Sparse Temporal Encoding of Visual Features for Robust Object Recognition by Spiking Neurons

Yajing Zheng, Shixin Li, Rui Yan, *Member, IEEE*, Huajin Tang, *Member, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

***Abstract*— Robust object recognition in spiking neural systems remains a challenging in neuromorphic computing area as it needs to solve both the effective encoding of sensory information and also its integration with downstream learning neurons. We target this problem by developing a spiking neural system consisting of sparse temporal encoding and temporal classifier. We propose a sparse temporal encoding algorithm which exploits both spatial and temporal information derived from an spike-timing-dependent plasticity-based HMAX feature extraction process. The temporal feature representation, thus, becomes more appropriate to be integrated with a temporal classifier based on spiking neurons rather than with nontemporal classifier. The algorithm has been validated on two benchmark data sets and the results show the temporal feature encoding and learning-based method achieves high recognition accuracy. The proposed model provides an efficient approach to perform feature representation and recognition in a consistent temporal learning framework, which is easily adapted to neuromorphic implementations.**

***Index Terms*— Robust object recognition, sparse temporal encoding, spatiotemporal patterns, spiking neural networks (SNNs), temporal classifier.**

I. Introduction

OBJECT recognition gains its importance in many fields. Primates are able to recognize objects rapidly and precisely [1]. Different transformations, including scale, lightness, and position are not likely to influent judge. Nowadays, even though in some data sets computer vision algorithms have outperformed human beings [2], there still exist big challenges, such that the performance of the biological visual system still outperforms the best computer vision systems in real sceneries.

Robust object recognition is believed to originate from the invariant representation of visual features. The idea has proved to be very successful as demonstrated by the well-known HMAX model, which models the hierarchical processing in ventral pathway and the feature extraction plays the essential role before recognition takes place. Reference [3] shows such a feedforward architecture gives rapid and yet a robust recognition performance. However, in such kind of feature representation, only spatial information was obtained and carried forward for a classifier for recognition, and it was unable to learn and represent temporal information. Importantly, temporal feature encoding is underexplored, as it is not only a key process for rapid visual recognition in neural systems [4], [5] but also an important technique to achieve highly efficient object recognition in dynamic visual scenes as shown in recent studies [6], [7]. Masquelier and Thorpe [8] introduce a new method of transforming HMAX into temporal domain using spike-timing-dependent plasticity (STDP)-based unsupervised learning, i.e., a temporal representation model based on spike-based encoding. The temporal model is able to obtain sparse temporal features from the input images which leads to invariance representation of visual inputs, and leads to high performance on object recognition as demonstrated in a recent work [9].

In this paper, we take advantage of the unsupervised learning method proposed in [8] to obtain the temporal features. Here, we develop a new encoding method to improve the representation invariance. In addition, in previous works such as [3], [8], and [9], feature and prototype learning is implemented by a nontemporal classifier such as support vector machine (SVM), while temporal feature encoding with spatiotemporal patterns requires new form of classifier. The main challenge of constructing such an architecture is how to integrate the spatiotemporal feature extraction with the spike-based learning algorithms [10]–[13]. The main contribution of this paper is proposing a unified systematic model with fully spike processing, effective combination, and encoding of extracted information through unsupervised learning. As a single neuron can be seen as an arithmetic operation unit [14], we use an algebraic transformation on features-spikes relationship to put specific incoming spatiotemporal spike train into a spiking neural network (SNN).

The rest of this paper is organized as follows. In Section II, we begin by presenting the feedforward SNN, which consists of feature extraction, encoding, learning, and readout part. Section III briefly introduces some supervised learning algorithms with spiking neuron which can learn the association between any incoming spike patterns with target spike patterns. Detailed investigation and analysis of the system are conducted in Section IV. This section shows the performances of our system through numerical simulations. Discussions are presented in Section V, followed by a conclusion in Section VI.

Manuscript received March 27, 2017; revised October 30, 2017; accepted February 10, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61673283. *(Corresponding author: Huajin Tang.)*

Y. Zheng, S. Li, R. Yan, and H. Tang are with the Neuromorphic Computing Research Center, College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: htang@scu.edu.cn).

K. C. Tan is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2812811

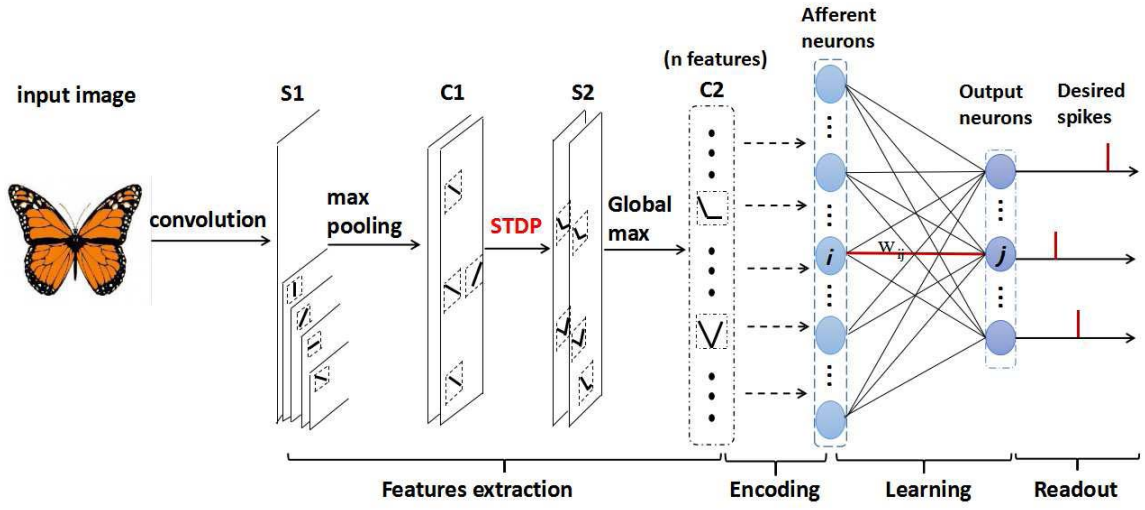


Fig. 1. Schematic of the feedforward computational model for object classification. It contains four functional parts: features extraction, encoding, learning, and readout. Image features are extracted by the STDP-based four-layer hierarchical network. These features are converted into spatiotemporal spikes by our encoding method. This spiking pattern is passed to the next layer for learning. The final decision is represented by the readout layer.

II. Spiking Neural Network Model

In this section, we begin by presenting the feedforward SNN. The model is composed of four functional parts: feature extraction, encoding, learning, and readout. Fig. 1 shows a general architecture of the system model. The information stream flows in a feedforward way. The unified system extracts input image features through a STDP-based four-layer hierarchical network. These features are encoded as inputs to learning layer for further computation.

Feature extraction layer gains the spatiotemporal information of spike patterns. The encoding part uses these features to generate a set of specific activity patterns that represent various attributes of external stimuli. Learning part tunes neurons' weights so as to make sure particular neuron can respond to certain patterns correctly. Readout part extracts information of stimuli from given neural responses.

A. STDP-Based HMAX Spiking Model

Feature extraction serves as an important factor in object recognition. Here, we use a hierarchical bioinspired model to conduct it. Human beings are able to recognize surrounding objects effectively no matter from which angle or in what kind of environment with different light conditions and complex backgrounds. It is believed that this ability is originated from hierarchical processing in ventral pathway. A four-layer network consisting of $S_1 - C_1 - S_2 - C_2$ is adopted to emulate processing in visual cortex. In traditional HMAX, S cells gain selectivity from linear sum operation while C cells are equipped with the ability of nonlinear maximization operation. Masquelier and Thorpe [8] introduce a new method of transforming HMAX into temporal domain.

The first layer of S_1 cells detect edges using convolution by Gabor Filter. After transforming RGB format into gray image, the algorithm uses 5×5 convolution kernels, which roughly correspond to Gabor filters with the wavelength of 5, an effective width of 2, and four preferred orientations: $\pi/8, \pi/4 + \pi/8, \pi/2 + \pi/8, and 3\pi/4 + \pi/8$. Image scales are set to be 100%, 71%, 50%, 35%, and 25%. So there are $4 \times 5 = 20$ S_1 maps. After that, intensity of image is transformed into spikes by taking reciprocal of the value after convolution step, which means higher intensity corresponds to earlier spike.

Each C_1 cell propagates the first spike emitted by the S_1 cells in a 7×7 square neighborhood of one orientation and one scale. There is one row and column of overlapped S_1 cells when constructing C_1 map, which means that there are $6 \times 6 = 36$ times fewer C_1 cells than S_1 cells. The result after doing max pooling is a new convolution map with different scales and orientations. In C_1 layer, the algorithm sorts cells in a descendent way in about four orientations. The approach keeps the biggest one and records its place in the image and corresponding scale. The latency of this element is the reciprocal of the intensity, meaning the value of new convolution map. Then, it performs a general ascendant ranking over rows and columns under the same scale. In [8], they also take advantage of local lateral inhibition mechanism.

In the process of forming S_2 layer, the algorithm propagates multiple scales. Each S_2 feature has a prototype S_2 cell which is a weighted combination of bars (C_1 cells) with different orientations in a 16×16 square neighborhood. Within those maps, S_2 cells can integrate spike only from four C_1 maps of their chosen scales. Duplicated cells in all positions of all scale maps integrate spike chain, and the first one reaching the threshold would be the winner. For each prototype, the winner S_2 cell triggers unsupervised STDP rule and its 16×16 weight matrix will update.

To employ the STDP rule, the latency of C_1 cells and spike time of S_2 cells are assumed as the firing time of presynaptic neuron and postsynaptic neuron, respectively. A simplified version of STDP is used to learn $C_1 - S_2$ weights as follows:

$$\begin{cases} \Delta w_{ij} = a^+ \cdot w_{ij} \cdot (1 - w_{ij}), \text{ if } t_j - t_i \le 0 \\ \Delta w_{ij} = a^- \cdot w_{ij} \cdot (1 - w_{ij}), \text{ if } t_j - t_i \ge 0 \end{cases} \quad (1)$$

where i and j refer to the index of postsynaptic and presynaptic neurons, and t_i and t_j are spike times. Δw_{ij} is the synaptic weight, and a^+ and a^- are learning rates. Note that the exact time difference between two spikes does not affect the weight change, but only its sign is considered. These simplifications are equivalent to assuming that the intensity-to-latency conversion of S_1 cells compresses the whole spike wave in relatively short time interval.

The last layer is in charge of recording firing of S_2 cells. In [8], it is described as the C_2 layer because it captures the first spike related to the corresponding feature of the neuron which executes the function of max pooling. Once a neuron of the last layer already fires, it is stored with four kinds of data that we need in the next step. From this network, we extract useful information as follows.

1) *Firing Spike S_i:* The index of the fired cell. It represents the spatial occupation of the selected feature and neuron. It needs to be divided by the real pixel number of the corresponding scale to indicate the relative position in the whole image.
2) *Local Firing Spike L_i:* It stands for the number of iterations at this position before the corresponding neuron of last layer fires. Because the array is sorted before processed by the weight matrix, so the higher the local firing spike, the neighborhood of this position is more crucial.
3) *Firing Time T_i:* The value after the first convolution and latency coding. The lower the firing time, the more critical the pixel of this feature.
4) *Size A:* The size of a certain image after being converted into spike form.

An important contribution of our paper is to adopt and combine those extracted information which are not used in [8] or [9]. In [8], only local firing spike was used as an input to a classifier like SVM. In this paper, we develop a new encoding method to show that the appropriate usage of above variables turn out to enhance the effectiveness for object classification.

B. Sparse Temporal Feature Encoding

The encoding part aims at generating spiking patterns using features extracted from the above part. We transfer the temporal information obtained from the unsupervised training to more appropriate spatiotemporal patterns for each image.

Reference [14] discusses various biophysical mechanisms that enable individual neuron to rapidly perform an arithmetic operation on signals; it receives through synaptic inputs, thereby allowing information to be transformed and combined before it is converted into neuronal output. Besides, there are predications that the most efficient codes for representing information are sparse, as observed experimentally [15], [16]. So the main focus of this encoding part will be plausible mechanisms that use results obtained from the unsupervised procedure to obtain sparse representation of original images, which allows rapid arithmetic operation on input stimuli.

Inspired by [14], we propose an encoding method transforming features to spikes. After the unsupervised learning phase, we obtain several C_2 features corresponding to intermediate-complexity visual features which are optimum for object classification. We propose an encoding method to make use of spike information mentioned earlier to represent a feature of image within an encoding time window.

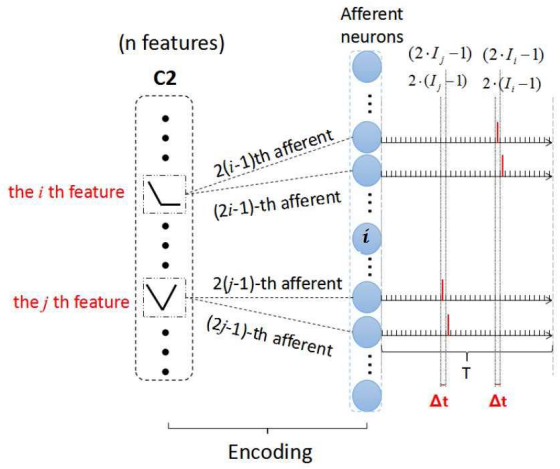


Fig. 2. Illustration of the encoding scheme. Top: generating $n \cdot 2$ spikes based on n C_2 features, every C_2 cell has two afferents for emitting spikes to represent its temporal and spatial information, respectively, the order of C_2 features in trains not necessarily corresponding to the order of its firing time in encoding time window. Here, it shows that the spikes emitted from two afferents representing jth feature which is later than spikes representing ith feature, yet afferents of ith feature are prior to jth feature in index of C_2 neurons. Red vertical bars: timings of spikes.

To evaluate temporal codes, the encoding window T is usually subdivided into smaller time bins(Δt) and the absence or presence of spikes in each bin is represented by a binary sequence [17]. We divide the encoding time window into twice the number of C_2 features and each time bin is Δt. Each C_2 features correspond to two components-temporal and spatial-so each C_2 feature corresponds to two small time bins. As the smaller of the firing time, the more important of this C_2 feature, we begin by sorting C_2 features by their mean firing time in an ascendant sequence. Our encoding scheme is shown in Fig. 2.

The encoding mechanism in [11] is a simple but an efficient example of coding, each input neuron fires once. In our model, an input neuron corresponds to one small time bin. In other words, each input neuron emits one spike within a time bin. In the first time slot, which represents the temporal information for a C_2 cell, both the firing time and local firing spike of each C_2 cell are utilized to determine when the input neuron emits a spike, and an emit time of this spike is computed as follows:

$$t_i^1 = 2 \cdot (I_i - 1) \cdot \Delta t + \frac{T_i \cdot \sigma}{L_i} \tag{2}$$

where t_i^1 is the emit time in the first time slot of the ith C_2 feature, I_i is the index of the C_2 feature representing the C_2 characteristic order in encoding window, Δt is the length of each time bin, T_i is the firing time of the ith C_2 feature, L_i is the local firing spike of the ith feature, and σ is a scaling parameter to ensure that the result of the firing time divided by local firing spike would not be too small.

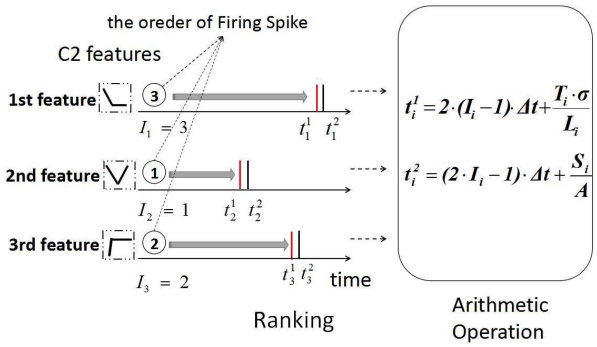


Fig. 3. Arithmetic-based encoding strategies. In the ranking part of the encoding phase, we use spike-order coding to determine the C_2 characteristic order in the encoding window. When the firing time is smaller, the two-time slot of the C_2 feature is earlier in encoding time window. After sorting C_2 features by the firing time, we compute the certain spatial and temporal time of C_2 feature through the proposed arithmetic operation. Red vertical line: temporal time information of the C_2 feature. Black vertical line: spatial time information of the C_2 feature.

In the second time bin, which represents the spatial information for a C_2 cell, we use firing spike of C_2 features to determine the emit time of its corresponding input neurons. As the firing spike of different image will change with the size of images, so the size of an image will be taken into consideration, the emit time of the second spike of a C_2 cell is computed as follows:

$$t_i^2 = (2 \cdot I_i - 1) \cdot \Delta t + \frac{S_i}{A} \tag{3}$$

where t_i^2 is the emit time in the second time slot of the ith C_2 feature, S_i is the firing spike of the ith C_2 feature, and A is the size of an image after being converted into spike trains through unsupervised learning phase. In Fig. 3, we give an example of three C_2 features to describe a detailed process of the encoding method, which consists of the ranking and arithmetic operation.

The encoding time window is the most important timescale characterizing a neural code. It is defined as the window containing the particular response patterns that are considered as the basic elements. Its length must be shorter than both the behavioral reaction time of the animal and the timescales on which the relevant stimulus features change. The length of the encoding time window depends on the time scale of the stimuli being encoded, and also on the reaction time of the system. It must be less than or equal to the duration of the integration window [18].

By simplifying the complex situation, where natural stimuli contain many features that vary independently on several timescale or it is unclear which features are represented by the neuron, we consider a feature-independent approach and compute information about all features using putative encoding windows whose length is varied relying on parameters, and then characterize the range of the encoding window length that maximizes the performance of the code [17], [19]. As the amount of features we use depends on the number of C_2 features, our encoding time window will change with the number of C_2 features. The length of time window will decrease when extracted C_2 features are less. Our encoding method not only maintains adequate information from original images but also facilitates later learning process.

C. SNN-Based Temporal Classifier

In the unified system, the learning part is formed from one layer of spiking neurons. As shown in Fig. 1, encoding neurons are fully connected to output neurons. Throughout this layer, the neurons are used to compute with precise spike timing.

Several temporal learning rules have been proposed to simulate processing and memorizing spatiotemporal patterns procedure of spike neurons. In this paper, we mainly use precise-spike-driven (PSD) learning rule [20], and we also test SPAN [21] and remote supervised method (ReSuMe) [22]. Later, we will prove that the PSD is faster and more effective and this is the reason why we adopt the PSD learning rule. As features have been extracted through the STDP-based HMAX unsupervised learning, the number of signal sources involved in learning decreases, which will directly benefit computation. Combined optimization with simple yet efficient synaptic adaptation of the learning algorithm, this network could transform streams of incoming spike trains into precisely firing spikes with high performance.

D. Readout Layer

The goal of readout part is to extract output spikes. As an example, we use two neurons to learn two different patterns generated by the encoding neurons. Each learning neuron corresponds to one category, and each of them is trained to produce a target spike train. The category of an input pattern will be determined by one of the neurons that generates the lowest spike distance. Here, we utilize Van Rossum metric [23] used in [20] to measure distance. The distance between two spike trains is measured as follows:

$$\text{Dist} = \frac{1}{\tau} \int_0^\infty [f(t) - g(t)]^2 \, dt \tag{4}$$

where τ is a free parameter, and $f(t)$ and $g(t)$ are filtered signals of the two spike trains that are considered for distance measurement. This distance parameter Dist is used for measuring and analyzing the performance of the learning rule. This rule is used in learning phase of SPAN only for measurement of criterion in ReSuMe and PSD.

III. Supervised Temporal Learning Rules

In the central nervous system, neuronal communication is mainly executed by the propagation of action potential or spikes. The fundamental computation of single neuron is transformation of incoming spike trains into an appropriate spike output. There are an increasing number of reports showing that the precise timing of an action potentials (spikes) carries significant information [4], [24]–[27], and for many relevant tasks, temporal precision in the neuronal responses is essential. We will mainly consider the supervised learning algorithms that are capable to learn the association between precise spike patterns. We have compared three widely adopted temporal learning algorithms for object recognition, i.e., PSD [20] rule,

SPAN [21], and ReSuMe [22]. We mainly consider PSD for our studies in the spatiotemporal pattern recognition due to its higher efficiency against the other two algorithms.

The neuron model is leaky integrate-and-fire (LIF), which is driven by an exponential decaying synaptic currents generated by its afferent synapses. According to the model, the equivalent circuit of the neuron is described as follows:

$$\tau_m \frac{du}{dt} = -[u - u_{\text{reset}}] + (I_{\text{ns}} + I_{\text{syn}}) \cdot R_m \tag{5}$$

where τ_m is the membrane time constant, u is the membrane potential, u_{reset} is the resting potential, I_{ns} and I_{syn} are the background current noise and synaptic current, respectively, and R_m is the resistance value of the membrane. But as we have segmented the object from background in the features extraction phase, so the I_{ns} is zero. For the postsynaptic neuron, the input synaptic current is computed as

$$I_{\text{syn}} = \sum_t w_i I^i_{\text{PSC}}(t) \tag{6}$$

where w_i is the synaptic efficacy of the ith afferent neuron, and I^i_{PSC} is the unweighted postsynaptic current from the corresponding afferent.

$$I^i_{\text{PSC}}(t) = \sum_{t^f} K(t - t^j) H(t - t^j) \tag{7}$$

where t^j is the time of the jth spike emitted from the ith afferent neuron, $H(t)$ refers to the Heaviside function, and K denotes the normalized kernel

$$K(t - t^j) = V_0 \cdot \left(\exp\left(\frac{-(t - t^j)}{\tau_s} \right) - \exp\left(\frac{-(t - t^j)}{\tau_f} \right) \right) \tag{8}$$

where V_0 is a normalization factor such that the maximum value of the kernel is 1. τ_s and τ_f are slow and fast decay constants, respectively, and their ratio is fixed as $\tau_s/\tau_f = 4$.

Motivated by Widrow-Hoff rule, synaptic weights of PSD rule are modified according to the following equation:

$$\frac{dw_i(t)}{dt} = \eta[s_d(t) - s_o(t)] I^i_{\text{PSC}}(t) \tag{9}$$

where η is a positive constant referring to the learning rate. $s_d(t)$ and $s_o(t)$ are desired outputs and actual output spike trains.

The spike train is defined as a sequence of impulses triggered by a particular neuron at its firing time. A spike train has the following form:

$$s(t) = \sum\nolimits_f \delta(t - t^f) \tag{10}$$

where t^f is the fth firing time, and $\delta(x)$ is the Dirac function: $\delta(x) = 1$ (if $x = 0$) or 0 (otherwise).

According to this rule, the polarity of synaptic changes depends on three cases: 1) a positive error, corresponding to a miss of the spike, when the neuron does not spike at the desired time; 2) a zero error, corresponding to a hit, when the neuron spikes at the desired time; and 3) a negative error, corresponding to false-alarm, where the neuron spikes when it is not supposed to.

With the PSD learning rule, after a learning trial, the total synaptic change is

$$\Delta w_i = \eta \left[\sum_g \sum_f K\left(t^g_d - t^f_i\right) H\left(t^g_d - t^f_i\right) - \sum_h \sum_f K\left(t^h_o - t^f_i\right) H\left(t^h_o - t^f_i\right) \right] \tag{11}$$

where $H(t)$ refers to the Heaviside function [$H(t) = 0$ if $t < 0$ and $H(t) = 1$ if $t \geq 0$]. The PSD provides an efficient way to processing spatiotemporal patterns, where the exact time of each spike is used for information transmission.

IV. Simulation Results

In this section, several simulations are performed to test the performance of the network and different learning rules. As features have been extracted through the STDP-based HMAX unsupervised learning, the number of signal sources involved in learning decreases, which will directly benefit computation.

A. Data Set and The Classification Problem

The data set that this paper mainly works on is 3D-Object data set provided by Savarese *et al.* at CVGLab, Stanford University [28]. The data set consists of 10 object categories (bike, car, shoe, iron, mouse, stapler, head, toaster, cellphone, and monitor). For each category, the data set contains images of 10 individual classes from eight different angles, three scales, and three tilts with various backgrounds which are similar to usual scenery.

Five instances of each object category are selected for the training set in learning phase and the rest of the categories serve as test set only used in test phase. All of the original sizes are preserved. The challenge of this data set is complex background and various scales and angles.

To verify the generalization ability, we also test the algorithm using Caltech101 data set. It contains 101 categories and each category has 40 to 800 images. The size of each image is roughly 300 × 200 pixels. Those images also have complicated backgrounds, and because some categories have limited number of pictures, the training set is comparatively small. This data set can prove the ability of taking advantage of small training set to achieve an object classification in complex environment and different conditions.

B. Encoding Images

Each image is converted into spatiotemporal pattern using strategy discussed. For the purpose of simplicity, we follow the illustration in [8] and set the Gabor Filter parameters. As mentioned in the previous part, each image goes through the learning phase and output abstract information. If the image is saturated, the program will skip it. After 20 000 iterations, features are clear to see. To illustrate that this step does extract features, here are features corresponding to the original image.

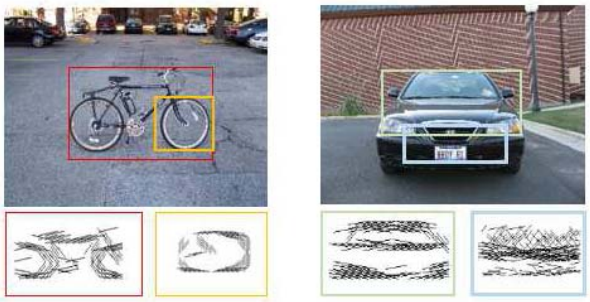


Fig. 4. Features selected from bicycles and cars. There are 50 feature prototypes learned from images. This is just for a better understanding of corresponding features.

In Fig. 4, it is obvious that the features represented in segments in different orientations are from images. In the later process, we will use the data related to those features to apply to the supervised learning phase.

We use those features exploited from the unsupervised procedure to multiple repeats of simulated stimuli. These stimuli contain a wide variety of spatial and temporal patterns, none of which dominated the stimulus but some of which are typically effective stimuli for learning neurons. The stimuli are, therefore, well studied to probe a neurons ability to convert spatial-temporal information into spike trains.

In the simulation experiments, the length of encoding time window is set as 25 ms. In Fig. 5, we show some spike pattern samples of bicycle and car, which are generated by the encoding method. From the two stable, periodic patterns shown in Fig. 5, we can see that after implementing the encoding method (which is presented in detail in Section II), even if the object instances are photographed in different conditions with different view angles, distances, and tilts, the spike patterns corresponding to one object instance are similar. This result suggests that the stimulus of one object is invariant to light, position, and scale. Experiments in [29] provide the empirical support for the hypothesis that a representation of object based on their bounding contours is well suited to mediate such an invariant coding.

To test findings about the spatial–temporal encoding method, we compare real data with the results of feature extraction. Feature extraction exactly preserves the firing-rate and firing-order in the original data, as described earlier. In fact, this procedure yields the only ensemble of spike trains that match the time-varying firing rate of real data and are rigorously consistent with the precise-firing model.

To quantify changes of encoding spike trains. We define *similarity index* to be the correlation coefficient between vectors derived from two different spike trains

$$\text{similarity index} = \frac{< s_1, s_2 >}{||s_1|| \cdot ||s_2||} \tag{12}$$

where s_1, s_2 are two subset vectors of input spike patterns from some afferent neurons. $< \cdots, \cdots >$ denotes inner product, and $||\cdots||$ denotes the norm, or the square root of a vector's inner product with itself. *similarity index* has a value close to 1 for maps that have similar shapes, 0 for maps that are nearly orthogonal, and -1 for maps that have opposite polarities. The similarity is an omnibus measure that averages over both space and time.

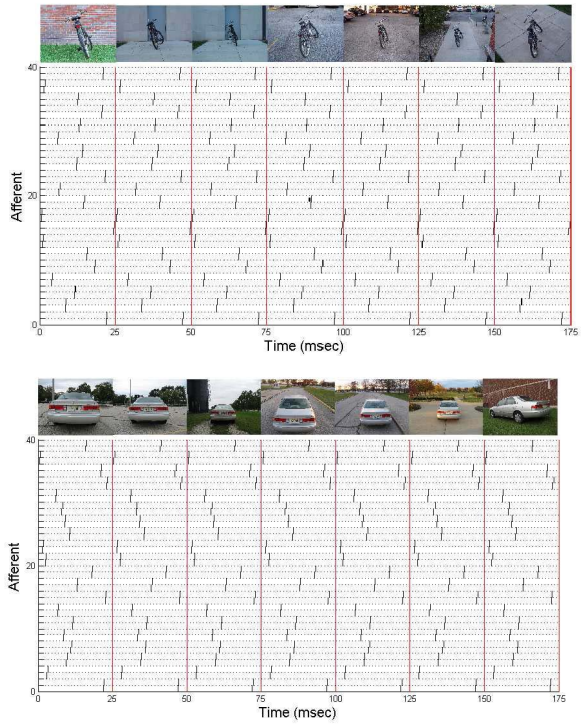


Fig. 5. Raster plots of spikes times of the afferent neurons. The spike trains of the first 40 afferent neurons are displayed (vertical lines). Top and Bottom: seven examples of the bicycle car images from the 3D-Object data set, respectively.

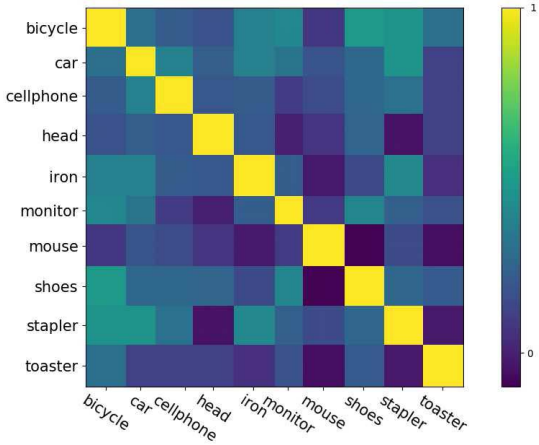


Fig. 6. Spike similarity analysis. We compute the *similarity index* of two spike trains selected from our set of spike patterns. Different *similarity index* values corresponding to different colors. The relationship between the *similarity index* and the color value is described at the right of the figure—the color bar.

After applying our encoding method to those images from 3D-Object data set [28], the input spike patterns of images from different categories are distinguishable. Fig. 6 shows a series plots that describe the relationship between two input spike patterns selected from spike patterns. As seen in Fig. 6, similarity indices of two spike patterns from one category are

TABLE I
TABULAR DESCRIPTION OF THE COMPARISON EXPERIMENT SETUP

	Neural Model	Synaptic Model
Type	Membrane time constant τ_m	Synaptic time constant τ_s, τ_f
Description	Membrane resistance R Spike threshold V_{thr} Reset potential u_r Time resolution dt	Synaptic weight initializing ω learning rate λ Simulation Time T Refractory period τ_{ref}
Algorithm		
PSD	$\tau_m = 10ms$ $R = 1M\Omega$ $V_{thr} = 10mV$ $u_r = 0mV$ $dt = 0.1ms$	$\tau_s = 10ms$, $\tau_s/\tau_f = 4$ $\omega \sim N(0.5, 0.2^2)\ nA$ $\lambda = 0.06$ $T = 25ms$ $\tau_{ref} = 0.75ms$
SPAN	$\tau_m = 10ms$ $R = 1M\Omega$ $V_{thr} = 10mV$ $u_r = 0mV$ $dt = 0.1ms$	$\tau_s = 1.25ms$ $\omega \sim N(0.5, 0.2^2)\ nA$ $\lambda = 0.03$ $T = 25ms$ $\tau_{ref} = 1ms$
ReSuMe	$\tau_m = 10ms$, $R_m = 10M\Omega$ $u_r = -65mV$ $T = 25ms$ $dt = 0.1ms$ $V_{thr} = -55mV$	$\tau_s = 3ms$ $\omega \sim N(0.5, 0.2^2)\ nA$ $\lambda = 0.1$ $\tau_{ref} = 0.75ms$

all almost equal to 1, while similarity indices of two spike trains from different categories are all close to 0, which means that the shapes of an input spike patterns from different classes are more distinguishable and spike patterns of one class have same shape.

Our encoding scheme extracts the basic information and encodes it to a spatiotemporal spiking pattern. Through the whole encoding structure, a sparse representation of the original incoming image is eventually obtained.

C. Comparison Among Different Temporal Learning Algorithms

In order to associate input spike trains with desired output spike trains effectively, we specify the parameters in the temporal learning rules–PSD rule, SPAN rule, and ReSuMe rule, and compare the learning speed and performance of these three algorithms. For the comparison purpose, we use the LIF model as the spiking neuron model in these three temporal learning rule. Before this comparison, we have tuned these three temporal rules with different sets of parameter. Therefore, each learning rule can reach the optimal performance with its set of parameters. Table. I explains the neural and synaptic models along with their parametrization of the temporal learning rules.

To compare the learning speed and performance of these three learning rules, we use input patterns obtained from the encoding phase. The learning purpose of these rules is converting an input spike trains into a target spike output. In this trial, we select 200 images belonging to each category, and test both the learning performance and speed on the 10 categories from the 3D-Object data set [28]. We record the accuracy of the image classification of different learning rule and the converged epochs of iterations.

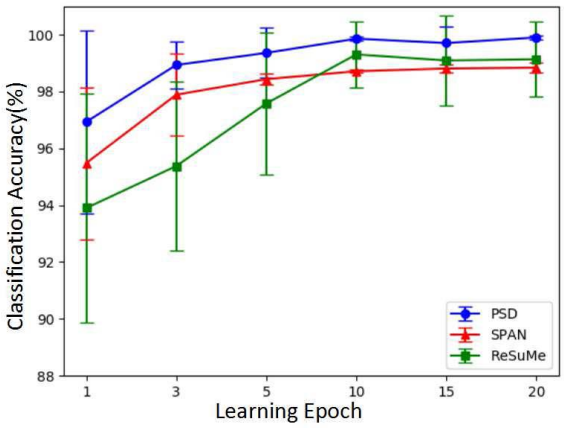


Fig. 7. Learning speed and performance of the PSD, SPAN, and ReSuMe learning algorithms.

Under 10-fold cross validation, we perform the comparing experiment for 20 times and the results are shown in Fig. 7. The final measurements of performance for each learning rule have been calculated by averaging the values of performance measures for each fold of the 10-fold cross validation process.

According to Fig. 7, the classification accuracy is not significantly different between PSD and ReSuMe, and both the accuracies are higher than the SPAN learning rule. This is because the input spike patters obtained through our encoding method are distinguishable, thereby significantly reducing the learning difficulty and improve the learning speed. As the weight adapting rule of PSD is briefer than ReSuMe and SPAN, it took less computation time in each epoch. The computation for each PSD learning epoch will finish within 4.2 s, while the SPAN and ReSuMe require 10.2 s and 7.8 s, respectively. Compared with the SPAN and ReSuMe, each learning epoch of PSD rule will take much less time, and the learning performance is much more stable. Besides, the learning speed of ReSuMe changes enormously for different initial conditions. Therefore, in order to realize the image recognition task quickly and stably, we choose PSD rule as algorithm in learning phase.

D. Performance Analysis of the Integrated System

According to the result of the comparison among PSD, SPAN, and ReSuMe, it is obvious that PSD could be more effective in learning phase. In this section, the function of our system for object classification is discussed. To analyze the ability of network to solve recognition task, we use spatiotemporal spike patterns we obtain from 3D-Object data set [28]. As described previously, all neurons are supposed to fire only once during the entire time window ($T = 25$ ms). Every two adjacent afferent neurons represent the spatial and temporal information of a C_2 features. The encoding method illustrated in Fig. 2 is applied to converting C_2 features into the spatiotemporal spike patterns.

We select 10 learning neurons trained by PSD, with each learning neuron corresponding to one category (bike, car, and so on). The learning parameter in PSD is set to be $\eta = 0.06$ and $\tau_s = 10ms$. All the learning neurons are trained to fire at

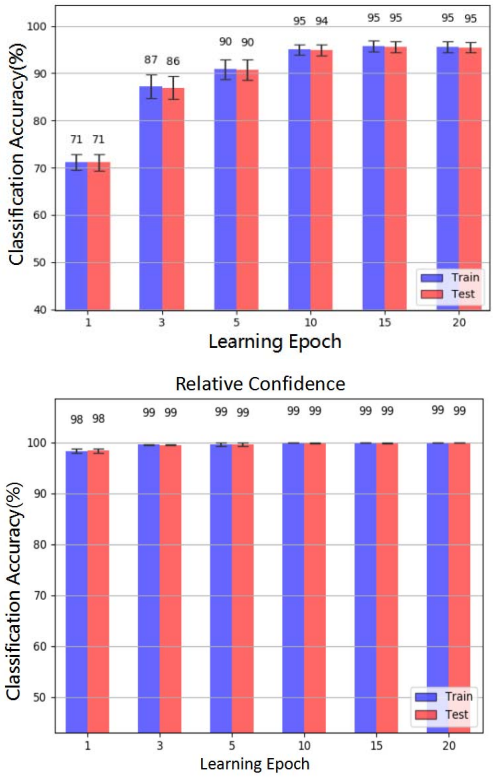


Fig. 8. Illustration of performance of PSD rule during the learning process. There are 10 categories from 3D-Object data set (to be classified). The average accuracies are represented by shaped bars. All the data are averaged over 10 runs.

target spike train with the corresponding category. The target spike train is set to be evenly distributed over the time window T (best set is 25 ms for the 3-D Object data set after many experiments, we will use a different length of time window for Caltech101 data set), with a specified number of spikes n ($n = 1$ we used here, considering that the more spikes in the target spike train, the smaller the memory capability would be). We set target time according to the method proposed in [30].

After training, classification is performed on both training and testing set. To evaluate the performance of the learning rule based on our encoding method, we use two decision-making criterions proposed in [20]: absolute confidence and relative confidence. However, in our learning scheme, with the absolute confidence criterion, only if the distance between the desired spike train and the actual output spike train is close to a specified value (0.05 used in our scheme), the input pattern will be regarded as correctly classified.

Fig. 8 shows the average classification accuracy for all categories under two proposed decisions. From the absolute confidence criterion, the average accuracy for the training set and testing set is 95.57% and 95.48%. Under the relative confidence, the average accuracies for the training and testing set are 99.95% and 99.82%. The performance of the classification task is, therefore, significantly improved by the relative confidence decision making criterion. With the absolute confidence criterion, the trained neuron strives to find a good match with the memorized patterns.

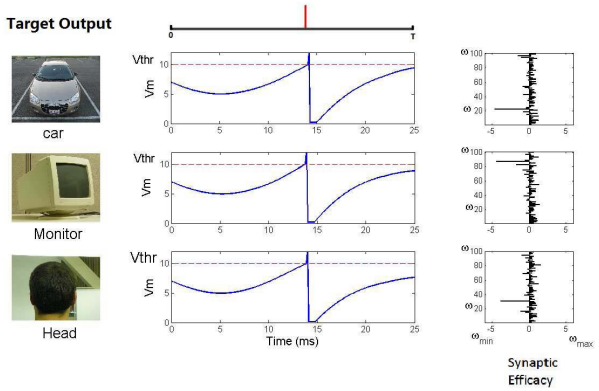


Fig. 9. Illustration of the voltage trace and synaptic efficacy for each category, three sample images from these three categories (left), voltage traces (middle) along with the synaptic weights (right) calculate for the input of the three images.

With the relative confidence criteria, the learning neuron (each corresponding to one category) attempts to find the most likely category through competition. The average accuracy for each category under two proposed decision criterions are shown in Table II (Acc stands for Accuracy; RC and ac stand for relative confidence and absolute confidence, respectively).

Fig. 9 shows the voltage trace and synaptic weight of the three learning neurons corresponding to car, monitor, and head. With Fig. 9, it is obvious that after successful learning, spikes occurs at desired times and membrane potential is below threshold in the rest of time.

Moreover, to test our system robustness to noise, we have tested our method with adding Gaussian jitter to each input spike to generate the noise pattern. If we train with the initial spike trains (no noise, deterministic training), with the jitter strength of test data sets increased, the generalization accuracy will decrease quickly. But if we train with noise training data sets, the learning neuron can successfully fire a target spike train with a relatively high accuracy when the noise strength is not higher than the one used in the training. In general, the model is less sensitive to the noise if the noisy training is performed. The robustness of our system to input jitter is shown in Fig. 10.

V. Discussion

In this paper, we have built a temporal feature encoding model by employing an asynchronous HMAX model. The whole system for pattern recognition task goes through unsupervised STDP-based learning, encoding, and supervised learning. PSD supervised learning algorithm is extremely fast with small number of iterations of 5–10, corresponding to fast reaction in primate perception. As stated in [3], this feedforward architecture remains in debate since the anatomical back projections are found abundantly present between almost all areas of visual cortex. However, considering the fact that primates are remarkably good at recognizing objects in a short time and activities of small neuronal populations in IT cells appear after short time span from stimulus onset (around 100 ms) [31], the hypothesis of a basic feedforward architecture for processing is highly supported.

TABLE II

MULTICATEGORY CLASSIFICATION OF SPATIOTEMPORAL PATTERNS

Acc(%)	bicycle		car		cellphone		head		iron		monitor		mouse		shoe		stapler		toaster	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test	train	test	train	test	train	test
RC	100	100	99.93	99.79	99.86	99.84	99.90	99.80	99.95	99.85	99.95	99.78	99.98	99.86	99.97	99.85	99.96	99.84	99.89	99.86
AC	67.58	66.06	98.33	97.92	99.53	99.35	91.44	91.49	99.47	98.86	99.08	98.01	98.47	98.26	98.87	98.77	99.28	99.21	99.55	99.28

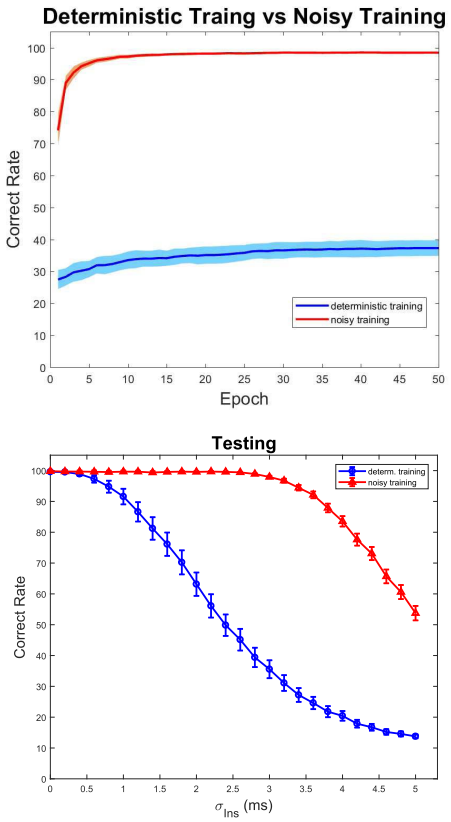


Fig. 10. Robustness of the learning rule against jittering noise of input stimuli. The neuron is trained under noise-free conditions (denoted as deterministic training), or is trained under noisy condition (denoted as noisy training). In the training phase (top row), the neuron is trained for 50 epochs. Along the training process, the average classification accuracy is shown. The standard deviation is denoted by the shaded area. In the testing phase (bottom row), the generalization accuracies of the trained neuron on different levels of noise patterns are presented. Both the average value and the standard deviation are shown. All the data are averaged over 100 runs.

Besides, our algorithm puts an emphasis on PSD learning mechanism. The eventual output is set to be precise spikes rather than classification index. Thus, the proposed SNN combined with STDP-based HMAX sparse temporal feature encoding can be considered a biological plausible model, which could explain the extremely rapid visual information recognition process in neural systems.

To test the validity of the algorithm, we choose another data set, considering that in 3-D data set, there are only 10 categories. Fig. 11 shows the accuracy of each category in 10-classification task on Caltech101. Training pictures in each class we use around 30 images and testing images are around 10–20, which also depend on the total number of images in the certain category. The accuracy of classification on both Caltech101 and 3-D data set are shown in Table III. The number of iterations is larger in Caltech101 because

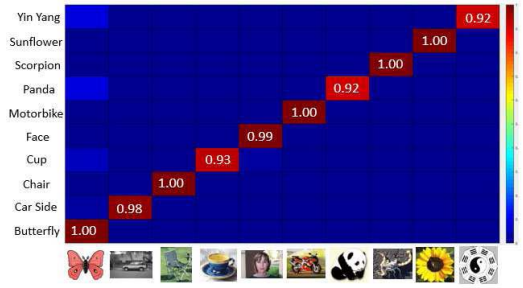


Fig. 11. Accuracy of each category classification for the 10-category Caltech101 data set.

TABLE III

RESULT COMPARISON

Data	Feature	Time	Iteration	Accuracy	
				Training	Testing
Caltech-20	20	10.0	150	94.3%	93.0%
Caltech-10	20	10.0	150	97.8%	96.3%
3D-10	50	25.0	30	99.8%	99.6%

training data in Caltech101 are relatively smaller than in 3-D data set. More iterations are required to achieve good feature extraction for better classification.

As shown in Table III, our classification accuracy is higher than the method proposed in [9]. By features extraction and the encoding phase, we could get a low dimension spatial–temporal representation of an original picture, and the PSD learning rule could associate input spike trains with desired output spike trains effectively. However, in [9], a one-versus-one multiclass linear SVM should learn $n * (n-1)/2$ hyperspace to divide n classes, which needs spending lots of computation time. Moreover, for the ETH 3D-Object data sets, our integrated system could attain the higher classification accuracy (99%) than that of [9] (96%).

The combination of spatial and temporal message is a critical factor for classification. The work indicates that with proper encoding scheme to represent the temporal features, SNN could perform as well as, or even better than the state-of-art learning method. We also characterize the ability of spiking neurons to implement desired transformation between input and output spike patterns. The speed is fast which could run within a minute on a machine with Core i7-4790, 3.60 GHz CPU, and 8G RAM. Implementation of temporal feature encoding and learning on neuromorphic hardware will give more advantages to SNN-based feature encoding and recognition, as demonstrated in recent works [6], [7]. Temporal encoding and learning as a consistent model has also been successful in emulating the memory formulation mechanisms [32]. Our proposed temporal encoding and learning model will provide new efficient methods for robust object recognition, which are easily adaptable for neuromorphic implementations.

Arithmetic operations allow neurons to combine and transform different signal and to distinguish both spatial and

temporal correlation in their synaptic input. We show that the encoded input spike trains are high cohesion and low coupling between different classes. Thanks to this strength of proposed encoding mechanism, there is little weight interference among different learning neurons corresponding to one category (10 learning neurons for 3D-Object data set, 20 learning neurons for Caltech101 data set), during the input–output association learning procedure. Otherwise, weights adaption will be interfered if there is high coupling between categories.

SNN-based algorithms compatible with hardware implementation are highly desirable for fabricating neuromorphic computing hardware, such as the neuromorphic chips introduced in [33]. The availability of an efficient SNN-based encoding and learning algorithms will directly benefit the computation with neuromorphic chips, rather than resorting to unfavorable mapping processes from deep learning models to SNN that are configured for many of the current neuromorphic hardware applications [34], [35].

VI. Conclusion

This paper demonstrates an efficient computational feedforward architecture using the hybrid of supervised and unsupervised learning. Visual features are represented as temporal information throughout the architecture. The effectiveness of our feedforward SNN has been verified by simulation results on 3D-Object data set and Caltech101 data set. We argue that the spare encoded visual features help to make a dimension reduction of the original data and leads to a more efficient SNNs-based supervised leaning and robust recognition.

Arithmetic operations allow neurons to combine and transform different signal and to distinguish both the spatial and temporal correlation in their synaptic input. We show that the encoded input spike trains are high cohesion and low coupling between different classes. Thanks to this strength of proposed encoding mechanism, there is little weight interference among different learning neurons corresponding to one category (10 learning neurons for 3D-Object data set and 20 learning neurons for Caltech101 data set), during the input–output association learning procedure. Otherwise, weights adaption will be interfered if it is high coupling between categories.

References

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

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[3] T. Serre, A. Oliva, and T. Poggio, "A feedforward architecture accounts for rapid categorization," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 15, pp. 6424–6429, 2007.

[4] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, Feb. 2008.

[5] C. Hung, G. Kreiman, T. Poggio, and J. DiCarlo, "Fast readout of object identity from macaque inferior temporal cortex," *Science*, vol. 310, no. 5749, pp. 863–866, 2005.

[6] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "HFirst: A temporal approach to object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2028–2040, Oct. 2015.

[7] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "HOTS: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1346–1359, Jul. 2017.

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

[9] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382–392, Sep. 2016.

[10] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, "A spike-timing-based integrated model for pattern recognition," *Neural Comput.*, vol. 25, no. 2, pp. 450–472, Feb. 2013.

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

[12] X. Xie, H. Qu, Z. Yi, and J. Kurths, "Efficient training of supervised spiking neural network via accurate synaptic-efficiency adjustment method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1411–1424, Jun. 2017, doi: 10.1109/TNNLS.2016.2541339.

[13] R. Gopalakrishnan and A. Basu, "Triplet spike time-dependent plasticity in a floating-gate synapse," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 778–790, Apr. 2017.

[14] R. A. Silver, "Neuronal arithmetic," *Nature Rev. Neurosci.*, vol. 11, no. 7, p. 474, 2010.

[15] W. B. Levy and R. A. Baxter, "Energy efficient neural codes," *Neural Comput.*, vol. 8, no. 3, pp. 531–543, 1996.

[16] D. Attwell and S. B. Laughlin, "An energy budget for signaling in the grey matter of the brain," *J. Cerebral Blood Flow Metabolism*, vol. 21, no. 10, pp. 1133–1145, 2001.

[17] S. P. Strong *et al.*, "Entropy and information in neural spike trains," *Phys. Rev. Lett.*, vol. 80, no. 1, pp. 197–200, 1998.

[18] F. Theunissen and J. P. Miller, "Temporal encoding in nervous systems: A rigorous definition," *J. Comput. Neurosci.*, vol. 2, no. 2, pp. 149–162, 1995.

[19] S. Panzeri, N. Brunel, N. K. Logothetis, and C. Kayser, "Sensory neural codes using multiplexed temporal scales," *Trends Neurosci.*, vol. 33, no. 3, pp. 111–120, 2010.

[20] Q. Yu, H. Tang, K. Tee, and H. Li, "Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns," *PloS ONE*, vol. 8, no. 11, p. e78318, 2013.

[21] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *Int. J. Neural Syst.*, vol. 22, no. 4, pp. 1250012-1–1250012-17, 2012.

[22] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, 2010.

[23] M. C. W. van Rossum, "A novel spike distance," *Neural Comput.*, vol. 13, no. 4, pp. 751–763, Apr. 2001.

[24] V. J. Uzzell and E. J. Chichilnisky, "Precision of spike trains in primate retinal ganglion cells," *J. Neurophysiol.*, vol. 92, no. 2, pp. 780–789, 2004.

[25] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *J. Neurosci.*, vol. 20, no. 14, pp. 5392–5400, 2000.

[26] W. Bair and C. Koch, "Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey," *Neural Comput.*, vol. 8, no. 6, pp. 1185–1202, 1996.

[27] R.-M. Memmesheimer, R. Rubin, B. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, 2014.

[28] S. Savarese and L. Fei-Fei, "3D generic object categorization, localization and pose estimation," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[29] Y. El-Shamayleh and A. Pasupathy, "Contour curvature as an invariant code for objects in visual area V4," *J. Neurosci.*, vol. 36, no. 20, pp. 5532–5543, 2016.

[30] Q. Yu, R. Yan, H. Tang, K. C. Tan, and H. Li, "A spiking neural network system for robust sequence recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 621–635, Mar. 2016.

[31] P. I. Perrett, J. K. Hietanen, M. W. Oram, P. J. Benson, and E. T. Rolls, "Organization and functions of cells responsive to faces in the temporal cortex," *Phil. Trans., Biol. Sci.*, vol. 335, no. 1273, pp. 23–30, 1992.

[32] J. Hu, H. Tang, K. C. Tan, and H. Li, "How the brain formulates memory: A spatio-temporal model," *IEEE Comput. Intell. Mag.*, vol. 11, no. 2, pp. 56–68, May 2016.

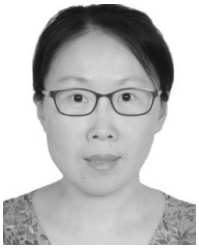
[33] N. Qiao *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers Neurosci.*, vol. 9, p. 141, Apr. 2015.
[34] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2015.
[35] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, pp. 11441–11446, Aug. 2016.

Yajing Zheng received the B.S. degree from Sichuan University, Chengdu, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Computer Science, Peking University, Beijing, China.

Her current research interests include neuroscience, brain-inspired computing, machine learning, and spiking neural networks.

Shixin Li received the bachelor's degree from the Department of Electronics Engineering, Sichuan University, Chengdu, China, in 2017. She is currently pursuing the master's degree with the Computer Science Department, University of California at San Diego, San Diego, CA, USA.

Her current research interests include perception and control for robotics using learning algorithms.



Rui Yan (M'11) received the bachelor's and master's degrees from the Department of Mathematics, Sichuan University, Chengdu, China, in 1998 and 2001, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, in 2006.

She was a Post-Doctoral Research Fellow with the University of Queensland, Brisbane, QLD, Australia, from 2006 to 2008, and a Research Scientist with the Institute for Infocomm Research, A*STAR from 2009 to 2014. She is currently a Professor with the College of Computer Science, Sichuan University. Her current research interests include intelligent robots, brain-inspired computing, and cognitive systems.

Huajin Tang (M'01) received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 1998, the M.Eng. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005.

He was a System Engineer with STMicroelectronics, Singapore, from 2004 to 2006. From 2006 to 2008, he was a Post-Doctoral Fellow with the Queensland Brain Institute, St Lucia, QLD, Australia, and the University of Queensland, Brisbane, QLD, Australia. Since 2008, he has been the Head of the Robotic Cognition Laboratory, Institute for Infocomm Research, A*STAR, Singapore. He is currently a National Youth-1000 Talent Distinguished Professor with Sichuan University, Chengdu, China. His research work on Brain GPS has been reported by MIT Technology Review in 2015. His current research interests include neuromorphic computing, neuromorphic hardware and cognitive systems, and robotic cognition. Dr. Tang received the 2016 IEEE Outstanding TNNLS Paper Award. He was the Program Chair of the 6th and 7th IEEE CIS-RAM and Chair of the 2016 and 2017 IEEE Symposium on Neuromorphic Cognitive Computing. He has served as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, and *Frontiers in Neuromorphic Engineering*.

Kay Chen Tan (SM'08–F'14) received the B.Eng. degree (Hons.) in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Full Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. He has authored or co-authored over 200 refereed articles and six books.

Dr. Tan is an elected member of the IEEE CIS AdCom from 2017 to 2019. He was the Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2010 to 2013. He is the Editor-in-Chief of the IEEE *Transactions on Evolutionary Computation*. He currently serves as the Editorial Board Member of over 20 journals.