

Mapping Temporal Variables Into the NeuCube for Improved Pattern Recognition, Predictive Modeling, and Understanding of Stream Data

Enmei Tu, Nikola Kasabov, *Fellow, IEEE*, and Jie Yang

Abstract—This paper proposes a new method for an optimized mapping of temporal variables, describing a temporal stream data, into the recently proposed NeuCube spiking neural network (SNN) architecture. This optimized mapping extends the use of the NeuCube, which was initially designed for spatiotemporal brain data, to work on arbitrary stream data and to achieve a better accuracy of temporal pattern recognition, a better and earlier event prediction, and a better understanding of complex temporal stream data through visualization of the NeuCube connectivity. The effect of the new mapping is demonstrated on three benchmark problems. The first one is the early prediction of patient sleep stage event from temporal physiological data. The second one is the pattern recognition of dynamic temporal patterns of traffic in the Bay Area of California and the last one is the Challenge 2012 contest data set. In all the cases, the use of the proposed mapping leads to an improved accuracy of pattern recognition and event prediction and a better understanding of the data when compared with traditional machine learning techniques or SNN reservoirs with an arbitrary mapping of the variables.

Index Terms—Early event prediction, NeuCube architecture, spatiotemporal data, spiking neural network (SNN).

I. INTRODUCTION

TEMPORAL data have been collected in various fields, such as brain science, ecology, geophysics, and social sciences. Temporal data may contain complex temporal patterns that would need to be learned and extracted in a computational model. In some cases, the variables describing the temporal data have spatial attributes, e.g., the 3-D location of the channels in electroencephalography (EEG) data [1] and the patterns that need to be learned become spatiotemporal. Learning dynamic patterns from temporal and spatiotemporal

Manuscript received August 14, 2014; revised February 21, 2016; accepted February 28, 2016. Date of publication March 15, 2016; date of current version May 15, 2017. This work was supported in part by the Education New Zealand, in part by the Tripartite Project within the Auckland University of Technology, New Zealand, Shanghai Jiao Tong University (SJTU) (East China) and Xinjiang University (West China) China, in part by the Knowledge Engineering and Discovery Research Institute, SJTU University, in part by the National Natural Science Foundation of China under Grant 61572315, and in part by the 973 Plan China under Grant 2015CB856004.

E. Tu and J. Yang are with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: hellotem@hotmail.com; jieyang@sjtu.edu.cn).

N. Kasabov is with the Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Auckland 1010, New Zealand (e-mail: nkasabov@aut.ac.nz).

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.2016.2536742

data is the challenging task, as the temporal features may manifest complex interaction that may also change dynamically over time. The time-windows of important temporal or spatiotemporal patterns may change over time, which is seldom covered by traditional machine learning methods, such as regression techniques, support vector machines (SVMs), or multilayer perceptrons (MLPs).

Many recurrent models have been proposed to learn spatiotemporal relationship from signal, such as the recursive self-organizing network models [2], [3] and recurrent neural network [4], [5]. Recently, NeuCube [1], [6] has been proposed to capture time and space characteristics of spatiotemporal brain data in a spiking neural network (SNN) architecture. The NeuCube architecture consists of: input data encoding module, that encodes multivariable continuous temporal stream data into spike trains; a 3-D recurrent SNN cube (SNNcube), where input data are mapped and learned in an unsupervised mode; and an SNN classifier that learns in a supervised mode to classify the spatiotemporal patterns of the SNNcube activities that represent patterns from the input data. The effectiveness and the superiority of NeuCube to model brain data (such as EEG) have been demonstrated in the previous works. Kasabov and Capecci [7] study the modeling ability of NeuCube for EEG spatiotemporal data measuring complex cognitive brain processes. In [8], NeuCube is used for modeling and recognition of complex EEG spatiotemporal data related to both physical and intentional (imagined) movements. Doborjeh *et al.* [9], [10] present the results of using NeuCube to classify/cluster fMRI data, and Capecci *et al.* [11] use NeuCube for the analysis of EEG data recorded from a person affected by absence epileptic. Capecci *et al.* [12] study how NeuCube can help understand brain functional changes in a better way.

While in these studies, NeuCube is used to model brain data, and the features are manually mapped into the system according to the spatial location where the signals are sampled from, and in this paper, we present a fully automatic input mapping method based on the graph matching technique that enables NeuCube to model any spatiotemporal data and leads to better spatiotemporal pattern recognition, early event prediction and model visualization, and understanding.

The main contributions of this paper are as follows.

- 1) We propose a new graph matching algorithm to optimize the mapping of the input features (variables) of arbitrary temporal stream data into a 3-D SNNcube with the

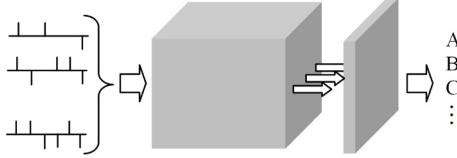


Fig. 1. Simplified representation of the NeuCube architecture [1], [6]. Spike trains (left) are fed to the SNNcube, and the neuron firing state vectors of the SNNcube are used to train an SNN classifier.

goal to achieve a better pattern recognition accuracy, an earlier event prediction from temporal data, and a better understanding of the data through model visualization.

- 2) We proposed a graph-based semisupervised learning algorithm to automatically analyze the neuronal cluster structure inside the trained SNNcube and develop various dynamic functions for the visualization of the neuronal activity states and synaptic evolving progress during learning process, and we prove the convergence of the new algorithm.
- 3) We present two algorithms, spike density correlation and maximum spike coincidence, for spike trains similarity measure that can be used in the graph-matching algorithm for two commonly encountered types of spike trains: bipolar spike trains and unipolar spike trains.
- 4) We introduce an adaptive threshold-based (ATB) encoding algorithm by using mean and standard deviation information of the signal gradient to calculate its threshold and thus make ATB self-adaptive to signal changes.
- 5) We develop an optimization component using genetic algorithm (GA) that can automatically optimize all NeuCube parameters in once and thus enable users to find optimal parameters' value easily and achieve good results.

The remainder of this paper is organized as follows. Section II describes the NeuCube architecture. Section III proposes a new method for the optimization of the mapping of the input temporal features into the SNNcube. Experiments on three benchmark data sets are reported in Section IV, followed by the conclusions and discussions in Section V.

II. NeuCube ARCHITECTURE

The main parts of NeuCube are: an input encoding module; a 3-D recurrent SNN reservoir/cube (SNNcube); and an evolving SNN classifier. Fig. 1 shows the block diagram of the NeuCube architecture. The SNNcube is a scalable module. Its size is controlled by three parameters: n_x , n_y , and n_z , representing the neuron number along the x , y , and z axes, respectively. The total number of neurons in the SNNcube is $N = n_x \times n_y \times n_z$. As a first implementation here, we use a probabilistic leaky integrate and fire (LIF) model [13].

The encoding module converts continuous data streams into discrete spike trains, suitable to be processed in the SNNcube, because spike neural networks can only process discrete spike trains. After encoding, the information contained in original continuous signal is transformed into the forms of spikes.

The NeuCube is trained in a two-stage learning process. The first stage is the unsupervised learning that makes the

SNNcube learn spatiotemporal relations from the input data by adjusting the connection weights in SNNcube. The second stage is the supervised learning that aims at learning the class information associated with each training spatiotemporal sample.

To be more specific, the unsupervised learning stage is intended to encode hidden spatiotemporal relationships from the input data into neuronal connection weights. According to the Hebbian learning rule, if the interaction between two neurons is persistent, then the connection between them will be strengthened. In particular, we train the SNNcube using spike-timing-dependent synaptic plasticity (STDP) learning rule [14]: if neuron j fires before neuron i , then the connection weight from neuron j to neuron i will increase, and if the spikes are in a reverse order, the connection from neuron i to neuron j will decrease. This ensures that the time difference in the input spiking trains, which encode the temporal patterns in the original input signals, will be captured by the neuron firing state and the unsymmetrical connection weights in the reservoir.

The second training stage is to train an output classifier using class label information associated with the training temporal samples. The dynamic evolving SNNs (deSNNs) [15] is employed here as an output classifier, because deSNN is computationally efficient and emphasizes the importance of both first spikes arriving at the neuronal inputs (observed in biological systems [16]) and the following spikes (which in some stream data are more informative).

Once a NeuCube model is trained, all connection weights in the SNNcube and in the output classification layer are established. These connections and weights can change based on further training (adaptation) because of the evolvable characteristic of the architecture.

III. METHOD FOR AN OPTIMAL MAPPING OF TEMPORAL INPUT VARIABLES INTO A 3-D SNNCUBE BASED ON THE GRAPH MATCHING ALGORITHM

A. Method for Mapping Input Temporal Variables Into a 3-D SNNcube Based on Graph Matching Optimization Algorithm

Given a particular spatiotemporal data set, it is important to optimize the mapping of the data into the 3-D SNNcube for an optimal learning, a better understanding of the spatiotemporal patterns in the data. For some spatiotemporal data, such as brain EEG, there is prior information about the location of each channel (input feature), and this information can be readily utilized for mapping the EEG temporal signal into the SNNcube [6]. But for other common applications, such as climate temporal data, we do not have such spatial mapping information. In addition, the way that the temporal data are mapped into the SNNcube would significantly impact the results. Here, we introduce a new method to map temporal variables into the SNNcube for a better pattern recognition, a better and earlier event prediction, and a better visualization of the model to explain the data.

Suppose there are s temporal samples in the data set, measured through v temporal variables, the observed time

length of each sample is t . We first choose randomly v input neurons from the SNNcube. Then, we map the variables into the SNNcube following the principle: input variables/features that, after the input data transformation into spike trains, represent highly correlated spike trains are mapped to nearby input neurons. Because high correlation indicates that the variables are likely to be more time-dependent with each other, this relationship should also be reflected in the connectivity of the 3-D SNNcube. Spatially close neurons in the SNNcube will capture in their connections more temporal interactions between the input variables mapped into these neurons. The principle of mapping similar input vectors into topologically close neurons is known from the Self-Organizing Maps (SOM) [17], but in the SOM, these are static vectors and the similarity is measured by the Euclidean distance. Now, we address the problem of mapping temporal sequences, rather than the static vectors, into a 3-D SNN.

Specifically, we construct two weighted graphs: the input neuron similarity graph (NSG) and the time series/signals similarity graph (SSG). In the NSG, the input neuron's spatial 3-D coordinates are denoted by $V_{NSG} = \{(x_i, y_i, z_i) | i = 1..v\}$, and the graph edges are determined in the following way: each input neuron is connected to its k nearest input neurons, and the edges are weighted by the inverse of the Euclidean distance between them.

In the SSG, we denote the spike train corresponding to the feature i as

$$s_i = \sum_{k=1}^{N_i} \delta(t - t_k) \quad (1)$$

where $\delta(t)$ is the Dirac delta function. The set of spike trains corresponding to all the features is $V_{SSG} = \{s_i | i = 1..v\}$, and it forms the graph vertex set. The graph edges are constructed in this way: each spike density function is connected to its k highest correlated neighbors, and the edges are weighted by the similarity between them. Measuring the similarity between spike trains is an important problem in neural science and has been studied for a long history [18], [19]. Here, we propose two simple but effective spike trains similarity measuring algorithms for two most commonly encountered spike train types, respectively. The first one is spike density correlation, which is suitable for measuring unipolar spike trains with two states (firing or unfiring). This kind of spike train is commonly observed in glutamatergic neurons and produced by some spike encoding algorithms, such as Bens Spiker Algorithm [20]. The other one is maximum spike coincidence, which is suitable for measuring bipolar spike trains with three state (excitatory, rest, or inhibitory). This kind of spike train is commonly produced by serotonergic neurons, as well as some encoding algorithms, such as address event representation (AER) [21].

1) Spike Density Correlation: Given a unipolar spike train s_i , we first use the kernel density estimation method to estimate its spike density function by

$$p_i(t) = \frac{1}{N_i} K_h(t) * \sum_{k=1}^{N_i} \delta(t - t_k) = \frac{1}{N_i h} \sum_{k=1}^{N_i} K\left(\frac{t - t_k}{h}\right) \quad (2)$$

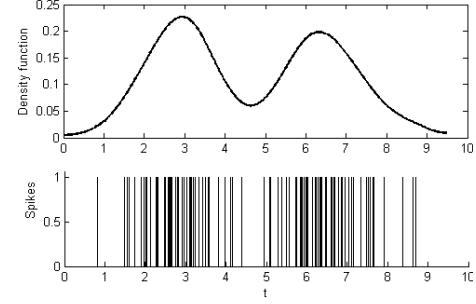


Fig. 2. Spike density estimation result.

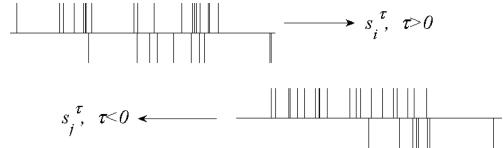


Fig. 3. Process of computing maximum spike coincidence.

where $*$ is the function convolution operator and $K(t)$ is a kernel function, with kernel bandwidth h . Fig. 2 shows the spike density estimation result. Thereafter, the similarity between spike train s_i and s_j is defined as Pearson's linear correlation coefficient

$$\omega(s_i, s_j) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad (3)$$

where μ_X and σ_X are the mean and the standard deviation of X , respectively. E is the expectation. Here, X and Y are the random variables corresponding to s_i and s_j , respectively.

2) Maximum Spike Coincidence: Given two bipolar spike trains s_i and s_j , we define the similarity between them as the maximal number of spikes that coincide, while one spike train moves over the other spike train

$$\omega(s_i, s_j) = \max_{\tau} \sum_{k=1}^{N_i} I(s_i(k), s_j^\tau(k)) \quad (4)$$

where s_i^τ is the result of translating s_i with τ time unit, that is

$$s_i^\tau = \sum_{k=1}^{N_i} \delta(t - \tau - t_k). \quad (5)$$

Note that while τ is negative, s_i is translated backward, and while τ is positive, s_i is translated forward, as shown in Fig. 3. $I(x, y)$ is the indicator function, equal to 1 if $x = y$ and 0 elsewhere.

After the construction of NSG and SSG graphs, we adopt the graph matching technique, which is a powerful tool to solve mapping problems and has been widely used in computer vision and pattern recognition, to determine an optimal mapping between the two weighted graphs under the mapping rule. For these two graphs, we can compute their adjacency matrices, written as A_n and A_s . The graph matching method is aimed to find out a permutation matrix P that minimizes the following objective function:

$$\min_P ||A_n - P A_s P^T||_F^2 \quad (6)$$

TABLE I
GRAPH MATCHING-BASED INPUT MAPPING ALGORITHM

Step	
1	Input spike trains and input mapping neurons
2	Construct graphs NSG using Euclidean distance
3	Construct graphs SSG using equation (3) or (4)
4	Compute vertex similarity using equation (7)
5	Compute edge similarity using equation (8)
6	Solve problem (6) by Factor Graph Matching

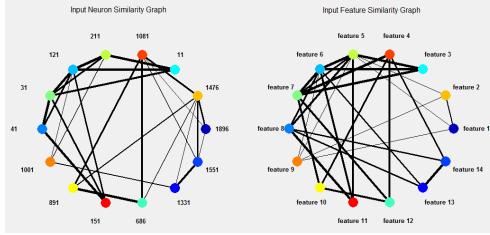


Fig. 4. Input mapping result obtained by the proposed method for exemplar temporal data.

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. Solving this problem exactly is known to be an NP-hard problem due to its combinatorial optimization property. Many algorithms have been proposed to find an approximated solution.

Among these algorithms, the factor graph matching (FGM) algorithm [22] has been demonstrated to produce the state-of-the-art results. Therefore, here we utilize the FGM algorithm to solve the SSG to the NSG matching problem in (6). Suppose in the NSG, the sum of graph edge weights of an vertex, say vertex $i_{NSG} \in V_{NSG}$, to all other vertices is $d(i_{NSG})$, and, similarly, in the SSG, the sum of graph edge weights of vertex $i_{SSG} \in V_{SSG}$ to all other vertices is $c(i_{SSG})$, then the difference between the normalized weight sum $\tilde{d}(i_{NSG})$ and $\tilde{c}(i_{SSG})$ reflects the similarity of the positions of i_{NSG} and i_{SSG} in the corresponding graphs, where $\tilde{d}(i_{NSG})$ [and similarly $\tilde{c}(i_{SSG})$] is the normalized weight sum of vertex i_{NSG} that is obtained by dividing $d(i_{NSG})$ by the largest weight sum in graph NSG. Therefore, we define the vertex similarity as

$$\exp(-|\tilde{d}(i_{NSG}) - \tilde{c}(i_{SSG})|^2/2\sigma_n^2); \quad i_{NSG}, i_{SSG} = 1 \dots v \quad (7)$$

and the edge similarity

$$\exp(-|a_{ij}^{NSG} - a_{kl}^{SSG}|^2/2\sigma_e^2); \quad i, j, k, l = 1 \dots v \quad (8)$$

where a_{ij}^{NSG} and a_{kl}^{SSG} are the graph edge weights in the NSG and the SSG, respectively. σ_n and σ_e are the parameters to control the affinity between neurons and edges, respectively. Table I gives the algorithm procedure.

Fig. 4 shows the matching results for an exemplar temporal data represented by 14 features (this is actually the case study data from experiment 1, presented in [23, Sec. 4]). The left graph is the input NSG, and the right graph is SSG. We can see that after matching, highly correlated features are mapped to nearby input neurons.

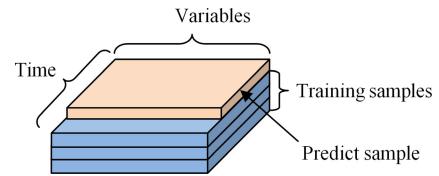


Fig. 5. Temporal data model used for early event prediction.

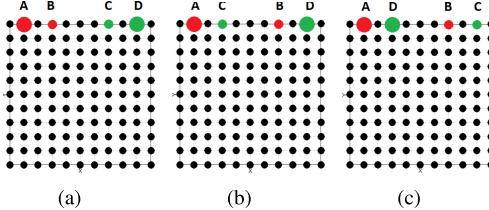


Fig. 6. Different spatial patterns of input mapping. Colored circles: input neurons. Black dots: normal neurons. The color indicates correlation and the shape size indicates dominance. (a) optimal mapping given by the proposed algorithm; (b) non-optimal mapping; (c) non-optimal mapping.

B. Optimal Input Mapping Enables Early and Accurate Event Prediction From Temporal Data

In many applications, such as pest population outbreak prevention, natural disaster warning, and financial crisis forecasting, it is important to know the risk of event occurrence as early as possible in order to take some actions to prevent it or make adjustment in time, rather than waiting for the whole pattern of temporal data to be entered into the model. The main challenge in the early event prediction task is that the time length of the recall samples should be smaller than that of training samples, because training samples are collected in the past and all the data are already known. This is shown in Fig. 5. Traditional data modeling methods, such as SVM, k -nearest neighbor (k NN), and MLP, are no longer applicable for early event prediction task, because they require prediction sample and training samples to have the same time length. Furthermore, the traditional methods cannot also model both the close interaction and interrelationship between time and space components of the spatiotemporal data.

In contrast, NeuCube with the proposed new mapping method would enable a better early event prediction, as the connectivity of a trained SNNcube would reflect on the temporal relationship in the temporal data. As a result, if part of a new sample is presented, this would fire a chain of activities in the SNNcube based on the established connections. To be more specific, let us consider mapping four features A–D into a 2-D SNNcube.¹ Assuming that feature A is more correlated with feature B than with others, but feature A is a more dominant (important) feature than feature B, e.g., feature A could be sun radiation and feature B could be temperature as in the aphids study in [23]. Similarly, assuming feature C is more correlated with feature D, but feature D is a more dominant feature than feature C. The result of optimal graph mapping is shown in Fig. 6(a). The highly correlated features are mapped nearby, while less correlated features are mapped far away. Recall that the STDP learning rule adjusts synaptic

¹Recall that the size of a 3-D SNNcube is $N = n_x \times n_y \times n_z$. 2-D SNNcube is a special case, where $n_z = 1$.

weight between nearby neurons according to their firing time difference. During the training process, the neurons around feature A will have more interaction with neurons around feature B than with other features, and these interactions cause unsymmetrical weight adjustment according to the STDP rule. After training, the connection patterns between neurons around these two features will encode the temporal characteristics in the training spike trains and, hence, the temporal relationship in original signals.

During testing phase, once similar temporal pattern appears, early firing is triggered and then propagated in a neuron population (hence, a firing chain of a neuron cluster) before full data are presented, i.e., the neurons and their connections form a functional cluster, which becomes selective to the spatiotemporal pattern and tends to fire at the start of the signal pattern. As demonstrated in [24] and [25], LIF neurons equipped with STDP learning rule can unsupervisedly learn an arbitrary spatiotemporal pattern embedded in complex background spike trains, and when its preferred spike sequence appears, the neuron can emit a spike very early at the start of the pattern. Biologically, the population firing in the SNNcube chain-fire phenomenon was observed in zebra finches high vocal center area to control the precise temporal structure in birdsong [26], in which the neural activity is propagated in chain network to form the basic clock of the song rhythm. In the SNNcube, we have also observed a similar chain-fire phenomenon, while spike trains are presented to the network. These features endow the SNNcube with a powerful potential to encode complex spatiotemporal patterns contained in the input spike trains used for training and to respond early to the presence of a specific spatiotemporal pattern in a recall/prediction mode. This is especially important when a result is the consequence of several highly correlated key factors, as further demonstrated by the study of stroke occurrence [27].² This phenomenon is similar to that of associative memories in Hopfield networks [28], but here, we deal with temporal patterns rather than with static input vectors.

In contrast, Fig. 6(b) and (c) shows the nonoptimal mapping, where uncorrelated features are mapped together. The neurons around uncorrelated features can hardly learn the original signal temporal relationship, because nearby neurons are presented with less even noncorrelated signals. The interactions between these nearby neurons can be much lower and do not capture any meaningful temporal correlation. As a result, while new samples are partially presented during testing, similar temporal pattern as contained in training samples cannot be detected correctly, as will be demonstrated by experimental results in Section IV.

C. Optimal Input Mapping Enables Better Network Structure Analysis and Visualization and a Better Data Understanding

After it is trained, the SNNcube has captured spatial and temporal relationships from the temporal data. It is helpful to know how the neurons in the SNNcube are related to the input features and what patterns the SNNcube have learned from the

input signals. This information is important to understand the model and the temporal data set [12]. While in the previous work the neuronal clusters are manually labeled according to the synaptic weights and this costs plenty of time and is less accurate, we propose the following algorithm to unveil the SNNcube structure through automatically analyzing neuronal clusters in the SNNcube.

The neurons in the cube are indexed from 1 to N according to ascendant order of their x , y , and z coordinates. We mark the input neurons as the information source in the SNNcube and define a source matrix $F_{\text{src}} \in R^{N \times v}$ as follows: if neuron i is the input neuron of variable j , then the entry (i, j) of F_{src} is 1, otherwise is 0. The affinity matrix $A \in R^{N \times N}$ of the SNNcube is defined in the following way. The entry (i, j) of A is the total spike amount transmitted between neuron i and neuron j . Note that more spike means stronger interaction and more information shared between the neurons. Then, the ratio of information for each neuron received from the input information sources is computed by the following algorithm.

- 1) Compute $S = D^{-1/2}AD^{-1/2}$.
- 2) Evaluate $\tilde{F} = I_{\text{rate}}S\tilde{F} + (I - I_{\text{rate}})F_{\text{src}}$ to converge.
- 3) Normalize $F = G^{-1}\tilde{F}$.

I is the identity matrix and I_{rate} is a diagonal matrix defining the propagation rates on different directions. D and G are also diagonal matrices with $D_{ii} = \sum_{j=1}^N A_{ij}$ and $G_{ii} = \sum_{j=1}^N \tilde{F}_{ij}$, $i = 1, 2, \dots, N$, respectively. In the first iteration, $\tilde{F} = F_{\text{src}}$. Step 1 computes the normalized adjacency matrix which can fully encode the connection information of SNNcube by a square matrix, according to the network theory [29]. Step 2 propagates information from input neurons to other neurons in an iterative way. The main principle behind the algorithm is that the information (or activation) is propagated in the network, and the propagation process is dominated by the network structure. Imagining that each input neuron is a source and has some kind of information (i.e., electricity, fluid matters, and so on), but the information type possessed by different input neurons is different (i.e., different ions or different liquids). The information is propagated from sources to other neurons in each iteration, and the propagation amount is proportional to the connection weight between each pair of neurons. At the beginning, only the input neurons (the sources) have the information, and the other neurons do not have any information. As the propagation process continues, all neurons receive some information from one or more input neurons, and the information amount corresponding to different input neurons is different. The amount of a particular type of information (from the corresponding input neuron) received by a neuron reflects the intimacy relationship of this neuron with that input source neuron. The more amount of information it received, the closer it is with that input neuron. Finally, according to the network theory when the whole network reaches to a stable state, entry F_{ij} represents the relative information amount neuron i received from input neuron j [30], [31]. Finally, Step 3 normalizes the information amount corresponding to different input neurons, and this facilitates to classify neurons into different input clusters by $k = \arg \max_{i=1..v} F_{ij}$, $j = 1 \dots N$, where v is the number of input neuron and N is the number of

²<http://www.kedri.aut.ac.nz/neucube/stroke>.

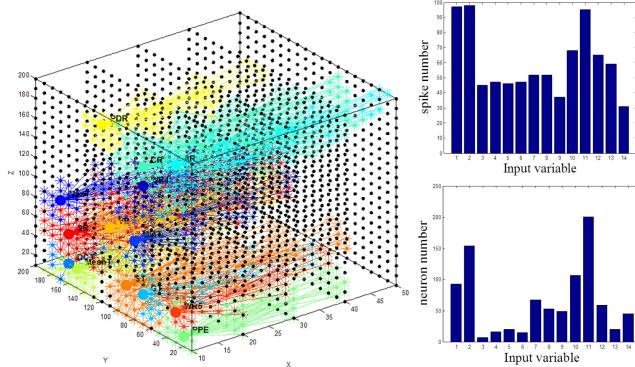


Fig. 7. Left: neuronal clusters in SNNcube after unsupervised training. Right: input spike number of each feature (top) and neuronal connections of each input neuron (bottom).

total neurons in SNNcube. Matrix F is the basis of extracting neuronal clusters.

The propagation factor matrix I_{rate} controls the convergence of the propagation algorithm and the amount of the information being propagated to other neurons in the SNNcube. Here, I_{rate} is computed by $(I_{\text{rate}})_{ii} = \exp(-\bar{d}_i^2/2\sigma^2)$, where \bar{d}_i is the mean affinity value between a neuron and its eight neighboring neurons, so that the information propagation between strongly connected neurons is large, while the information propagated through weakly connected neurons is small.

We now show the propagation algorithm that converges. Because I_{rate} is a diagonal matrix and $(I_{\text{rate}})_{ii} = \exp(-\bar{d}_i^2/2\sigma^2) < 1$, the spectral radius of I_{rate} is $\rho(I_{\text{rate}}) < 1$. After the i th iteration, we have

$$\tilde{F}^{(i)} = A_{sc}F_{\text{src}} + A_{nb}(I - I_{\text{rate}})F_{\text{src}} \quad (9)$$

where $A_{sc} = (I_{\text{rate}}S)^{i-1}$ and $A_{nb} = \sum_{k=0}^i (I_{\text{rate}}S)^k$, representing the information amount acquired from sources and neighborhood, respectively. Let $P = D^{-1}A$ be the random walk matrix on the graph. Then, for the spectral radius of P , we have $\rho(P) \leq \|P\|_\infty \leq 1$. Because $S = D^{1/2}PD^{-1/2}$, S is similar to P , so $\rho(S) = \rho(P) \leq 1$. Since the spectral radius $\rho(I_{\text{rate}}) < 1$, and $\rho(AB) \leq \rho(A)\rho(B)$, so $\rho(I_{\text{rate}}S) < 1$. Therefore, $\lim_{i \rightarrow \infty} A_{sc} = 0$, $\lim_{i \rightarrow \infty} A_{nb} = (I - I_{\text{rate}}S)^{-1}$, and

$$F^* = \lim_{i \rightarrow \infty} \tilde{F}^{(i)} = (I - I_{\text{rate}}S)^{-1}(I - I_{\text{rate}})F_{\text{src}}. \quad (10)$$

It is worth mentioning that Step 2 can also be replaced by (10). But since the matrix A is highly sparse, it will be much more efficient to evaluate the equation in Step 2 than to invert a matrix in (10), regarding to both space and time consumption.³ Another advantage of using Step 2 is that during the iterative process, it is more interesting and intuitive to observe how the information is propagated in the SNNcube, rather than jumping directly to the resultant clusters view given by (10).

Fig. 7 (left) shows the network structure after unsupervised training for the study of aphids data presented in our previous work [23]. The big solid dots represent the input neurons, and other neurons are labeled in the same intensity

³Note that $I_{\text{rate}}S$ is a highly sparse matrix which keeps constant during iterations, and Step 2 usually converges after several iterations.

as the input neuron from which it receives the most spikes. The unconnected dots mean no spike arrived at that neuron. In Fig. 7 (right), the top pane is the spike number for each variable after encoding, and the bottom pane is the neuron number belonging to each input variable cluster. From Fig. 7, we can see the consistency between the input signal spike train and the SNNcube structure. Note that variable 11 (solar radiation) is emphasized in the SNNcube that suggests a greater impact of the solar radiation on the aphid number. This was observed also in another work [32]. This is very different from the traditional methods, such as SVM, which have been used for the same tasks but without offering facilities to trace the learning processes for the sake of data understanding. It is worth mentioning that the spatial pattern of the input mapping, e.g., in Fig. 4, is embedded in the source matrix F_{src} and has a direct influence on the visualization results and the interpretation of the results. These mapping and visualization are demonstrated to be useful for high level cause–results interpretation in some studies, such as EEG signal study, for identifying the differences between people with opiate addiction and those undertaking substitution treatment for opiate addiction [12] and the fMRI data study for brain activity, while the subjects are presented with different pictures.⁴ It is important to note that while in [12], the input mapping and the neuronal clustering were performed by hand, the proposed graph mapping and structure analysis algorithms enable all these to be finished automatically.

IV. EXPERIMENTAL RESULTS ON THREE CASE STUDY PROBLEMS

In this section, we present three case studies to demonstrate the validity of the proposed architecture for both early even prediction based on temporal data and spatiotemporal pattern recognition. The first case study is conducted on a benchmark physiological data set SantaFe⁵ to demonstrate the ability of the proposed method to classify temporal data and to make early event prediction. We demonstrate the validity of the mapping method for early event prediction. The second case study is conducted on a spatiotemporal data set PEMS, which can be downloaded from California Department of Transportation Caltrans Performance Measurement System (PEMS) Web site,⁶ to perform pattern recognition, in which each feature is sampled from a fixed location. We demonstrate that the proposed mapping in Section III improves the accuracy of spatiotemporal pattern recognition. The third case study is carried on a contest physiological data set Challenge 2012⁷ to demonstrate the superiority of the proposed graph mapping over randomly mapping.

While in [1], [6], and [33], AER encoding with fixed threshold is used and the threshold has to be tuned manually each time for each feature individually, here we introduce a self-adaptive bidirectional thresholding method, ATB encoding algorithm. ATB is simply self-adaptive to all features: for an input time series/signal $f(t)$, we calculate the mean m and the

⁴<http://www.kedri.aut.ac.nz/neucube/fmri>.

⁵<http://www.physionet.org/physiobank/database/santa-fe/>.

⁶<http://pems.dot.ca.gov>.

⁷<http://physionet.org/challenge/2012/#rules-and-dates>.

TABLE II
PARAMETERS TO BE INCLUDED IN THE OPTIMIZATION PROCESS

Parameter	Meaning	Value range
Spike Threshold	Threshold of ATB encoding	[0.1, 0.9]
Firing Threshold	LIF neuron firing threshold	[0.01, 0.8]
STDP Rate	The learning rate in STDP algorithm	[0.001, 0.5]
Refractory Time	Refractory time after firing	[2, 9]
Mod	Modulation factor in [15]	[0.00001, 0.5]
Drift	Synaptic drift parameter in [15]	[0.1, 0.95]
K	Number of nearest neighbors in deSNN	[1, 10]

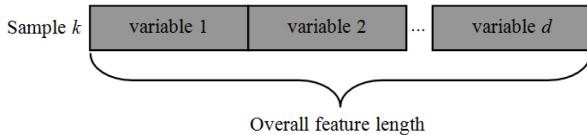


Fig. 8. Convert spatiotemporal data into static vector for baseline algorithms.

standard deviation s of the gradient df/dt , then the threshold is set to $m + \alpha s$, where α is a parameter controlling the spiking rate after encoding. After this, we obtain a positive spike train, which encodes the segments in the time series with increasing signal and a negative spike train, which encodes the segments of decreasing signal.

Because the NeuCube system is a complex system and contains many tunable parameters, manually tuning these parameters might be time consumption for a nonexperienced user. We have implemented a GA optimization component to optimize all the system parameters at once in a k -fold cross validation way with respect to a data set. The objective function of the GA is the overall validation error rate. We use the optimization component to simultaneously optimize seven key parameters of the NeuCube system in Table II. After optimization, GA outputs the optimal parameters' value that attained the smallest cross-validation error rate.

Baseline algorithms consist of the following.

- 1) Multiply linear regression (MLR) is simple, extensively studied, and can achieve relative stable result in many applications.
- 2) SVM is probably the most widely used and has been demonstrated to be successful in various applications.
- 3) MLP is a classic neural network model and has advantages in processing multivariate data.
- 4) k NNs and weighted k NNs (wkNNs) [34] are among the most popular classification algorithms [35] and especially while processing high-dimension data.
- 5) Global alignment kernel (GAK) [36] is a recently developed method to process time series and can achieve the state-of-the-art results in many applications. So, there are six baseline algorithms⁸: MLR, SVM, MLP, k NN, wkNN, and GAK.

Because the baseline algorithms process static vectors, we prepared their data set in the following way. We concatenated each temporal variables one after another to form a long feature vector for each sample, as shown in Fig. 8. Since the

⁸The first three methods are implemented in NeuCom system: <http://www.kedri.aut.ac.nz/areas-of-expertise/data-mining-and-decision-support-systems/neucom>. k NN and wkNN are implemented based on the MATLAB kNN search function. GAK is available at: <http://www.iip.ist.i.kyoto-u.ac.jp/member/cuturi/GA.html>.

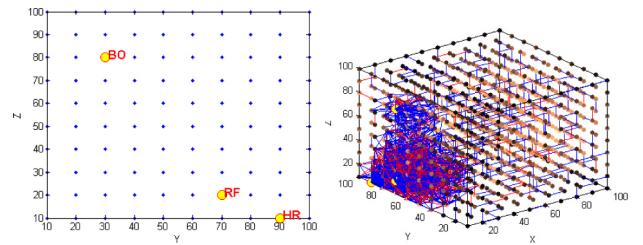


Fig. 9. Left: input mapping for SantaFe data set (yz plane of the SNNcube). Right: synaptic connections after training. Blue connections have positive weight. Red connections have negative weight.

baseline algorithms have very few tunable parameters (one or two parameters), the parameters are tuned in a grid search method. This process is simple and quick, usually just a few steps to locate the optimal value of the parameters.

A. Predict Events by Classifying Time Series Data Based on Physiologic Signals

In this case study, we conduct experiments for complex physiologic time series classification on SantaFe data set. The data set contains three physiologic features: heart rate (HR), respiration force (RF, chest volume), and blood oxygen (BO). Each of the feature was measured twice in 1 s and the measurement lasted 5 h. The aim of this case study is to predict the sleep stage of a patient by using the collected physiologic signals. There are three sleeping stages: awake, sleep, and waking, and the sleep stages are read from the EEG signals by neurologist.

The original data set contains 34 000 data points for each feature (approximate 4.7 h long) and 103 sleep stage labels, and the sleep stage labeling interval varies from 60 to 4600 s. Each label indicates a different sleep stage happening at that moment, i.e., whenever the patient transits from one sleep stage to another different sleep stage, a corresponding label is assigned at that time. So, the signals recorded during the labeling interval reflect the happening process of the next sleep stage. Consequently, by classifying these temporal data into different classes, we actually predict the next sleep stage of the patient. In this case study, we only use the labeling data with 60-s time interval to make sure that the feature length is the same for every sample for fair comparison, because the traditional methods (such as SVM) cannot process data which have uneven feature length between samples. In this case, we use a $10 \times 10 \times 10$ SNNcube⁹ and deSNN network with a wkNN classifier as the output layer [15]. We build a $60 \times 3 \times 42$ data matrix, where 60 is the time length of each signal, 3 is the feature number, and 42 is the sample number.

Fig. 9 (left) gives the mapping result found by the proposed input mapping algorithm. It can be seen that the RF is mapped between BO and HR, but it is much closer to HR.

⁹Currently, we determine the reservoir size by trying several different ones. But as a general rule, the larger the reservoir is, the more powerful computational ability the reservoir has, and thus the more complex patterns the model can learn and recognize. However, a larger reservoir consumes longer time and more memory space and needs more training data for training.

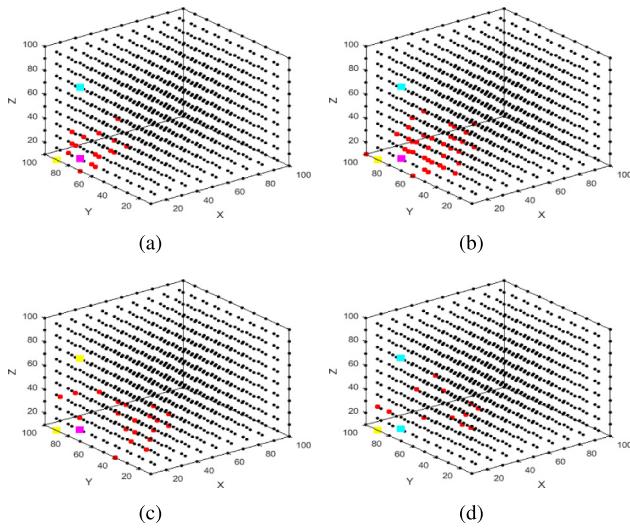


Fig. 10. Firing state spreading in the SNNcube during training stage. (a)–(d) Four consecutive frames.

This indicates that the respiration signal more correlates with HR than BO. This result is consistent with the actual observation described in the three major research questions, which are raised on the data set Web site.¹⁰ This is also further demonstrated in Fig. 9 (right), in which we can see clearly that the connections between RF and HR are much denser than connections in other places. This indicates that the interaction between RF and HR is more active and much stronger than other parts. Therefore, the temporal relationship in the original data is fully captured by the connections in the SNNcube after training.

Fig. 10 shows the four consecutive snapshots of the instantaneous firing state of the neurons during the training process. Squares are input neurons (purple means that a positive spike is an input at this moment. Cyan means negative spike, and yellow means no spike). Red dots are the firing neurons, and black dots are the unfiring neurons. Note that the firing neuronal cluster grows from small to large and the propagation of the spike states (hence, the information the spike carried) has a trajectory in the SNNcube. With the meaning of the firing pattern and spreading trajectory, how they reveal the significance of the SNNcube modeling ability are the questions to be answered; the study of these is out of the scope of this paper and we leave them in our future papers.

Fig. 11 shows the evolving process of the connections during different times in the training stage. From this, we can see how the connections between different neurons are created dynamically as the spike trains input to the reservoir. At the beginning, the connections in the reservoir are sparse and nearly randomly distributed [Fig. 11(a)]. As the training process continues, some new connections appear around each input neuron, but BO input neuron does not have interaction with the other two input neurons [Fig. 11(b)]. At the last half training process, more connections are created, and the

¹⁰How do the evolution of the HR, the respiration rate, and the BO concentration affects each other? (A correlation between breathing and the HR, called respiratory sinus arrhythmia, is almost always observed.)

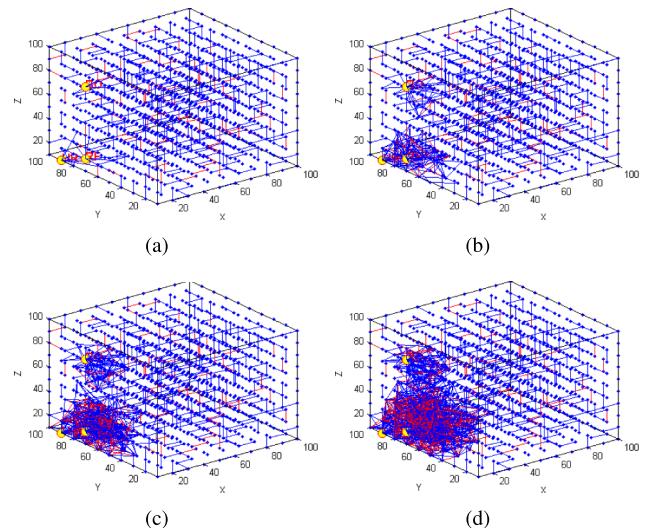


Fig. 11. Connection evolving process during training stage. (a)–(d) Four status from the beginning to the end of training.

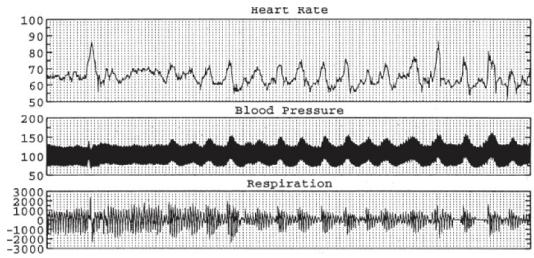


Fig. 12. Physiological signals in SantaFe data set (courtesy from [39]).

interaction between each pair of input neurons increases, especially for BO [Fig. 11(c) and (d)]. The synaptic weight evolving process is tightly connected with the physiological signal trend in Fig. 12, from which we can see clearly the correlation between the BO and the other two signals in the last half signal segment. It is worth mentioning that while previous method to study the SantaFe data set mainly focused on the analysis of the nonlinear dynamics of the respiration, HR, and blood pressure [37], [38], they can hardly provide such an intuitive and direct way to observe the dynamics of the signals and, hence, the interaction between the physiological signals during a sleep stage.

We designed two experiments on this data set to show the predictive ability of NeuCube with different mappings and how early the model can predict sleep stage using part of the data. In these experiments, we trained NeuCubes of both random mapping and graph mapping using 100% of the time length samples (60 s), but temporal data of only 75% and 50% of the time length of the samples was used to predict the sleep stage.

Experimental results of early event prediction are shown in Table III. In order to compare the impact of testing time length upon accuracy, we also include the result of full time length in the second column. NeuCube(R) is random mapping and NeuCube(G) is graph mapping.

From these results, we can see that the NeuCube(G) can perform better for early event prediction. In contrast, the

TABLE III
EARLY PREDICTION ACCURACY OF SantaFe DATA SET (%)

	Accuracy of each training and testing time length (sec)		
	60 (full)	45 (early)	30
MLR	42.50	32.29	32.29
SVM	43.75	22.29	28.96
MLP	38.96	39.17	26.25
kNN	35.85	38.75	32.08
wkNN	48.45	48.13	25.83
GAK	45.21	40.07	38.43
NeuCube(R)	51.62	43.5	35.12
NeuCube(G)	67.74	63.06	51.61

performance of the NeuCube(R) for early prediction is much lower, because SNNcube with randomly mapping fails to acquire the temporal pattern in the data for early firing during testing stage. Furthermore, a realistic early event prediction should be that as the time length of the observed data increases, the prediction accuracy will also increase. But from Table III, we can see that as the time length of the training data increases, traditional modeling methods do not necessarily produce better results (some even worsen), because they cannot model the whole spatiotemporal relationship in the prediction task. They can only model a certain time segment. Temporal data typically exhibits complex interrelationship and interaction among different features, and traditional methods are proposed to process static vector data, and thus, they can hardly model the complex relationship contained in temporal data. The GAK and wkNN can produce better results, but their accuracies are still much lower than that of NeuCube. Because the NeuCube is trained on the whole spatiotemporal relationship in the data, even a small amount of input data can trigger spiking activities in the SNNcube that will correspond to the learned full temporal pattern resulting in a better prediction.

B. Spatiotemporal Pattern Recognition Based on Spatiotemporal Data

In this case study, we consider a benchmark traffic status classification problem of spatiotemporal data from the PEMS database. In freeways, vehicle flow is monitored by traffic sensors with fixed spatial locations, and the data collected by these sensors exhibit spatial and temporal characteristics. Discovering spatial-temporal patterns can be very meaningful for traffic management and a city traffic plan.

The study area is San Francisco bay area, which is shown in Fig. 13. There are thousands of sensors distributed over the road network, and the sensor distribution is indicated in Fig. 13 (right), in which each black dot represents a monitoring sensor. These sensors monitor lane occupation rate 24 h every day. Measurements are taken every 10 min and normalized between [0 1], where 0 means no car occupation and 1 means full occupation of the lane in the monitoring region. So, there are 144 (24 × 6) data points per day. In this case study, we collect traffic data over a period

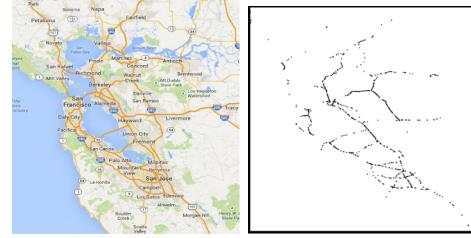


Fig. 13. Left: map of the study area (from the Google map). Right: reconstructed topology of the traffic sensor network. Each black dot inside represents a sensor.

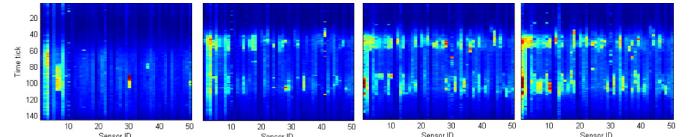


Fig. 14. Spatial and temporal patterns of the sample traffic data from Monday to Thursday.

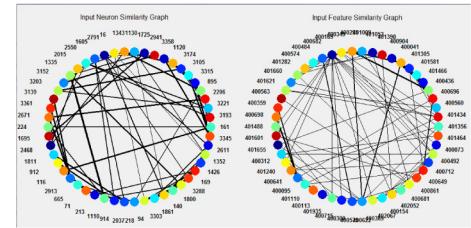


Fig. 15. Final input mapping result used in this case study. Left: input NSG (the number beside each vertex is the input neuron ID). Right: input feature similarity graph (the number beside each vertex is the traffic sensor ID).

of 15 months, and thus, after removing public holidays and sensor maintenance days, there are 440 days to be classified. We did some preprocessing of the data: 1) we removed the data of outlier sensors from the data set, e.g., sensors producing always 1 or 0 in 24 h, and sensors flip from 0 to 1 or 1 to 0 suddenly; 2) nearby sensors that produce almost the same data sequence are combined into one sensor; 3) the total occupation rate of each sensor is calculated as a sum of all measurements over 440 days; and 4) 50 sensors corresponding to the largest occupation rate are selected as the final features (variables) to represent the data set. Fig. 14 shows the samples of the spatial and temporal distribution of the traffic status of the road network from Monday to Thursday.

In this case study, we construct a $15 \times 15 \times 15$ reservoir. Fig. 15 shows the final input mapping result found by the graph matching algorithm. We can see the similar connection pattern between the two graphs.

Fig. 16 shows the overall neuron firing state matrix of the SNNcube corresponding to the four data samples, from top to bottom Monday until Thursday. In each figure, the horizontal axis is neuron ID, and the vertical axis is time tick, from 0 at the top to 144 at the bottom. One should note that while in the plot, it seems the firing state matrix is very dense, it is actually very sparse. Take the Thursday (the fourth one) as an example. There are 20416 firing entries and the firing state matrix size

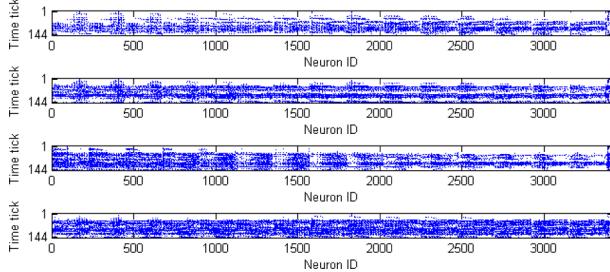


Fig. 16. Neuron firing state of the reservoir. Bars from top to bottom correspond to Monday to Sunday.

TABLE IV
COMPARATIVE ACCURACY OF SPATIOTEMPORAL
PATTERN CLASSIFICATION (%)

	MLR	SVM	MLP	k NN	wkNN	GAK	NeuCube
Param.	-	$d:2$	$n:100$	$k:10$	$k:10$	$\sigma:5$	-
Acc.	56.82	43.86	68.18	66.82	71.36	72.27	75.23

is 486 000 (144×3375 , where 3375 is the total neuron number in the SNNcube), so the firing entries is $\sim 4.20\%$. We can see that these sparse firing matrices have different patterns related to the input data. Meanwhile, since the size of the SNNcube can be specified according to the problem, the SNNcube with highly sparse firing rate has a great power to encode input signals and patterns, and thus, it can potentially model any complex spatial and temporal relationship.

After obtaining the firing state matrix of each sample, we transmit them to the output deSNN layer [15], which creates an output neuron for each sample and connects the output neuron to each neuron in the SNNcube. Then, the weights of the connections are established by rank order learning rule [16]. For a testing sample, its class label is determined by comparing its output neuron weights with those training samples' output neuron weights, which are established at training stage, using the wkNN rule [34]. In Table IV, we compared the twofold cross validation experimental result between NeuCube and baseline algorithms: MLR, SVM, MLP, k NN, wkNN, and GAKs [36].

The parameters' values used in the classical machine learning methods are: d —degree of polynomial kernel; n —number of neurons in the hidden layer of MLP; k —number of nearest neighbors in k NN and wkNN; and σ —Gaussian kernel width. From these results, we can see that the proposed NeuCube model achieves better classification results. This is because the traditional machine learning methods are designed to process static vector data, and they have limited ability to model spatially correlated and temporally varied data. Meanwhile, MLR, SVM, and MLP also show disadvantages while modeling high-dimensional data (e.g., there are 7200 features in each sample of this case study). k NN and wkNN have been widely used in high-dimensional data processing, such as document classification, because they can approximately reconstruct the underlying manifold whose dimension is usually much lower than its ambient space, and thus, they can produce better

TABLE V
PHYSIOLOGICAL SIGNALS IN CHALLENGE 2012 DATA

Feature	Meaning	Unit
DiasABP	Invasive diastolic arterial blood pressure	mmHg
GCS	Glasgow Coma Score (3-15)	-
HR	Heart rate	bpm
MAP	Invasive mean arterial blood pressure	mmHg
NIDiasABP	Non-invasive diastolic arterial blood pressure	mmHg
NIMAP	Non-invasive mean arterial blood pressure	mmHg
NISysABP	Non-invasive systolic arterial blood pressure	mmHg
RespRate	Respiration rate	bpm
SysABP	Invasive systolic arterial blood pressure	mmHg
Temp	Temperature	°C
Urine	Urine output	mL
Weight	-	kg

results than MLR and SVM. While the recently proposed GAK algorithm is shown to be very efficient and effective in processing time series, its performance is still lower than that of NeuCube. It should be mentioned that although in this case study, classifying weekday traffic patterns does not have an obvious application situation now, this PEMS database has been established as a benchmark data set to test an algorithm's ability for spatiotemporal data processing [36], [40].

C. Comparison of Random Input Mapping and Graph Input Mapping Method for Physionet Data Classification

To fully evaluate the effect of input mapping, comparison experiments are conducted on both Challenge 2012 and SantaFe data sets. Different from SantaFe data set described in Section IV-A, Challenge 2012 is a more complex and challenging data set. The goal of Challenge 2012 data is to predict mortality of 4000 intensive care unit (ICU) patients using their 37 physiological signals (see Table V), which are measured at irregular time interval within the total time length being 48 h. The irregularity of the measurement time interval and the complexity of the physiological signal make this prediction task highly challenging.

We first apply the following preprocessing procedure to the Challenge 2012.

- 1) *Patient Selection:* The original data contain records of 12 000 ICU patients, and the records are divided into two parts, training set A with 4000 patients' records and testing set B with 8000 patients' records. Because testing set B does not contain mortality state, and thus, we cannot calculate exactly its classification accuracy, we just conduct experiments on training set A. Some of the patients only have several measurements, for example, due to an early in-hospital death. So, we select the patients who have more than 40 records from set A, and this results into a subset contains 3470 patients.
- 2) *Feature Selection:* The original data contain some features which have too few measurements during the observation period to enable effective temporal encoding, such as the TropT and TropI features are only measured several times during 48 h. We first calculate the overall measurement amount of every feature and then select those features with average measurements more than 5. As a result, the 12 features are used in the experiments, as shown in Table V.

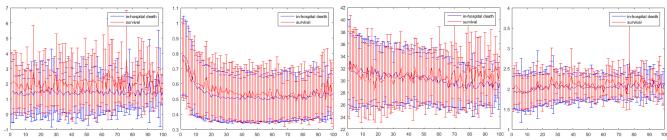


Fig. 17. Mean and standard deviation of the first four selected features for in-hospital death and survival patients. From left to right: DiasABP, GCS, HR, and MAP. Note how similar the signals of the two classes are.

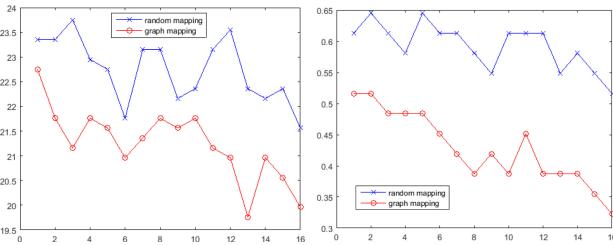


Fig. 18. Error rate of random mapping and graph mapping on Challenge 2012 data set (left) and SantaFe data set (right).

Fig. 17 shows the mean and the standard deviation of the first four selected features for the two classes. From Fig. 17, we can see that the physiological signals corresponding to the two classes have very similar trends, and the difference is almost indistinguishable to human eyes [41].

The comparison experiments are conducted in the following way. The GA optimization component is running with two modes: one is randomly mapping and the other mode is graph-matching mapping. The optimized parameters are shown in Table II. GA parameters are: generation number: 16, population size: 50, crossover function: scattered, crossover fraction: 0.2, selection function: roulette, and elite count 5. The results Challenge 2012 and SantaFe are shown in Fig. 18, in which the horizontal axis represents the GA generation and the vertical axis represents the error rate (fitness value of GA). From these results, we can see that the graph matching can achieve lower error rate than randomly mapping. As the generation number increases, we can see more clearly decreasing trend for graph-based mapping, while the trend of random mapping curve is not so obvious and has greater fluctuation. It has to be mentioned that while the fitness value of GA algorithm decreases monotonically for optimizing deterministic system, it usually has fluctuation and does not always decrease for stochastic system. Because in a stochastic system even the system parameters are the same, different initial states may yield different results. This is the reason why the error rate (fitness value of GA) in our NeuCube system dose not monotonically decrease.

To further compare the performance, Fig. 19 shows the mean error rate and the standard deviation of error rate of the 10 times running on Challenge 2012. From these results, we can see that the performance of the graph-based mapping method is more effective and stable, regarding to the lower mean error rate and the smaller standard deviation.

Since Challenge 2012 is a contest data set that highly challenges traditional classification algorithms, it might be interesting to compare the performance of NeuCube with

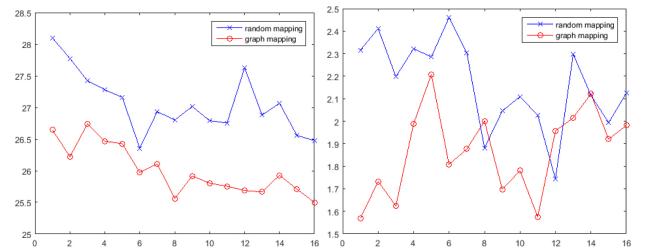


Fig. 19. Mean (left) and standard deviation (right) of error rate of random mapping and graph-based mapping in GA parameter optimization for Challenge 2012 data set.

TABLE VI
COMPARATIVE ACCURACY OF SPATIOTEMPORAL
PATTERN CLASSIFICATION (%)

	MLR	SVM	MLP	kNN	wkNN	GAK	NeuCube
Param.	-	$d:2$	$n:40$	$k:6$	$k:6$	$\sigma:5$	-
Acc.	47.31	46.81	51.3	49.3	49.3	51.6	76.95

traditional methods. The results are shown in Table VI. Note the highest accuracy of baseline algorithms is 51.6% from GAK, but the NeuCube can achieve 76.95%, a significant improvement to the state-of-the-art results. The optimal values of the parameters given by GA that attain this result are: spike threshold—0.800, firing threshold—0.500, STDP rate—0.073, refractory time—6, mod—0.687, and drift—0.095.

V. CONCLUSION

In this paper, we proposed a new mapping method to map spatiotemporal input variables into a 3-D SNN architecture called NeuCube that enable NeuCube models any spatiotemporal data. The weighted undirect graph matching technique is adopted here, so that similar input variables based on their temporal similarity are mapped into spatially closer neurons. The closer the neurons in the SNNcube are, the more temporal relationships they learn from data. This automatic optimal mapping algorithm greatly reduces the work load for a user to find out the optimal mapping by hand in the previous system and can yield better results regarding to spatiotemporal pattern recognition and early event prediction. A comparison study of the proposed mapping and randomly mapping has been conducted on a popular contest physiological data set Challenge 2012, and the results demonstrated the superiority of the proposed method over randomly mapping method.

To have a better understanding of the SNNcube model and the data modeled in it, we also proposed an algorithm based on network activation spreading to automatically reveal neuronal clusters, which in the previous system were also determined by hand according to the learned connection weights [12] and thus costed lots of time. The algorithm can divide the SNNcube into different neuronal clusters based on either the spike amount communication or the connection weight between neurons. These neuronal clusters can help us to better understand the learning mechanism and the modeling ability of the SNNcube, as well as the data.

It should be mentioned that due to the scalability of the SNNcube, NeuCube is a deep architecture, and it has the potential to model any complex spatiotemporal data. Our future work includes the following.

- 1) Improvement of the mapping method for complex temporal and spatial data, including spatiotemporal data with moving spatial coordinates.
- 2) Extension to including semisupervised learning ability into the NeuCube model, because semisupervised learning has been demonstrated in various applications to be useful [42]–[44] and is one of the basic ability of human neural system [45].
- 3) Statistical analysis of the firing pattern and spreading trajectory in SNNcube with mining algorithms [46] and how they reveal the significance of the SNNcube modeling ability.
- 4) More experiments on other ecological-, environmental-, financial-, and business-temporal data with large-scale SNNcube to further explore the deep modeling and deep prediction capability of the proposed NeuCube on complex spatiotemporal data.

ACKNOWLEDGMENT

This work was initiated in the Knowledge Engineering and Discovery Research Institute (KEDRI) during the visit of E. Tu in KEDRI and continued in the Shanghai JiaoTong University in a collaborative way.

REFERENCES

- [1] N. K. Kasabov, “NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data,” *Neural Netw.*, vol. 52, pp. 62–76, Apr. 2014.
- [2] T. Voegtlin, “Recursive self-organizing maps,” *Neural Netw.*, vol. 15, nos. 8–9, pp. 979–991, 2002.
- [3] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert, “Recursive self-organizing network models,” *Neural Netw.*, vol. 17, nos. 8–9, pp. 1061–1085, 2004.
- [4] T. Mikolov, S. Kombrink, L. Burget, J. H. Černocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 5528–5531.
- [5] I. Sutskever, J. Martens, and G. Hinton, “Generating text with recurrent neural networks,” in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 1017–1024.
- [6] N. Kasabov, “NeuCube evospike architecture for spatio-temporal modelling and pattern recognition of brain signals,” in *Artificial Neural Networks in Pattern Recognition*. Berlin, Germany: Springer, 2012, pp. 225–243.
- [7] N. Kasabov and E. Capecci, “Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes,” *Inf. Sci.*, vol. 294, pp. 565–575, Feb. 2015.
- [8] D. Taylor *et al.*, “Feasibility of NeuCube SNN architecture for detecting motor execution and motor intention for use in BCI applications,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 3221–3225.
- [9] M. G. Doborjeh, E. Capecci, and N. Kasabov, “Classification and segmentation of fMRI spatio-temporal brain data with a NeuCube evolving spiking neural network model,” in *Proc. IEEE Symp. Evol. Auto. Learn. Syst. (EALS)*, Dec. 2014, pp. 73–80.
- [10] M. G. Doborjeh and N. Kasabov, “Dynamic 3D clustering of spatio-temporal brain data in the NeuCube spiking neural network architecture on a case study of fMRI data,” in *Neural Information Processing*. Switzerland: Springer International Publishing, 2015, pp. 191–198.
- [11] E. Capecci *et al.*, “Modelling absence epilepsy seizure data in the NeuCube evolving spiking neural network architecture,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.
- [12] E. Capecci, N. Kasabov, and G. Y. Wang, “Analysis of connectivity in NeuCube spiking neural network models trained on EEG data for the understanding of functional changes in the brain: A case study on opiate dependence treatment,” *Neural Netw.*, vol. 68, pp. 62–77, Aug. 2015.
- [13] N. Kasabov, “To spike or not to spike: A probabilistic spiking neuron model,” *Neural Netw.*, vol. 23, no. 1, pp. 16–19, 2010.
- [14] S. Song, K. D. Miller, and L. F. Abbott, “Competitive Hebbian learning through spike-timing-dependent synaptic plasticity,” *Nature Neurosci.*, vol. 3, no. 9, pp. 919–926, 2000.
- [15] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, “Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition,” *Neural Netw.*, vol. 41, pp. 188–201, May 2013.
- [16] S. Thorpe and J. Gautrais, “Rank order coding,” in *Computational Neuroscience*. New York, NY, USA: Springer, 1998, pp. 113–118.
- [17] T. Kohonen, “The self-organizing map,” *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [18] M. C. W. van Rossum, “A novel spike distance,” *Neural Comput.*, vol. 13, no. 4, pp. 751–763, Apr. 2001.
- [19] J. Dauwels, F. Vialatte, T. Weber, and A. Cichocki, “On similarity measures for spike trains,” in *Advances in Neuro-Information Processing*. Berlin, Germany: Springer-Verlag, 2009, pp. 177–185.
- [20] B. Schrauwen and J. Van Campenhout, “BSA, a fast and accurate spike train encoding scheme,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 4. Piscataway, NJ, USA, Jul. 2003, pp. 2825–2830.
- [21] K. Dhoble, N. Nuntalid, G. Indiveri, and N. Kasabov, “On-line spatiotemporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order- and temporal spike learning,” in *Proc. WCCI*, 2012, pp. 554–560.
- [22] F. Zhou and F. De la Torre, “Factorized graph matching,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 127–134.
- [23] E. Tu *et al.*, “NeuCube^(ST) for spatio-temporal data predictive modelling with a case study on ecological data,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2014, pp. 638–645.
- [24] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains,” *PLoS ONE*, vol. 3, no. 1, p. e1377, 2008.
- [25] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Competitive STDP-based spike pattern learning,” *Neural Comput.*, vol. 21, no. 5, pp. 1259–1276, 2009.
- [26] Y. Ikegaya *et al.*, “Synfire chains and cortical songs: Temporal modules of cortical activity,” *Science*, vol. 304, no. 5670, pp. 559–564, 2004.
- [27] M. Othman, “Spatial-temporal data modelling and processing for personalised decision support,” Ph.D. dissertation, School Comput. Math. Sci., Auckland Univ. Technol., Auckland, New Zealand, 2015.
- [28] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.
- [29] F. R. Chung, *Spectral Graph Theory*, vol. 92. Providence, RI, USA: AMS, 1997.
- [30] J. Shrager, T. Hogg, and B. A. Huberman, “Observation of phase transitions in spreading activation networks,” *Science*, vol. 236, no. 4805, pp. 1092–1094, 1987.
- [31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16. 2004, pp. 321–328.
- [32] S. P. Worner, G. O. Lankin, S. Samarasinghe, and D. A. J. Teulon, “Improving prediction of aphid flights by temporal analysis of input data for an artificial neural network,” in *Proc. New Zealand Plant Protection*, 2002, pp. 312–316.
- [33] N. Kasabov *et al.*, “Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke,” *Neurocomputing*, vol. 134, pp. 269–279, Jun. 2014.
- [34] K. Hechenbichler and K. Schliep, “Weighted k -nearest-neighbor techniques and ordinal classification,” Sonderforschungsbereich der Ludwig-Maximilians-Universität München, Munich, Germany, Tech. Rep., 2004.
- [35] X. Wu *et al.*, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
- [36] M. Cuturi, “Fast global alignment kernels,” in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 929–936.
- [37] M. Korürek and A. Nizam, “Clustering MIT-BIH arrhythmias with ant colony optimization using time domain and PCA compressed wavelet coefficients,” *Digit. Signal Process.*, vol. 20, no. 4, pp. 1050–1060, 2010.
- [38] M. Engin, “ECG beat classification using neuro-fuzzy network,” *Pattern Recognit. Lett.*, vol. 25, no. 15, pp. 1715–1722, 2004.

- [39] Y. Ichimaru and G. B. Moody, "Development of the polysomnographic database on CD-ROM," *Psychiatry Clin. Neurosci.*, vol. 53, no. 2, pp. 175–177, 1999.
- [40] A. Skabardonis, P. Varaiya, and K. Petty, "Measuring recurrent and nonrecurrent traffic congestion," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1856, pp. 118–124, Jan. 2003.
- [41] M. T. Bahadori, D. Kale, Y. Fan, and Y. Liu, "Functional subspace clustering with application to time series," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 228–237.
- [42] E. Tu, J. Yang, J. Fang, Z. Jia, and N. K. Kasabov, "An experimental comparison of semi-supervised learning algorithms for multispectral image classification," *Photogram. Eng. Remote Sens.*, vol. 79, no. 4, pp. 347–357, 2013.
- [43] S. Li, Z. Wang, G. Zhou, and S. Y. M. Lee, "Semi-supervised learning for imbalanced sentiment classification," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1826–1831.
- [44] E. Tu, J. Yang, N. Kasabov, and Y. Zhang, "Posterior distribution learning (PDL): A novel supervised learning framework using unlabeled samples to improve classification performance," *Neurocomputing*, vol. 157, pp. 173–186, Jun. 2015.
- [45] B. R. Gibson, T. T. Rogers, and X. Zhu, "Human semi-supervised learning," *Topics Cognit. Sci.*, vol. 5, no. 1, pp. 132–172, 2013.
- [46] E. Tu, L. Cao, J. Yang, and N. Kasabov, "A novel graph-based k -means for nonlinear manifold clustering and representative selection," *Neurocomputing*, vol. 143, pp. 109–122, Nov. 2014.



Enmei Tu was born in Anhui, China. He received the B.Sc. and M.Sc. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2007 and 2010, respectively, and the Ph.D. degree from the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China, in 2014. He is currently a Research Fellow with the Rolls-Royce@NTU Corporate Laboratory, Nanyang Technological University, Singapore. He had worked in Knowledge Engineering and Discovery Research Institute, New Zealand for one year before joining into NTU. His current research interests include machine learning, computer vision, and neural information processing.



and image processing, discovery.

Prof. Kasabov is a fellow of the Royal Society of New Zealand and the New Zealand Computer Society.

Nikola Kasabov (F'10) is currently the Founding Director and Chief Scientist with the Knowledge Engineering and Discovery Research Centre, Auckland University of Technology, Auckland, New Zealand, and the Personal Chair of Knowledge Engineering with the School of Computing and Mathematical Sciences. He has authored over 450 works in international journals and conferences, and books/chapters. His current research interests include computational intelligence, neurocomputing, bioinformatics, neuroinformatics, speech novel methods for data mining, and knowledge



Jie Yang received the bachelor's and master's degrees from Shanghai Jiao Tong University, Shanghai, China, in 1985 and 1988, respectively, and the Ph.D. degree from the University of Hamburg, Hamburg, Germany, in 1994. He is currently the Professor and Director of the Institute of Image Processing and Pattern recognition with Shanghai Jiao Tong University. He is the Principal Investigator of more than 30 national and ministry scientific research projects, including two national 973 research plan projects, three national nature foundation projects, and five international cooperative projects with France, South Korea, Japan, and New Zealand. He has authored around 500 articles in national or international academic journals and conferences.