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

Predicting Temporal Sequences Using an Event-based Spiking Neural Network Incorporating Learnable Delays

Tingting (Amy) Gibson, James A. Henderson, and Janet Wiles, *member*

Abstract— **This paper presents a novel paradigm for a spiking neural network to forecast temporal sequences. The key to the approach is a new model of a spiking neuron that can make multi-step predictions, using learnable temporal delays at both dendrites and axons. This model is able to learn the temporal structure of space-time events, adaptable to multiple scales, with the neurons able to function asynchronously to predict future events in a video sequence. This approach contrasts with conventional neural network approaches that use fixed time steps and iterative prediction. Simulations were conducted to compare the new model to a conventional iterative paradigm on motion sequences from a frame-free event-driven Dynamic Vision Sensor (DVS128, 16k pixels), showing that the new approach consistently has a low prediction error while the iterative paradigm is affected by propagated errors.**

spiking neural networks; transmission delays; delay learning; spike-delay-variance learning; dynamic vision sensor

I. Introduction

SPIKING neural networks (SNNs) are beginning to be investigated in engineering for use as bio-inspired computational devices, adding temporal precision to the power of rate-coded neural networks [1]. SNNs differ from traditional neural networks in that spikes are typically modeled with instantaneous responses, no temporal duration, and binary states; enabling a high level of temporal precision and complex dynamics [2]. However, SNNs have primarily been used for modeling neural dynamics rather than as computational devices. In this paper, we address problems in forecasting sensory sequences, by extending SNN modeling to incorporate learnable delays on dendrites and axons (§3).

Sensory data may show little change in the input for a period, followed by rapidly presented dense event streams of new information. For example, in vision, an unchanging scene, such as a white wall, a stationary image, or darkness, can all carry little new information for long periods of time, and can then be followed by rapid scene changes. Such data is amenable to representation as an *event steam*, which is a list of events ordered temporally. Each event indicates a change in state [3], localized in time. Durations between events can vary, enabling an event stream to represent multi-scale phenomena. Event-based sensors are beginning to be developed in neuromorphic engineering, with benefits of low bandwidth, low power, and the ability to span orders of magnitude in temporal precision. The inherently multi-scale nature of event sequences presents a challenge in prediction, especially in forecasting not only *what* events will occur in the future but also *when* they will occur.

Artificial neural networks (ANNs) have had considerable success in time series prediction with results comparable to and in some cases better than classical statistical models [4]-[6], making them good candidates for modeling event sequences. However, consistent with conventional time series prediction paradigms, most ANNs sample time series data at fixed temporal resolutions [7] with the states of their neurons updated synchronously at every simulated time step, and prediction at fixed discrete future time steps. If such ANNs are used with event sequences, the input stream first needs to be converted to a time series with uniform time bins. The resulting series can become large and sparse due to long durations between events; greatly increasing computational cost. Prolonged exposure to an absence of inputs (or unchanging values) while neuron states are iteratively updated can also affect prediction accuracy. Additionally, the sampling time resolution limits the fastest temporal dynamics that can be learnt from the input sequence [8], and updating every part of the network at the same pace increases the difficulty of learning multiple temporal dynamics, especially slow and long-term dynamics.

To predict asynchronous event streams effectively, we consider that a new SNN framework that supports variable durations and event-based processing is needed, based on viewing a sequence of spikes as an event stream. The remainder of this section reviews prediction paradigms in rate-coded ANNs (§1.1), then describes SNN architectures with delays (§1.2). The event-based data used in this study is derived from a 128x128 pixel Dynamic Vision Sensor (DVS128, described in §2). The novel SNN model with learnable delays is described in §3; with simulation results in §4. The paper finishes with a discussion of the model for event-based computation (§5) and conclusions (§6).

A. Neural network architectures for prediction

Traditional neural networks for prediction include Elman's simple recurrent network [9] and Jordan's [10] output-to-hidden recurrent network. Both models are three-layer recurrent networks, with input, hidden and output layers of rate-coded neurons, which process one input at each time step and predict the next input. Elman's architecture has recurrent connections at the hidden layer, which is connected both to the input and to a copy of itself delayed by one time step. Jordan's network has recurrent connections from the output layer to the hidden layer, also delayed by one time

All authors are with the School ITEE, Uni of Queensland, QLD, 4072 Australia (corresponding author J Wiles: j.wiles@uq.edu.au). This work was supported by an APA to TG and AOARD Grant FA2386-12-1-4050.

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

step. Other architectures, such as time-delay neural networks (TDNNs) [11], use multiple past inputs at each time step to predict the next value.

To predict more than one step ahead, two main approaches have been proposed: direct prediction and indirect prediction [12]. Both approaches receive multiple inputs at each time step. The direct approach simultaneously predicts multiple time steps: The simplest network predicts a single output at a desired prediction horizon [13]-[15]; multiple versions of this network each with a different prediction horizon can be combined to form an ensemble to predict multiple steps in the future [16]. A more complex variation is a direct multi-input-multi-output architecture in which a sequence of outputs at predetermined future time steps can be simultaneously predicted [17]. The indirect prediction approach utilizes an iterative prediction mechanism similar to a Jordan network, in which each output is fed back into the network as an input for the next time step [18]. This paradigm has been used in a wide range of time series including chaos [19], macroeconomics [20], traffic [21], wind speed and power [22].

Each method has advantages and disadvantages: Direct prediction has been claimed to be more accurate under some conditions [12] but training multi-level networks can be difficult because all levels must be optimized simultaneously [16]; Indirect prediction can be optimized at each level, but prediction errors propagate to future states and errors accumulate over long periods with no input.

B. SNNs and modeling of delay times

Of the few SNNs that have been designed for prediction, the majority use one-step-ahead prediction. Berthouze and Tijsseling [23] utilized a spiking network architecture akin to Elman's simple recurrent network [9] to predict context-dependent sequences. Object trajectories have also been tracked via a spiking one-step prediction network [24], [25].

Event-based SNN simulators (not for prediction) have also been developed using queues of events and asynchronous updating of neurons [26]-[29]. While these packages enable simulation of spiking neurons with event-based mechanisms, they were not designed for engineering functional tasks.

An open question in neuroscience is how timing and temporal delays are represented in the brain. The majority of models use implicit methods for incorporating temporal properties using neural dynamics, with synaptic weights that have an indirect relationship to delay times. An alternative is to explicitly represent delays, which provides opportunities for more effective and efficient learning algorithms. Temporal delays have been used to encode information [30] and used as temporary memory [31]. Delays can regulate the synchrony of information processing [32], aid pattern recognition [33] and handle long-term dependencies [34]. In a previous project, our group developed a learning algorithm for SNNs with explicitly tunable input delays and variances, called Spike Delay Variance Learning (SDVL), presented at a previous IJCNN [35]. This approach introduced learnable delays at dendrites, but only predicted a single future time point. These methods are yet to be tested on sensory event data such as that produced by DVS128.

II. DATA: FRAME-FREE VISUAL DATA STREAM

The development of new SNN architectures for cognitive systems requires new ways of representing input/output data based on the principle of spikes as asynchronous events, rather than time-locked to uniform sampling. Visual and auditory neuromorphic systems have pioneered this new way of representing sensory stimuli [36], [37]. The DVS128 is a frame-free visual sensor. As noted in the introduction, it has a 128x128 array of pixels (16k pixels), each corresponding to an independent sensor with fast latencies in the microsecond range. Unlike conventional frame-based image sensors, the DVS128 transmits information asynchronously whenever the illumination of a pixel changes more than a threshold amount. This communication is event-based, and results in bandwidth only being used by active pixels. The DVS128 data stream can be accumulated over a period of time in order to approximate frames to any level of temporal resolution. The data stream can also be intensity-coded to enable static images to represent pixel motion (see Fig. 1).

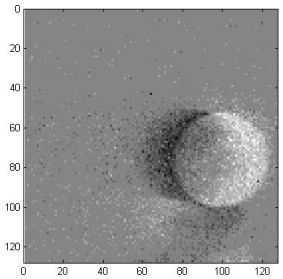

Fig. 1. Image of a rolling ball, created by accumulating DVS128 data stream. White pixels indicate increases in luminosity, black pixels indicate decreases, with intensity indicating the event's recency – whiter pixels indicate more recent positive events and blacker pixels indicate more recent negative events.

A variety of datasets from the DVS128 have been developed for open access, including a moving version of MNIST postcode data [38]. Our lab has developed a range of benchmark datasets including that of laser pointers drawing on a screen, object motion and whole-of-field motion generated by the DVS128 camera mounted on a robot moving through space [39].

A. Laser dot dataset and prediction task

For this study, we used the DVS128 to capture the movement of a spot of light on a plain wall created by a laser pointer oscillating horizontally at approximately 0.44 cycles/sec. The use of a laser dot enables clear visualization of the data over time (see Fig. 2). Over the course of 48s, 21 cycles (moving left then right) were recorded, generating 204,195 events (104,391 positive and 99,804 negative).

DVS128-based studies to date have used classification paradigms [40], [41]; feature recognition based on laser line extraction [42]; particle tracking [43]; and optic flow [44]. Our approach contrasts with these, predicting movement at multiple future time points, and over a range of time scales.

B. Multi-scale representations of DVS128 data

The DVS128 data is represented in raw form as individual

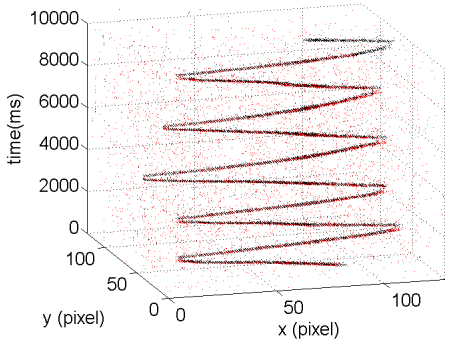

Fig. 2. Raw event stream generated by the DVS128 camera, capturing a laser dot oscillating horizontally (x-axis) over 10s (z-axis). Each dot represents a change of luminosity (red is positive, black is negative), with a temporal resolution of 15μs.

spikes at microsecond resolution. However, many visual tasks of interest require aggregation over large regions of time and space. For example, a cognitive system may need to track a single object moving over time. The variables of interest can be represented by the centroid of the object, rather than an explicit representation of its boundaries. (Note that this is not a universal format: In other tasks, the motion of the boundaries may be the variables of interest, and other representations will be more appropriate.)

In this study, we are interested in predicting the movement of the laser dot over time, so the important feature in the input stream is the change of the position of the centroid of the laser dot. As the raw data stream includes events for all changes in illumination at any pixel over the course of the recording, there is no labelled information about which events correspond to the movement of the laser dot. We assume events that correspond to the laser dot are spatially and temporally close to each other, and determine the position of the laser dot by finding clusters of events that match these criteria. If the object to be tracked is of a complex shape, or if there is more than one object to be tracked, multiple clusters over space and time will be needed. In this study, the laser dot is the only object in the recording and it has very minor movement along the *y* axis; we cluster the events based on time, covering the whole *x* axis and the section of the *y* axis that contains the laser dot, with each cluster encapsulating 200 events.

Representing clusters by centroids provides sufficient information for tracking movement, but loses information about variance. To track the distribution of events, a Gaussian is fit to each cluster (see Fig. 3). Theoretically any distribution could be used to describe these clusters; Gaussians were chosen for computational tractability and simplicity.

In summary, the raw data stream was re-represented by a sequence of clusters, each specified by a Gaussian N with mean $\boldsymbol{m}$ indicating the space-time centroid ($\langle x, y, t \rangle$ in pixel coordinates, x, y, and time since the start of the video, t) and the covariance matrix C is the 3x3 covariance matrix $\langle \sigma_{xx}, \sigma_{xy}, \sigma_{xt}, \ldots, \sigma_{tt} \rangle$.

$$N(\boldsymbol{m}, C) = exp\left[-\frac{1}{2}(\boldsymbol{x} - \mathbf{m})^T C^{-1}(\boldsymbol{x} - \mathbf{m})\right] \qquad (1)$$

where $\boldsymbol{x}$ is the location vector $\langle x, y, t \rangle$ of a data point.

This representation format has several advantages: the sequence of Gaussians forms an event stream in its own right; it can be tailored to any level of down-sampling in space, time or both (in our study, the event stream size reduced from an initial 104,391 positive events to 435 Gaussian events); it is grid-free and scale-free as each Gaussian can represent a region of any shape and size; and it can be used as a scalable data structure in algorithms with a mixture of widely varying spatial scales.

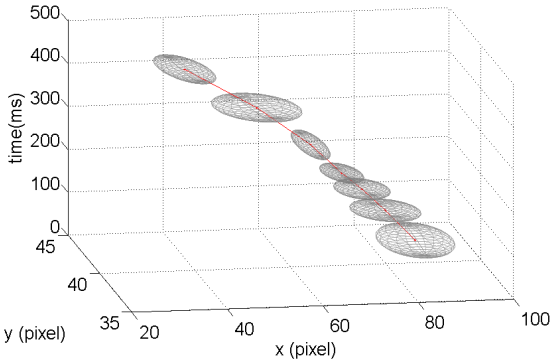

Fig. 3. Scale-free representation of the first 500ms of the moving laser data using Gaussians (boundaries show 1 standard deviation).

III. Methods

A. Spiking neuron model and network architecture

The goal of the new SNN is to learn temporal patterns from an input event stream and predict future event sequences. The network architecture consists of a single layer of hidden spiking neurons that process the input stream and predict an output stream.

The input event stream is presented to the network event by event. Each event, i, can be viewed as approximating the distribution of a region of the DVS space-time data, defined by a Gaussian, N_i ($\boldsymbol{m}_i$, C_i), with mean, $\boldsymbol{m}_i$, and covariance matrix, C_i, (encoding 3 mean and 9 covariance values as described in §2). Each hidden neuron has multiple dendrites sampling different input events and an axonal tree with branches predicting different future event distributions (see Fig. 4). The distribution of each dendrite, j, is summarized by a Gaussian, N_j ($\boldsymbol{m}_j$, C_j), with mean, $\boldsymbol{m}_j$, and covariance matrix, C_j. The number of dendrites per neuron is fixed throughout training but the number of axonal branches changes via recruitment and pruning. Hidden neurons are recruited incrementally throughout training, in proportion to the number of patterns recognized in the input.

When an event is input to the network, all dendrites calculate their match to the event, and all neuron states are updated. Unlike the dendrites in conventional neuron models which passively conduct current to the soma (cell body of the neuron), in our model, dendrites take their inspiration from the growing field of dendritic computation, and are capable of processing the input events with appropriate temporal delays,

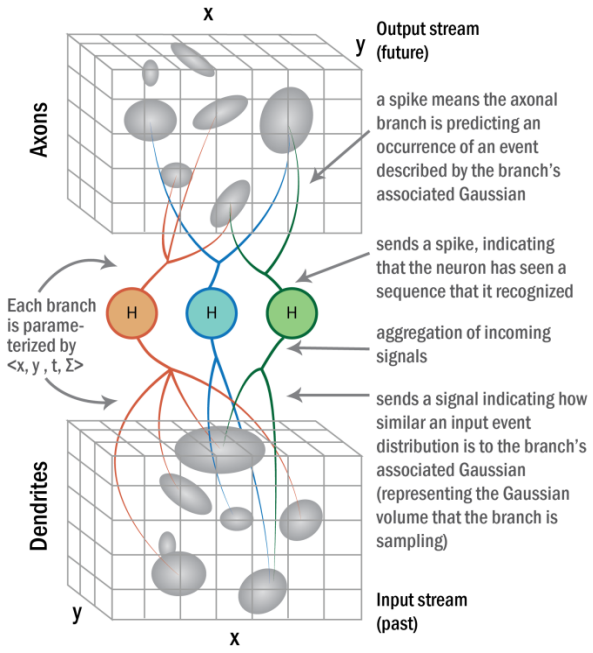

Fig. 4. The architecture of the SNN. Dendrites of the hidden neurons sample the input stream and axonal branches predict the output stream. Each of these branches is summarized by a set of parameters describing a Gaussian distribution. Each dendrite processes a queued event at the appropriate time delay (see text for details), by computing the similarity between its associated Gaussian and the input event, which is integrated at the soma. If a hidden neuron fires, it sends a spike along the axonal branches, which cast predictions based on their learnt Gaussian parameters.

enabling them to represent and match space-time distributions of DVS data. Each branch calculates the match between the input, N_i, and its own distribution, N_j, resulting in a new Gaussian N_p with mean $\boldsymbol{m}_p$ and covariance matrix C_p:

$$N_p(m_p, C_p) = N_i(\boldsymbol{m}_i, C_i)N_j(\boldsymbol{m}_j, C_j) = q_{i,j}(\boldsymbol{m}_i, \boldsymbol{m}_j, C_iC_j)\, N_{i,j}(\boldsymbol{m}_{i,j}, C_{i,j}) \quad (2)$$

where

$$q_{i,j} = exp\left[-\tfrac{1}{2}(\boldsymbol{m}_i - \boldsymbol{m}_j)^T(C_i + C_j)^{-1}(\boldsymbol{m}_i - \boldsymbol{m}_j)\right],$$
$$\boldsymbol{m}_{i,j} = \left(C_i^{-1} + C_j^{-1}\right)^{-1}\left(C_i^{-1}\boldsymbol{m}_i + C_j^{-1}\boldsymbol{m}_j\right), \text{and}$$
$$C_{i,j} = \left(C_i^{-1} + C_j^{-1}\right)^{-1}.$$

The signal, s, from the dendrite to the soma is calculated as the normalized integral of N_p over space and time:

$$s = \frac{\int_{-\infty}^{\infty} N_p(\boldsymbol{m}_p, C_p)dxdydt}{\int_{-\infty}^{\infty} N_j(\boldsymbol{m}_j, C_j)^2\, dxdydt} \quad (3)$$

Although all dendrites are presented with input events simultaneously, calculation of s depends on each dendrite's time delay, t (the time mean of the dendrite's Gaussian parameters). For computational efficiency, the following process is used in our model: When an input is available, all dendrites with no time delay immediately process the input event. For dendrites with a non-zero time delay, the input event is added to a queue associated with the dendrite. As the network is an event-based system without a global clock, the dendrites only continue processing when the next event arrives. As each input event is presented, the delays for all queued events are updated relative to the most recent event. When an event has been delayed to a time that matches or exceeds the time delay associated with the dendrite, the input is processed and the result propagated to the soma.

For each hidden neuron, the membrane potential, v, at the presentation of each new input event is calculated by summing the signals from its dendrites:

$$v = \sum_j s_j \quad (4)$$

where j indexes the neuron's dendrites.

All neurons whose membrane potentials exceed their thresholds (see Table 1 for numerical details) have the potential to fire. To force neurons to learn different input patterns, k-winner-take-all is used to restrict the number of neurons firing for each input event, with the winners based on the highest membrane potentials (for this study, $k = 1$).

When a neuron fires, it generates a spike which propagates along its axonal tree. The axonal tree comprises multiple axonal branches, each associated with a Gaussian distribution parameterized similarly to the dendrites. The time, t, of these parameters indicates the delay from the firing of the hidden neuron to a predicted output event. The future output stream is estimated by summing the predicted events from all neurons over all time (see Fig. 5).

B. *Learning Rules*

The model uses online, self-supervised learning. The learnable variables are the Gaussian parameters of the event distributions associated with the dendrites and axonal branches. At the beginning of training, the network has a pool of embryonic hidden neurons, each with a fixed number of dendrites and an allocation of axonal branches not yet associated with event distributions. When a new input event is fed to the network and no hidden neurons fire in response to the input, an embryonic neuron is recruited to one-shot-learn

TABLE I
NETWORK PARAMETERS

	Direct network		Iterative network	
	Initial	After training	Initial	After training
No. of hidden neurons	0	23	0	50
No. of dendrites/neuron	10	10	10	10
No. of axonal branches/neuron	20	avg 15	1	1
Threshold	2	varies	9	varies
- adjustment rules	1.1 when the neuron fires 0.9 when the neuron does not fire			
Direct network only parameters				
Prediction confidence	Initialized to 1			
- adjustment rules	1.01 if the prediction does not add new axon 0.9 if the prediction adds new axon 1 if result is greater than 1			
- for axon pruning	0.9			
Distance for adding new axon	20 bits in information loss			

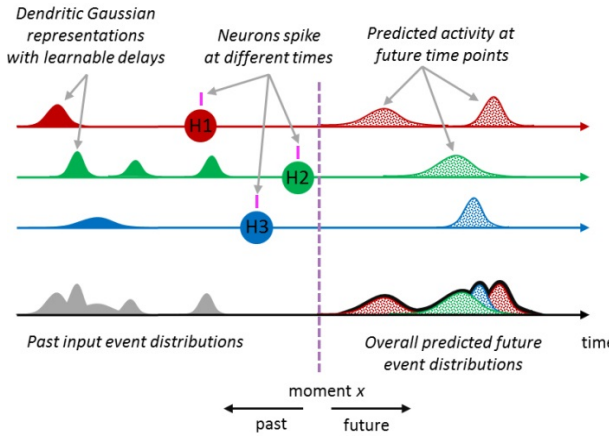

Fig. 5. Diagrammatic view of delays on the dendrites and axons. The x-axis represents time. Shown are three neurons H1, H2, and H3 (one row per neuron) and the input sequence (grey distribution, bottom row). Each neuron fires after matching a particular part of the input (distributions shown in solid colours) and predicts a future sequence (stippled colours). To calculate the total predicted future event stream at the time point marked by the vertical dashed line, predictions are summed from all neurons that fired previously (resulting in the distribution shown by the black outline, bottom row).

this input pattern; its dendrites are initialized to match the input distributions (means and covariance values of the nearest events in the past) and its axonal branches are initialized to a subset of randomly chosen future events within a given time range (5s).

Learning occurs at dendrites and axons only if their associated hidden neuron fires. When a hidden neuron fires in response to an input sequence, its dendrites adjust their distribution parameters closer to those of the input event. The distribution that best matches their estimation is found using the Kullback–Leibler divergence D_{KL} (also known as relative entropy). D_{KL} measures the information lost when the normal distribution of an input event N_i^n (normalized N_i) is approximated by that of a dendrite N_j^n:

$$N^n(\boldsymbol{m}, C) = (1/\sqrt{(2\pi)^3 \det(C)})\, N(\boldsymbol{m}, C) \qquad (5)$$

$$D_{KL}(N_i^n || N_j^n) = \frac{1}{2}\begin{pmatrix} (\boldsymbol{m}_j - \boldsymbol{m}_i)^T C_j^{-1} (\boldsymbol{m}_j - \boldsymbol{m}_i) \\ +tr(C_j^{-1} C_i) - k - \ln\left(\frac{\det(C_i)}{\det(C_j)}\right) \end{pmatrix} \qquad (6)$$

where $\boldsymbol{m}_i$, $\boldsymbol{m}_j$, C_i and C_j are the means and covariances of the normal distributions N_i^n and N_j^n, and k is the dimension of the distribution. For computational tractability, after a dendrite has identified the best-matched input event, its parameters are updated numerically as follows: A new distribution is calculated by generating d_1 data samples from the dendrite's old Gaussian distribution, and d_2 data samples from the input event to be learnt. The ratio d_1:d_2 is given by

$$(1 - 1/r) : 1/r \qquad (7)$$

where r is the number of times the hidden neuron has fired since the beginning of training. This adjustment ensures that a frequently firing neuron becomes more stable over time.

Axonal branches of the hidden neurons learn using a similar algorithm to the dendrites. After prediction of a future event, the nearest event from the real data source is matched (using Eqn 6). If the match is close (this study used an information loss of 20bits, calculated by $D_{KL}(N_i^n||N_j^n)/\log 2$), the axonal branch learns as described for the dendrites. Otherwise, a new axonal branch is added with parameters matching the targeted prediction. Each of the axonal branches is also associated with a confidence value (see Table 1 for implementation values), which is increased when the branch matches an input, and decreased when not. After training, axonal branches with low prediction confidences are pruned (see Table 1 for values).

When a neuron fires, its threshold is increased to make it harder to fire in response to other patterns in the future. This mechanism forces the neuron to specialize in its learned pattern. If a neuron does not fire over a period of time, its threshold is reduced to aid generalization (see Table 1 for adjustment rules).

IV. Results

A. Directly predicting a sequence

After training on 33 seconds of laser motion data (approx. 15 oscillations of the laser pointer), the SNN reliably predicted the future movement of the laser dot for unseen data (tested for 5 seconds). Figure 6 provides a segment of this network output showing the shape of the predicted future sequence (red contours), the input events (blue contours) and the source data (black dots). The predicted trajectory closely follows the source data.

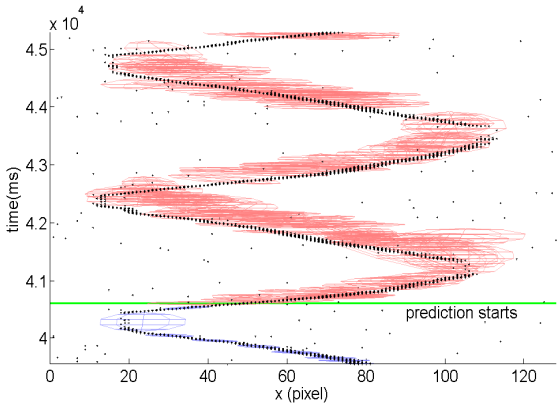

Fig. 6. Predicting future motion with spiking neurons. An SNN was trained on the moving laser pattern (see Fig. 2). Red contour lines: the shape of Gaussian predictions to 1 standard deviation. Blue contour lines: the shape of the input Gaussians to 1 standard deviation. Black dots: the actual spikes from the DVS128 recording.

B. Iterative prediction

For comparison, a second network architecture was implemented based on the traditional time series approach of predicting the value of the next input. The technique uses a recurrent mechanism – iteratively feeding the prediction for the next time step into the input for the subsequent prediction. To compare the prediction performances of the SNN (direct) and iterative approaches, a traditional iterative prediction network was trained on the same laser pointer data. Iterative

networks are usually fixed time-step based, so the data was processed into frames by accumulating all spikes within 100ms intervals. A Gaussian was then fitted to each frame, forming a frame sequence comparable to its event counterpart. The network was trained to predict the next time step in the form of a Gaussian. The learning mechanism for adapting the Gaussians was as described in §3B. During testing, at each time step the network predicted the next event, which then became part of the subsequent input sequence. This iterative prediction can be continued for as long as required.

C. Error comparison

Both the direct and iterative SNNs were trained on 33 seconds of the data (each network using its corresponding pre-processed inputs), then tested with 20 novel sequences, each consisting of 1.1 seconds of previously unseen input data and 5 seconds of forecasting data. Each neuron in the direct network was initialized with 10 dendrites and 20 axonal branches; while the iterative network was initialized with 10 dendrites (equal to an input sequence of 10 frames), but only one axonal branch for predicting the next time step. (See Table 1 for network settings).

For comparison with the source data stream, 10000 points were sampled from each network's predicted future distributions. The sampled points and the raw data were binned into discrete *time* bins of 200ms and discrete *x* position bins of 8 pixels, forming a 25x16 grid (the variation in the *y* axis was ignored as the laser dot motion was restricted to *x* axis). Each cell in the grid was then represented by the event count of that cell. The prediction error of a cell $e_{n,x,t}$, of network model n at position bin x and time bin t, was calculated as follows:

$$e_{n,x,t} = abs\left(p_{n,x,t} - p_{a,x,t} \right) * w_{n,x,t} \qquad (8)$$

where $p_{n,x,t}$ is the normalised prediction results of the network model m at cell (x,t); $p_{a,x,t}$ is that of the actual data events; and w is the weight of the error of a cell, calculated as the ratio of the normalised prediction results and that of the actual data:

$$w_{n,x,t} = \max\left(p_{n,x,t}, p_{a,x,t}\right) / \min\left(p_{n,x,t}, p_{a,x,t}\right) \qquad (9)$$

To prevent undue skew in the weights, the maximum weight was capped at 50.

Prediction errors for both networks were calculated as a function of time and averaged over the 20 test cases (see Fig. 7). The error of the direct model remained low and was consistent over time; the iterative model showed comparable performance over the first 200msec, but then consistently increased with time. The contrast between direct and iterative prediction paradigms illustrates the sensitivity of the iterative paradigm to prediction error in earlier steps, with errors in past predictions propagated to subsequent predictions. It also shows the direct multi-step paradigm is consistent throughout prediction. The slight downward trend of the prediction error of the direct network is due to increasing sparsity of the prediction at later times (the performance measure favours sparse predictions over dense predictions that cover a large area).

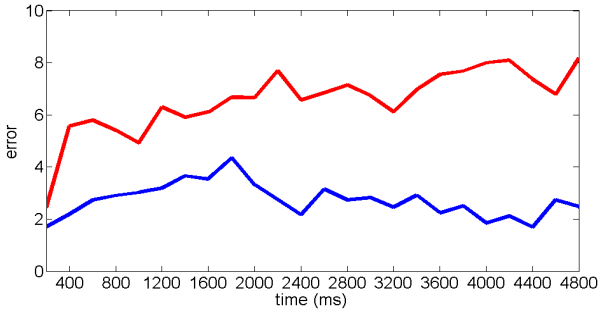

Fig. 7. Prediction errors over time for two contrasting paradigms (average of 20 test cases). *Direct prediction paradigm* (blue): The errors remain low throughout the entire prediction. *Iterative prediction paradigm* (red): Initially good with increasing trend in time.

V. Discussion

In this paper, we presented a new approach to forecasting using SNNs with event-based inputs. It required rethinking how to represent inputs, and augment spiking neurons with learnable delays to enable processing of event streams. The new designs have both benefits and costs, which we discuss in the following sections.

A. Using distributions to represent raw input data

New types of neuromorphic sensors such as the DVS128, require correspondingly new processing paradigms. In this study we have developed an event-based scale-free data structure involving Gaussians that can sample any degree of precision in time and space, free from fixed step sizes of a spatial or temporal grid.

The distribution of the events generated by the DVS128 can be viewed as a density function with a time resolution that has microsecond precision. We re-represented this density function as a set of Gaussians, which can be scaled from one Gaussian per spike, at the most detailed level, to a single Gaussian for the entire space-time stream, with any mixture of Gaussians in between. These Gaussians can have varying spatial and temporal scales, and the covariance of each Gaussian can be parameterized differentially along the spatial and temporal axes. Such re-represented data forms an event stream in its own right, with the advantage of greatly reducing the number of events and enabling mixtures of differentially scaled space-time events.

B. An SNN that processes input distributions as events

The hidden neurons modeled in our network differ from conventional spiking neurons by adding time delays to axonal arbors as well as dendrites, enabling the simultaneous prediction of multiple future events. The advantage of this approach is that predictions are not required for all future time steps, only those for which data is available. The Gaussian parameters of the dendritic and axonal branches can be seen as similar to synaptic weights in traditional neural networks: they affect the signal received by the neurons and are learned during training. One of these parameters, the time delay variable, plays a key role in multiple time-scale prediction.

C. Capturing temporal structures in time delays

The goal of this SNN model is to learn the temporal structures of an event-based sequence and predict how space-time events are related to each other. Temporal information is represented in the time delays of the dendrites, as each time delay encodes the temporal position of its associated distribution in relation to other distributions in the same dendritic tree. Each dendrite is free to encode the temporal scale that best suits the events it is trained on. This feature provides great flexibility in learning temporal structures in a time step free environment.

In contrast to traditional models of dendrites as passive structures, the main role of which is propagating signals, the dendrites in this study have considerable computational ability. In this model, active dendritic computation is postulated as a temporo-spatial sequence memory. The time delays of the dendrites are implemented as delays placed on computation rather than transmission, contrasting with conventional models [31]-[34]. Further studies are needed to determine if this mechanism impacts on the quality of prediction.

The axonal model used in this network is an extension of dendritic computation, and to our knowledge has not been studied in this form in neuroscience (see [45] for a review). Combining the use of time delays with dendritic and axonal computation is the key mechanism that makes learning event-based multi-time-scale sequences possible and future studies are needed to compare this axonal branching model with other approaches.

D. Application to real data

When applied to real data recorded from the DVS128, this model successfully predicted future events with high accuracy. Comparison was made between the direct and iterative predictions, with the former consistently more accurate, as the latter was affected by propagated errors.

The data used in this study is very simple, consisting of a single object with a simple oscillating trajectory. Although not demonstrated, the network should be able to handle multiple trajectories by training different sets of hidden neurons to learn each trajectory. For complex real world scenes with multiple objects, pre-processing the data with multiple smaller Gaussians would more suitable to enable segmenting individual objects and features of complex objects. Smaller Gaussians could also help to identify atmospheric noise which tends to be uncorrelated with other motion in scenes.

The new SNN model is not limited to processing data from the DVS camera. Any data that can be converted to a space-time sequence of events could be processed.

E. Comparing this model to traditional ANNs

Many aspects of our network and prediction paradigm differ from traditional time series predicting ANNs:

1) Our system is event-based and clock-free, learns and predicts multiple time-scale data.
2) Events are represented as Gaussians as opposed to single-valued inputs. These Gaussians represent the distributions of events in space and time; and the Gaussians associated with the dendrites and axons indicate the temporo-spatial volume the branches sample or predict. This is a different way of thinking about data compared to traditional ANNs.
3) Our system incorporates dendritic computation and axons with learnable delays contrasting with traditional transmit-only connections.

F. Potential improvements to this model

This network shows promising results when predicting the laser dot dataset. However, the model design is still at a preliminary stage and there are many aspects that could be improved and further explored:

1) The laser dot data was selected deliberately as a simple demonstration of the Gaussian event-based network model. Studies will be done with more complex data, such as visual flow from self-motion, multiple moving objects and other complex scenes.
2) Currently the dendrites learn the most recent sequences. Future extensions should enable them to learn subsequences sampled from multiple non-consecutive time points. Growing, pruning and merging branches based on the information encoded by each branch could help to evolve a more elegant network structure.
3) In the current system, processing is updated with input events, potentially causing mismatches with the learned delays of the dendrites. Updating time points could be dynamically generated to correspond to dendritic delays.

VI. Conclusions

Neuromorphic engineering is developing powerful computational hardware for spiking networks. However, paradigms that advance the theory of computation with spikes have been slower to develop. This paper demonstrates how a novel model of a spiking neuron with explicitly learnable delays on both dendrites (inputs) and axons (outputs) can efficiently and effectively connect pasts and futures represented as event streams which have multiple scales.

The DVS128 datasets for this study are available as open access data from http://www.itee.uq.edu.au/cis/.

Acknowledgments

We thank students and colleagues in the Complex and Intelligent Systems group at UQ, especially Rob Quinn, David Tingley, Joshua Arnold and Tharun Sonti for the video sequences used in these studies. The authors are grateful to Tobi Delbrück for his valuable advice and for the jAER software infrastructure.

References

[1] H. Adeli and S. Ghosh-Dastidar, "Spiking neural networks," International Journal of Neural Systems, vol. 19, pp. 295-308, 2009.

[2] P. Stratton and J. Wiles, "Self-sustained non-periodic activity in networks of spiking neurons: The contribution of local and long-range connections and dynamic synapses," NeuroImage, vol. 52, pp. 1070-1079, 2010.

[3] K. M. Chandy, "Event-driven applications: costs, benefits and design approaches," Presented at the Gartner Application Integration and Web Services Summit, San Diego, CA, Jun. 2006.

[4] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," Management Science, vol. 42, pp. 1082-1092, Jul. 1996.
[5] G. P. Zhang, B. E. Patuwo, and M. Y. Hu, "A simulation study of artificial neural networks for nonlinear time-series forecasting," Computers & Operations Research, vol. 28, pp. 381-396, 2001.
[6] D. Chaturvedi, "Artificial neural network and supervised learning," in Soft Computing. vol. 103, ed: Springer Berlin Heidelberg, 2008, pp. 23-50.
[7] E. Kayacan, B. Ulutas, and O. Kaynak, "Grey system theory-based models in time series prediction," Expert Systems with Applications, vol. 37, pp. 1784-1789, 2010.
[8] J. Reutimann, M. Giugliano, and S. Fusi, "Event-driven simulation of spiking neurons with stochastic dynamics," Neural computation, vol. 15, pp. 811-830, Apr. 2003.
[9] J. L. Elman, "Finding structure in time," Cognitive science, vol. 14, pp. 179-211, 1990.
[10] M. I. Jordan, "Supervised learning and systems with excess degrees of freedom," 1988.
[11] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 37, pp. 328-339, 1989.
[12] C.-T. Cheng, J.-X. Xie, K.-W. Chau, and M. Layeghifard, "A new indirect multi-step-ahead prediction model for a long-term hydrologic prediction," Journal of Hydrology, vol. 361, pp. 118-130, 2008.
[13] A. Palmer, J. José Montaño, and A. Sesé, "Designing an artificial neural network for forecasting tourism time series," Tourism Management, vol. 27, pp. 781-790, 2006.
[14] F. A. Guerra and L. d. S. Coelho, "Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis neural network with learning by clustering and particle swarm optimization," Chaos, Solitons & Fractals, vol. 35, pp. 967-979, 2008.
[15] H.-T. Pao, "Forecasting electricity market pricing using artificial neural networks," Energy Conversion and Management, vol. 48, pp. 907-912, 2007.
[16] F.-J. Chang, Y.-M. Chiang, and L.-C. Chang, "Multi-step-ahead neural networks for flood forecasting," Hydrological Sciences Journal, vol. 52, pp. 114-130, Feb. 2007.
[17] M. Campolo, A. Soldati, and P. Andreussi, "Artificial neural network approach to flood forecasting in the River Arno," Hydrological Sciences Journal, vol. 48, pp. 381-398, Jun. 2003.
[18] R. Boné and M. Crucianu, "Multi-step-ahead prediction with neural networks: a review," 9emes rencontres internationales: Approches Connexionnistes en Sciences, vol. 2, pp. 97-106, 2002.
[19] H. Mirzaee, "Long-term prediction of chaotic time series with multi-step prediction horizons by a neural network with Levenberg–Marquardt learning algorithm," Chaos, Solitons & Fractals, vol. 41, pp. 1975-1979, 2009.
[20] T. Teräsvirta, D. van Dijk, and M. C. Medeiros, "Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: a re-examination," International Journal of Forecasting, vol. 21, pp. 755-774, 2005.
[21] J. M. P. Menezes Jr and G. A. Barreto, "Long-term time series prediction with the NARX network: An empirical evaluation," Neurocomputing, vol. 71, pp. 3335-3343, 2008.
[22] T. G. Barbounis, J. B. Theocharis, M. C. Alexiadis, and P. S. Dokopoulos, "Long-term wind speed and power forecasting using local recurrent neural network models," Energy Conversion, IEEE Transactions on, vol. 21, pp. 273-284, 2006.
[23] L. Berthouze and A. Tijsseling, "A neural model for context-dependent sequence learning," Neural processing letters, vol. 23, pp. 27-45, Feb. 2006.
[24] C. Boucheny, R. Carrillo, E. Ros, and O. M. D. Coenen, "Real-time spiking neural network: an adaptive cerebellar model," in Computational Intelligence and Bioinspired Systems. vol. 3512, ed: Springer Berlin Heidelberg, 2005, pp. 136-144.
[25] H. Burgsteiner, M. Kröll, A. Leopold, and G. Steinbauer, "Movement prediction from real-world images using a liquid state machine," Applied Intelligence, vol. 26, pp. 99-109, Apr. 2007.
[26] L. Watts, "Event-driven simulation of networks of spiking neurons," Advances in neural information processing systems, pp. 927-927, 1994.
[27] A. Delorme and S. J. Thorpe, "SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons," Network: Computation in Neural Systems, vol. 14, pp. 613-627, Jan. 2003.
[28] S. J. Thorpe, R. Guyonneau, N. Guilbaud, J.-M. Allegraud, and R. VanRullen, "SpikeNet: real-time visual processing with one spike per neuron," Neurocomputing, vol. 58–60, pp. 857-864, 2004.
[29] M. Mattia and P. Del Giudice, "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses," Neural Computation, vol. 12, pp. 2305-2329, 2000.
[30] S. Tolnai, B. Englitz, J. Scholbach, J. Jost, and R. Rübsame, "Spike transmission delay at the calyx of Held in vivo: rate dependence, phenomenological modeling, and relevance for sound localization," Journal of neurophysiology, vol. 102, pp. 1206-1217, 2009.
[31] P. Sterne, "Information recall using relative spike timing in a spiking neural network," Neural Computation, vol. 24, pp. 2053-2077, 2012.
[32] C. Panchev, "A spiking neural network model of multi-modal language processing of robot instructions," in Biomimetic Neural Learning for Intelligent Robots, ed: Springer, 2005, pp. 182-210.
[33] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 1017-1024.
[34] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," Neural computation, vol. 19, pp. 2881-2912, 2007.
[35] P. W. Wright and J. Wiles, "Learning transmission delays in spiking neural networks: A novel approach to sequence learning based on spike delay variance," in Neural Networks (IJCNN), The 2012 International Joint Conference on, 2012, pp. 1-8.
[36] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," Solid-State Circuits, IEEE Journal of, vol. 43, pp. 566-576, 2008.
[37] L. Shih-Chii, A. Van Schaik, B. A. Minch, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, 2010, pp. 2027-2030.
[38] T. Serrano-Gotarredona and B. Linares-Barranco. MNIST-DVS Database [Online]. Available: http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html
[39] T. Gibson, S. Heath, R. P. Quinn, A. H. Lee, J. T. Arnold, T. S. Sonti, A.Whalley, G. P. Shannon, B. T. Song, J. A. Henderson and J. Wiles, "Event-based visual data sets for prediction tasks in spiking neural network", To appear in the Proceedings of the 2014 International Conference on in Artificial Neural Networks (1CANN), 2014
[40] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," Frontiers in neuroscience, vol. 7, Oct. 2013.
[41] A. Eun Yeong, L. Jun Haeng, T. Mullen, and J. Yen, "Dynamic vision sensor camera based bare hand gesture recognition," in Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), 2011 IEEE Symposium on, 2011, pp. 52-59.
[42] C. Brandli, T. Mantel, M. Hutter, M. Höpflinger, R. Berner, R. Siegwart, and T. Delbruck, "Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor," Frontiers in neuroscience, vol. 7, Jan. 2014.
[43] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," Experiments in Fluids, vol. 51, pp. 1465-1469, Nov. 2011.
[44] F. Koeth, H. G. Marques, and T. Delbruck, "Self-organisation of motion features with a temporal asynchronous dynamic vision sensor," Biologically Inspired Cognitive Architectures, vol. 6, pp. 8-11, 2013.
[45] D. Debanne, "Information processing in the axon," Nat Rev Neurosci, vol. 5, pp. 304-316, 2004.