Event-driven Continuous STDP Learning with Deep Structure for Visual Pattern Recognition

Daqi Liu and Shigang Yue, *Senior Member, IEEE*

***Abstract*—Human beings can achieve reliable and fast visual pattern recognition with limited time and learning samples. Underlying this capability, ventral stream plays an important role in object representation and form recognition. Modeling the ventral steam may shed light on further understanding the visual brain in humans and building artificial vision systems for pattern recognition. The current methods to model the mechanism of ventral stream are far from exhibiting fast, continuous and event-driven learning like the human brain. To create a visual system similar to ventral stream in human with fast learning capability, in this study, we propose a new spiking neural system with an event-driven continuous spike timing dependent plasticity (STDP) learning method using specific spiking timing sequences. Two novel continuous input mechanisms have been used to obtain the continuous input spiking pattern sequence. With the event-driven STDP learning rule, the proposed learning procedure will be activated if the neuron receive one pre- or post-synaptic spike event. The experimental results on MNIST database show that the proposed method outperforms all other methods in fast learning scenarios and most of the current models in exhaustive learning experiments.**

***Index Terms*—Spiking neural network, event-driven STDP, continuous learning, deep learning, visual pattern recognition.**

I. Introduction

HUMAN beings are capable of processing complex visual data effortlessly in an extremely short period of time with extremely limited learning samples. Visual cortex within the brain is the part of the cerebral cortex responsible for processing visual information. There are two information processing pathways originating in the occipital cortex: ventral stream and dorsal stream, which are also known as what (object recognition) and where (spatial vision) pathways. The ventral stream plays an important role in object representation and form recognition. Therefore, modeling the underlying processing mechanism of the ventral stream becomes an essential step for automatic visual pattern recognition.

Methods to model the underlying processing mechanism of the ventral stream using spiking neural networks (SNNs) have been proposed in the last decade. According to spiking encoding mechanisms, those methods can be divided into two main categories: spiking rate-based methods [1], [2], [3], [4], [5], [6], [7], [8], [9] which count spikes within a time period, and spiking time-based methods [10], [11], [12] which prefer early spikes. Research showed that mammalian brains use only millisecond scale time windows to process complicated real life visual recognition tasks [13]. In such a short time window, it will not be enough to generate meaningful spiking rate if using spiking rate-based coding methods. It is also clear that an identical spiking rate may come from different spiking timing sequences. Therefore, it is reasonable to choose time-based coding methods in the model for fast information processing.

For the few publications on visual pattern recognition using spiking time-based coding methods, the authors in [10], [11] proposed a SNN framework and supervised Tempotron rule to train the input visual stimuli. However, the supervisory error signal, which is critical in Tempotron rule, is hard to obtain in real scenarios. In [12], Thorpe et. al use SNN architecture to simulate the processing procedure of HMAX model [14], [15], [16]. The STDP learning rule used in their method is only for local intermediate feature extraction instead of pattern recognition [12].

Another issue for most of the SNN with time-based or rate-based coding methods on pattern precognition is continuous learning. It is obvious that neurons within ventral stream could receive spiking patterns continuously without any resetting involved. Continuous stimulus presentation is a significant feature in generating a versatile and general spiking neural network [17]. However, with the current methods [7], [10], [11], to learn the spatiotemporal structures, a new input spiking pattern is only allowed to feed into the learning system when the membrane potential generated by the previous spiking pattern has been reset. This means the learning procedures have to be rested during learning and the time (gaps) between adjacent spiking patterns are wasted.

Event-driven mechanism plays an important role in ventral stream, which make the learning procedure more physiologically realistic and efficient. For a neuron within ventral stream, event-driven mechanism means its learning procedure will be activated when the neuron receives one pre- or post-synaptic spike event. This implies that the neurons within ventral stream can only, and need only, to remember local spike events with the event-driven mechanism. However, with the current SNN methods using STDP learning rule [1], [2], [3], neurons need to integrate all the related spikes within the learning window, which is obviously inefficient and physiologically unrealistic.

To build an efficient and biologically plausible ventral steam model, we propose a novel SNN structure with an event-driven continuous STDP (ECS) learning method using specific spiking timing sequences in this study. For the areas like machine learning or computer vision, the proposed ECS method provides a fast, continuous and event-driven learning procedure similar to the ventral stream within the human brain, which is efficient and biologically plausible. To obtain the continuous input spiking pattern sequence, two different

Manuscript accepted to publish on IEEE transactions on Cybernetics at January 30, 2018.

continuous input sequence mechanisms have been used. The first mechanism adds intervals between spiking patterns and updates the synaptic efficiency sequentially, while the second one divides the whole sequence into two subsequences and updates the synaptic efficiency by alternating between those two subsequences. Moreover, event-driven STDP learning method has been employed by involving two on-line, local learning rules that are activated only in response to occurrences of spike events. For this spiking time-based model, neurons use the first spike instead of average spiking rate to classify the input images. The experiment results on MNIST [18] database demonstrate the capability of the proposed ECS method - it outperforms all other methods in fast learning scenarios and most of the current models in exhaustive learning experiments.

The layout of this paper is summarized as follows: the related works are illustrated in Section II. Section III introduces the framework of the proposed feed-forward spiking neural network, along with its neuron model and event-driven STDP learning method. The experimental results and its analysis are depicted in Section IV. Finally, Section V summarizes this paper, discuses the advantages and other potential applications of the proposed method.

II. Related Works

Ventral stream is a hierarchical system in which each layer extracts different level of abstractions [19]. Through such hierarchical structure, the input visual stimuli will be transformed to high level generalized abstractions. Deep learning, a branch of machine learning, has often been used to accomplish the above scenarios [20]. HMAX [14], [15], [16], a type of convolutional neural networks (CNN) [21], [22] within deep learning methods, has been proposed to provide the much-needed framework for summarizing and integrating input visual stimuli, and thus obtaining the high level abstractions eventually.

Instead of analog values, neurons within ventral stream use spikes to represent high level abstractions. To increase the level of realism in neural simulation, spiking neural network (SNN) [23], [24] is often used as the neural network modeling tool. For SNN, neuron model is essential since it defines how the activities of the neurons change in response to each other. In this paper, conductance-based leaky integrate-and-fire (LIF) neuron model [25], [26] has been used to regulate the behaviors of the neurons as it improve the realism of the neuron simulation.

For the neurons within SNN, the spike coding scheme is also critical as it changes the format of information processing, e.g. from analog feature values into spiking patterns (specific spiking timing sequences). In this paper, rank order coding (ROC) [27], a simple yet powerful spiking time-based coding scheme, has been employed to generate the first spike wave, which has been used to represent the high level abstractions. ROC scheme considers the first spike wave conveys enough significant structural information for further visual pattern recognition application. ROC is proved to be more efficient compare to other coding schemes [27]. Within SNN, neurons use spiking patterns to convey the structural information from one layer to another.

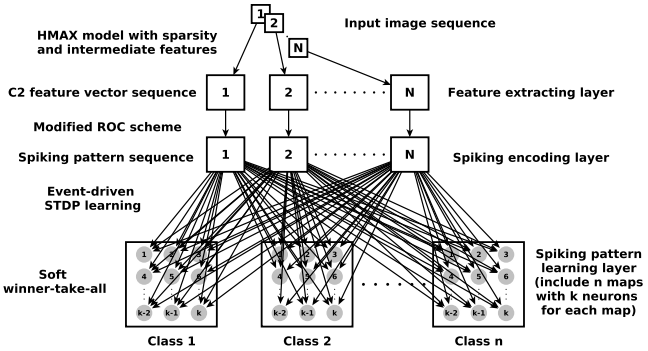


Fig. 1: The framework of the proposed timing-based feed-forward spiking neural network. For simplicity, the lateral inhibition connections of the last layer have not been included.

To learn the spiking patterns, spike timing dependent plasticity (STDP) [28], [29], [30], a temporally asymmetric form of Hebbian learning [31], has often been used within SNN. It is widely believed that it underlies learning and information storage in the brain, as well as the development and refinement of neuronal circuits during brain development [32]. For SNN, spiking learning rules are quite essential as they can "recall" or "recognize" different spiking patterns. Other methods such as back propagation (BP) (e.g. [7]) algorithm or Tempotron rule can also be used to learn the spiking patterns. However, for BP algorithm, there are three main drawbacks: 1) The vanishing/exploding gradient problem will make the learning procedure inefficient and time-consuming; 2) The global error information incorporated within the learning procedure have no strong biological supports and difficult to obtain in real world; 3) Overfitting problem will emerge if increasing the number of hidden layers, which will make the learning procedure not robust to interferences. For Tempotron rule, the supervisory error signal, which is critical in the learning procedure, is hard to obtain in real scenarios. Our previous works [33], [34] have demonstrated the potential of STDP learning in real time learning situation.

III. Proposed Framework and Learning Method

Ventral stream is a hierarchical visual processing pathway, which aims to build an invariance to position and scale first, and then to viewpoint and other transformations [35]. To simulate the hierarchical ventral stream, in this paper, a novel feed-forward SNN framework has been proposed, which includes three layers: feature extracting layer, spiking encoding layer and spiking pattern learning layer. Fig.1 shows the framework of the proposed feed-forward SNN with several keywords highlighting the key methodology used in each layer.

To provide an overview of the proposed ECS learning method, we have shown its pseudo code in the Algorithm 1. Given an input image sequence, feature extracting layer extracts its corresponding C2 feature vector sequence through five sub-layers. Within the spiking encoding layer, the above C2 feature vector sequence is transformed into spiking patterns using the modified ROC scheme, and these spiking patterns are further linked into a continuous spiking pattern sequence

Algorithm 1 ECS Learning Method

$(I_1, I_2, \ldots, I_N) \leftarrow image\_sequence$
$w \leftarrow random\_values\_within\_w^{min}\_and\_w^{max}$
Feature Extracting Layer:
Input image: $I^{\sigma}_{(x,y)}$
S1 features: $F^{\sigma,\theta}_{(x,y)} = \exp\left(-\frac{\left(x_0^2+\gamma^2 y_0^2\right)}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}x_0\right)$
C1 features: $r^{\sigma,\theta}_{(x,y)} = \underset{j=1\cdots m}{max} F^{\sigma,\theta}_{(x_j,y_j)}$
S2 features:$R(X,P) = exp\left(-\frac{\|X-P\|^2}{2\sigma^2\alpha}\right)$
C2 feature vector: $(r_1, r_2, \ldots, r_d)$
Spiking Encoding Layer:
Modified ROC scheme: $t = p(max_r - r)$
Continuous sequence mechanism: sequential or parallel
Spiking Pattern Learning Layer:
Weight sharing mechanism: applied for parallel sequences
Event-driven STDP learning method:
when fired a pre-synaptic spike
$g_{ex} = g_{ex} + w$
$A_+ = A_+ + \alpha_+$
$w = hardbound(w + A_-, w^{min}, w^{max})$
when fired a post-synaptic spike
$A_- = A_- + \alpha_-$
$w = hardbound(w + A_+, w^{min}, w^{max})$
when received an inhibitory pre-synaptic spike
$g_{in} = g_{in} + w^{in}$
Adaptive thresholding:
$\tau_v \frac{d}{dt} V_t = V_{thr} - V_t; V_t = V_t + V_i$
if Fed spiking pattern sequence once **then**
    End
else
    Goto the spiking pattern learning layer again
    return w

through the proposed continuous input sequence mechanism. Spiking pattern learning layer employs a weight sharing mechanism, an event-driven STDP learning and an adaptive thresholding strategy to train the synaptic weights from the input continuous spiking pattern sequence. The learning procedure is fast because we only need to feed the continuous spiking pattern sequence once. The details of each layer will be explained in the following subsections.

A. Feature Extracting Layer

Feature extraction is an essential capability of a visual system. From the computational model point of view, the visual system is an information processor performing computations on internal symbolic representations of visual information [36]. The computational model plays an important role in obtaining the invariant high level features from the input visual stimuli and the high level features should strike a balance state between invariance and distinguishability [20].

To extract balanced high level features, Riesenhuber and Poggio [14] proposed a feed-forward processing computational model (named HMAX model) based on the knowledge of the visual cortex and achieved promising results on some of the standard classification databases. Such a model focuses on the object recognition capabilities of the ventral stream in an "immediate recognition" mode, independent of attention or other top-down effects [14]. Inspired by the simple and complex cells within V1 (discovered by Hubel and Wiesel [37]), Serre, Wolf and Poggio [15], [16] extended the original HMAX model and thus built an increasingly complex and invariant feature representation by alternating between a template matching and a maximum pooling operation. Increasing the sparsity of basis functions is equals to reduce the capacity of the classifier [38], [39]. Localized intermediate approaches retain some coarsely-coded location information [40] or record the locations of features relative to the object center [41]. After incorporating some additional biologically-motivated properties of the visual cortex, Mutch and Lowe [42] proposed a novel model by adding sparsity and localized intermediate-level features into the model proposed by Serre et al. Such a model achieves a significant improvement in final classification performance.

Inspired by the computational model proposed by Mutch and Lowe [42], this paper tries to build the feature extracting layer based on the base computational model proposed by Mutch and Lowe, as depicted in Fig.2. The aim is to build a feature dictionary or feature vector for each input image. The framework contains five hierarchical layers, besides the input image layer, each built from the previous layer by alternating template matching and max pooling operations. Note, cortical network simulator (CNS) [43], a GPU-based framework, has been used to simulate the feature extracting layer. Below, we will briefly introduce each layer of the framework used in the feature extracting layer:

1) Input image layer: All input images have been converted to grayscale and scale the shorter edge to 140 pixels while retaining the aspect ratio. An image pyramid $I^{\sigma}_{(x,y)}$ (σ depicts the scale and (x, y) denotes its location) with 10 scales has been built, each a factor of $2^{1/4}$ smaller than the last.

2) Gabor filter (S1) layer: Basically, at each possible position and scale, S1 layer applies the Gabor filters with four different orientations $(0^o, 45^o, 90^o, 135^o)$. For an input image $I^{\sigma}_{(x,y)}$, its corresponding Gabor response $F^{\sigma,\theta}_{(x,y)}$ can be used to mimic the simple cell within the primary visual cortex V1, which can be described as follows:

$$F^{\sigma,\theta}_{(x,y)} = \exp\left(-\frac{\left(x_0^2+\gamma^2 y_0^2\right)}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}x_0\right), \quad \text{s.t.} \quad x_0 = xcos\theta + ysin\theta; \quad y_0 = -xsin\theta + ycos\theta \tag{1}$$

where x_0 and y_0 represent abscissa and ordinate after rotating θ, respectively. γ represents aspect ratio and λ the wavelength. The Gabor filters are 11×11 in size. The components of each filter are normalized so that their mean is 0 and the sum of their squares is 1. The same size filters have been used for all scales.

3) Local invariance (C1) layer: This C1 layer pool over retinotopically organized afferent S1 units from the previous S1 layer with the same orientation and from the same scale band. Basically, the response $r^{\sigma,\theta}_{(x,y)}$ of a complex C1 unit

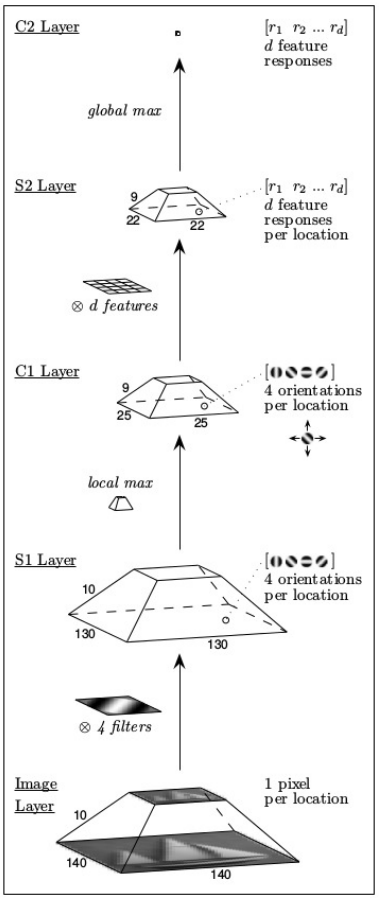


Fig. 2: The computational base model proposed by Mutch and Lowe [42]. This base model consists of 5 layers, besides the input image layer, each built from the previous layer by alternating template matching and max pooling operations. $\otimes$ means the template matching operation. For each input image, a $C2$ feature vector with d elements will be generated.

corresponds to the maximum response of its m afferents $\left(F^{\sigma,\theta}_{(x_1,y_1)}, \cdots F^{\sigma,\theta}_{(x_m,y_m)}\right)$ from the previous S1 layer:

$$r^{\sigma,\theta}_{(x,y)} = \underset{j=1\cdots m}{max}\ F^{\sigma,\theta}_{(x_j,y_j)} \tag{2}$$

Through the maximum pooling operation, the generated C1 units will obtain certain local invariance.

Unlike the traditional HMAX model [15], a lateral inhibition mechanism has been used between S1/C1 units encoding different orientations at the same position and scale. Basically, such a mechanism ensures these units are competing to describe the *dominant* orientation (maximally responding C1 unit) at their location. By doing this, those non-dominant orientations will be ignored.

4) Intermediate feature (S2) layer: Within the computational model proposed by Serre. et al [15], for every position and scale, the template matching operations have been conducted between the patch of C1 units centered at that position/scale and each of d prototype patches. Here, a patch means a set of processing units and the prototype patches can be considered as templates within the template matching operations, as described in [15]. Those prototype patches are randomly sampled from the C1 layers of the training images in an initial feature-learning stage, which represent the intermediate-level features of the base model. During the feature learning stage, sampling is performed by centering a patch of vary sizes at a random position and scale in the C1 layer of a random training image. Therefore, a prototype consists of all the C1 units within the patch. Note, for each position, there are units representing each of the four orientations. A Gaussian radial basis function has been used to compute the response of a patch of C1 units X to a particular S2 prototype P with size of $n \times n$:

$$R(X,P) = exp\left(-\frac{\|X-P\|^2}{2\sigma^2\alpha}\right) \tag{3}$$

with X and P have dimensionality $n \times n \times 4$, where $n \in \{4, 8, 12, 16\}$. The standard deviation σ is set to 1 and the parameter α represents a normalizing factor for different patch size.

However, real neurons are likely to be more selective among potential inputs. Therefore, by storing the identity and magnitude of the dominant orientation at each of the $n \times n$ positions in the patch, the number of inputs to an S2 feature has been reduced to one per C1 position. Trough this, the dense prototype in the old model has been reduced to sparse prototype, which makes the S2 units less sensitive to local clutter and thus improves the generalization.

5) Global invariance (C2) layer: Within the traditional HMAX model [15], by pooling the maximum response from one of d prototype patches, one element of the d-dimensional vector (C2 features vector) will be obtained. All position and scale information will be removed if using this mechanism. However, neurons in visual areas like V4 and inferior temporal cortex (IT) do not exhibit full invariance and are known to have receptive fields limited to only a portion of the visual field and range of scales. Therefore, in this paper, certain limits has been incorporated into the global position/scale invariance mechanism used in [15].

B. Spiking Encoding Layer

Within SNN, spiking encoding scheme plays an important role since it transforms the analog feature values into real spikes. The generated spikes are also known as spiking patterns, which are used by the neurons within ventral stream to represent the spatiotemporal structural information. Moreover, neurons need neuron model to transmit information from one layer to another. Note, Brian simulator [44] has been used to model the proposed SNN framework within this paper. In this section, the proposed spiking encoding scheme and continuous input sequence mechanism will be illustrated firstly, followed by the neuron model.

1) spiking encoding scheme: Spiking encoding scheme transforms the analog feature values into spike sequences, and thus accomplishes the information format transformation task of the ventral stream. Both spiking rate and spiking time can be used to represent the spatiotemporal structural information within ventral stream.

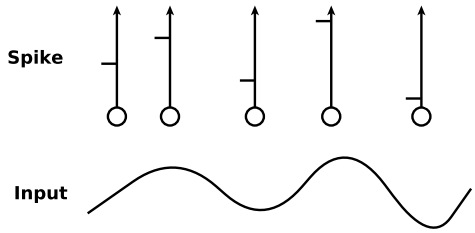


Fig. 3: Rank order coding scheme diagram. The short horizontal line within the spike part represents the latency of firing a spike. It can be seen that the higher the intensity of the input visual stimulus, the earlier the spike will be fired.

Traditionally, spiking rate is considered to represent most, if not all, information of the input visual stimuli. Spiking rate-based coding scheme assumes that as the intensity of a stimulus increases, the rate of spikes increases. However, electrophysiological studies [45], [46] indicate the neurons within the inferotemporal cortex (IT) respond selectively to faces only 80-100 ms after presenting the visual stimulus. Such short time window is not enough to generate a meaningful spiking rate. Moreover, the specific spiking timing sequence may convey much more important information than the spiking rate itself. Therefore, spiking time-based coding scheme is superior than the spiking rate-based coding scheme.

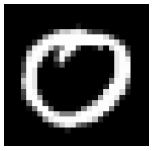
Within the spiking time-based coding schemes, rank order coding (ROC) [27], a simple yet powerful time-to-first-spike temporal coding scheme, stands out from the competition. Basically, this coding scheme can be summarized as the higher the intensity of the input visual stimulus, the earlier the spike will be fired, as shown in Fig.3. ROC scheme only allows the neuron to fire at most once and considers the first spike wave conveys enough information for further visual processing. Furthermore, it only uses the relative firing orders to represent the input visual stimuli.

However, countless spiking patterns may have an identical relative firing order and the traditional ROC scheme, in such scenarios, cannot distinguish these spiking patterns. In this paper, we use specific absolute spiking timings to replace the relative orders. Similarly, within the proposed spiking encoding scheme, the higher the intensity of the input visual stimulus, the earlier the spike will be fired. For a global invariant $C2$ feature response r, the corresponding spiking timing t (with the unit s) can be computed as follows:

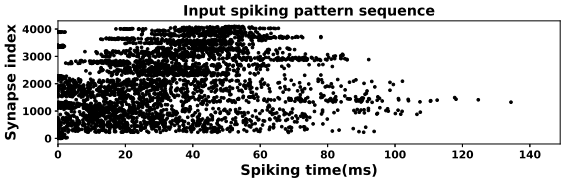
$$t = p(max_r - r) \tag{4}$$

where max_r is the maximum value of all related $C2$ features in the receptive filed and p is a positive constant within the range from 0 to 1. Here, p is used to control the length of the processing time window of a specific spiking pattern. As mentioned in the above section, the $C2$ feature response r has been normalized to $[0, 1]$. If p takes 1, then the maximum processing time window of a specific spiking pattern will be 1 s. In this paper, the processing time window of a specific spiking pattern is set to 0.2 s ($p = 0.2$).

Fig.4 shows one input image and its corresponding spiking pattern. Note, the C2 feature vector has 4096 (d feature responses in Fig.2) elements, and thus there are 4096 neurons to fire spikes. Only those neurons with spiking timings less than 150 millisecond will fire spikes. By feeding 10 input images into the proposed feed-forward SNN, Fig.5 shows spiking pattern sequence without interference and with interference, respectively. Specifically, the interference includes time jitter to the input spiking pattern itself and background neural noise between adjacent spiking patterns, which will be discussed in section IV-B2.

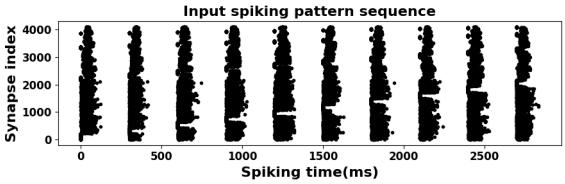


(a) Input image

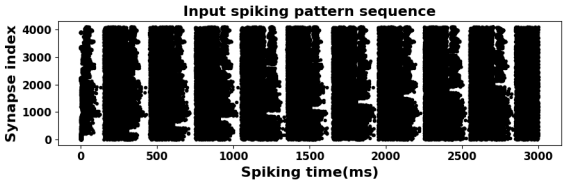


(b) Spiking pattern

Fig. 4: One input image and its corresponding spiking pattern. Given an input image (a), the feature extracting layer extracts its C2 feature vector, and the modified ROC scheme further transforms this feature vector into the spiking pattern shown in (b) using the Equation 4.



(a) Spiking pattern sequence with no interference



(b) Spiking pattern sequence with time jitter and neural noise

Fig. 5: Spiking pattern sequence (10 input images) without interference and with interference. The interference includes time jitter to the input spiking pattern itself and background neural noise between adjacent spiking patterns.

2) Continuous input sequence mechanism: Within ventral stream, neurons receive the spiking patterns continuously without any resetting procedures involved. Continuous stimulus presentation is a significant feature in generating a versatile and general spiking neural network [17]. However, with the current input mechanisms, a new spiking pattern is only allowed to feed into the learning system after the membrane

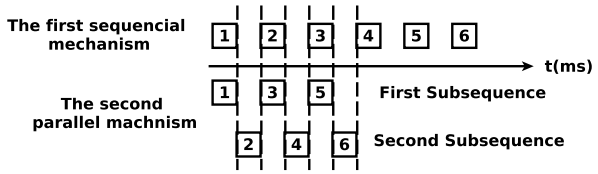


Fig. 6: The proposed two continuous input sequence mechanisms. The squares $(1, 2, 3, 4, 5, 6)$ represent spiking patterns with same time span and interval. The second parallel mechanism generates two subsequences in which the intervals of one subsequence can be used by another to increase the processing efficiency.

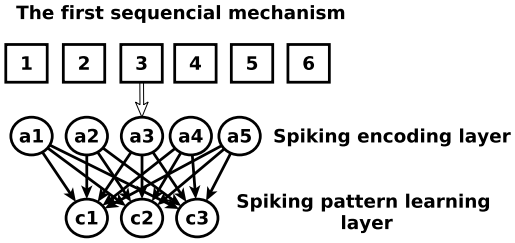


Fig. 7: Learning mechanism for the first sequential mechanism. Each spiking pattern within the sequence will be sequentially generated in the spiking encoding layer, i.e. (1,2,3,4,5,6)→(a1,a2,a3,a4,a5).

potentials generated by the current spiking pattern have been reseted to their resting values. Otherwise, the previous spiking pattern will influence the learning procedure of the current spiking pattern. Such input mechanism is very inefficient and not biologically plausible.

To address the above drawbacks, two different continuous input sequence mechanisms have been proposed. The first mechanism applies a sequential strategy while the second one follows a parallel fashion. Fig.6 shows the proposed two continuous input sequence mechanisms in which the spiking patterns have the same time span (T_s) and interval ($T_i = T_s$). Unlike the first sequential mechanism, the second parallel mechanism generates two subsequences in which the intervals of one subsequence can be used by another to increase the processing efficiency.

3) Neuron model: For SNN, neuron model is a critical building block since it defines how the activities of the neurons change in response to each other. To obtain a compromise between biological plausibility and computational complexity, conductance-based leaky integrate-and-fire (LIF) model [25], [26] is used as the neuron model in this paper.

Similar to [25], [26], within the proposed SNN, the postsynaptic membrane potential V (with the unit mV) of a neuron is determined by

$$dV/dt = (g_{ex}(E_{ex} - V) + g_{in}(E_{in} - V) + V_r - V)/\tau_m \tag{5}$$

where τ_m is the postsynaptic neuron membrane time constant. E_{ex} and E_{in} represent the membrane potential of excitatory

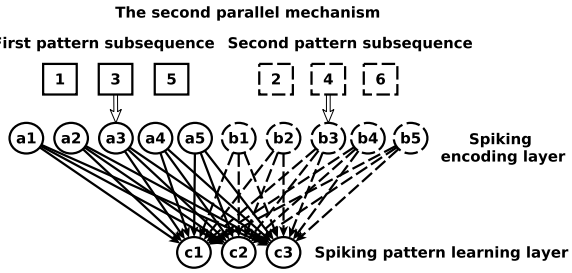


Fig. 8: Learning mechanism for the second parallel mechanism. Here, two neuron groups are created and each group corresponds to different spiking subsequence, i.e. (1,3,5)→(a1,a2,a3,a4,a5), (2,4,6)→(b1,b2,b3,b4,b5). The corresponding synaptic connections within the two groups share the same weights: when update the synaptic weight W_{a1c1}, the associated synaptic weight W_{b1c1} will be immediately updated to the same value as W_{a1c1}.

synapse and inhibitory synapse, respectively. V_r depicts the resting membrane potential. The excitatory/inhibitory synaptic conductance g_{ex}/g_{in} and its related peak conductance are measured in units of the leakage conductance of the neuron and are thus dimensionless. When the postsynaptic membrane potential of a neuron reaches the threshold V_t (with the unit mV),

$$V \geq V_t \tag{6}$$

the neuron fires a spike, and then enters the absolute refractory period, in which the postsynaptic membrane potential is reseted to V_r. The absolute refractory time is T_{rf}. The excitatory/inhibitory synaptic conductance decays exponentially:

$$\begin{aligned} dg_{ex}/dt &= -g_{ex}/\tau_{ex} \\ dg_{in}/dt &= -g_{in}/\tau_{in} \end{aligned} \tag{7}$$

where τ_{ex}/τ_{in} represents the excitatory/inhibitory synaptic conductance time constant.

C. Spiking pattern learning layer

Within the proposed SNN framework, spiking pattern learning layer is essential since it can distinguish different continuous input spiking pattern sequences after learning the corresponding synaptic weight matrices (selectivities). Specifically, within the spiking pattern learning layer, there are n maps corresponding to n classes within the database. Each map corresponds to one possible class. The neurons within each map are fully connected to the previous layer. Within each map, there are k neurons corresponding to k possible sub-classes (intra-class variances). To achieve a competitive learning, a soft winner-take-all strategy is used by adding lateral inhibition connections within the spiking pattern learning layer. Each neuron has lateral inhibition connections to all the other neurons. The input image belongs to the map (class) with the largest number of neurons firing the first spikes.

In this section, the weight sharing learning mechanism for the second parallel continuous input sequence has been introduced first, followed by the event-driven STDP learning rule and the adaptive thresholding method.

1) Weight sharing learning mechanism: For the first sequential continuous input sequence, the learning mechanism is simple and intuitive: when receiving an incoming spiking pattern, the associated synaptic weights of the neurons within spiking pattern learning layer will be updated. Within the interval between adjacent spiking patterns, the membrane potentials of these neurons will be gradually reduced to their initial values and then these neurons can start to receive the next spiking pattern, as shown in Fig.7.

Unlike the above learning mechanism, for the second parallel continuous input sequence, two neuron groups are created and each group corresponds to different spiking pattern subsequence, as demonstrated in Fig.8. The number of neurons within the spiking encoding layer has been expanded to twice of the original. Basically, this weight sharing learning strategy can be summarized as: when neurons within one group update their synaptic weights, the corresponding neurons within another group will be in an idle state and then update their synaptic weights to the same values as the first group right after the neurons within the first group finishes the update procedure, and vice versa. For instance, within the Fig.8, when the synaptic efficiency W_{a1c1} updates, its associated synaptic efficiency W_{b1c1} within the second neuron group will also be immediately updated to the same value as W_{a1c1}.

2) Event-driven STDP learning rule: To learn the synaptic weight matrices from the continuous input spiking pattern sequences, spike timing dependent plasticity (STDP) [28], [29], [30], a temporally asymmetric form of Hebbian learning [31], has been applied within the proposed SNN. STDP requires no prior information or teaching signals since it is essentially an unsupervised learning rule.

In neuroscience, long-term potentiation (LTP) is a persistent strengthening of synapses based on recent patterns of activity, while long-term depression (LTD) is an activity-dependent long-lasting reduction in the efficacy of neural synapses. Thus, STDP learning rule can be described as: when a presynaptic spike fires slightly earlier than the post-synaptic spike, the associated synaptic efficacy will be potentiated (LTP); Otherwise, the associated synaptic efficacy will be depressed (LTD). The STDP function W (t) can be expressed as follows (t is the time difference between pre and post-synaptic spikes):

$$\begin{aligned} w\left(\mathrm{t}\right) &= A_{+}\exp\left(-\frac{t}{\tau_{+}}\right) \quad for \;\; t>0 \\ w\left(\mathrm{t}\right) &= -A_{-}\exp\left(\frac{t}{\tau_{-}}\right) \quad for \;\; t<0 \end{aligned} \tag{8}$$

where A_{+} and A_{-} represent amplitude of LTP part and LTD part of the *learning window*, respectively. τ_{+} and τ_{-} are time constant for LTP and LTD, respectively. Fig.9 shows one example of STDP learning window.

For biological reasons, it is desirable to keep the synaptic efficacy in a predefined range. A hard bound strategy has been used to ensure the synaptic efficacy remains in the desired range $w^{min} \leq w \leq w^{max}$, where w^{min} and w^{max} are minimum and maximum value, respectively. Basically, if w^{min} and w^{max} are specified, values smaller than w^{min} become

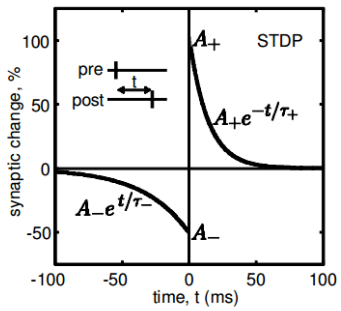


Fig. 9: STDP learning window. When a presynaptic spike fires slightly earlier than the post-synaptic spike, the associated synaptic efficacy will be potentiated; Otherwise, the associated synaptic efficacy will be depressed.

w^{min}, and values larger than w^{max} become w^{max}, which can be described as:

$$hardbound(w, w^{min}, w^{max}) = \begin{cases} w^{max}, \; if \; w > w^{max} \\ w^{min}, \; if \; w < w^{min} \\ w, \; if \; w^{min} \leq w \leq w^{max} \end{cases} \tag{9}$$

Traditional STDP learning rule needs to sum over all pairs of spikes to update the learning synaptic weight matrix, which is very inefficient. Moreover, it is also physiological unrealistic since the neurons, in real scenarios, cannot remember all its previous spike times within the learning time window.

To address the above problems, in this paper, an event-driven STDP learning rule has been applied to the proposed SNN framework by involving two on-line, local learning rules that are applied only in response to occurrences of spike events. Two new variables α_{+} and α_{-} are used in the above event-driven STDP learning rule, which represent the "traces" of pre and postsynaptic activity.

Within the traditional STDP learning procedure, a postsynaptic neuron will only update its synaptic weights after it fires a postsynaptic spike. However, by tuning the "traces" variables, the neuron can update the synaptic weights whenever it receive a presynaptic spike or firing a postsynaptic spike. These "traces" can be computed by the following equations:

$$\begin{aligned} \tau_{+}\frac{d}{dt}\alpha_{+} &= -\alpha_{+} \\ \tau_{-}\frac{d}{dt}\alpha_{-} &= -\alpha_{-} \end{aligned} \tag{10}$$

here, τ_{+} and τ_{-} represent the pre- and post-synaptic activity traces time constant, respectively. The update procedure of the traditional STDP learning involves summing pairs of pre- and post-synaptic spike events, while the event-driven learning mechanism has divided the above learning procedure into two separated stages: presynaptic event learning and postsynaptic event learning. Within the training procedure, the learning procedures of these two events do not affect each other.

When received a presynaptic spike, g_{ex}, A_+ and w will be updated as follows:

$$\begin{aligned} g_{ex} &= g_{ex} + w \\ A_+ &= A_+ + \alpha_+ \\ w &= hardbound(w + A_-, w^{min}, w^{max}) \end{aligned} \tag{11}$$

where the $hardbound$ function will make the synaptic weight remains in the desired range $w^{min} \leq w \leq w^{max}$, when fired a post-synaptic spike, A_- and w will be modified according to following equations:

$$\begin{aligned} A_- &= A_- + \alpha_- \\ w &= hardbound(w + A_+, w^{min}, w^{max}) \end{aligned} \tag{12}$$

and when received an inhibitory presynaptic spike, g_{in} will be updated as follows:

$$g_{in} = g_{in} + w^{in} \tag{13}$$

where w^{in} represents the fixed inhibitory synaptic weight, which can be used to achieve a soft winner-take-all strategy. The inhibition should set to the right level, since strong inhibition will limit the number of neurons trained while weak inhibition will lead to same synaptic weights. In other words, this soft winner-take-all strategy ensures each neuron can learn different input spiking pattern.

3) Adaptive thresholding method: As an asynchronous neural network, the proposed SNN suffers a major drawback: the neurons fired the first spikes will tend to fire spikes more easier than other neurons. This is because these neurons will immediately start integrating incoming spikes right after firing the first spikes. Hence, the membrane potentials of these neurons will be activated most and then fire spikes earlier than other neurons. The lateral inhibition will further accelerate such situation and this will in turn affect the final classification performance.

To achieve a stable network, we incorporate an adaptive thresholding method [2] into the spiking pattern learning layer. By incorporating such method, the more a neuron fires, the higher its membrane threshold will be. In other words, to fire a spike, the neuron needs to integrate more presynaptic spikes. Specifically, for each neuron, instead of using the predefined membrane threshold V_{thr}, an adaptive membrane threshold V_t will be incorporated within the spiking pattern learning layer, which can be computed by the following equation:

$$\tau_v \frac{d}{dt} V_t = V_{thr} - V_t \tag{14}$$

where τ_v is the time constant of the adaptive membrane threshold V_t. V_t will increase every time the neuron fires a spike, as shown in the follows:

$$V_t = V_t + V_i \tag{15}$$

where V_i represents a predefined increment (unit: mV).

IV. Experimental Results and Analysis

To verify the proposed ECS learning algorithm, in this paper, we compare it with two types of method: the deep SNN learning methods and the conventional deep learning methods. Essentially, the proposed ECS learning algorithm is a deep SNN learning method. Unlike the conventional deep learning methods which employ gradient descent mechanism (especially back propagation) to train the spatial structures, the deep SNN learning methods could support other efficient biologically plausible learning mechanisms to train the spatiotemporal structures.

Fig. 10: Random examples of MNIST database.

TABLE I: Parameter settings of the proposed SNN.

Parameter	Description	Value
d	the number of elements in a $C2$ vector [2]	4096
τ_m	membrane time constant [1]	10 ms
τ_v	potential threshold time constant [1]	20 ms
τ_{ex}	excitatory conductance time constant [1]	5 ms
τ_{in}	inhibitory conductance time constant [2]	10 ms
τ_+	presynaptic trace time constant [1]	20 ms
τ_-	postsynaptic trace time constant [1]	20 ms
p	a positive constant for ROC [2]	0.2
T_s	real processing time window [2]	150 ms
T_i	interval time window [2]	150 ms
T_{rf}	absolute refractory time [1]	1 ms
E_{ex}	excitatory membrane potential [1]	0 mV
E_{in}	inhibitory membrane potential [2]	-85 mV
V_r	resting membrane potential [1]	-74 mV
V_{thr}	membrane potential threshold [2]	-45 mV
V_i	increment for adaptive potential threshold [2]	5 mV
w^{min}	minimum synaptic weight [1]	0
w^{max}	maximum synaptic weight [2]	0.01
w^{in}	fixed inhibitory synaptic weight [2]	0.05
α_+	the presynaptic trace [1]	$0.01w^{max}$
α_-	the postsynaptic trace [1]	$-\alpha_+(\tau_+/\tau + -)1.05$

[1] Take the same value as [25].
[2] Optimized to achieve the best classification performance.

To accomplish the above comparison, two datasets - MNIST [18] and Caltech 101 [47] - are used in this paper. MNIST handwritten digital dataset includes 10 classes and it contains 60,000/10,000 training/testing samples (with the size of 28×28 for each sample image). Fig.10 shows some random examples of the MNIST database. Caltech 101 contains a total of 9,146 images, split between 101 distinct object categories (faces, watches or pianos, etc.) and a background category. In the following subsections, the parameter settings and the convergence/robustness verification of the proposed ECS learning method are introduced first, followed by experimental comparison with the deep SNN learning methods and the conventional deep learning methods, respectively.

A. Parameter settings

Within the feature extracting layer, all parameters besides d take the same values as [42]. Normally, to achieve a realistic

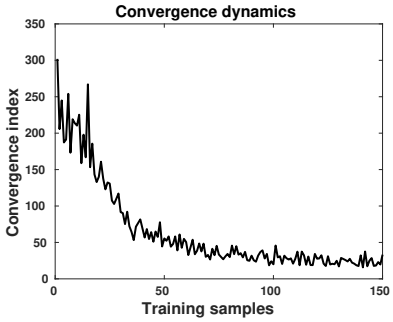


Fig. 11: Convergence dynamics of the proposed ECS method. The learning procedure enters a relatively stable stage after feeding around 50 random samples extracted from a randomly chosen class.

neural simulation, the parameters often take the values within a limited predefined range. Table I shows the parameter settings used in the proposed SNN framework. Specifically, some of the parameters used in the proposed SNN framework take the the same values as [25] while the values of other parameters, like [26], are chosen by optimizing the whole classification performance. Note, the time resolution of this experiment is 0.1 ms.

B. Convergence and robustness of the proposed ECS method

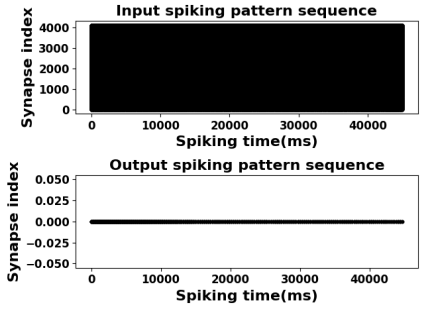
In this section, we will validate the convergence and robustness of the proposed ECS learning method. On one hand, the proposed method should ensure it can achieve convergence so that the synaptic weights with selectivity can be learned eventually. On the other hand, the proposed method should be robust enough to ignore the interferences and only concentrate on learning the input visual stimuli.

1) Convergence analysis: In this section, we will verify the convergence property of the proposed ECS method since the learned synaptic weights would become random if the learning method does not achieve the convergent state.

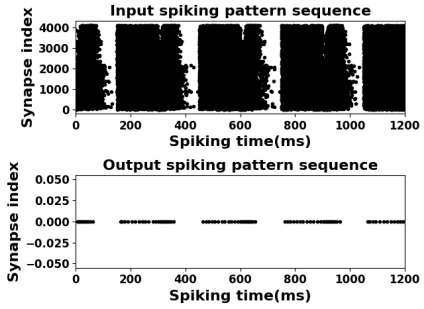
Ideally, when the learning procedure reaches convergence, the number of spiked neurons and their spiking timings would remain at a fixed state with most learned synaptic weights taking 0 and the rest taking 1 [48]. However, in reality, it is impossible to achieve the fixed state since the learning methods, in most cases, cannot fully learn the abundant intra-class variances that exist in the input visual stimuli. Instead, they would enter a a stable stage (not fixed) with some of the learned synaptic weights scattering between 0 and 1.

To verify whether a learning method becomes convergent or not, in this paper, a convergence index f has been proposed by summing the spiking timings of all fired neurons within a spiking pattern period. For each spiking pattern period (including spiking pattern time window and the interval between adjacent spiking patterns) within the whole spiking pattern sequence, f can be computed as follows:

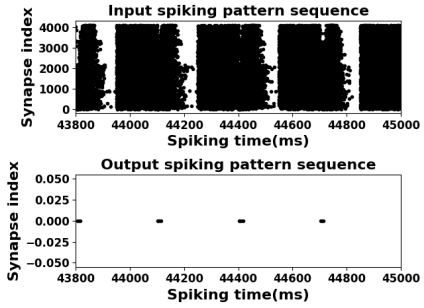
$$f = \sum_{i=1}^{n} (st_i - init) \tag{16}$$



(a) The whole input and output spiking pattern sequences using 150 random samples within the same randomly chosen class



(b) The first four input and output spiking pattern sequences extracted from (a)

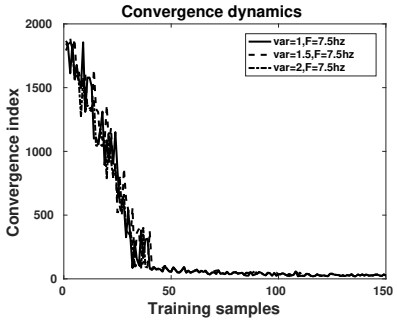


(c) The last four input and output spiking pattern sequences extracted from (a)

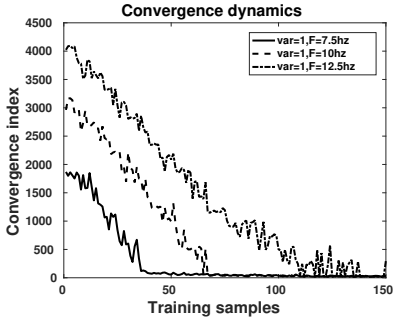
Fig. 12: Robustness of the proposed ECS method to the interferences. In (b), there are spikes fired within the background neural noise time window while no spikes fired within that period after learning, as shown in (c). This proves that the proposed ECS method will gradually ignore the interferences and only concentrate on learning the real spiking patterns.

where n means the number of spiked neurons within the current spiking pattern period, st_i represents the spiking time (unit: ms) of that specific fired neuron and $init$ depicts the starting time (unit: ms) of the current spiking pattern period.

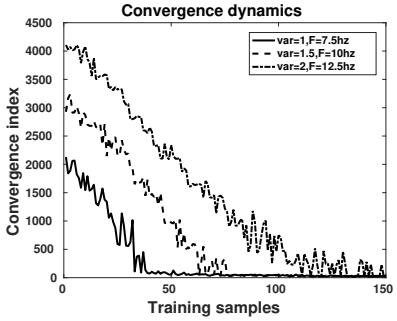
We then use the proposed ECS method to train 150 random samples extracted from a randomly chosen class and compute the convergence dynamics, as demonstrated in Fig.11. It can be seen that the convergence index f sharply decreases until around 50 and then enters a relatively stable stage. It is noted that f, as expected, still fluctuates slightly after the learning method becomes convergent.



(a) Same frequency and different variance



(b) Same variance and different frequency



(c) Different variance and different frequency

Fig. 13: Dynamic convergence index under different interferences settings, in which var represents the variance of the time jitter and F is the frequency of the background neural noise. It can be seen that the convergence is harder to achieve if increasing the level of the interferences. Moreover, the background neural noise has more influence on the convergence performance.

2) Robustness analysis: In this section, we will verify whether the proposed ECS method is robust enough to ignore the interferences and only concentrate on learning the input visual stimuli. Here, interferences include the time jitter and the background neural noise, which are ubiquitous in the ventral stream and would deteriorate the information transmission quality. To accomplish the above verification, we will investigate the dynamic spiking status of the output neurons by incorporating the time jitter and the background neural noise into the learning procedure.

Specifically, the time jitter obtained from a standard normal distribution will be added into each spiking pattern itself. The probability density function ($\phi(x)$) of the standard normal distribution is as follows:

$$\phi(x) = \frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}} \tag{17}$$

Meanwhile, the background neural noise generated from an uniform poisson point progress with frequency of 7.5 Hz (simulating a common α brainwave) will be inserted into the interval between the adjacent spiking patterns.

Given a compact set K, a point process X is defined as a mapping from a probability space to configurations of points of K [49]. Note, $N(A)$ is the number of point of a point process X falling in the Borel set A. Let $v(.)$ be a Borel measure on K, then a point process X on K is a poisson point process with intensity $v(.)$ if $N(A)$ is poisson distributed with mean $v(A)$ for every bounded Borel set A included in K and the random variables $N(A_1), ..., N(A_k)$ are independent for any k disjoint bounded Borel set $A_1, ..., A_k$. Uniform poisson point process [50], the most simple poisson point process, is the point process with intensity measure being proportional to the Lebesgue measure on K:

$$v(.) = \beta\lambda_k(.) \tag{18}$$

The mean number of points falling into K is then:

$$\mathbb{E}[N(K)] = \beta\lambda_K(K) \tag{19}$$

where $\mathbb{E}$ represents expectation symbol and λ_K depicts the frequency of the poisson law. β is a positive constant value. A two steps procedure can be used to generate points in K with the distribution of this point process: Firstly, simulate N according to poisson law with mean given by the latter equation (it gives $N = n$); Secondly, sample each of the n points according to a uniform law on K.

With the above interferences settings, we then train 150 random samples extracted from a randomly chosen class and investigate the dynamic spiking status of the output neurons, as demonstrated in Fig.12. Specifically, (a) shows the whole dynamic learning procedure using 150 random samples, (b) represents the dynamic learning procedure of the first four spiking pattern period and (c) depicts the dynamic learning procedure of the last four spiking pattern period. From Fig.12 (b), it can be seen that there are several output spikes during the background neural noise time window, while in Fig.12 (c), the output spikes only appear within the input spiking pattern time window and there are no spikes generated within the background neural noise time window.

The possible reason for the above behaviors is that the integral area of LTD is larger than the integral area of LTP within the proposed ECS method. If there are no repeating inputs fed into the proposed ECS method, all learned synaptic weights would saturate to 0. Moreover, those background neural noise patterns are irrelevant since they are independent and identically distributed. Thus, for the proposed ECS method, only the repeated input visual stimuli can be learned eventually, as demonstrated in Fig.12.

To further investigate the influences of the interferences to the proposed ECS learning procedure, the dynamic convergence indexes with three different interferences settings have been used in this subsection, as shown in Fig.13. With the

TABLE II: Correct classification performance using different random training samples with two different methods, in which CNN[42] without interferences (var=0, F=0hz) and ECS with interferences (var=1, F=7.5hz)

Condition	CNN[42]:**var=0, F=0hz**; ECS:**var=1, F=7.5hz**					
Random test	Number of training samples					
	50		100		150	
	CNN[42]	ECS	CNN[42]	ECS	CNN[42]	ECS
1	0.8	0.86	0.87	0.89	0.82	0.85
2	0.83	0.83	0.86	0.85	0.88	0.86
3	0.83	0.86	0.82	0.84	0.87	0.91
4	0.82	0.86	0.83	0.87	0.88	0.87
5	0.8	0.81	0.85	0.87	0.82	0.87
6	0.84	0.87	0.86	0.88	0.83	0.88
7	0.8	0.87	0.87	0.89	0.87	0.88
8	0.84	0.86	0.85	0.83	0.89	0.84
9	0.81	0.79	0.84	0.85	0.82	0.86
10	0.87	0.8	0.85	0.88	0.85	0.82
average	0.824	0.841	0.85	0.865	0.853	0.864
running time	22.8s	23.1s	23.1s	23.3s	23.1s	23.4s

• Note: 0.8 in this table means 80% correct classification rate. To be fair, for each random test with specific number of training samples, CNN [42] and ECS use the same random training samples and the same 100 random testing samples.

TABLE III: Correct classification performance comparison between CNN [42] and ECS using 100 training samples without any interferences (var=0, F=0hz).

Condition	CNN [42]:**var=0, F=0hz**; ECS:**var=0, F=0hz**	
Random test	CNN [42]	ECS
1	0.85	0.88
2	0.87	0.86
3	0.83	0.87
4	0.87	0.89
5	0.82	0.90
6	0.9	0.91
7	0.83	0.86
8	0.86	0.95
9	0.8	0.91
10	0.84	0.88
average	0.847	0.891
running time	23.15s	22.75s

• Note: 0.85 in this table means 85% correct classification rate. To be fair, CNN [42] and ECS use the same random training samples and the same 100 random testing samples.

increase of the variance var of the time jitter and the frequency F of the background neural noise, more training samples are needed for the proposed ECS learning method to enter the convergence state. Besides, the background neural noise has more influence than the time jitter.

C. Comparison with deep SNN learning methods

In this section, we use MNIST dataset to accomplish the comparison task. From the conventional deep learning point of view, it may seem small for evaluating the scaling of architectures and learning methods to larger applications. However, we are only in the initial stage of understanding the connections between the temporal dynamics of biologically realistic networks, and mechanisms of temporal and spatial credit assignment [51]. For deep SNN learning methods, MNIST dataset still remains important since almost all recently published SNN papers are evaluated on this benchmark, meaning it remains the only dataset allowing for comparisons.

Two types of experiments are investigated in this section. One type of experiment is using a simple random sampling scheme in which only a small part of training/testing samples has been used to learn/test the selectivities. Another one is employing an exhaustive scheme in which all training/testing samples have been used to learn/test the selectivities. Unlike the ideal exhaustive scheme which remains in a static state (with no new training/testing samples added into the dataset), the simple random sampling scheme requires the learning methods to generate acceptable performance based on quite limited learning resources (such as running time or computational capacity).

1) Simple random sampling experiments: To verify the performance of the proposed ECS method on the real learning scenarios, simple random sampling experiments have been used in this section, which randomly select limited samples for training and testing. This is because it is often hard to fully exploit a huge database (such as MNIST database) with limited time, let alone the size of database is changing over time in some cases.

In this section, two types of experiments are conducted to compare the proposed ECS method with a typical CNN method [42] - one adds interferences ($var = 1$ for time jitter, $F = 7.5hz$ for background neural noise) into the proposed ECS method and no interferences for the CNN method, while another incorporates no interferences to both methods. Within the training period, 50/100/150 random training samples are employed for each class, meaning totally 500/1000/1500 samples are extracted from the whole 60000 training samples. Within the testing period, we randomly extract 100 testing samples from the whole 10,000 testing database.

Table II shows the correct classification performances of the first type of experiment. To be fair, for each random test with specific number of training samples, CNN [42] and ECS use the same random training samples. It can be seen that the proposed ECS method with 100 random training samples strike the balance state between the processing time and final classification performance, which only use 23.3s to achieve 86.5% correct classification performance. More importantly, this performance is achieved when incorporating interferences into the proposed ECS method.

For the second type of experiment, we compare the dynamic convergence index of three random tests (without adding any interferences) using the proposed ECS method, as shown in Fig.14. It can be seen that the proposed ECS method would roughly hit the stable period around 100 training samples. For the sake of simplicity, we only demonstrate the classification

TABLE IV: Classification accuracy performance using different methods on MNIST database.

Spiking Coding-type	Architecture	Preprocessing	(Un-)supervised	Learning Rule	Performance	
					Simple random sampling [a]	Exhaustive [b]
Time-based	Spiking convolutional neural network	Modified HMAX	Supervised	**ECS(this paper)**	**89%**	**93.0%**
	Two layer network[10]	Simplified HMAX	Supervised	Tempotron rule	79.0%	N/A
	Two layer network[11]	Simplified HMAX	Supervised	Tempotron rule	N/A	91.3%
Rate-based	Dendritic neurons[4]	Thresholding	Supervised	Morphology learning	N/A	90.3% [d]
	Spiking RBM[5]	None	Supervised	Contrastive divergence, linear classifier	N/A	89.0%
	Spiking RBM[6]	Enhanced training set to 120,000 examples	Supervised	Contrastive divergence	N/A	89.0%
	Spiking convolutional neural network[7]	None	Supervised	Backpropagation	N/A	99.1%
	Spiking RBM[8]	Thresholding	Supervised	Contrastive divergence	N/A	92.6% [c]
	Spiking RBM[8]	Thresholding	Supervised	Contrastive divergence	N/A	91.9% [c]
	Two layer network[9]	Edge-detection	Supervised	STDP with calcium variable	N/A	96.5% [e]
	Multi-layer hierarchical neural network[1]	Orientation-detection	Supervised	STDP with calcium variable	N/A	91.6%
	Two layer network[2]	None	Unsupervised	Rectangular STDP	N/A	93.5%
	Two layer network[3]	None	Unsupervised	Exponential STDP	N/A	95.0%

[a] Simple random sampling performance has been generated by averaging 10 random tests using 50 random training samples per class and 100 random testing samples, which is suitable for real-time learning since the whole database is impossible to obtain in most real scenarios.
[b] Exhaustive performance shows the ideal experimental results by using whole 60000 training samples and 10000 testing samples within MNIST database.
[c] The authors only use 1000 testing samples to obtain the performance
[d] The authors only use 5000 testing samples to obtain the performance
[e] The authors use 10000 randomly chosen samples from MNIST database instead of the dedicated testing database

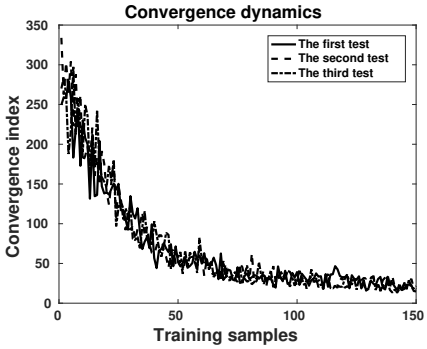


Fig. 14: Dynamic convergence index of three random tests using the proposed ECS method without adding any interferences (var=0, F=0hz). It shows the proposed ECS method will enter the stable stage after feeding 100 training samples if adding no interferences.

performance comparison using 100 training samples within the learning procedure, as shown in Table III. Compared with the CNN [42] method, the proposed ECS method uses slightly less time (22.75s) to generate a quite higher correct classification performance (89.1%).

2) Exhaustive experiments: For fair comparison with the current deep SNN learning methods, exhaustive experiments using all training/testing samples have also been conducted in this paper. Unlike the simple random sampling experiments, exhaustive experiments use the whole 60,000 training samples to learn the synaptic weights and then distinguish the whole 10,000 testing samples based on the above learned synaptic weights.

Table IV shows the simple random sampling and exhaustive classification accuracy performance using the proposed ECS method and different state-of-the-art deep SNN learning methods. According to their spiking coding type, the learning methods within Table IV have been divided into two categories: time-based and rate-based. From Table IV, it can be seen that, within the time-based spiking learning methods, the proposed ECS method achieves the best classification performances in both simple random sampling and exhaustive experiments. Moreover, it achieves this performance by only using one run (feeding the whole training samples into SNN framework once), which greatly improve the learning efficiency and thus reduce the whole processing time. For instance, with the same experimental platform (Intel Corporation Xeon E3-1200, 16 GB DDR3 RAM, 120 GB SSD, NVIDIA GeForce GT 640), it takes around one day for the SCNN [7] method to finish the training, while the proposed method only takes less than one hour. Furthermore, we use various biologically plausible models to build the proposed SNN framework, such as conductance-based LIF model, adaptive thresholding method and event-driven continuous STDP learning rule. The above experiments have demonstrated the capacity and potential of a biologically plausible model in efficient learning for pattern recognition.

It is noted that few rate-based methods achieve better performance. However, they suffer two main drawbacks: 1) To achieve the best performance, all those methods need to feed the whole training samples into their SNN frameworks at least several times (several runs) since one run is not enough to generate a meaningful spiking rate. This is very time consuming. For instance, since the two layer network [3] needs to run at least 15 times to obtain the performance, its running time is at least 15 times longer than the proposed ECS method. The actual running time differences should be much larger since we have incorporated efficient event-driven continuous mechanisms into the proposed method. 2) Some methods incorporate global information into the learning procedure which is not feasible in real world conditions. For instance, based on back propagation (BP) algorithm, a spiking convolutional neural network [7] used a global error signal to update the synaptic weights however such global error signal is

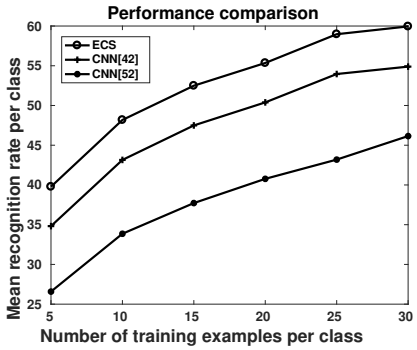


Fig. 15: Performance comparison of the three methods using different numbers of training samples per class.

impossible to obtain within a biological system. Based on the current knowledge in neuroscience, the special computational circuit for obtaining such global back propagation errors seems unlikely to have arisen in an evolved organism.

D. Comparison with conventional deep learning methods

For fair comparison, in this section, we challenge the proposed ECS learning rule with two conventional CNN methods [42], [52] on a more complicated Caltech 101 dataset. The reason why we choose these specific methods to compare is because their architectures have the same depth as the proposed ECS learning method. Specifically, we use different number of samples per class (5/10/15/20/25/30) to train the synaptic weights and then test the classification performance on the remaining samples, as shown in Fig.15. Compared with the conventional CNN methods [42], [52], the proposed ECS learning method clearly achieves better performance when using different number of training samples per class.

While some of the current conventional deep learning methods may achieve better performance, they are often challenged by two main issues: 1) Inefficiency. They are inefficient in terms of computational resource and running time, i.e. the hardware mapping analysis in [53] demonstrated that SNN implementation on a spike-based hardware is two orders of magnitude more energy-efficient than the similar CNN implementation on off-the-shelf FPGA-based hardware. 2) Biologically implausible. In visual cortex, neurons use spikes to transmit information and the gradient descent mechanism is almost impossible to train the spiking neural network due to the non-differentiable nature of spike events.

Even though the biologically plausible spiking neural network learning methods are still in their initial stage, the existing evidence suggests that it is possible to efficiently optimize complex functions of temporal history in the context of spiking networks of biologically realistic neurons, which indicates they have the great potential to accurately and efficiently train the complicated datasets [51].

V. Conclusion

In this paper, an event-driven continuous STDP (ECS) learning method using specific spiking timing sequences has been proposed. Within the proposed SNN framework, we use image sequence as the input and extract the high level abstractions by using the modified HMAX model with sparsity and intermediate variables. Through the modified ROC scheme, the extracted high level abstractions have been transformed into separated spiking patterns. Two novel continuous input sequence mechanisms have been proposed to connect these separated spiking patterns into spiking pattern sequences. For different continuous input sequence mechanism, we use different event-driven STDP learning procedure to train the final synaptic efficiency matrix. A soft winner-take-all strategy has been used within the above learning procedure so that neurons can compete with each other to represent the input images. The experimental results on MNIST database show that the proposed ECS method outperforms all other methods in fast learning scenarios and most of the current models in exhaustive learning experiments. For future work, a new comprehensive spiking encoding scheme may be integrated to generate more than one spike per synapse connection and new feature extracting layers may be added to extract more features.

Acknowledgment

The authors have been supported by EU FP7 project HAZCEPT(318907) and EU Horizon 2020 project STEP2DYNA(691154).

References

[1] M. Beyeler, N. D. Dutt, and J. L. Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule," *Neural Networks*, vol. 48, pp. 109–124, Dec. 2013.

[2] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to Device Variations in a Spiking Neural Network With Memristive Nanodevices," *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 288–295, May 2013.

[3] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, p. 99, 2015.

[4] S. Hussain, S. C. Liu, and A. Basu, "Improved margin multi-class classification using dendritic neurons with morphological learning," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 2640–2643.

[5] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011, pp. 1–4.

[6] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Neuromorphic Engineering*, vol. 7, p. 178, 2013.

[7] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[8] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Neuromorphic Engineering*, vol. 7, p. 272, 2014.

[9] J. M. Brader, W. Senn, and S. Fusi, "Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics," *Neural Computation*, vol. 19, no. 11, pp. 2881–2912, Sep. 2007.

[10] Q. Yu, H. Tang, K. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Trans. Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1539–1552, 2013.

[11] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1963–1978, Sep. 2015.

[12] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Computational Biology*, vol. 3, no. 2, p. e31, 2007.

[13] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, no. 6582, pp. 520–522, 1996.

[14] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, Nov. 1999.

[15] S. Thomas, W. Lior, B. Stanley, R. Maximilian, and P. Tomaso, "Robust object recognition with cortex-like mechanisms," *IEEE Trans. Pattern Anal. Mach. Intelli.*, vol. 29, no. 3, pp. 411–426, 2007.

[16] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio, "A quantitative theory of immediate visual recognition," *Progress in Brain Research*, vol. 165, pp. 33–56, 2007.

[17] T. Rumbell, S. Denham, and T. Wennekers, "A Spiking Self-Organizing Map Combining STDP, Oscillations, and Continuous Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 894–907, 2014.

[18] [Online]. Available: http://yann.lecun.com/exdb/mnist/

[19] T. Poggio and E. Bizzi, "Generalization in vision and motor control," *Nature*, vol. 431, no. 7010, pp. 768–774, Oct. 2004.

[20] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[21] Q. Wang, J. Gao, and Y. Yuan, "Embedding structured contour and location prior in siamesed fully convolutional networks for road detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 230–241, 2018.

[22] ——, "A joint convolutional neural networks and context transfer for street scenes labeling," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[23] E. Nichols, L. J. McDaid, and N. Siddique, "Biologically Inspired SNN for Robot Control," *IEEE Transactions on Cybernetics*, vol. 43, no. 1, pp. 115–128, Feb. 2013.

[24] Q. Kang, B. Huang, and M. Zhou, "Dynamic Behavior of Artificial Hodgkin Huxley Neuron Model Subject to Additive Noise," *IEEE Transactions on Cybernetics*, vol. 46, no. 9, pp. 2083–2093, Sep. 2016.

[25] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, no. 9, pp. 919–926, Sep. 2000.

[26] S. Song and L. F. Abbott, "Cortical development and remapping through spike timing-dependent plasticity," *Neuron*, vol. 32, no. 2, pp. 339–350, Oct. 2001.

[27] A. Delorme and S. Thorpe, "Face identification using one spike per neuron: resistance to image degradation," *Neural Networks*, vol. 14, no. 6-7, pp. 795–803, 2001.

[28] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 386, pp. 76–78, 1996.

[29] R. Kempter, W. Gerstner, and J. L. van Hemmen, "Hebbian learning and spiking neurons," *Phys. Rev. E*, vol. 59, pp. 4498–4514, 1999.

[30] P. J. Sjstrm, G. G. Turrigiano, and S. B. Nelson, "Rate, timing, and cooperativity jointly determine cortical synaptic plasticity," *Neuron*, vol. 32, pp. 1149–1164, 2001.

[31] D. O. Hebb, *The Organization of Behavior: a neuropsychological theory*. New York: Wiley, 1949.

[32] P. J. Sjstrm, E. A. Rancz, A. Roth, and M. Husser, "Dendritic excitability and synaptic plasticity," *Physiological Reviews*, vol. 88, no. 2, pp. 769–840, Apr. 2008.

[33] D. Liu and S. Yue, "Visual pattern recognition using unsupervised spike timing dependent plasticity learning," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 285–292.

[34] ——, "Fast unsupervised learning for visual pattern recognition using spike timing dependent plasticity," *Neurocomputing*, vol. 249, pp. 212–224, Aug. 2017.

[35] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio, "A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex," 2005.

[36] D. . T. P. Marr, "From understanding computation to understanding neural circuitry," *Neurosciences Res. Prog. Bull.*, vol. 15, pp. 470–488, 1977.

[37] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, Oct. 1959.

[38] M. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1150–1159, Sep. 2003.

[39] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: fast algorithms and generalization bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[40] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1475–1490, Nov. 2004.

[41] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 1, 2005, pp. 26–33.

[42] J. Mutch and D. G. Lowe, "Object Class Recognition and Localization Using Sparse Features with Limited Receptive Fields," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 45–57, Jan. 2008.

[43] J. Mutch, U. Knoblich, and T. Poggio, "CNS: a GPU-based framework for simulating cortically-organized networks," Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2010-013 / CBCL-286, February 2010.

[44] D. F. Goodman and R. Brette, "The brian simulator," *Frontiers in neuroscience*, vol. 3, no. 2, p. 192, 2009.

[45] S.-I. Amari and N. Kasabov, Eds., *Brain-like Computing and Intelligent Information Systems*, 1st ed. Springer-Verlag Singapore Pte. Limited, 1998.

[46] L. C. Jain, U. Halici, I. Hayashi, S. B. Lee, and S. Tsutsui, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*. CRC Press, Jun. 1999.

[47] [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[48] R. Guyonneau, R. VanRullen, and S. J. Thorpe, "Neurons tune to the earliest spikes through STDP," *Neural Computation*, vol. 17, no. 4, pp. 859–879, Apr. 2005.

[49] V.-J. D. Daley, D.J, *An Introduction to the Theory of Point Processes*. New York: Springer, 1988.

[50] W. S. K. D. Stoyan and J. Mecke., *Stochastic geometry and its applications*. Wiley, 1988, vol. 2.

[51] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an integration of deep learning and neuroscience," *Frontiers in computational neuroscience*, vol. 10, 2016.

[52] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. Ieee, 2005, pp. 994–1000.

[53] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.