2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

NeuCube$^{(ST)}$ for Spatio-Temporal Data Predictive Modelling with a Case Study on Ecological Data

Enmei Tu, Nikola Kasabov, Muhaini Othman, Yuxiao Li, Susan Worner, Jie Yang and Zhenghong Jia

***Abstract*— Early event prediction challenges most of existing modeling methods especially when dealing with complex spatio-temporal data. In this paper we propose a new method for predictive data modelling based on a new development of the recently proposed NeuCube spiking neural network architecture, called here NeuCube$^{(ST)}$. The NeuCube uses a Spiking Neural Network reservoir (SNNr) and dynamic evolving Spiking Neuron Network (deSNN) classifier. NeuCube$^{(ST)}$ is an integrated environment including data conversion into spike trains, input variable mapping, unsupervised learning in the SNNr, supervised classification learning, activity visualization and network structure analysis. A case study on a real world ecological data set is presented to demonstrate the validity of the proposed method.**

Keywords—NeuCube architecture; early event prediction; spatio-temporal data; Ecological data processing

I. INTRODUCTION

Early event prediction is very crucial when solving important ecological and social tasks described by temporal- or/and spatio-temporal data, such as pest population outbreak prevention, natural disaster warning and financial crisis prediction. The generic task is to predict early and accurately whether an event will occur in a future time based on already observed spatio-temporal data. The time length of the training data (samples, collected in the past) and the test data (samples used for prediction) can be different as illustrated in Fig. 1.

Predictive modeling of spatio-temporal data (SSD) is a challenging task because it is difficult to model both time and space components of the data because of their close interaction and interrelationship. Traditional machine learning methods, such as support vector machine (SVM) and multi-layer perceptron (MLP), have limited success modeling such data. Firstly, traditional machine learning methods are suitable for classifying vector based and static types of data but not spatio-temporal data [1]. Secondly, most of them use global modeling techniques where a model is derived from all available data that covers the whole problem space and is represented as a single function. This function is then applied to new samples anytime and anywhere regardless of possible contradictory personalized features, thus causing inaccurate decisions. Even though personalized modeling techniques such as *k* nearest neighbors (*k*NN), weighted *k* nearest neighbors (w*k*NN) [2] and weighted distance, weighted variables *k* nearest neighbors (ww*k*NN) [3] could overcome the drawbacks of global modeling, they are still only suitable for vector based classification and static types of data. Thirdly, traditional methods demand the training samples and testing samples to have same length of input features. For early event prediction, a pre-processing technique has to be applied in order to keep all features the same length, but this pre-processing will either cause information loss (by cutting longer features to be short) or introduce residual false information (by padding shorter features to be long).

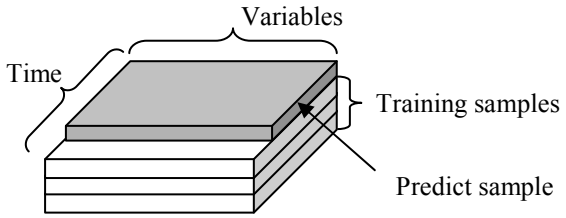


Fig. 1. A spatio-temporal data model used for early event prediction

In this paper we introduce a new method for predictive modeling based on spatio-temporal data using spiking neural networks. Our method can work on both even-feature-length data sets and uneven-feature-length data sets. The proposed method first maps the input features onto the neuron firing state space and the firing state vectors of each sample are fed to a spike neuron network classifier. The potential benefits of the proposed modeling method are: (1) in the proposed modeling method, the feature length (no the sample number) of training samples can be different from that of testing samples. This is rather common in applications of early event prediction, because training samples are usually collected in the past, and thus they have been fully observed; but predicting samples are only partially observed and the feature length increases along with time. This enables our modeling method to make *earlier* prediction, not having to wait for the whole predicting sample to be observed; (2) by introducing a general mapping strategy which maps the similar features

Enmei Tu and Jie Yang are with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200240, China. (corresponding author to provide phone: 0086-021-34204033; e-mail: jieyang@sjtu.edu.cn)

Nikola Kasabov, Muhaini Othman and Yuxiao Li are with the Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Auckland 1010 New Zealand (e-mail: nkasabov@aut.ac.nz, muothman@aut.ac.nz, dawnliwatts@gmail.com).

Muhaini Othman is with Universiti Tun Hussein Onn Malaysia (e-mail: muothman@aut.ac.nz).

Susan Worner is with Bio-Protection Research Centre, Lincoln University, Canterbury, New Zealand (e-mail: worner@lincoln.ac.nz)

Zhenghong Jia is with School of Information Science and Engineering, Xinjiang University, Urumqi, 830046, China (e-mail: jzhh@xju.edu.cn)

The work is sponsored by the Tripartite Collaboration Programme between China (Shanghai Jiao Tong University and Xinjiang University) and New Zealand (Auckland University of Technology) funded by both Education NZ and the Chinese Ministry of Education. This work is partly supported by NSFC China (No: 61273258) and Ph.D. Programs Foundation of Ministry of Education of China (No.20120073110018).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

onto nearby neurons, the modeling method can work on any spatio-temporal data. Meanwhile, by doing so, the model not only can capture the signal changes with time, but also can take the correlations (hence the potential interactions) between features into consideration; (3) the modeling method provides an intuitive way to visualize the inside activities of model during training and predicting stages. This facilitates user to have a better understanding of the model and the data being processed. A case study on a real world ecological data set is presented to demonstrate the validity of the proposed method.

II. NeuCube$^{(ST)}$ for Spatio-Temporal Data Predictive Modeling

To capture the time and space characteristics of spatio-temporal brain data in a spiking neural network (SNN) architecture, NeuCube was proposed and experimented in [1, 4]. The main parts of NeuCube are a three-dimension spiking neural network reservoir (SNNr) and an evolving SNN classifier. In this paper we extend the NeuCube architecture with some new features that make it usable not only for brain data, but for any spatio-temporal data of any type. A block diagram of the architecture is shown in Fig. 2. It contains three parts: an input encoding module, a three-dimension SNNr and an output dynamic evolving spike neural network (deSNN) classifier [4]. The size of the SNNr is controlled by three parameters: n_x, n_y and n_z , representing the neuron numbers along x, y and z direction. And the total neuron number in the reservoir is $N = n_x \times n_y \times n_z$. We call our computing model NeuCube$^{(ST)}$, where *ST* stands for *Spatio-Temporal.*

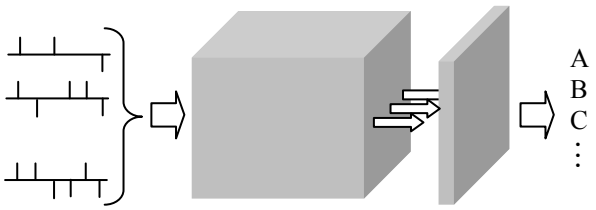


Fig. 2. The architecture of NeuCube$^{(ST)}$. Spike trains (left) are fed to the SNNr and the neuron firing states vectors of this reservoir are used to train a deSNN classifier.

A SNNr of 1000 (10 x 10 x 10) neurons is shown in Fig. 3. The size of the SNNr can vary depending on the prediction task and the data.

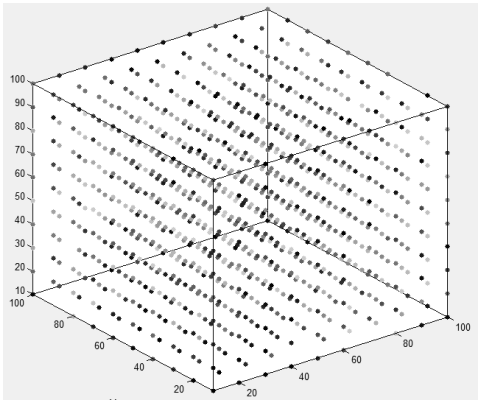


Fig. 3. A spiking neural network reservoir (SNNr) of 1000 neurons

The NeuCube$^{(ST)}$ is trained in a two-stage learning process. The first stage is unsupervised learning that makes the SNNr learn spatio-temporal relations from the input data by adjusting the connection weights in SNNr. The second stage is supervised learning that aims at learning the class information associated with each training spatio-temporal sample. The modeling process contains five components: data encoding, reservoir initialization, unsupervised training of the reservoir, supervised training of the classifier and new sample testing.

A. Data Encoding

In real world applications, spatio-temporal data are often collected in the form of real value sequences. Such sequences cannot be readily used by spiking neural networks. So first of all, we need to convert the continuous signal into discrete spike trains. We use the Address Event Representation (AER) encoding method to discretize the continuous signal. This encoding method was applied successfully for the artificial retina sensor [5].

Specifically, we perform a bi-direction thresholding to the signal gradient with respect to time, d/dt. The threshold is self-adaptive and is determined in the following way: for a signal $f(t)$, we calculate the mean m and standard deviation s of the gradient d/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 places of the increasing signal and a negative spike train which encodes the places of the decreasing signal.

B. NeuCube$^{(ST)}$ *Initialization*

We initialize the SNNr following the *small world connection* rule where: each neuron in the reservoir is connected to its nearby neurons which are within a distance d, where d is equal to the longest distance between any pair of neurons in the reservoir multiplied by a parameter r. The initial weights of the connections are set to the product of a random number within [-1, 1] and the inverse of the distance between them. We randomly select 80% of the connection weights to be positive and 20% to be negative.

The connection weights between the input neurons and other neurons are doubled in order to emphasize these neurons' significance in the reservoir.

C. Training Stage I: Unsupervised Reservoir Training

The unsupervised learning stage is intended to encode 'hidden' spatio-temporal relationships from the input data into neuronal connection weights. According to Hebbian learning rule, if the interaction between two neurons is persistent, then the connection between them will be strengthened. Specifically, we train the reservoir using spike-timing dependent synaptic plasticity (STDP) learning rule [6]: if neuron j fires before neuron i, then the connection weight from neuron j to neuron i will increase and, on the other hand, the connection from neuron i to neuron j will decrease (Fig. 4). 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.

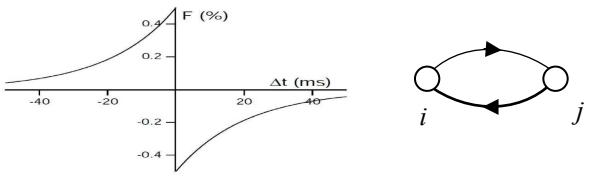


Fig. 4. STDP learning rule [6]. Left: the connection weight changes as a function of firing time difference between two connected neurons. Right: neuron *j* fires before neuron *i* fires, so connection from *j* to neuron *i* increase while connection from *i* to *j* decrease.

In the SNNr, when a neuron fires, it emits a spike and then its potential is reset to 0. Each neuron connecting to this firing neuron will receive a spike and its potential increases with respect to its connection weight to current firing neuron. The potential of each neuron has a small constant rate of leakage over the time unless it becomes 0. After learning, the connection weights in the reservoir encode temporal relationships from the input spatio-temporal data.

D. Training Stage II: Supervised Classifier Training

The second training stage is to train the output classifier using class label information associated with the training samples. The dynamic evolving Spike Neural Networks (deSNN) [7, 8] is used here as an output classifier, because deSNN is computationally efficient and emphasizes the importance of the first spike, which has been observed in biological systems. For each training sample, we create an output neuron and connect it with each neuron in the reservoir. The initial connection weights are all zero. The connection weights are then established by the Rank-Order (RO) learning rule [9]. Specifically, the potential of neuron i at time t is computed using

$$P(i,t) = \sum \mathrm{mod}^{order(j)} w_{j,i} \quad (1)$$

where mod is a modulation factor; *order*(*j*) is the arriving order of the spikes to connection *j*, *i*, among all spikes from all connection to the neuron *i*. This learning rule endows a higher priority to the first spike coming to the output neuron. For the first spike in the reservoir excited by training sample *k*, the connection weights of corresponding output neuron are set by

$$w_{j,i} = \mathrm{mod}^{order(j)} \quad (2)$$

After the first spikes, the connection weight is modified according to firing state of the corresponding neuron in the reservoir: if a neuron fires, it emits a spike to all output neurons that have a connection with it and, as a result, the connection weight between this neuron and the corresponding output neurons strengthens and the potentials of these output neuron increase; otherwise the connection weight weakens and the potential of the output neuron leaks as time elapses. When the potential of an output neuron exceeds a firing threshold, a spike is emitted by it. After training, the connection weights between the output neurons and the reservoir neurons encode both the spike order and the support neurons of each training sample.

E. Testing with New Samples

Once the NeuCube$^{(ST)}$ is trained, all connection weights in the reservoir and in the output classification layer are established. These connections and weights can change based on further training (adaptation), because the evolvable characteristic of the architecture. For a given new sample without any class label information, the trained NeuCube$^{(ST)}$ can be used to predict its class label. For the deSNN classifier, there are two algorithms that can be used to determine the class label of the new sample [7, 8]:

a) deSNNs: it creates an output neuron for the new test sample and after the connection synaptic weights of this neuron are calculated, it compares these weights with the synaptic weights of those training neurons that were established during the supervised training process; the new testing sample is labeled with the label of the existing neuron whose weight vector is closest to that of the new neuron, created for the new sample.

b) deSNNm: it does not create any new output neuron. The new sample is fed to NeuCube$^{(ST)}$ and its class label is set to be same as the training sample whose corresponding output neuron fires first.

III. Input Variable Mapping Method And Visualization of Reservoir Activity

A. Input variable mapping

Given a particular spatio-temporal data set, it is important to know how to map the data into the reservoir. For some special data such as brain EEG data, there is prior information about the location of each signal channel being collected and this information can be readily utilized for mapping the signal channels into the reservoir [4]. But for much more common applications such as climate data and ecological data, we do not have such mapping. Here we introduce a new method to map the input variables into the reservoir.

Suppose there are *s* samples in the data set and there are *v* variables for each sample and the observed time length of each variable is *t*. We first choose *v* input neurons from the SNNr. Then we map the variables into the computing SNNr following this principle: *high 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, and this relationship should also be reflected in the reservoir. Spatially close neurons in the SNNr will capture in their connections more interactions between the input variables mapped into these neurons.

Specifically, we construct two weighted graphs: the input neuron distance graph (NDG) and the signal correlation graph (SCG). In NDG, the input neurons' coordinate set, denoted by $V_{NDG} = \{(x_i, y_i, z_i) \mid i = 1..N\}$, is the graph vertex set and the graph edges are determined in this 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 SCG, we first use the Parzen window method to estimate the spike density function corresponding to each variable and then the graph vertex set, denoted by $V_{SCG} = \{f_i \mid i = 1..N\}$, is the spike density function. The graph edges are constructed in this way: each spike density function is connected to its *k* highest correlated neighbours and the edges are weighted by the statistical correlation between the spike density functions of the input variables.

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 good mapping between any two weighted graphs under the mapping rule. In our case, the two graphs are NDG and SCG. 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:

$$\min_P \| A_n - PA_sP^T \|_F^2 \qquad (3)$$

where $\|\bullet\|_F$ denotes the Frobenius matrix norm. Exactly solving this problem is known to be NP hard due to its 0-1 combinatorial optimization property. Many algorithms have been proposed to find an approximated solution. Among these algorithms the Factor Graph Matching (FGM) algorithm [10] has been demonstrated to produce state-or-art results. So here we utilize the FGM algorithm to solve problem (3) with the following settings: suppose in NDG the sum of graph edge weights of an vertex, say vertex $i_{NDG} \in V_{NDG}$, to all other vertices is $d(i_{NDG})$, and, similarly, in SCG the sum of graph edge weights of vertex $i_{SCG} \in V_{SCG}$ to all other vertices is $c(i_{SCG})$, then the difference between $d(i_{NDG})$ and $c(i_{SCG})$ reflects the similarity of the positions in this graph. So we compute the similarity of the two vertices by using the formula:

$$\exp\left(-|\, d(i_{NDG}) - c(i_{SCG})\,|^2 / 2\sigma_n^2\right);\ \ i_{NDG}, i_{SCG} = 1...v \qquad (4)$$

For the graph edge similarity, we use the formula:

$$\exp\left(-|\, a_{ij}^{NDG} - a_{kl}^{SCG}\,|^2 / 2\sigma_e^2\right);\quad i,j,k,l = 1...v \qquad (5)$$

where: $a_{ij}^{NDG}, a_{kl}^{SCG}$ are graph edge weights in NDG and SCG, respectively; σ_n and σ_e are two parameters to control the affinity between neurons and edges, respectively.

Fig. 5 shows the matching result. The left graph is the input NDG and the right graph is SCG. We can see that after matching, highly correlated features are mapped to nearby input neurons.

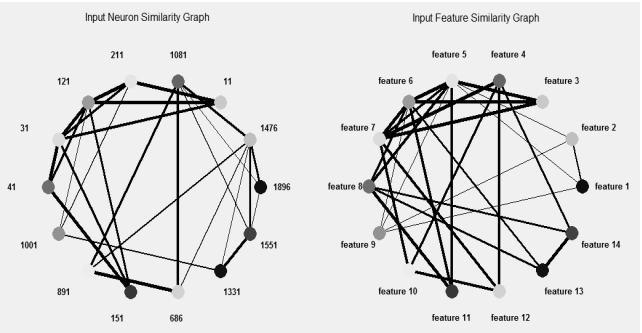


Fig. 5. An input mapping result obtained by the proposed method in the paper using graph matching technique.

B. NeuCube$^{(ST)}$ *visualization*

Visualization of the neuron activities, the connection weight changes and the structure of the SNNr are important for understanding the data and processes that generated it. In NeuCube$^{(ST)}$ we can visualise at each moment the spiking state of the neurons in the SNNr and their connection adjustment. Fig. 6 shows snapshots of the instantaneous neuron spiking state, the connections between neurons and the weight adjustment during a SNNr training. This is very different from traditional methods such SVM which have been used for same tasks but without offering facilities to trace the learning processes for the sake of data understanding.

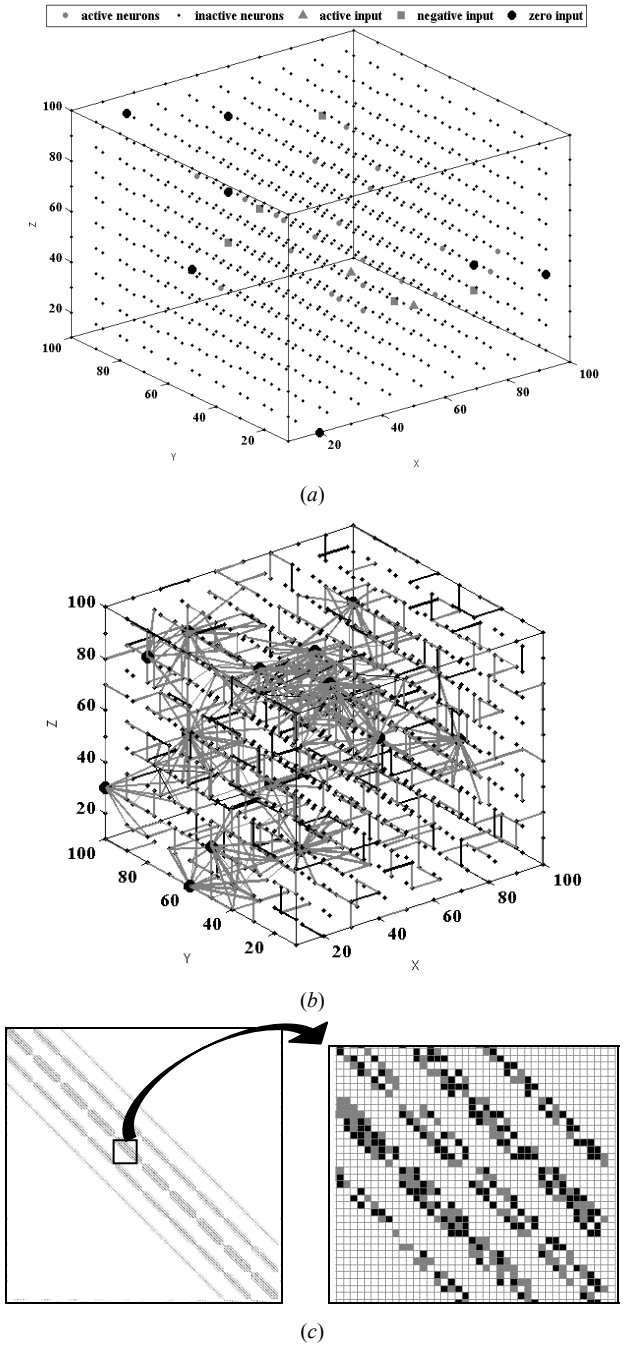


Fig. 6. Snapshots from a dynamic visualisation of a SNNr: (*a*) Neuron spiking state; (*b*) Connections with weights larger than 0.08; (*c*) Connection weight changes. Left: the whole weight matrix; right: magnified illustration of the indicated small region (black indicates decreasing and grey indicate increasing).

After it is trained, the SNNr has captured spatial and temporal relationships from the data. We use the information spreading algorithm [11], which has been demonstrated to be powerful for analysis of dynamic activity spreading (here spike spreading) in networks [12],

to analyze network structure and to explore the relation between the learnt structure and the data set used for training. Further analysis will be presented in the experimental section.

IV. Case Study of Aphids Population Prediction

The NeuCube$^{(ST)}$ environment described above is illustrated here on a case study involving predictive modelling in ecology. Specifically, a model is created to predict aphid abundances in autumn in a geographical location over 11 years.

As major pests, some species of aphids can pose significant damage to autumn agricultural produce particularly wheat crops. Thus, the prediction of a possible abundance of aphids in the autumn seasons becomes important for growers to manage these pests. The numbers of a particular species of aphids (in this study *Rophalosiphum padi*) in autumn are considered to be correlated with environmental conditions and the numbers of aphids in previous spring [13]. The raw data which will be used in the prediction study includes two sets of temporal data. One set is weekly aphid trap catches recorded by Crop & Food Research, Lincoln, Canterbury, New Zealand. The other is made of weekly records for weather variables at the Canterbury Agricultural research centre, Lincoln, New Zealand.

The weather variables include: 1) average rain fall (mm); 2) cumulative weekly rainfall (mm); 3) maximum air temperature (°C); 4) minimum air temperature (°C); 5) mean air temperature (°C); 6) cumulative degree days for the week (°C); 7) grass temperature (°C); 8) soil temperature at 100 cm below ground (°C); 9) Penman potential evaporation (mm); 10) vapour pressure (hPa); 11) potential deficit of rainfall (i.e. accumulated excess of Penman over rainfall); 12), solar radiation (MJ/m2); and 13) wind run (km/day).

Some pre-processing was applied to the raw dataset used in this experiment. Firstly, following[14], some errors in the original data records are addressed, such as the negative records for solar radiations. Secondly, the pre-processing is carried out to take into consideration the relevant results from previous research on aphid prediction. Specifically, for each of the 13 environmental variables, six derived variables were calculated. According to [13], during the flight period for each year, the number of aphids caught in a week was correlated with the mean value of each weather variables over the previous N week, N can be from 1 week to a maximum of 10 weeks. In this study, for each environmental variable, four variables were generated for the averages of previous 1, 2, 3, and 4 weeks. Further, according to [15], it also appears that the numbers of aphids caught in the suction trap (A suction trap is an apparatus catching aphids automatically when they fly into it) are closely related to the changes of some weather variables, rather than the exact measurements in the corresponding weeks, thus, two other variables, the first order derivative and second order derivative are also generated. Among these variables, 14 of them are selected for the current study, based on two rules. Rule 1, for each original weather variable, either the original variable itself, or one of the six generated variables should be selected. Thus, the attribute dimension which is represented by them can be kept in the following study. Rule 2, to select a variable from either the original variable or its six derived variables, the one with the maximum correlation co-efficient with aphid counts will be chosen. Thus, for the following analysis, there are total of fourteen variables: 1) average rainfall (AR, mm); 2) cumulative rainfall, the average of 4 weeks (CR, mm); 3) cumulative degree days (DCT, °C); 4) grass temperature, average of four weeks (GT, °C); 5) maximum air temperature (MaxT, °C); 6) mean air temperature (MeanT, °C); 7) minimum air temperature, average of two weeks (MinT, °C); 8) Penman potential evaporation (PPE, mm); 9) potential deficit of rainfall (PDR), first order derivative; 10) soil temperature (ST, °C); 11) solar radiation (SR, MG/m2); 12) vapour pressure, average of five weeks (VP, hPa); 13) wind run (WR4), average of four weeks (km/day); 14) wind run (WR5), average of five weeks (km/day).

Thirdly, the pre-processing needs to coordinate the data with the experimental design (which will be discussed in detail in below). Instead of predicting weekly aphid counts as in previous studies on the aphid data set [13-15], the research here will predict the pattern of aphids number over the autumn season as a whole. While the time period covered by the original data is from week 27 of 1981 to week 50 of 2000, the whole time series will be parsed into yearly based periods. Each of the time periods will start from week 31 to week 30 the following year. Each of the time segments will be treated as an individual case. For such cases, the associations of spring aphid patterns, the following autumn patterns, and the weather variables for the corresponding periods were the focus.

Based on the examination of the aphid patterns for these time segments, data from 11 years were selected. These 11 years have their spring aphid spikes happening around week 46 of the Julian year. The autumn aphid patterns of these 11 years can be classified into two categories based on the original raw data - the high aphid pattern (class 1) or low aphid pattern (class 2) in autumn. The autumn patterns of aphids of these 11 years will be learnt by the NeuCube$^{(ST)}$. For any future year, when the spring aphid population peaks at about week 46, the potential autumn aphid pattern in the following year can be predicted.

V. Experimental Design and Result

A. Design of Experiment

We conduct three experiments on this real world aphid data set to show the validity of the NeuCube$^{(ST)}$ architecture and how soon before the event our model can achieve a good prediction.

In the first experiment, we aimed to show the predictive ability of NeuCube$^{(ST)}$ for full time length variables. We use the whole time available, i.e. 52 weeks, for both training and testing under the assumption that a perfect weather forecast for the autumn season can be obtained which is an ideal case, but not a realistic one. In the following experiments two and three, we aimed to show the predictive ability of NeuCube$^{(ST)}$ and how early the model can predict the autumn pattern. In these two experiments, we trained NeuCube$^{(ST)}$ using 100% of the time length samples, but only 80% and 75% of the time length to predict the aphid population pattern in the last 25% time period, as illustrated in Fig. 7. The white bars

represent training data length and the grey bars validation data length.

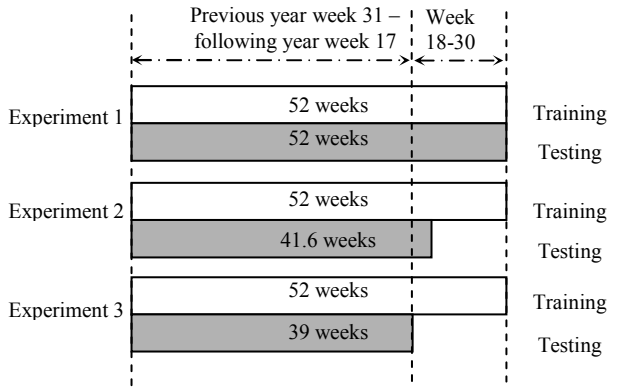


Fig. 7. Experimental design. White bars represent the time length of training samples and the grey bars represent the time length of testing samples.

In all these experiments, the size of the SNNr is 2000 neurons (5x20x20). It is trained and tested in a leave-one-out cross validation mode. This is based on the assumption that the climate change trend during these 11 years can be ignored and the weather variables are independent between different years.

B. Experimental Result

Fig. 8 shows the input variable mapping result on the minimal *x* coordinate face of the cubic reservoir. Note the two main groups of weather variables, in other words, temperature (MaxT, minT, MeanT, DCT, GT, ST) and rainfall (AR, CR, PDR), are mapped to nearby neurons. The solar radiation (SR) is mapped in the middle of temperature variables because temperature is greatly determined by solar radiation.

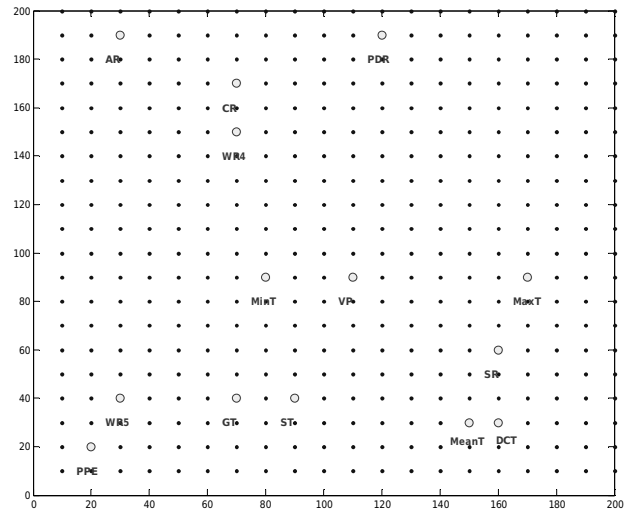


Fig. 8. Input variable mapping result by graph matching

Fig. 9 (*a*) shows the neuron connections with absolute weights larger than 0.07 after the reservoir is trained in one run. Fig. 9 (*b*) plots the total number of spikes each neuron emitted during the training process.

In Fig. 9 (*a*) grey lines represent connections with positive weight and black lines represent connections with negative weights. The line thickness indicates the strength of the connection. The neurons are coloured to indicate their connection strength with other neurons in the reservoir. Brighter neurons have stronger connections with other neurons while darker neurons have weaker connection with others.

From the visualisation, we can derive some conclusions achieve a better understanding of the data and the problem:

a) Connections between nearby input neurons are denser than the connections between far away input neurons. This indicates that there are more interactions between input variables mapped to closer neurons.

b). Neurons in the middle of the reservoir are generally more active than those on the edges. This indicates that the neurons in the reservoir are not uniformly activated that reflects on the input data.

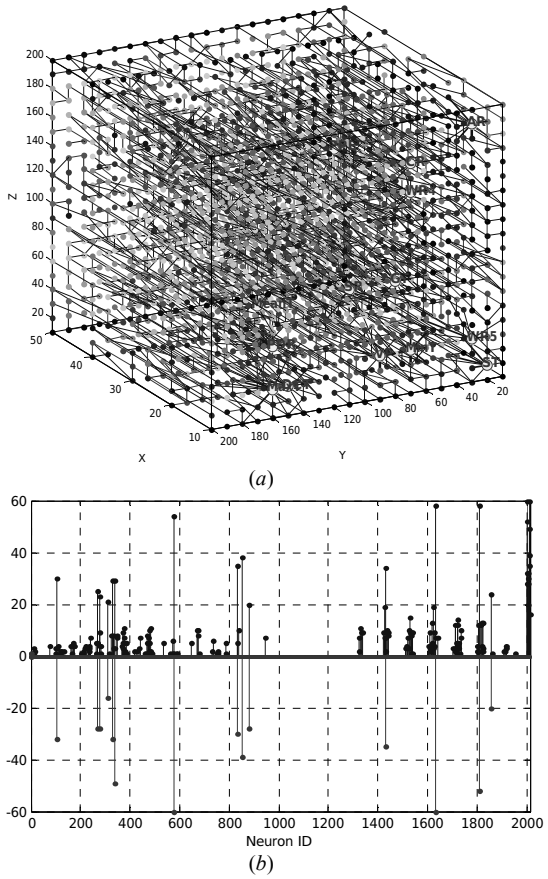
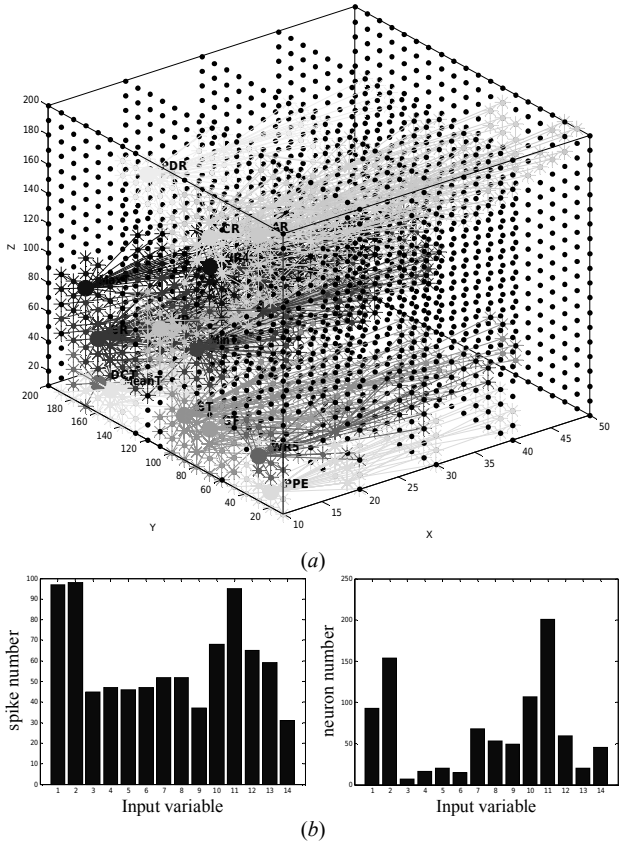


Fig. 9. (*a*) Reservoir connections after training. The black are negative weight connections and the grey are positive weight connections; (*b*) the total spikes amount emitted by each neuron during training process. Negative stems mean negative spikes.

We can also analyze the SNNr structure in terms of information distribution and spread [12]. We first treat the input neurons as the information sources in the network and then use the spreading algorithm from [11] to determine neuron clusters belonging to each information source based on the spike transmission in the network. So the more spikes that are transmitted between two neurons, the tighter they will be connected with each other. After this we link a neuron to the one from which it receives most spikes. Fig. 10 (*a*) shows the network structure after unsupervised training. The big solid dots represent input neurons and other neurons are labeled in the same intensity as the input neuron from which it receives most spikes. The unconnected dots mean no spike arrived at that neuron. In Fig. 10 (*b*), the left pane is spike number

for each variable after encoding and the right pane is the neuron number belonging to each input variable cluster. From this figure we can see the consistency between the input signal spike train and the reservoir structure. It is worth noting that variable 11 (solar radiation) is emphasized in the reservoir that suggests a greater impact of the solar radiation on the aphid number. This was observed also in a previous study [13].



(*a*)

(*b*)

Fig. 10 (*a*) The SNNr structure after unsupervised training; (*b*) Input spike number of each feature (left) and neuronal connections of each input neuron (right).

The early event prediction accuracy on the aphid data set is shown in table I, where the middle row is the time length of test data used for the prediction.

TABLE I AUTUMN APHID PREDICTION ACCURACY (%)

	Accuracy of each testing time length (weeks)		
	52 (full)	41.6 (early)	39
Accuracy	100%	90.91%	81.82%

In the third experiment, 75% of the temporal input data (39 weeks, out of total 52 weeks per year) is used comprising data from week 31 in the previous year to week 18 in following year, where week 18 is at the beginning of the autumn period. At this time of the year an accurate prediction of whether there will be a high aphid amount peaks in later weeks is very important for both ecology and agriculture, because early actions can be taken to prevent the spread of aphid population. For this time length, with 75% of time observed, our model can give 81.82% prediction accuracy. In the second experiment, 80% of time length (41.6 weeks of total 52 weeks per year) is from week 31 in previous year to week 20.6 in next year, where week 20.6 is at the head of the aphid flight period. At this time of the year, some first small aphid peaks have arrived. From this we can see that our model can make *early* decision before the peak appears. With 80% of data observed (early in the aphid flight period), we can have more than 90% confidence to make an early decision. Furthermore, as the time passes, if new data are collected, it can be added directly to the testing sample to give a better prediction, without re-training the model using both old and new data as it would be the case with SVM or MLP methods. This is the essential difference between the new method and traditional methods such as SVM and MLP.

VI. COMPARATIVE STUDY

We conducted experiments to compare between traditional modeling methods and our new modeling method for early event prediction. We used multiple-linear regression (MLR), support vector machine (SVM), multilayer perceptron (MLP), *k* nearest neighbors (*k*NN) and weighted *k* nearest neighbors (w*k*NN) as the baseline algorithms. We designed three experiments for these baseline algorithms, as illustrated in Fig. 11. Note that for these baseline algorithms, the time length of training samples and testing samples have to be the same as these methods cannot tolerate different lengths of feature vectors for training and recall.

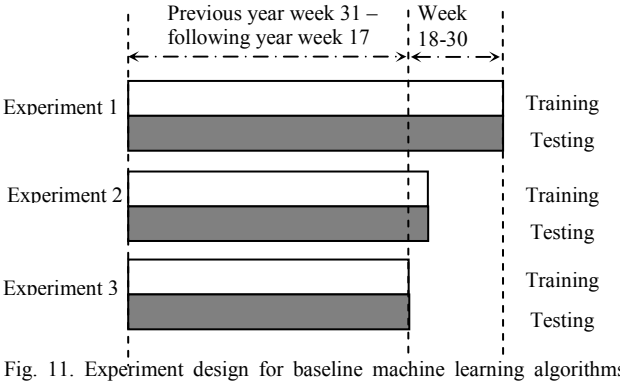


Fig. 11. Experiment design for baseline machine learning algorithms. White bars represent the time length of training samples and the grey bars represent the time length of testing samples.

We prepared the data set in this way: for experiment 1, we concatenated the weather variables one after another to form a long feature vector for each sample, as shown in Fig. 12. For each weather variable, there are 52 data points in one year (one data point per week), so the length of final feature, counting 14 variables data, will be 578. For experiments 2 and 3, we removed the corresponding percentage of data points from each year's data and then perform the same procedure as in experiment 1.

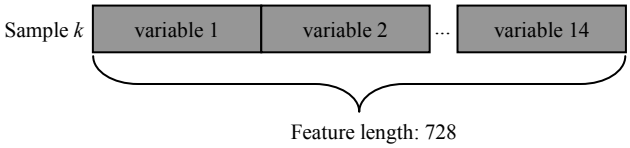


Fig. 12. Data set preparation for baseline algorithms.

We tune the parameters of the baseline algorithms in a grid search way and the final parameters are: a degree 2 polynomial kernel for SVM; 30 hidden neurons and 500 training cycles for MLP; *k*=5 for both *k*NN and w*k*NN. Experimental results are shown in table II.

TABLE II PREDICTION ACCURACY OF APHID DATA SET (%)

	Accuracy of each training and testing time length (weeks)		
	52	41.6	39
MLR	36.36	64.63	72.73
SVM	72.73	72.73	63.64
MLP	**81.82**	**81.82**	**81.82**
***k*NN**	72.73	63.64	63.64
w*k*NN	72.73	63.64	63.64
Max	81.82	81.82	81.82

Comparing table I and table II, we can see that NeuCube(ST) can perform better for early event prediction. A realistic early event prediction should be that as the time length of observed data increases, the prediction accuracy will also increase. But from table II we can see that as the time length of training data increase, traditional modeling methods do not necessarily produce better results (some even worsen), because they cannot model the whole spatio-temporal relationship in the prediction task. They can only model a certain time segment. Because NeuCube(ST) models the whole spatio-temporal relationship in the data, even a small amount of input data can trigger spiking activities in the SNNr that will correspond to the learned full temporal patterns resulting in a better prediction when more temporal data entered.

VII. CONCLUSION & FUTURE WORKS

The research presents a further development of a SNN architecture NeuCube, initially created for modelling spatio-temporal brain data [9,16]. The new architecture–NeuCube$^{(ST)}$ is suitable to model any spatio-temporal data. A method to map input variables onto neurons from the SNNr is proposed here, so that similar input variables based on their temporal similarity are mapped onto spatially closer neurons. The closer the neurons in the SNNr are, the more temporal relationships they learn from data. This is illustrated with analysis of ecological data comprising aphid pest abundance prediction.

Future work includes experiments on other ecological and environmental data collected in New Zealand, China and other countries related to a range of ecological and environmental event prediction, such as earthquakes, tsunami and volcanic activity.

ACKNOWLEDGMENT

This work was completed during the visit of Enmei Tu to the Knowledge Engineering and Discovery Research Institute (KEDRI, www.kedri.aut.ac.nz). The authors would like to thank the anonymous reviewers for their helpful comments to improve this paper.

REFERENCES

[1] N. Kasabov, V. Feigin, Z. G. Hou, C. Y., L. Liang, R. Krishnamurthi, et al., "Evolving Spiking Neural Network Method and Systems for Fast Spatio-Temporal Pattern Learning and Classification and for Early Event Prediction with a Case Study on Stroke," Neurocomputing, in press, 2014.

[2] K. Hechenbichler and K. Schliep, "Weighted k-nearest-neighbor techniques and ordinal classification," http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9, 2004.

[3] N. Kasabov, "Global, local and personalised modeling and pattern discovery in bioinformatics: An integrated approach," Pattern Recognition Letters, vol. 28, pp. 673-685, 2007.

[4] N. Kasabov, "NeuCube evospike architecture for spatio-temporal modelling and pattern recognition of brain signals", Artificial Neural Networks in Pattern Recognition, Springer, 2012, pp. 225-243.

[5] T. Delbruck, B. Christian, and L. Longinotti, "Real time sensory-motor processing for event-based sensors and systems," http://sourceforge.net/p/jaer/wiki/Home/, 2007.

[6] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," Nature neuroscience, vol. 3, pp. 919-926, 2000.

[7] K. Dhoble, N. Nuntalid, G. Indiveri, and N. Kasabov, "Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning," The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1-7.

[8] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition," Neural Networks, vol. 141, pp.188-201, 2012.

[9] S. Thorpe and J. Gautrais, "Rank order coding," Computational Neuroscience, Springer, 1998, pp. 113-118.

[10] F. Zhou and F. De la Torre, "Factorized graph matching," 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 127-134.

[11] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," NIPS, 2004, pp. 595–602.

[12] J. Shrager, T. Hogg, and B. A. Huberman, "Observation of phase transitions in spreading activation networks," Science, vol. 236, pp. 1092-1094, 1987.

[13] S. Worner, G. Lankin, S. Samarasinghe, and D. Teulon, "Improving prediction of aphid flights by temporal analysis of input data for an artificial neural network," New Zealand Plant Protection, pp. 312-316, 2002.

[14] M. J. Watts and S. P. Worner, Using multi-layer perceptrons to model the Lincoln aphid data set: Lincoln University. Bio-Protection & Ecology Division, 2007.

[15] M. J. Watts and S. P. Worner, Comparison of multi-layer perceptrons and simple evolving connectionist systems over the Lincoln aphid data set: Lincoln University. Bio-Protection & Ecology Division, 2007.

[16] Kasabov, N. "NeuroCube: A Spiking Neural Network Architecture for Mapping, Learning and Understanding of Spatio-Temporal Brain Data", Neural Networks, in press 2014