} \end{aligned} \tag{3}$$

The weight change Δw_{ij} of a synapse is then computed as a function of $x_{ij}(t)$ and $y_{ij}(t)$. The weight is updated by that change such that $w_{ij}(t) \leftarrow w_{ij}(t) + \Delta w_{ij}$. The final learning rules read:

$$\Delta w_{ij} = \begin{cases} \alpha_+\big(1 - x_{ij}(t) - w_{ij}(t) + w_{\text{offset}}\big), & \text{if neur. } j \text{ spikes and } x_{ij}(t) > \theta_{\text{stdp}} \\ \alpha_-\big(1 - y_{ij}(t)\big), & \text{if neur. } i \text{ spikes and } y_{ij}(t) > \theta_{\text{stdp}} \end{cases} \tag{4}$$

and are combined with the hard bounds $0 \leq w_{ij} \leq 1$. α_+ and α_- are the learning rates. w_{offset} is a positive offset that shifts the attracting fixed point up. We will discuss about the fixed point in the next paragraph. If $x_{ij}(t)$ ($y_{ij}(t)$) is smaller than the threshold θ_{stdp}, then it does not lead to a change. This is done to account for the time elapsed between the presentation of two successive input patterns. Without this condition, the memories $x_{ij}(t)$ (y_{ij}(t)) decaying to 0 after a first input presentation would automatically lead to a maximal weight change when a new input pattern is presented.

In the first adaptation case - that corresponds to $\Delta_t > 0$ - we can observe that the learned weight is an attractive fixed point by analyzing the equilibrium solution :

$$\begin{aligned} & \Delta_{w_{ij}} = 0 \\ \Leftrightarrow\ & 0 = 1 - x_{ij}(t) - w_{ij}(t) + w_{\text{offset}} \\ \Leftrightarrow\ & w_{ij}(t) = 1 - x_{ij}(t) + w_{\text{offset}} \end{aligned} \tag{5}$$

Consider a pattern that is repeatedly presented as an input. w_{ij} will converge to a value that depends on the memory $x_{ij}(t)$ and thereby on Δ_t. The higher Δ_t, the higher the learned value, as the memory $x_{ij}(t)$ relaxes to 0 for large Δ_t values. Thus the first presynaptic neuron that spike will have the highest weight and the last one that spike will have the lowest weight (see Fig. 3). This process happens for each pair of pre and postsynaptic neuron that meet the conditions of the first adaptation case. This behavior is reminiscent of first-spike latency code or of rank-order coding which can be considered as a special case of the former.

The second adaptation case - which corresponds to $\Delta_t \leq 0$ - implements depression. The weight gets depressed by a magnitude that grows with Δ_t, as the memory $y_{ij}(t)$ relaxes to 0 for large Δ_t. Thus in the limit, the weights w_{ij} takes the value 0, as the lower bound for the weight is set to 0.

3.4 Lateral Interactions

We describe in this section the lateral interaction profile in layer v, where the neurons are arranged in a grid. Instead of using a high global inhibition to implement a Winner-Take-All (WTA) behavior, we use a distance-dependent lateral interaction profile, similar to the Kohonen SOM [8]. This profile implements weak short-range inhibition and strong long-range inhibition [6]. In this way, spatially close neurons will tend to spike together and represent close input vectors, while distant neurons will not spike together and thus will learn uncorrelated codes. This also tends to accelerate the learning process, as a population of neurons spike in response to an input vector rather than a single neuron. The lateral interaction profile in layer v reads

$$w_{ij} = \begin{cases} c_{\min}\|i-j\|, & \text{if } \|i-j\| \leq r \\ c_{\max}, & \text{otherwise} \end{cases} \tag{6}$$

where w_{ij} is the lateral connection weight between the neurons at locations i and j in the grid. $c_{\min}$ scales the inhibitory level for neurons whose distance is less than a radius r, otherwise w_{ij} takes a maximum value $c_{\max}$.

In the learning phase of the Kohonen SOM, the interaction radius progressively decreases so as to first organize the map and then make the neurons learn more and more individualized receptive fields. When the interaction radius is equal to zero, we get a WTA behavior. We mimic this behavior by increasing

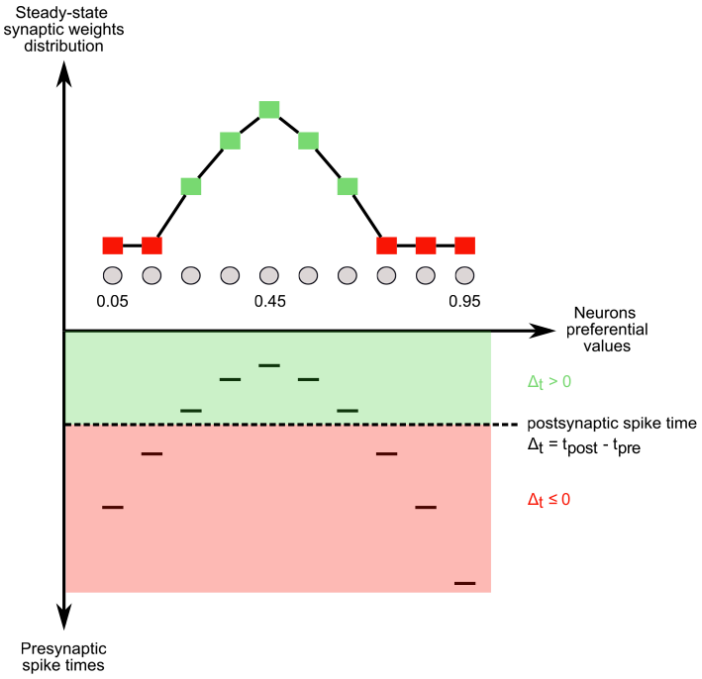

Fig. 3. Qualitative sketch of the steady-state learned weights when presenting an input value of 0.45. The value gets encoded in a spatio-temporal spike pattern with a bank of 10 neurons in layer u. The neuron whose preferential value is the closest to the input spikes first, followed by the other neurons of that bank. At some time, a postsynaptic spike is emitted by a neuron in layer v, here after the 5th spike and before the 6th spike. The sign of the time difference Δ_t between a post and presynaptic spike delimits two learning regimes for the STDP rule. If $\Delta_t > 0$, then the learned weight is an attracting fixed point. If $\Delta_t \leq 0$ the weight gets depressed by a magnitude that grows with Δ_t, and finally the weight tends to 0.

the lateral connection weights w_{ij} during the training phase. By increasing the inhibition level, we approach a WTA behavior. The growing inhibition during the training phase is implemented by the following homeostatic mechanism:

$$\tau_w \frac{dw_{ij}}{dt} = c_{\max} - w_{ij} \tag{7}$$

Thus w_{ij} will exponentially converge to the target value $c_{\max}$ with a timescale τ_w. In our experiments τ_w is equal to the period of the training phase.

4 Experiments and Results

This section presents the results of the conducted experiments to evaluate the performance of our architecture in terms of reconstruction error. The data are

Table 1. Network parameters used in all simulations

Neuronal parameters, used in (1)							
dt	τ_m^u	τ_m^v	$V_{\text{reset}}^{u,v}$	θ^u	θ^v	T_{refrac}^u	T_{refrac}^v
0.1 ms	10.0 ms	1.3 ms	0.0	0.5	1.3 n_{dim}	6 ms	4 ms
Synaptic parameters, used in (2)							
τ_f (u to v)	τ_f (v to v)						
2.8 ms	0.7 ms						
Learning parameters, used in (3) and (4)							
τ_+	τ_-	α_+	α_-	w_{offset}	θ_{stdp}		
2.2 ms	5.5 ms	0.005	0.045	0.2	0.35		
Neighborhood parameters, used in (6) and (7)							
r	c_{min}	c_{max}					
0.3	4 θ_v	37 θ_v					

reconstructed from the weights with a center of mass with periodic boundary conditions. This is a consequence of the circular nature of the gaussian receptive fields of neurons in layer u. For the two datasets, data were normalized in range [0.05, 0.95] to prevent the wrapping effect of the circular gaussian receptive fields covering the range [0,1]. Each input vector is encoded by the receptive fields in a vector of activation levels for the neurons of u layer. Activation levels are kept constant for a period of 12.5 ms, followed by a quiescent period of 12.5 ms (see Fig. 2B). Synaptic weights between layers u and v are randomly initialized in range [0.4,0.6]. Layer v is initialized with 100 neurons arranged in a 10*10 grid.

Table 1 show the network parameters and gives a simple strategy to scale the architecture to higher input data dimensions n_{dim}. Only three parameters need to be scaled: the threshold level θ_v and the inhibitory levels c_{min} and c_{max}.

4.1 Quality Assessment of the Reconstructed Images

We used the root mean squared (RMS) error to quantify the difference between an original image patch $\mathbf{y}_p$ and a reconstructed patch $\hat{\mathbf{y}}_p$.

$$RMS = \frac{1}{P}\sum_{p=1}^{P}\sqrt{\frac{1}{D}\sum_{i=1}^{D}(y_{i,p}-\hat{y}_{i,p})^2} \tag{8}$$

where P is the number of extracted patches from the images and D is the number of pixels of the patch.

4.2 Results on MNIST and Natural Images

We performed two experiments on two real datasets. For the testing phase and for both experiments we fixed the lateral inhibition in layer v to c_{max} and disabled the plasticity.

The first experiment was conducted on the MNIST dataset [9]. We randomly selected a subset of 15 000 and 1000 training and testing digits, respectively, from the MNIST dataset. The 28*28 digits were splitted in 4*4 patches. We scaled θ_v, c_{min}, c_{max} with the strategy presented in Table 1 for 16 dimensions. 150,000 randomly selected patches were used to train the network. Figure 4 shows the learned receptive fields and the reconstructed images for visual assessment.

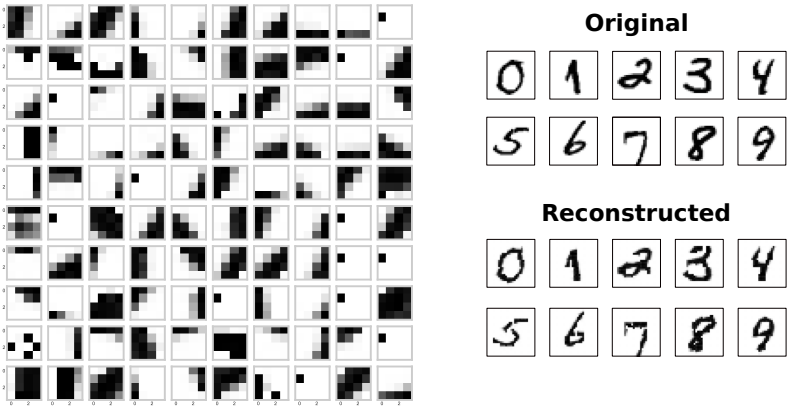

Fig. 4. Left: learned receptive fields of the neurons of v layer after the training phase. Right: original digits and reconstructed digits from the receptive fields.

Table 2. RMS reconstruction error values reported by Tavanei et al. [18]

Dataset	Tavanei [18]	Our model
MNIST	0.17	0.13
Natural images	0.24	0.10

The second experiment was performed on natural images [13]. 512*512 images were splitted in 16*16 patches. Again, we scaled θ_v, c_{min}, c_{max} with the strategy presented in table 1 for 256 dimensions. 150,000 randomly selected patches were used so as to train the network.

The RMS reconstruction error for the two datasets is reported in table 2 and is compared to the state-of-the art performances of Tavanei et al. [18]. Our architecture shows better performances for the two datasets.

5 Discussion

Related works make use of rate codes [18, 3, 7]. Our work makes use of first-spike latency code which brings significant advantages compared to rate code and rank-order code. Consider a bank of N neurons in which each neuron only spike once to

represent a feature. Rank-order codes used in conjunction with gaussian receptive fields will lead to a resolution of $1/N$ for the encoded feature. Thus increasing the resolution requires to increase the number of neurons and consequently the number of spikes. This is costly but the situation is getting worse for a rate code, as it requires N^2 neurons to achieve the same resolution. It is possible to be more efficient: by using the spike latencies we can theoretically decode a feature value with a resolution that is only limited by the time step resolution. What is then missing is a mechanism able to efficiently decode these latencies. This is provided by our novel STDP rule that learns a representation of the latencies in the synaptic weights. In our experiments only 10 neurons in layer u, generating 10 spikes, are able to represent a feature. The first neurons that spike in response to a feature value convey a maximal amount of information - as they are best tuned to the feature value - thus the resulting latencies are encoded by non-zero synaptic weights. The neurons that spike later convey few information, thus the resulting latencies induce weight depression and make the weights decay to 0. In this way a postsynaptic neuron in layer v learns a filter in its afferent synaptic weights that makes it able to quickly detect an input pattern. In other words the postsynaptic neuron spikes without having to wait for the whole set of presynaptic neurons to spike. Rumbell et al. [16] also developped a spiking SOM that uses temporal codes and mimics the original Kohonen SOM [8]. However, this architecture only performs categorization and not vector quantization.

A limitation of our architecture is the costly all-to-all lateral connectivity in layer v. The scalability of the architecture also needs further investigations, though preliminary results show that a simple strategy to adapt parameters to higher dimensions already provides satisfactory performances.

To conclude, we presented in this paper a SNN architecture able to perform for the first time vector quantization using temporal codes, more precisely first-spike latency code. This code brings a good theoretical compromise between the amout of neurons and/or spikes required to encode a continuous variable compared to rate code or rank-order code. What was missing was a mechanism able to decode the latencies. We solved this issue by developing a novel STDP rule able to efficiently learn a representation of the latencies in the synaptic weights. The locality in time and space of the computation brought by the STDP rule makes it neuromorphic-friendly. The learned synaptic weights fall in range [0,1] and the distribution is unimodal, allowing an easy subsequent decoding step. A soft competition is implemented in the lateral connections of layer v so that neighboring neurons of the BMU have a chance to spike and thus to represent close vectors in the input space. The lateral inhibition increases during the training phase with an homeostatic mechanism, so as to first organize the map and then make the neurons learn more and more individualized receptive fields.

Acknowledgments

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

References

1. Bohte, S., La Poutre, H., Kok, J.: Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. IEEE Transactions on Neural Networks **13**(2), 426–435 (Mar 2002)
2. Brette, R.: Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain. Frontiers in Systems Neuroscience **9** (Nov 2015)
3. Burbank, K.S.: Mirrored STDP Implements Autoencoder Learning in a Network of Spiking Neurons. PLOS Computational Biology **11**(12), e1004566 (2015)
4. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in Computational Neuroscience **9** (2015)
5. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In: Advances in Neural Information Processing Systems 7. pp. 625–632. MIT Press (1995)
6. Hazan, H., Saunders, D.J., Sanghavi, D.T., Siegelmann, H., Kozma, R.: Lattice map spiking neural networks (LM-SNNs) for clustering and classifying image data. Annals of Mathematics and Artificial Intelligence (Sep 2019)
7. King, P.D., Zylberberg, J., DeWeese, M.R.: Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1. The Journal of Neuroscience: The Official Journal of the Society for Neuroscience **33**(13), 5475–5485 (Mar 2013)
8. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics **43**(1), 59–69 (Jan 1982)
9. LeCun, Y., Cortes, C., Burges, C.: The MNIST database, `http://yann.lecun.com/exdb/mnist/`
10. MacQueen, J.B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: Cam, L.M.L., Neyman, J. (eds.) Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 281–297. University of California Press (1967)
11. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. Neural Networks: The Official Journal of the International Neural Network Society **15**(8-9), 1041–1058 (Nov 2002)
12. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: Neural-gas network for vector quantization and its application to time-series prediction. IEEE Trans. on Neural Networks **4**(4) (1993)
13. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (Jun 1996)
14. Rossum, M.C.W.v., Bi, G.Q., Turrigiano, G.G.: Stable Hebbian Learning from Spike Timing-Dependent Plasticity. Journal of Neuroscience **20**(23), 8812–8821 (Dec 2000)
15. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**(7784), 607–617 (Nov 2019)
16. 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 **25**(5), 894–907 (May 2014)
17. Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., Masquelier, T., Maida, A.: Deep learning in spiking neural networks. Neural Networks **111**, 47–63 (Mar 2019)
18. Tavanaei, A., Masquelier, T., Maida, A.: Representation learning using event-based STDP. Neural Networks **105**, 294–303 (Sep 2018)
19. Vasuki, A., Vanathi, P.: A review of vector quantization techniques. IEEE Potentials **25**(4), 39–47 (Jul 2006)