IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 July 2022, accepted 10 August 2022, date of publication 22 August 2022, date of current version 21 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3200699

RESEARCH ARTICLE

An Optimized Deep Spiking Neural Network Architecture Without Gradients

YESHWANTH BETHI, YING XU, (Member, IEEE), GREGORY COHEN, (Member, IEEE), ANDRÉ VAN SCHAIK, (Fellow, IEEE), AND SAEED AFSHAR, (Member, IEEE)
International Centre for Neuromorphic Systems, MARCS Institute, Western Sydney University, Sydney, NSW 2150, Australia

Corresponding author: Yeshwanth Bethi (y.bethi@westernsydney.edu.au)

This work was supported in part by the Commonwealth of Australia through the Next Generation Technology Fund (NGTF) Cyber Call 2019; and in part by the Defence Science and Technology Group, Department of Defence.

ABSTRACT We present an end-to-end trainable modular event-driven neural architecture that uses local synaptic and threshold adaptation rules to perform transformations between arbitrary spatio-temporal spike patterns. The architecture represents a highly abstracted model of existing Spiking Neural Network (SNN) architectures. The proposed Optimized Deep Event-driven Spiking neural network Architecture (ODESA) can simultaneously learn hierarchical spatio-temporal features at multiple arbitrary time scales. ODESA performs online learning without the use of error back-propagation or the calculation of gradients. Through the use of simple local adaptive selection thresholds at each node, the network rapidly learns to appropriately allocate its neuronal resources at each layer for any given problem without using an error measure. These adaptive selection thresholds are the central feature of ODESA, ensuring network stability and remarkable robustness to noise as well as to the selection of initial system parameters. Network activations are inherently sparse due to a hard Winner-Take-All (WTA) constraint at each layer. We evaluate the architecture on existing spatio-temporal datasets, including the spike-encoded IRIS, latency-coded MNIST, Oxford Spike pattern and TIDIGITS datasets, as well as a novel set of tasks based on International Morse Code that we created. These tests demonstrate the hierarchical spatio-temporal learning capabilities of ODESA. Through these tests, we demonstrate ODESA can optimally solve practical and highly challenging hierarchical spatio-temporal learning tasks with the minimum possible number of computing nodes.

INDEX TERMS Local learning, neuromorphic feature extraction, spiking neural networks, spike-timing-dependent plasticity, supervised learning.

I. INTRODUCTION

Over the last decade, Deep Learning using Artificial Neural Network (ANN)s has been adapted to almost every computational field. The vast adoption of Deep Learning can be attributed to the simple error back-propagation algorithm [1] that enabled auto-differentiation for the ANN models. The error back-propagation algorithm solves the credit-assignment problem for any arbitrary cascade of layers, activation functions, and any given loss function, as long as they are all differentiable. The power of such modularity gave rise to complex hierarchical models which could adapt to any domain of data such as numerical, images, audio, video, robotics, and natural language. Modular ANNs are also capable of handling multi-modal data from different domains. Yet, the error back-propagation rule which underpins ANNs does not fit well with our understanding of the form and function of biological SNNs, which were the original inspiration of ANNs. This troubling incongruity is a major focus of research in computational neuroscience and has motivated many attempts at adaptation, reconciliation, and re-interpretation of the computations involved in both artificial and biological neural networks [2].

There are several reasons why the error back-propagation algorithm can not plausibly be implemented in biological neural networks [3], [4]. First, error back-propagation requires a numerically precise error measurement for a batch of data. To date, no evidence for such batch processing or

The associate editor coordinating the review of this manuscript and approving it for publication was Zhipeng Cai.

numerically precise error measurements has been uncovered in the brain and given our mature understanding of the behaviour of biological neural networks, such evidence is unlikely to emerge. Second, successful error back-propagation requires global, precise, and repeated propagation of error measures through all the involved computational nodes in a network - beginning from the inputs to the highest layers and back. This has been dubbed the 'weight transport problem', as the weights of higher layers have to be made available to the lower layers for successful backpropagation of error values [4]. Again, no evidence for such processes has been, or is likely to be, found. The third and most crucial pillar of error back-propagation is the differentiability requirement for all the constituent components of a given network. Indeed, it is this very aspect of the error back-propagation algorithm which makes it difficult to use on non-differentiable data domains like spatio-temporal spikes patterns which the brains use as the primary mode of computation and communication. This leads to the biggest problem in the use of error back-propagation in computational neuroscience namely, the credit-assignment problem.

There have been multiple attempts at approximating error back-propagation and applying gradient descent to SNN architectures. SpikeProp [5] was among the first works to derive a supervised learning rule for SNNs from the error back-propagation algorithm. Tempotron [6] was introduced in which the error back-propagation was applied by defining loss functions based on the maximum voltage and the threshold voltage of the output neurons. Chronotron [7] was introduced as an improvement over Tempotron by using a new distance metric between the predicted and target spike trains. More recent works applied error back-propagation to SNN architectures by using different surrogate gradients for the hard thresholding activation functions of the spiking neurons [8], [9], [10], [11]. However, they don't address how biology can realize the computation of gradients and their access to the neurons involved in the computation. Furthermore, we don't have evidence on how batching of data happens in biology, and most of the gradient descent approaches rely on batching the data. Despite the lack of bio-plausibility, the error back-propagation based approaches have been adopted in computational neuroscience as useful alternative tools to discover the required connectivity in SNNs for a given specific task [12], [13], [14].

Feedback alignment has been used as an alternative to error back-propagation for SNNs in [15] to solve the 'weight transport' problem. Feedback alignment shows that multiplying errors by random synaptic weights is enough for effective error back-propagation without requiring a precise symmetric backward connectivity pattern. There have been parallel investigations of more bio-plausible local learning rules for SNNs which do not require access to the weights of other neurons in the network. This set of learning rules for SNNs can be characterized as synaptic plasticity rules which use Spike-Timing-Dependent Plasticity (STDP) in some form. STDP rules have more commonly been used for extracting features from spike trains in an unsupervised manner. There have also been probabilistic [16] and reinforced [17] variants of STDP to learn discriminative features which can assist classification. STDP rules are also commonly used along with WTA rules to promote competition among neurons and reduce the redundancy in the learnt features [18]. Paredes-Vallés *et.al.* [19] used a homeostasis parameter that controls the excitability of neurons in combination with STDP and WTA rules to promote stability. Supervised Hebbian Learning (SHL) [20] and ReSuMe [21] were one of the first works to use STDP rules to perform supervised learning in single layer SNN networks. ReSuMe implements a spiking version of Widrow-Hoff rule [22] for rate-coded SNNs and using STDP and anti-STDP processes. Spike Pattern Association (SPAN) [23] is another learning method that adopted the Widrow-Hoff rule for spike sequence learning. [24] extend the ReSuMe to multiple layers and approximate the gradient descent update step of the intermediate layers to an STDP process. Taherkhani *et.al.* [25] proposed another modification to the ReSuMe rule to extend it to multi-layer SNNs.

Another set of learning algorithms that gained momentum recently is applying evolutionary methods for SNN optimization. Some works have been solutions for SNN network structure optimization [26], [27], [28], [29] and others have been for synaptic weights optimization [30], [31], [32]. Apart from these broader themes, there have been other works that used practical mathematical solutions to the learning problem. For instance, SpikeTemp [33] used rank-order based learning for SNNs. Multi-Spike Tempotron (MST) [34] used a technique called aggregate label learning that can learn predictive cues or features. MST was followed up with a computationally simpler version called Threshold-Driven Plasticity (TDP) [35]. Both MST and TDP update weights by calculating the gradients with respect to the threshold of a neuron and finding the optimal threshold for a desired number of spikes from a neuron. Membrane Potential Driven Aggregate Learning (MPD-AL) [36] was proposed as an alternative version of aggregate learning which used gradients with respect to the membrane potential. The role of neuromodulators in synaptic learning was explored through three-factor learning rules in [37], [38], [39], [40], [41]. The key motivation to use three-factor learning rules is to extend the functionality of STDP beyond unsupervised learning, which by design, neglects any information regarding "reward", "punishment," or "novelty" during learning [42].

Alongside the regular SNN architectures, with the recent popularity of neuromorphic vision sensors [43], multiple event-based neural architectures have been proposed [44], [45]. Afshar *et.al.* [46] proposed Feature Extraction using Adaptive Selection Thresholds (FEAST) as an unsupervised feature extraction algorithm for event-based data using exponential time surfaces and spiking neuron-like units which have individual selection thresholds. In the FEAST algorithm, each feature unit is represented by a weight vector for all its inputs and a selection threshold that facilitates equal activation of the features during online learning. In this paper, we propose to use these adaptive selection thresholds for multi-layer supervised learning and propose our method as a simple solution to the credit assignment problem. Our proposed method is the first to not require the transport of weights across neurons in the network or random connections for error propagation. We achieve all feedback to earlier layers using precisely timed binary attention signals which signal "reward" and "punishment" of the recently active neurons.

II. BACKGROUND AND RELATED WORK

A. TIME SURFACES

Tapson *et al.* [47] proposed the use of exponentially decaying kernels for processing event data produced by neuromorphic vision sensors. We shall use the terminology introduced by Lagorce *et al.* [45] and refer to these kernels as time surfaces. The time surface is the trace of events per each input channel which is updated only when an event arrives at a channel. An event from an event-based sensor can be described as:

$$e_i = (x_i, y_i, t_i, p_i) \tag{1}$$

Equation (1) describes an event from a pixel location (x_i, y_i) as the coordinates on the sensor, with polarity p_i, arriving at time t_i. A similar notation can be used to represent a spike as an event:

$$s_i = (c_i, t_i) \tag{2}$$

Equation (2) represents a spike s_i from channel c_i at time t_i. We can use the time surfaces to keep the trace of spikes from a spiking source.

The exponential time surface $S_t[i]$ of a channel i at time t with a time constant of τ can be calculated as follows:

$$S_t[i] = P[i] * e^{\frac{-(t-TS[i])}{\tau}} \tag{3}$$

where $P[i]$ in Equation (3) is the last updated potential of the channel i and $TS[i]$ holds the timestamp of the last spike to have occurred at the channel i. If a new spike (i, t_i) occurs at the channel i then the potential of the channel $P[i]$ is updated according to Equations (4) and (5)

$$\underset{new}{P[i]} = \underset{old}{P[i]} * e^{\frac{-(t_i-TS[i])}{\tau}} + c \tag{4}$$

$$\underset{new}{TS[i]} = t_i \tag{5}$$

where c is the constant by which the potential is increased for each new spike at a channel. c is generally set to 1.

This formulation of time surfaces for spikes is simply a reformulation of the Excitatory Post Synaptic Potential due to an input spike to a biological neuron as described in the Steins model [48], [49], which jumps by an amount w_k on the arrival of a spike and decays exponentially thereafter.

$$\Delta u(t) = w_k * \epsilon(t - t_k^f) \tag{6}$$

$$\epsilon(t) = e^{\frac{-t}{\tau_m}} \tag{7}$$

A neuron n can then be parameterized by a weight vector W_n representing the synaptic weights to all the input channels N_c and a spiking threshold θ_n. The total contribution to the membrane value v_n of the neuron n by all input spikes arrived up until a time can be formulated as the dot product of the normalized time surface of all channels and the normalized weight vector. The normalization of both the potential and the weights ensures the membrane value v_n is between 0 and 1. The membrane value v_n is calculated according to Equations (8) and (9). C in Equation (8) represents the time surface context which is the normalized time surface of all input channels. The time surface context C represents the recent activity from all the input channels.

$$C = \frac{S}{\|S\|} \tag{8}$$

$$v_n = \underset{(1\times N_c)}{W_n} \cdot \underset{(1\times N_c)}{C} \tag{9}$$

Afshar *et al.* [46] introduced an algorithm to extract features from event-data using layers of such neuronal units in an unsupervised manner. In addition to the weights which represent the features, each neuron has a threshold as an additional parameter. For every input event, the dot product (v_n) of the time surface context (C) and the weight (W_n) of a neuron is calculated. The dot products of all neurons are then compared to their respective thresholds. Out of the neurons with the dot products greater than their respective thresholds$(v_n \geq \theta_n)$, the neuron with the largest dot product is considered the winner for the given input event. If none of the dot products crosses their respective neuron thresholds, the thresholds of all the neurons are reduced by a pre-defined fixed value. On the contrary, if a matching neuron is found, the feature/weight vector W_n is updated with the current event context C using an exponential moving average, and the threshold of the neuron is increased by a fixed value. The thresholds and weights of other neurons are left unchanged.

The adaptation of the selection thresholds promotes homeostasis and facilitates equal activation of the feature neurons in response to the data. This Feature Extraction using Adaptive Selection Thresholds (FEAST) is an online learning method that clusters the incoming event contexts of all the events into N clusters where N is the number of neurons used in a FEAST layer.

Lowering the threshold of a feature neuron makes that neuron more receptive to new event contexts, whereas increasing the threshold makes a feature neuron more selective. FEAST treats each incoming event with equal priority as there is no information regarding the importance of individual events. This results in the features representing the most commonly observed spatio-temporal patterns in the input data. However, learning the most commonly occurring features may not be ideal for tasks that depend on rarer task-specific features.

B. ABSTRACTION OF SPIKING NEURAL NETWORKS

The space of possible SNN architectures can be characterized by the different models of neurons, synapses, learning rules, and network architectures used in them. In this space, there is often a trade-off between the biological plausibility and practical applicability of the models. Network models attempting to demonstrate biological plausibility through detailed phenomenological modelling from the voltage-gated ion channels to the delays at the neuronal synapses tend to be limited in their performance and utility in the context of challenging machine learning tasks. The computational cost of these models increases with the bio-plausibility of the model. Different neuronal models have been proposed which approximate and abstract the details of these complexities with easy-to-handle mathematical and probabilistic models [50], [51], [52], [53]. Leaky-Integrate and Fire (LIF) neuron model [49] and specifically the Spike Response Model (SRM) [54] are among the most popular choices of neuron models in the SNNs, even though the degree to which they explain the neuronal dynamics is limited compared to other models like Hodgkin-Huxley [55] or Izhkevich [52] models. Their vast adoption can be attributed to their analytical tractability and computational simplicity compared to other neuronal models. But even the SNN models which use simpler neuronal models like LIF or Adaptive Leaky-Integrate and Fire (ALIF) neurons [56] require additional complexities such as Excitatory-Inhibitory Balance, and the right amount of lateral excitation and inhibition to instil behaviours like WTA. These complex processes make it difficult to scale up the simulations of the multi-layered SNNs and limit the exploration of broader system-level learning mechanisms of the SNNs as there are a lot of variables in the system. In the same way, that time surfaces represent simplified hardware friendly abstractions of the EPSP, the FEAST network can be best understood as a highly abstracted, functionally equivalent, modular implementation of a well-balanced excitatory SNN with inhibitory feedback leading to a winner take all operation at a single layer. In this way, a FEAST layer represents a neuron group. Picking only one winner in each layer of FEAST for any input event is a proxy for hard WTA motif in a neuron group, without requiring any forms of inhibition. Simpler and computationally easier abstract SNN models like FEAST can help us explore more system-level learning rules in SNNs without having to worry about problems like achieving EI balance and promoting or removing oscillations in the networks. Just like Address-Event Representations (AER) being used in Neuromorphic hardware to facilitate the communication in SNNs, we can use novel abstractions like FEAST to explore the space of local learning rules in Spiking Neural Architectures. Continuing in this approach and extending it, the Optimized Deep Event-driven Spiking neural network Architecture (ODESA) introduced in this paper, represents a method to locally train hierarchies of well-balanced EI networks on event-based data in a supervised manner. In this way, the abstracted SNN which ODESA represents can be used to rapidly explore a wide range of multi-layered SNN models for real-world online supervised learning applications.

III. MATERIALS AND METHODS

The aim of ODESA is to use a multi-layered spiking neural architecture and train it to map any input spatio-temporal spike pattern X_{train} to any output spatio-temporal spike pattern Y_{train} and to do so entirely using binary signals, and without having access to the weights of other neurons or batching of input data. The latter restriction not only makes ODESA a useful framework for studying local learning biological SNNs but also allows local online training in neuromorphic hardware implementations of such networks. The ODESA architecture can contain multiple hidden layers with different time constants to learn hierarchical spatio-temporal features simultaneously at different timescales to support an output layer consisting of classification readout neurons which generate the desired spike in Y_{train}.

A. CLASSIFICATION USING ADAPTIVE SELECTION THRESHOLDS

The output classification layer in ODESA has $k * N_c$ neurons ($k = 1, 2, 3, \dots$) for a classification task with N_c classes. The output layer is divided into N_c groups, each responsible for spiking for their respective classes. For any given input spike to the layer, only one neuron (out of $k * N_c$ neurons) can spike.

The threshold adaptation in ODESA's output layer is driven by the supervisory spike signal Y_{train} for a given input spike stream X_{train}. Considering that ODESA is event-driven, it is assumed that there exists an input spike i_t in X_{train} at time t for every output spike o_t in Y_{train}. The labelled input spike i_t in X_{train} which has an output label spike o_t associated with it, is treated with additional attention. For the labelled input spike i_t, if there is no spike from the respective class neuron group responsible for the current class of the supervisory spike o_t, the thresholds for all the neurons in the class group are lowered. If there is a spike from any of the neurons in the class group, the winner neuron's weights are updated with the input spike's event context, and its threshold is updated. Alternatively, in the absence of a spike from the correct class group, the thresholds of all the neurons in the group are reduced. This weight update and threshold increase in a neuron can be thought of as 'rewarding a neuron' for its correct classification. A decrease in the threshold of a neuron to make it more receptive can be viewed as 'punishing a neuron' for not being active.

The threshold increase step in ODESA is different from that proposed in Afshar *et al.* [46]. Rather than the fixed value used in the previous work, it is an adaptive value that depends on the dot product of the context with the weight according to Equations (15) and (16). The $\Delta\theta_n$ in Equation (15) is never negative because for any winner neuron n, v_n is always at least as high as θ_n to win. This new threshold adaptation ensures that threshold (θ_n) of a neuron moves asymptotically towards, but never reaches 1 as the model gets better at classification. This modification speeds up the threshold adaptation operation, while simultaneously improving the stability of the system by having a proportional increase in the threshold based on the membrane value (v_n) at the time of winning an input spike. Equations (10) to (16) show the weight and threshold adaptation of a winner neuron n if it belongs to the correct class group.

$$C = \frac{S(i_t)}{\|S(i_t)\|} \tag{10}$$

$$v_n = W_n \cdot C \tag{11}$$

$$\Delta W_n = C - W_n \tag{12}$$

$$W_n = W_n + \eta * \Delta W_n \tag{13}$$

$$W_n = \frac{W_n}{\|W_n\|} \tag{14}$$

$$\Delta\theta_n = v_n - \theta_n \tag{15}$$

$$\theta_n = \theta_n + \eta_{thresh} * \Delta\theta_n \tag{16}$$

Equation (17) shows the punishment of neuron group which has not spiked by lowering the thresholds of the group. Θ_{label} represents the thresholds of all the k neurons in the class group that corresponds to the label of the input spike. θ_{drop} is the fixed value by which the thresholds are reduced. Figure 1 shows the geometric representation of the reward and punishment of the neurons.

$$\Theta_{label} = \Theta_{label} - \theta_{drop} \tag{17}$$

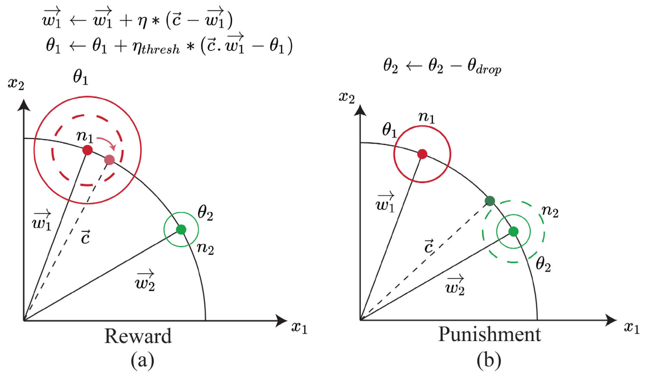
$\vec{w_1} \leftarrow \vec{w_1} + \eta * (\vec{c} - \vec{w_1})$
$\theta_1 \leftarrow \theta_1 + \eta_{thresh} * (\vec{c}.\vec{w_1} - \theta_1)$
x_2
θ_1
n_1
$\vec{w_1}$
$\vec{c}$
θ_2
n_2
$\vec{w_2}$
x_1
Reward
(a)
$\theta_2 \leftarrow \theta_2 - \theta_{drop}$
x_2
θ_1
n_1
$\vec{w_1}$
n_2
$\vec{c}$
θ_2
$\vec{w_2}$
x_1
Punishment
(b)

FIGURE 1. For a sample input space with 2 dimensions, the neurons (n_1 and n_2) can be represented by their unit weight vectors ($\vec{w}_1$ and $\vec{w}_2$) and their thresholds (θ_1 and θ_2). The input time surface context is also represented as a unit vector $\vec{c}$. The solid circles represent the original thresholds and dashed circles around the neurons represent the updated thresholds. The red and green colours represent different output classes/labels. (a) If the winner neuron n_1 belongs to the same class as the label of input context $\vec{c}$, then the weight w_1 is moved towards the context $\vec{c}$ and threshold θ_1 is increased proportionally (b) If the neuron n_2 for the corresponding class of an input context $\vec{c}$ does not spike, then the neuron is punished by dropping its threshold θ_2 and making it more receptive.

The WTA constraint in the ODESA layers ensures that there can be at most one winner neuron for each input spike to a layer. This creates competition between the neurons to capture regions of the input space that precede output labels. In this way, neurons in each layer attempt to only learn the spatio-temporal features that are crucial in discriminating one class from another class ensuring that the neuron groups don't learn features that are common between two different classes.

The rewarding and punishing of neurons based on their activity with respect to the label spike stream is the key element of learning in ODESA.

B. MULTI-LAYER SUPERVISION THROUGH SPIKE-TIMING-DEPENDENT THRESHOLD ADAPTATION

Decaying event kernels, time surfaces, and the EPSPs they represent are all imperfect as memory units since they can map a wide range of spike trains onto the same analog value at a channel, thus losing potentially critical information, especially when there are multiple spikes per input channel. This poses a serious problem for real-world tasks which depend on features that occur at multiple timescales and which generally contain an arbitrary number of information-carrying spikes per channel. Furthermore, in most real-world tasks, a shared collection of low level features, when combined in varying ways in time and feature-space, maps the input data to the desired output. Thus, hierarchical layers are often required to solve complex tasks that require associations at different feature levels and timescales.

Multiple spiking layers, however, pose a new problem to learning not present in simple one layer networks such as FEAST. This is a problem of credit assignment. Because the thresholding operation in SNNs is non-differentiable, other works have used various versions of gradient approximation and Back-Propagation Through Time (BPTT), [8], [9], [10], [11]. In contrast, ODESA solves this problem without the use of gradients, by using the activity of the next layer as the supervisory signal for the current layer.

Each neuron in the hidden layer of an ODESA network has a trace of its latest activation. Equation (18) describes the trace $\Lambda^l_t[n]$ of a neuron n in layer l at time t. Just like time surfaces, updates of the trace of a layer (Λ^l) are event-driven. t_n is the time of neuron n's most recent spike. The trace $\Lambda^l_t[n]$ acts as a measure that indicates the neuron n's recent activity at any time t. The trace of a layer Λ^l_t is used to find the neurons that participated in generating a spike in the next layer and reward or punish them accordingly. The time constant of the trace Λ^l is equal to the time constant of the neurons in its next layer τ_{l+1}, whereas the time constant τ_l for neurons in a layer l is used to decay the inputs to layer l via time surface S^l. Thus in general, the time constant of layer l's trace τ_{l+1} is not equal to that layer's time surface time constant τ_l. In our experiments, we have used the same time constant for all the neurons in a layer. When any neuron n_{l+1} in the next layer spikes (post-synaptic spike) for an input spike (pre-synaptic spike) from the current layer l, a local binary attention signal $A^{l+1}[i]$ is passed to the current layer l from the next layer $l+1$ indicating activity in the next layer. The current layer l uses the local attention signal to reward its recently active neurons (whose trace $\Lambda^l_t[n] \geq \Phi$) and punishes its inactive neurons ($\Lambda^l_t[n] < \Phi$) where Φ is the trace recency threshold. In this work, an arbitrarily chosen trace recency threshold value of $\Phi = 0.1$ is used throughout. The reward and punishment of the neurons in this layer are the same as described in Section III-A.

$$\Lambda^l_t[n] = e^{\left(\frac{-(t-t^f_n)}{\tau_{l+1}}\right)} \tag{18}$$

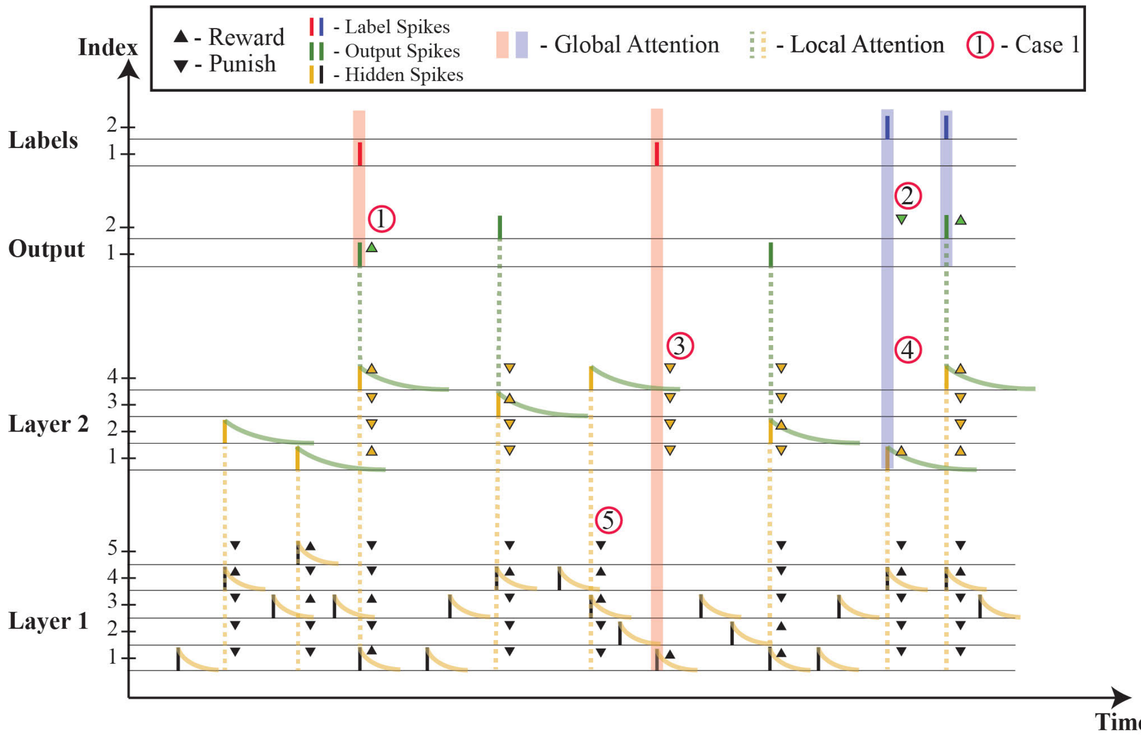

FIGURE 2. **Multi-layer supervision in ODESA using spike-timing-dependent threshold adaptation. The shaded vertical lines represent the binary global attention signal generated for each output label spike, and the dotted vertical lines represent the binary local attention signals sent to each layer from its next layer. The up and down arrows represent the reward and punishment of the individual neurons. Case 1: The predicted output spike matches the label spike, and the corresponding output neuron is rewarded. Case 2: The corresponding output neuron for the correct class is punished as it failed to spike in the presence of input from Layer 2. Case 3: All neurons in Layer 2 are punished as they failed to spike for an input spike from Layer 1 in presence of the global attention signal. Case 4: The active neuron in Layer 2 is rewarded in presence of the global attention signal. Case 5: The neurons with trace above the recency threshold are rewarded and the other neurons are punished in the presence of local attention signal from Layer 2.**

This local attention signal driven reward mechanism forces the neurons in each layer to learn features that best support the activity in the next layer. The neurons in the next layer in turn compete to support the activity of neurons in the following layer and so on. The last hidden layer of the network is rewarded by the output layer, and the output layer is rewarded by label spikes which are events that carry the ground truth labels. The supervision of the output layer can be considered the same as the supervision applied to the hidden layers but with a trace that decays instantaneously, i.e., $\tau_{o+1} \rightarrow 0$. The instantly decaying trace ensures the rewarding of the output layer only when it generates a spike precisely at the time of the actual label spike. The post-synaptic spike-timing-dependent threshold adaptation described is the key element to the learning in multi-layered ODESA. This threshold adaptation mechanism is an additional dimension to the learning process apart from the usual synaptic weight adaptation used in STDP-based SNNs. It helps in regulating the spike activity and utilises all the neuron resources available by promoting equal activation of all the hidden neurons involved in generating an expected output spike. Thus providing the required behavioural complexity needed to solve the credit assignment problem over multiple layers.

Since the ODESA network is event-driven, if a hidden layer l fails to spike for an input spike there will be no spike generated at the output layer that can be used for training. Therefore, in addition to the local attention signal $A^{l+1}[i]$, a Global Attention Signal $G[i]$ is also generated for every labelled input spike. Every layer is expected to spike for a labelled input spike such that an output spike can be generated. Hence, all the neurons in the silent layer l that failed to spike in presence of the Global Attention Signal and an input spike from its previous layer are punished. This rule isolates the layer where the failure to transmit spikes took place and punishes the neurons in that layer by making them more receptive. Additionally, the neurons in every layer which were active during the global attention signal are rewarded. In this way, every layer in a well trained network will generate a spike whenever an output spike is expected. Figure 2 shows the spike-timing-dependent threshold adaptation for a sample network activity. The exponential kernels show the trace of the spikes at each layer and the dotted lines show the local attention signals a layer passes to the layer below it when a spike is generated by one of its neurons. The upward arrows indicate the neurons that are rewarded and the downward arrows indicate the neurons that are punished. The red and blue shaded zones indicate the global attention signals generated when a labelled spike arrives.

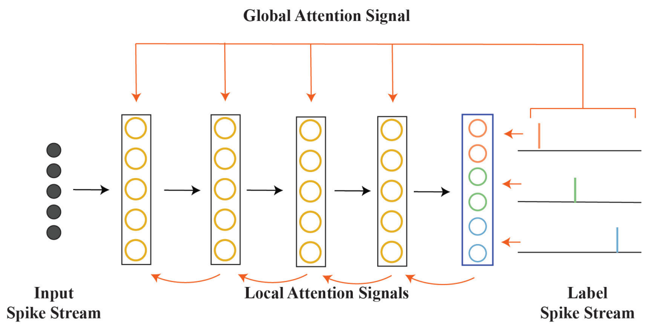

FIGURE 3. Overview of the supervision in multi-layered ODESA.

C. OVERVIEW OF LEARNING RULES IN ODESA

Figure 3 shows the operation of a multi-layered ODESA network. The Global Attention Signal $G[i]$ is a binary event that indicates the presence of a labelled spike. The first layer l in the network that stays inactive (Case 3 in Figure 2) while having an input spike from the layer below it $l-1$ is punished in the presence of the Global Attention Signal. The Local Attention Signals $A^{1:L}[i]$ are also binary events that indicate the activation of the next layer to the current layer. The recently active neurons ($\Lambda_t^l[n] \geq \Phi$) in the previous layer are rewarded (weight update and threshold adaptation as in Equations (13) to (16))) and the inactive ones are punished (threshold decreased as in Equation (17)). This supervision is different to traditional back-propagation techniques which require access to all the higher layers' activity to find the gradients for a given layer. ODESA training continues even when one of the later layers in the network goes silent. This spike-timing-dependent threshold adaptation equips ODESA with the ability to learn features simultaneously at all layers irrespective of the other layers' states.

D. ADDITIONAL OUTPUT LAYER ADAPTATION

Along with the threshold adaptation described in Section III-A, additional weight update steps were investigated to speed up the convergence of learning. The first addition is the use of negative weight updates for misclassified spikes. If a neuron n in the output layer belonging to a different class group than the label class group spikes for an input, the weights of the neuron are updated using a negative weight update according to Equation (19).

$$W_n = W_n - \eta * \Delta W_n \tag{19}$$

The second additional weight update step investigated was rewarding the closest neuron in the label class group when there is no winner from this group. The closest neuron is the neuron with the highest dot product value (v in Equation (9)) among the neurons in the group corresponding to the label. The weights and threshold of the closest neuron c with the highest dot product value among the label class group is

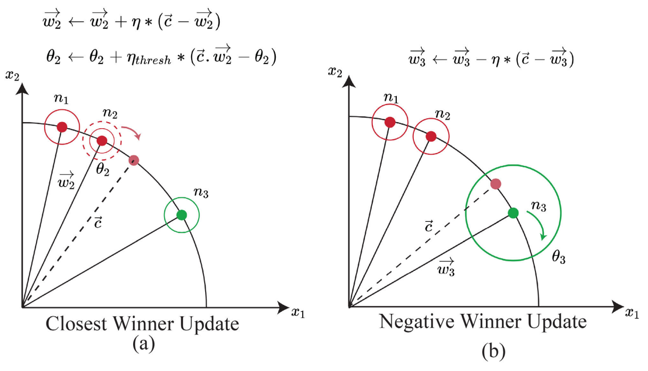
$\vec{w_2} \leftarrow \vec{w_2} + \eta * (\vec{c} - \vec{w_2})$
$\theta_2 \leftarrow \theta_2 + \eta_{thresh} * (\vec{c}.\vec{w_2} - \theta_2)$
x_2
n_1
n_2
n_3
$\vec{w_2}$
θ_2
$\vec{c}$
x_1
Closest Winner Update
(a)
$\vec{w_3} \leftarrow \vec{w_3} - \eta * (\vec{c} - \vec{w_3})$
x_2
n_1
n_2
n_3
$\vec{c}$
$\vec{w_3}$
θ_3
x_1
Negative Winner Update
(b)

FIGURE 4. For a sample input space with 2 dimensions, the neurons (n_1, n_2, and n_3) can be represented by their unit weight vectors ($\vec{w_1}$, $\vec{w_2}$ and $\vec{w_3}$) and their thresholds (θ_1, θ_2 and θ_3). The input time surface context is also represented as a unit vector $\vec{c}$. The solid circles represent the original thresholds and dashed circles represent the updated thresholds. The red and green colours represent different output classes. The colour of the context vector $\vec{c}$ represents the label. (a) The closest neuron n_2 in the correct class group is pushed towards the current input context and rewarded with it. (b) The neuron n_3 which spiked for the context $\vec{c}$ of a different class is pushed away from it by applying negative weight update.

updated according to Equations (20) to (21). This step is analogous to pulling the thresholds of the neurons in the label class group down until one of them spikes, and rewarding the first neuron that spikes. Figure 4 shows the geometric representation of the two additional adaptations.

$$W_c = W_c + \eta * \Delta W_c \tag{20}$$
$$\theta_c = \theta_c + \eta_{thresh} * \Delta\theta_c \tag{21}$$

The adaptation discussed in Section III-A along with the above two weight update steps together gave the best performance across all the tasks. The advantages in performance due to these weight update steps are discussed in Section V-A

IV. RESULTS

We tested ODESA on different spatio-temporal transformation and classification tasks. The method was tested on a random pattern association task where random input spatio-temporal patterns were mapped to a target output spike stream. The network can simultaneously learn a hierarchical representation of the input patterns using an optimal number of neurons at each layer. Traditional machine learning datasets can be converted to spiking dataset using a range of different techniques like rate coding, population encoding, intensity to latency encoding, etc. We used a variety of encoding techniques to convert popular machine learning datasets to spike-based datasets and evaluated ODESA on them. ODESA was tested on the IRIS dataset converted to precise temporal coded spike patterns using population coding proposed by Bohte *et al.* [5]. We also tested it on the Oxford spike pattern which was used to demonstrate the capabilities of SuperSpike [11], as well as the latency-coded MNIST dataset. We then show the capabilities of the architecture by testing it on more complex problems like decoding

Morse Code sequences and spoken digit classification using spikes from a Cochlea model.

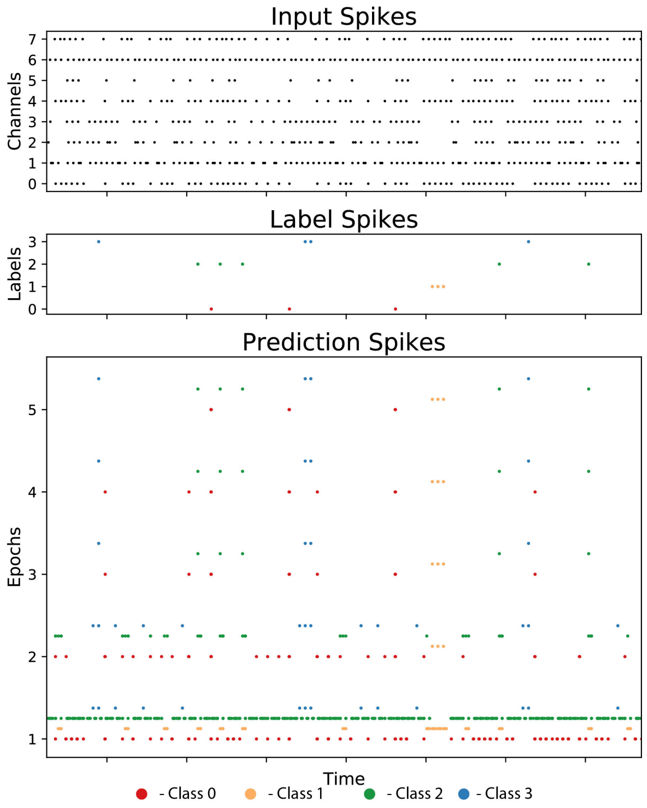

FIGURE 5. Evolution of output layer activity for the random pattern association task. Embedded in the input spike patterns are unique sequences of symbols made of unique random patterns. The label spikes denote the desired output spike pattern for the given input spike pattern. The prediction spikes show the change in the activity of the 4 output neurons across 5 epochs.

A. RANDOM PATTERN ASSOCIATION TASK

We first tested ODESA on a random pattern association task. Three random symbols (A, B, and C) were generated with 8 input channels that had a variable number of spikes per channel (6-8 spikes per symbol). We picked four target sequences randomly which are a combination of the constituent symbols (e.g., BBA, ACB, CAC, CCC). A stream of random combinations of the symbols ("….B A A C B B C…") with no gap in between is presented to a two-layered ODESA network. A global attention signal is raised indicating the class at the last spike of each picked target sequence if and when they occur in the stream. The goal of the experiment is for the output layer of the ODESA network to generate spikes precisely at the time of the target label spikes. The network starts with spiking for every input event and gets rewarded every time it gets one of the label spikes right. The rewarded neurons get more specific in their spiking with time, and the neurons in the hidden layer get more specific in the features that occur just before the label spikes. Figure 5 shows the evolution of the output layer's activity across 1 to 5 epochs for a given random pattern association task.

B. OXFORD SPIKE PATTERN

We also tested how the ODESA architecture can be adapted to other types of spike prediction problems such as the Oxford spike pattern, which was used to showcase the performance of the SuperSpike algorithm [11] in their supplementary code repository.[1] The Oxford spike dataset consists of an input spike train, and a target spike train. The input spike train consists of random spikes generated in 200 channels over a period of 1.89 seconds. The target spike train is an image of a building that has been converted to a spatio-temporal spike train over 200 channels and 1.89 seconds. The task is to predict the precisely timed target spike train based on the random, but fixed, input spike train. This is very similar to the random pattern association task in the previous experiment, but with higher dimensions and without any inherent sub sequences in the input spike train. It is different from other datasets we tested ODESA in this work, due to the need to generate multiple output spikes per input spike. So the ODESA output layer was slightly modified to accommodate this requirement. We removed the hard WTA step in the output layer of the network and allowed multiple neurons to spike as long as their membrane values crossed their respective thresholds. The threshold adaptation of the output layer remained unchanged, and whenever a neuron group responsible for an output spike failed to spike, the thresholds were reduced. Neurons that correctly predicted the target spikes were rewarded. Also, as the ODESA architecture in its current form does not have delays in its synapses, at least one input spike should exist for every label spike. To facilitate this, we modified the target by mapping each spike in the output spike train to the nearest spike in the random input spike train. As the input spike train was fairly dense, the structure of the overall image did not change much from this process, as can be seen from the side by side comparison of target spikes in Figure 6.

Figure 6 shows the prediction from a single layer ODESA network and a two layer network trained with SuperSpike. The SuperSpike algorithm [11] uses an error measure that is a function of temporal difference between the predicted spike train and the target spike train. However, as ODESA makes predictions for each input spike, we had to create a different evaluation metric to monitor the training. For each input spike, we calculated the Intersection Over Union (IOU) over the sets of target spikes and predicted spikes. A mean IOU (between 0 and 1) per input spike was then calculated per epoch to evaluate the algorithm. With one output layer and no hidden layers, ODESA could achieve a mean IOU score of 0.80 for the Oxford spike pattern. The number of neurons per each correct class group (k) was proportional to the number of spikes in each target class. Though the receptive fields of neurons in ODESA can accommodate local temporal jitter, ODESA primarily learns patterns through clustering similar input patterns together. Therefore, it requires more than one neuron for each target class in the output layer to solve a problem like the Oxford spike pattern, as most of the features

[1]https://github.com/fzenke/pub2018superspike

at each input spike are uncorrelated and unique. Two-thirds of the number of unique spikes in each target channel was used as the k (number of neurons in each class group) value for that class group in the final network. That was the minimum number of neurons required in the output layer to predict most of the output spikes in our experiments. Section V-B discusses the performance of ODESA with different network depths on the Oxford spike pattern.

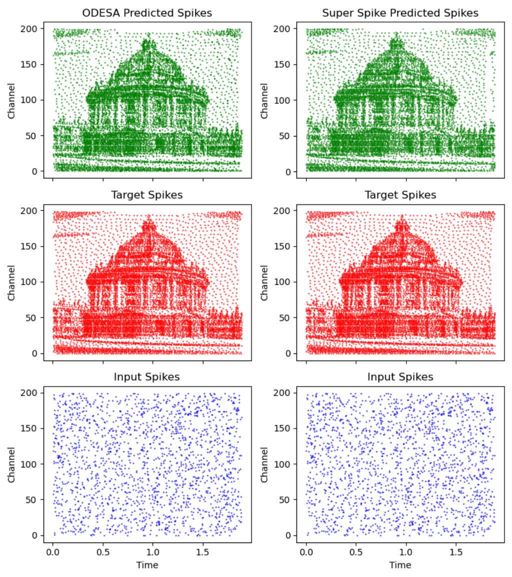

FIGURE 6. Prediction of Oxford spike pattern using ODESA and the original SuperSpike algorithm using symmetric feedback.

C. IRIS DATASET

To compare with other supervised learning methods on SNNs, we followed the same biologically plausible encoding scheme of traditional machine learning datasets as first proposed in Bohte *et al.* [5]. The Fisher's IRIS dataset contains 3 classes with 50 samples each and is known for having linearly non-separable classes. Each sample has 4 input features, and $m = 5$ gaussian receptive fields per input feature were used to convert the features into $4 \times 5 = 20$ spiking channels. We used $\beta = 1.5$ just like the original work [5] where it was first used. We added no additional input channels to spike at intermediate intervals like used in [25]. Each input spiking channel emits a spike only once in the time between $t = 0$ and $t = 1$ secs per example. A simple 2-layer ODESA network was used with the hidden layer having time constant of $\tau_1 = 0.6\,sec$ and an output layer with time constant of $\tau_2 = 0.9\,sec$. The hidden layer had 10 neurons and the output layer had 1 neuron per class. We evaluated the network by performing 2-fold and 4-fold cross-validation. We compared the results with other supervised learning algorithms in SNNs and some non-spiking methods for reference in Table 1. We can see that ODESA can achieve comparable performance in the task with significantly fewer neurons and trainable parameters than other SNN training algorithms.

TABLE 1. Comparison of ODESA network on Fisher's IRIS Dataset with other methods.

Method	Inputs	Hidden	Outputs	Iterations	Train-Test Split	Testing Accuracy
Spiking Methods						
RBF [57]	32	-	3	-	50%-50%	92.6%
SWAT [58]	40	208	3	500	50%-50%	95.3%
SpikeProp [5]	50	10	1	1000	50%-50%	96.1%
QuickProp [59]	50	10	1	100-200	50%-50%	92.3%
RProp [59]	50	10	1	Less than 100	50%-50%	93.2%
RBF [60]	-	-	3	-	50%-50%	89.0%
Bako et al. [24]	52	-	3	1000	50%-50%	83.4%
TEM [61]	48	-	3	Less than 100	50%-50%	92.5%
Sporea et al. [24]	48	200-300	3	100	75%-25%	94.0%
Taherkhani et al. [25]	169	360	3	100	75%-25%	95.7%
Proposed Method	20	10	3	200	75%-25%	95.6%
Proposed Method	20	10	3	200	50%-50 %	95.1%
Non-Spiking Methods						
K-Means [57]	-	-	-	-	50%-50%	88.6%
SOM [57]	-	-	-	-	50%-50%	85.3%
MATLAB BP [58]	50	10	3	2.6×10^6	50%-50%	95.7%

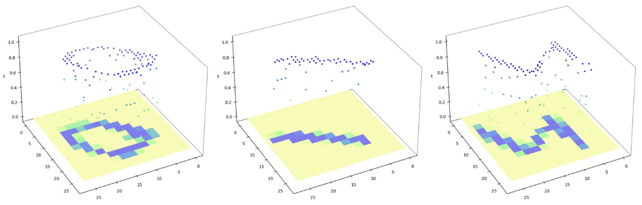

FIGURE 7. Examples of classes '0','1',and '5' of latency coded spikes from MNIST.

D. LATENCY-CODED MNIST

MNIST [62] consists of 60,000 training images belonging to 10 classes and 10,000 testing images. Each image is of size 28×28 which makes the input channels equal to 784. The MNIST images were preprocessed using latency coding to convert them into a spiking dataset. The brightness value of each pixel was linearly transformed from 0-255 to 0-1 seconds which was used as the timing of a spike from the corresponding input channel. All the spikes from pixels that had a timestamp of less than 0.3 seconds were eliminated to reduce the number of input spikes. As ODESA expects a precisely timed label spike, at the end of each example we generated a labelled spike that denotes the class of the example. The latency coding ensures that the input time surface context at the end of the example is similar to the original MNIST image. Figure 7 shows a few examples of input spikes generated this way. We have used an ODESA network with 1 hidden layer (6000 units) and 1 output layer ($k = 10$) to train on the latency-coded MNIST dataset to achieve a test accuracy of 93.23%. The time constants for each layer τ_1 and τ_2 was set to $1.0\,sec$ each. We can easily visualise the features of time surface contexts learnt by the neurons in the hidden layer by simply plotting the heat map of the weights like shown in Figure 8 (a).

We can see that different neurons have learnt different intensity regions of the digits in MNIST, as the respective spikes fall close to each other temporally. We can also approximately estimate the patterns learnt by later layers by multiplying the weight matrices of a layer with its previous layer and so on. For example, in a two layer ODESA network, the composite weights of 2nd layer can be estimated by a simple matrix multiplication (Equation (22)). Where W_2 is the weight matrix of the 2nd layer, and W_1 is the weight matrix of the 1st layer. n_i, n_1, and n_2 represent the number of input channels, the number of neurons in 1st layer, and the number of neurons in 2nd layer respectively. It should be noted that this is only a limited 2D visualisation of the higher dimensional spatio-temporal feature learnt by the output neurons. Estimating the original time surface patterns which trigger the neurons is not easily tractable due to the weight normalisation and the non-linear thresholding operation.

$$\underset{(n_2\times n_i)}{W_2^{comp}} = \underset{(n_2\times n_1)}{W_2} \cdot \underset{(n_1\times n_i)}{W_1} \tag{22}$$

Figure 8(b) shows the composite weights of the output layer of ODESA network trained on MNIST. Each row in Figure 8(b) represents the k neurons in each class. The composite weights show that each neuron in an output class group is learning a different cluster of patterns for the class. Table 2 shows the comparison of the test accuracy with various other SNN methods. ODESA performs on par with all the STDP-based methods without requiring an additional classifier at the end.

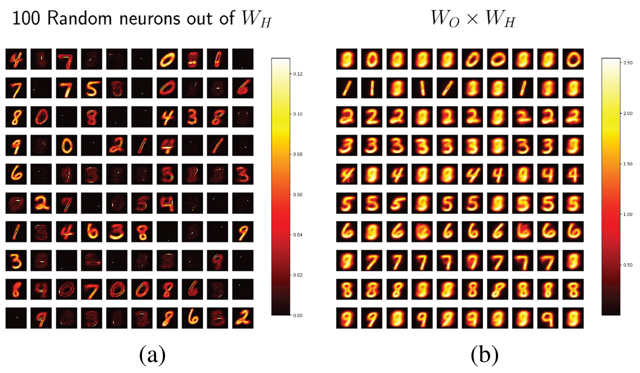

FIGURE 8. (a) Weights of 100 neurons randomly picked from hidden layer. (b) Composite weight visualisation of output neurons by multiplying the weight matrices of hidden layer and output layer. Each row represents the weights of neurons (k = 10) in each class group.

TABLE 2. Comparison of accuracy on the test set of MNIST with other SNN methods.

Method	Pre-Processing	Test Accuracy
Two Layer with Exponential STDP + *a posteriori* neuron selection [63]	Rate Coding	95.00%
Spiking RBM + Contrastive Divergence, Linear Classifier [64]	Thresholding + Rate coding	89.00%
Spiking RBM + Contrastive Divergence [65]	Thresholding + Spike Coding	91.90%
Dendritic neurons + Morphology Learning [66]	Rate Coding	90.30%
Multi-layer hierarchical network + STDP with Calcium variable [67]	Orientation Detection + Spike Coding	91.60%
Spiking CNN + ANN to SNN Conversion [68]	Rate Coding	98.20%
Spiking CNN + Tempotron Rule [69]	Scaling, Orientation, Thresholding + Spike Coding	91.30%
Three Layer SNN + STDP-Backpropagation [70]	Rate Coding	97.20%
Deep SNN + Backpropagation [71]	Rate Coding	98.60 %
Two Layer SNN + SGD [72]	Latency based Spike Encoding	96.92%
ODESA (Our method)	Latency based Spike Encoding	93.24%

E. INTERNATIONAL MORSE CODE

The problem with using the IRIS dataset and other machine learning datasets converted to spiking datasets using population encoding or latency coding is that each input channel can only spike once during the entire example. Furthermore, there is no hierarchy in the temporal features to be learnt. Hence, these datasets only test the spatial feature learning capabilities of an SNN. Even popular neuromorphic vision datasets like N-MNIST do not have a hierarchy of temporal features, i.e., the time of the spike patterns is not key to the task, which is the classification of the digits such that the removal of all timing information (by simply binning all spikes at the input channel) does not result in any loss of information for the task. Thus, which spiking datasets test the true temporal learning capabilities of SNNs [73] is still an open question. Models trained on the datasets like IRIS only learn one spike per channel, and don't have to learn a sequence of different states along the temporal dimension. Each example is a single spatio-temporal feature with no more than one spike per channel. Hence, we made a custom task that can test and show the hierarchical learning capabilities of ODESA in the temporal dimension. We used the International Morse Code to encode different letters and numbers into spikes from two channels: "dash" and "dot". Each of these channels spikes multiple times for a given letter, and words would have multiple occurrences of the constituent letters. This forces the models trained on such a dataset to learn not only the spatio-temporal features across the channels at a given time but also learn the sequences of occurrences of such spatio-temporal features. For example, to differentiate two numerical sequences like "0,0,1,0,0" and "0,0,0,1,0", the model would have to learn the internal representation of the constituent numbers "0" and "1" which translate to "**-----**", and "**.----**" in Morse code using dots (**.**) and dashes (**-**) as spikes. We generated multiple tasks which involve complex sequence learning to test the hierarchical spatio-temporal learning capabilities of ODESA.

The first task with Morse code was to classify four different names: "ANDRE", "GREG", "SAEED", "YESH", and "YING" in Morse code. Each letter in each word was converted into a spike stream using the dots and dash encoding of Morse Code. For example, letter 'Y' is encoded as "**-.--**" in Morse code and a spike is generated from the respective input channel for each dash '**-**' or dot '**.**'. A time gap was inserted between each letter to distinguish it from the next letter. At the last spike of the last letter of each word, a corresponding output class spike is generated in the output spike train. The task is to predict the output spike train for the sequence of input spikes that conveys the different names in the training set. We used a two-layered ODESA network, one layer to learn spatio-temporal features at the timescale of the letters, and the second layer to learn features at the timescale of words, which is then fed into the output layer to learn the representations of the exact words in the dataset.

The second task we tested ODESA on was the previously mentioned sequence of "0,0,1,0,0" and "0,0,0,1,0". Though the supervisory signal is only provided at the end of the last "0" in each sequence, the model would have to learn some intermediate representation of "1" to be able to solve the task. Along with learning the representation of the symbol "1", the network also has to learn the position of the same symbol in the sequence. We used a two-layered ODESA network, one layer to learn the symbols (0 and 1), and the second layer, which is the output layer, to learn the sequence of the symbols. The output layer has two neurons for the two classes. The ODESA network can learn an intermediate representation of the symbol "1" without relying on an explicit supervisory signal for symbol "1" due to the local supervision provided by the next layer. Figure 9 shows the network activity at each layer after training on both the digit sequence and name detection tasks.

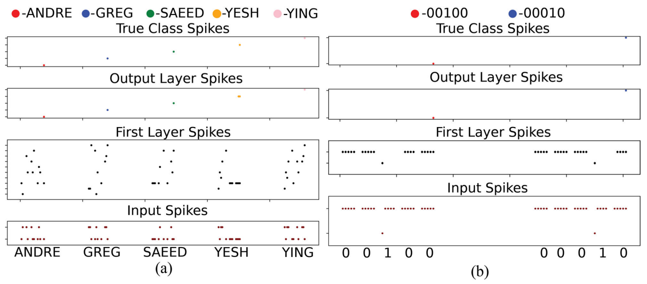

FIGURE 9. (a) Two-Layer ODESA activity after training to recognize the five author names "ANDRE", "GREG", "SAEED", "YESH", "YING" in Morse Code. (b) Two-Layer ODESA activity after training to recognize "0,0,1,0,0" and "0,0,0,1,0" in Morse Code.

The third Morse Code task we tested ODESA on was a more complex sequence that involved learning sentences in Morse code. We took Shakespeare's Sonnet 18 and used the first four lines as four different sequences to be learnt by the model. The four sentences were "shall I compare thee to a summers day", "thou art more lovely and more temperate", "rough winds do shake the darling buds of may", and "and summer's lease hath all too short a date". The reason for selecting a poem for the sequence learning task was that poems have line endings that rhyme with each other. The last two letters of line 1 and line 3 are "ay", and the last three letters of line 2 and line 4 are "ate". This makes sure that the model cannot just learn the ending letters of each line to differentiate different lines. A three-layer ODESA network was trained for the task such that the first layer can learn the letters, the second layer can learn the words, and the third layer can learn the sentences. The time constant for the first layer was at the scale of the letters (5 timesteps), the second layer's time constant was at the scale of words (50 timesteps), and the output layer has a time constant appropriate for the scale of the sentences (200 timesteps). Figure 10 shows the evolution of the activity at each layer at different epochs of the training. It shows how the specificity and the precise prediction of labels at the output layer with more training. Figure 11 shows the final activity of the network for a single example after the training.

F. TIDIGITS

Next, we tested ODESA on the TIDIGITS corpus [74] for an isolated spoken digits recognition task. The TIDIGITS corpus includes isolated digits and digit sequences from both female and male speakers in different age groups. It thus provides sufficient speaker diversity. In [75], the corpus was converted into a spike version using a threshold coding mechanism [76]. Each utterance from the corpus is firstly pre-processed by a 20-channel Constant-Q Transform (CQT) cochlear filter bank ranging from 200 Hz to 8 kHz. The generated spectrogram is then further encoded into spikes using the threshold coding. For each cochlear output channel, 15 onset thresholds and 15 offset thresholds are set for the normalised amplitude. The upward and downward threshold crossing events of the channel represent an afferent to form a 30-afferent spike sequence. In this work, we use the generated spiking TIDIGITS dataset from [75] to test ODESA, and only isolated digits (11 classes) from adult female and male speakers are used, which includes 2464 digit utterances for training and 2486 for testing. MPD-AL [36] was used to classify the same spiking dataset using 10 neurons for each of the 11 classes in the dataset. MPD-AL is a version of aggregate-label learning which aims to achieve a desired spike count from a post-synaptic neuron based on the feedback signal. MPD-AL uses an iterative method to find the easiest modifiable time instant during the course of an input spike pattern based on the membrane potential traces of a neuron. The synaptic weights are then adjusted to add or remove post-synaptic spikes until the number of spikes matches the desired number of spikes. The 10 neurons assigned for each class are then trained to generate the desired number of spikes only for spike patterns that belong to their corresponding class and remain silent for other classes. Two different decoding schemes were used, based on the desired number of spikes to be generated, for the correct class neurons in [36]. The original work labelled the entire spike pattern as a single class and the number of spikes generated throughout an input spike pattern was used to determine the predicted class of the pattern. In our method, we labelled the last spike of each input spike pattern for a class with a label spike as ODESA requires a precisely timed label spike. We have used a three layer network with each layer learning spatio-temporal patterns at different timescales, i.e., 15 ms, 30 ms, and 35 ms respectively. Table 3 compares the results between the proposed method and MPD-AL. [36] on the same dataset.

Unlike MPD-AL, ODESA networks do not access the entire history of the activity of a neuron during an input spike pattern. The supervisory label signal which indicates the class of an input spike pattern is only available to the neuron at the last spike in the spike pattern. It is possible to label every input spike with the label of the whole example, but that can often be misleading to the network, as two different digits can have similar sounds/phonemes in parts of the example. The framework of ODESA expects a correlation between the input pattern and the label spike. Hence we decided to only provide the labelled spike at the end of each example. In contrast, MPD-AL has access to the label of the input spike pattern at every instance of time. A three layer (2000-4000-11) ODESA network could achieve a training accuracy of 96.80% and a test accuracy of 91.4% on the

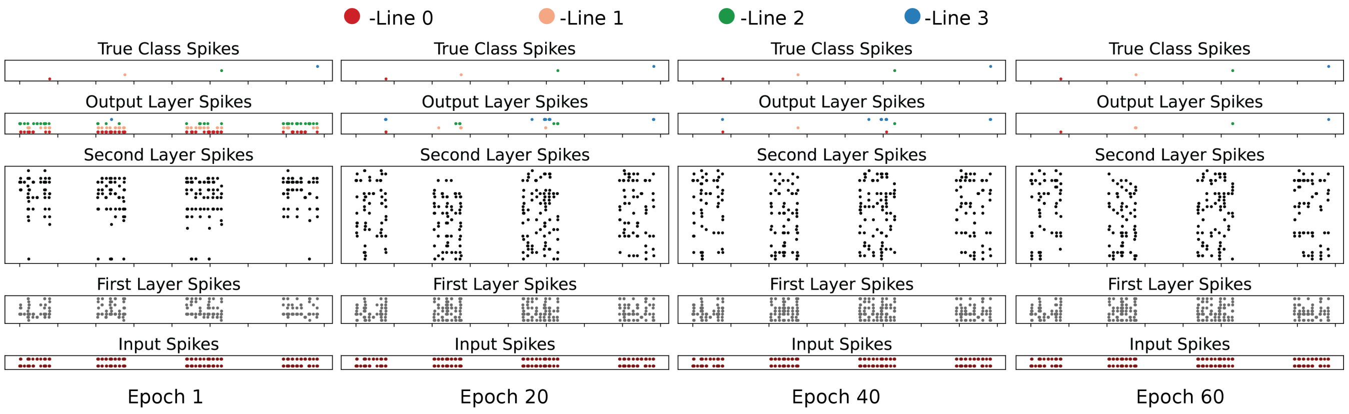

FIGURE 10. Evolution of network activity over the epochs in training for the Sentence classification task.

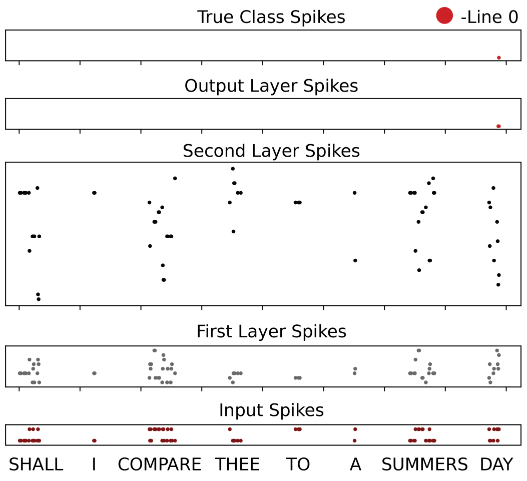

FIGURE 11. Zoomed in network activity at each ODESA layer for the input line "SHALL I COMPARE THEE TO THE SUMMERS DAY" in Morse Code.

spiking TIDIGIT dataset. Table 3 shows the comparison of the performances with MPD-AL.

TABLE 3. Test accuracy results on TIDIGIT dataset.

Method	Test Accuracy
MPD-AL with $N_d = 3$ [36]*	95.3%
MPD-AL with Dynamic Decoding [36]*	97.5%
Proposed Method	91.4%

(The scores are reported as presented in [36]. However, they have not been verified as there was not enough information to reproduce and no code to do so was available.)*

V. DISCUSSION

A. ROLE OF WEIGHT UPDATES

As described in Section III-A, threshold adaptation and weight update (Equations (13) to (16)) in the output layer are the key steps that facilitate timely spiking of the neurons in the correct class group. The learning algorithm resulting from these updates in itself is capable of learning the features required to solve the benchmarks presented in this paper. However, the network failed to converge in some scenarios depending on the initial conditions. We investigated additional weight update steps for the output layer (Section III-D) which can speed up the convergence of the network performance. One of the additional weight update steps was to use a negative weight update (Equations (20) to (21))) which is similar to anti-STDP to disincentivise neurons of wrong class groups from spiking. We observed that this step improved the final mean training accuracy on many datasets. The other weight update step investigated was rewarding the closest neuron in the correct class group in case none of the neurons in the correct class group spiked. This step improved the speed of convergence of the model more than the final accuracy. The closest winner update helps by rewarding the probable candidate output neuron in the correct class group immediately, without having to wait until the exact example reappear in the training data. Both the additional weight update steps individually improved the final accuracy of the models, and the combination of all the weight updates gave the best results in terms of the mean accuracy and its variance across multiple trials and initial conditions. We used the same IRIS dataset as in Section IV-C to compare the effects of the additional weight update steps in improving the mean accuracy and the variance of the networks across 20 trials of 2-fold cross validation. Figure 12 shows the mean of training and testing error across multiple trials at each epoch along with the standard deviation in the training and testing errors at each epoch. We don't yet fully understand why the models see a slight drop in the performance after reaching their peak and plan to investigate this in future work. We suspect some form of over-fitting as one of the reasons.

B. DEEP LEARNING IN ODESA

As the learning of ODESA inherently did not have any limitations on the depth of the models, we wanted to investigate the effect of deeper networks on the performance of the model. Unfortunately, none of the current spiking benchmarks available has enough temporal hierarchy to require much deeper networks that can utilise the hierarchy of temporal features. Nevertheless, we used the Oxford spike pattern to test any vanishing effects of the feedback attention signals used in

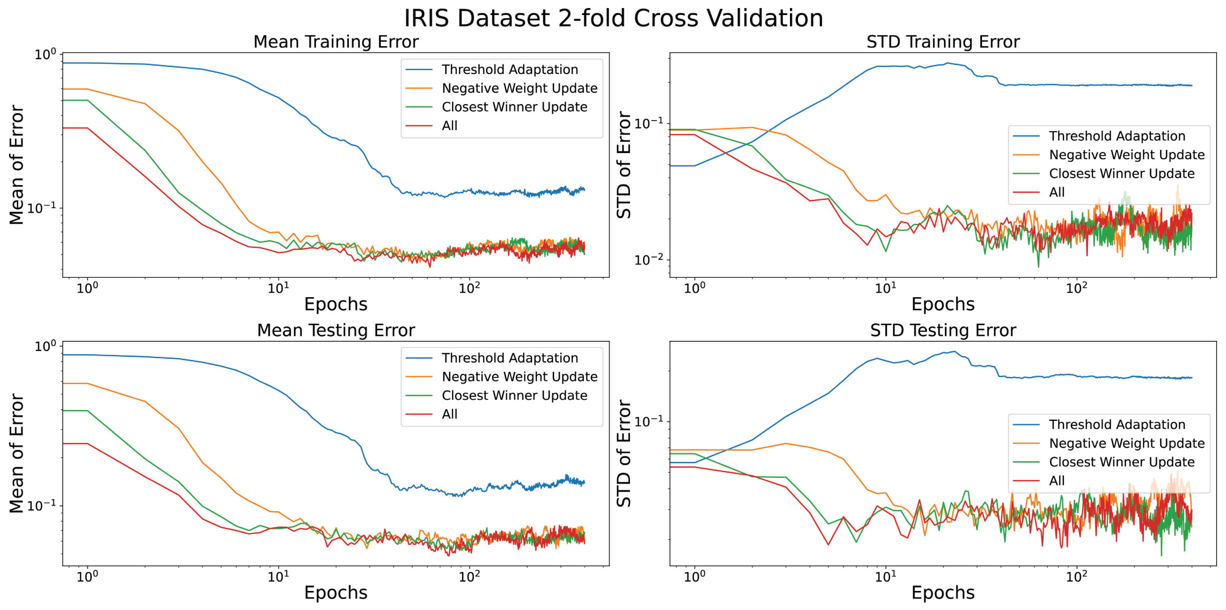

FIGURE 12. **Comparison of the model performance with different weight and threshold adaptation on the output layer.**

ODESA with increasing depth of the models. We used networks of different depths from 1 layer to 10 layers (including the output layer) and all of them converged to a final accuracy after a sufficient number of training epochs. Different time constants (τ_l) were used for each layer ranging from 0.001s to 0.0075s in the networks. The number of neurons also ranged from 300 to 1000 with the depth of the network. Figure 13 shows the qualitative output of networks of different depths for the Oxford spike pattern. We have used the mean IOU per input spike as the measure of performance for the Oxford spike pattern. Figure 14 shows the trend of the mean IOU per input spike per epoch during the training of these networks. We observed that deeper networks required more epochs to converge to their final performance. This is expected because the latter layers of a network depend on the features of the earlier layers. We also noticed that there is a drop in performance when the networks get deeper, but there was no clear trend in this. Some deeper networks performed better than other shallower networks and vice versa. This can be due to multiple reasons including that each layer has to learn spatio-temporal features at different time scales depending on the time constant τ_l of the layers. Depending on the input spike data, the number of possible spatio-temporal features can be different at different time constants. These effects require a more detailed study to fully investigate.

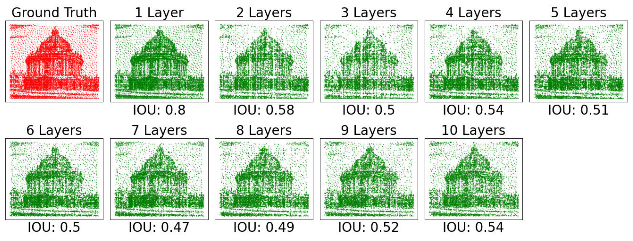

FIGURE 13. **Prediction of Oxford spike pattern from networks with different number of layers ranging from 1 to 10 layers including the output layer.**

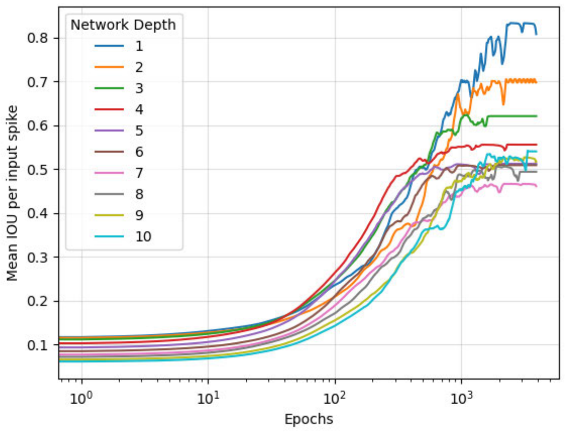

FIGURE 14. **Mean IOU per spike over Epochs for different depths of ODESA Networks on Oxford Spike Pattern.**

C. ROLE OF THRESHOLDS IN ODESA

There have been previous works that have used the neuronal spike thresholds in the learning process. Most commonly ALIF neurons with dynamic thresholds have been shown to improve performance in recurrent SNNs [9], [77] and promote homeostasis in the networks with STDP [63]. Aggregate label learning methods like Multi-Spike Tempotron (MST) [34] and Threshold-Driven Plasticity (TDP) [35] used a Spike Threshold Surface to map the threshold of a neuron and the number of spikes it generates for a given spike train. Optimizing the weights using gradients with respect to the threshold enabled neurons to output the desired number of spikes per input pattern. Alternatively, Membrane Potential

Driven Aggregate Learning (MPD-AL) [36] used gradients with respect to the membrane potential, unlike MST and TDP. All the aggregate learning methods involve iterations to find the optimal threshold, or membrane voltage, to generate the desired number of spikes per neuron. But these learning algorithms cannot influence the precise timing of the spikes. Furthermore, the aggregate learning algorithms are single neuron algorithms and don't have solutions to learn hierarchical spatio-temporal features using multiple layers of neurons. Learning in ODESA doesn't involve finding the correct target threshold or membrane potential. Instead, ODESA finds the right weights and thresholds by online adaptation of the thresholds and facilitates neurons to spike precisely at the time of a global attention signal created by a supervisory signal. ODESA networks are capable of generating precisely timed spike patterns that can be learnt. In addition to that, ODESA can use spike-timing-dependent threshold adaptation to learn hierarchies of spatio-temporal features at multiple timescales by driving the adaptation of the previous layer's thresholds based on the current layer's activity. Evidence shows that rapid threshold adaptation in pre-synaptic neurons occurs with correlated spiking in the post-synaptic neuron as a form of spike-timing-dependent intrinsic plasticity in biology [78].

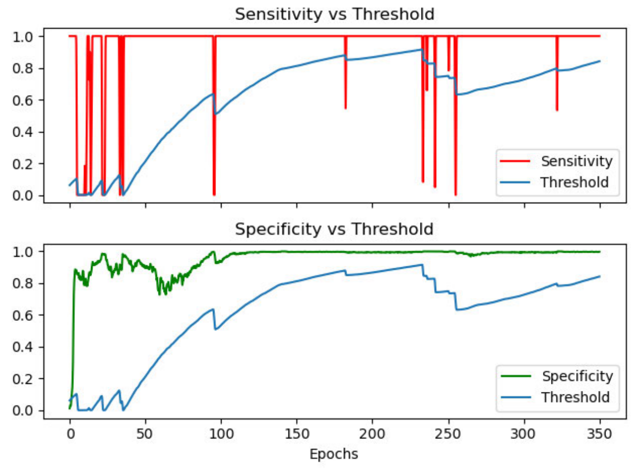

FIGURE 15. The trend of sensitivity and specificity of a class in the Sentence classification task with respect to the threshold of the corresponding output neuron.

The thresholds in ODESA networks maintain neuronal homeostasis while also acting as gates to the adaptation of synaptic weights. When an output neuron does not spike for an important input spike, the threshold of the neuron is decreased until the neuron starts spiking. When any of those spikes match with the target spikes, the threshold of that neuron is increased while its weights are adjusted simultaneously. Higher thresholds make neurons more selective in their spiking. Every time the neuron weights are updated for an input time surface context, the likelihood of the neuron spiking again for the same input context increases, and the threshold increase makes it difficult for the neuron to spike for other contexts. If a neuron gets too specific, its thresholds are lowered again to let the neuron spike for a broader range of input spike patterns. This continues until the neuron reaches a balance between the desired activity and its activity. The desired activity for the output layer is the label output spike train. Similarly, the goal of the hidden neurons is to support the spikes in the subsequent layer. Increasing the thresholds of the participant neurons and lowering the thresholds of inactive neurons for a spike in the next layer forces all the neurons in the layer to participate equally for every spike generated in the next layer. The combination of threshold adaptation and weight updates makes the neurons gradually become more specific in their spiking and match the spiking behaviour to the desired spike train. This can be seen in the evolution of layer activity throughout the training in Figure 10.

Figure 15 shows the trend of sensitivity and specificity of a class in the Sentence classification task. The drops in the threshold of an output neuron always occur with the drops in the sensitivity of a class. This is due to the punishment of the output neuron when it misses a ground truth label spike. An output neuron can miss the spikes when its threshold gets too high. On the other hand, the specificity of the class improves with a higher threshold of a neuron. When the threshold of a neuron is lowered, it also affects the specificity of the neuron as it spikes for other input spikes as well. This interplay between the rising and falling of the threshold helps the neuron to get more precise with its spiking and predict as many labelled spikes as possible. The threshold reaches the maximum value when no label spikes are missed and it precisely spikes only for the labelled input spikes. The thresholds can also be viewed as the confidence of a neuron for its corresponding class.

D. OUTPUT DECODING IN ODESA

Different output decoding schemas are used in SNN models depending on output neurons using a Latency code or a Rate code. The time to first spike would require resetting to some default state from which the time of the spikes is supposed to be calculated. Rate coding on the other hand offers some error tolerance as a few missed spikes may not affect the firing rate of the output neurons. The output label spikes in ODESA are precisely timed with respect to the input spikes. The output spike from an ODESA model indicates the best prediction given all the evidence until that instant in time. ODESA performs the best when there exists a strong temporal correlation between the label spike and the labelled input spike. This sometimes can lead to an erroneous classification in some cases when the labelling is not precise. Tasks like Morse code detection, generally don't have this problem as they have a precise ending to a signal that can trigger the output classification. But there are very few openly available datasets that have temporally precise labelling. Spike-based TIDIGITS dataset examples usually have a few noisy input spikes at the end of each example which can make it challenging to learn the temporal correlation between the labelled input spike and the learnt label spike of the example.

E. ADVANTAGES OF ODESA

Previous works have used multiple layers in SNNs to learn a spatial hierarchy of features at the same timescale. ODESA uses multi-layer networks to learn a spatio-temporal hierarchy of features. ODESA achieves sequence learning without having to rely on recurrent connections and only uses feed-forward connections in the network. This is achieved by calculating traces of neurons using longer time constants as we go deeper into the multi-layered network. Many of the previous event-driven feature extraction architectures [44], [45], [46] also require a classifier at the the end of the SNN to achieve classification in SNNs. ODESA eliminates the requirement of such a separate classifier module and provides a solution using a spiking neural architecture from end to end. This architecture not only allows classification by the SNN but also allows them to learn arbitrary transformations between two spike streams.

ODESA does not rely on surrogate gradients or real continuous-valued error feedback signals. The entirety of learning is achieved by only using binary attention signals, which are bio-plausible. This way, even when a deeper layer in the network goes silent, the earlier layers continue learning without having to wait for feedback signals from the silent layer. Another benefit of not using surrogate gradients and only using binary feedback signals is that ODESA doesn't have to rely on differentiable loss functions. In error back-propagation methods, the weights of a layer cannot be updated until an entire forward pass of prediction and backward pass of error calculation occurs. The learning of any hidden layer of ODESA only depends on the global attention signal from the labelled output spike train and the local attention signal from the layer above it. In essence, the learning of hidden layers in ODESA can continue even when a higher layer has gone completely silent. Furthermore, the computations in ODESA networks are all causal in the sense that no signals are propagated back in time, and no form of error BPTT occurs in the network. ODESA only uses computations that are event-driven and not clock-driven. This is valuable for energy-efficient neuromorphic hardware and can work well with neuromorphic sensors which may not generate data at all times. As the learning algorithm is online in nature, ODESA can also handle concept-drift in the data and keep updating its parameters in the light of any new changes in the input data distributions. But this can also lead to catastrophic forgetting in cases where the training data is not uniformly distributed.

The supervision in ODESA networks is not coupled with a fixed neuron model. The weight update steps in Section III-A only perform the function of maximising the dot product of the winning input time surface context and the synaptic weights of the winner neuron. ODESA can be easily extended to other spiking neuronal models that allow threshold adaptation, with an equivalent synaptic weight update step that increases the likelihood of the neuron to spike again for the same input spike pattern. For neuron models with synaptic delays, the time constant of the trace of neurons has to be increased to take into account the delayed input to the next layer which may be crucial for future implementation in neuromorphic hardware. Similar modifications to the trace function (currently instantaneous exponential decay) would have to be made for higher-order dynamics of neuronal synaptic responses(e.g Alpha PSP). This opens door to more novel abstract spiking neuronal units that can be specialised to the task and input data (e.g., Neuromorphic vision data, Neuromorphic audio data, etc.). It also paves the way to use task-specific network architectures like convolutional layers and skip connections as used in ANNs which can induce task-specific priors to the features. In future work, the assumption of having an input spike for every ground truth spike can be relaxed by including delays in the neuron synaptic models.

F. LIMITATIONS OF ODESA

The learning in each layer of ODESA is essentially a spike-based clustering algorithm similar to K-Means clustering. The neurons in ODESA learn features by learning the centres of the cluster of time surface contexts for which each neuron fires along with an acceptance boundary around this centre. Negative weights are not present in ODESA neurons due to the nature of the weight updates which are a form of the exponential moving average. Even the negative weight update step when added does not typically result in the generation of negative weights. The negative weight update step only helps in moving the weights away from the given time surface context in the high dimensional space. As can be seen in the Figure 8, none of the weights is negative. This can potentially limit the performance compared to some other SNN methods which use neurons with negative weights. One possible solution is using an inhibitory neuron group that learns complimentary features using the same principles of ODESA in each layer. Alternate weight update paradigms which allow negative weights can also be one of the possibilities to explore in future work. Here we focused on extending the optimised SNN concept to multiple layers without resorting to error back-propagation which is not bio-plausible and challenging to implement in neuromorphic hardware.

VI. CONCLUSION

In this work, we introduced a novel optimised abstraction to SNNs that can be used to solve practical machine learning problems in the spiking domain. We conducted a comprehensive evaluation of the architecture on various spiking datasets available. Unlike other methods that try to approximate error back-propagation to SNNs using techniques like surrogate gradients, we explore using a simple local spike-timing-dependent adaptation of thresholds and weights as the only way to train spiking neural architectures. The learning rule in ODESA networks is an event-driven supervised learning algorithm that enables learning transformations between arbitrary input and output spike trains under the single assumption that there exists a spike in the input train for every spike in the output spike train. We show that learning can be achieved in deep spiking neural architectures using binary attention signals between neuron groups and one global binary attention signal along with a simple three-factor adaptation of weights and thresholds. ODESA networks have sparse activity due to the hard WTA constraint on each layer, making it energy efficient to implement on neuromorphic hardware. The local learning rules of ODESA also help reduce the memory accesses required, as the weight updates do not depend on the parameters of other computational nodes in the network. This enables on-chip online learning capabilities for neuromorphic hardware that can exploit the afore-mentioned characteristics. This work is an attempt at bridging the gap between machine learning and SNN models using a novel abstraction of biological spiking networks without relying on back-propagation of gradients.

AUTHOR CONTRIBUTIONS

Yeshwanth Bethi, Saeed Afshar, and André van Schaik conceived the overall theory and design of the algorithm. Yeshwanth Bethi, Saeed Afshar, and Ying Xu performed the experiments. Yeshwanth Bethi wrote the initial draft of the manuscript. Saeed Afshar, André van Schaik, Ying Xu, and Gregory Cohen edited and provided feedback on the manuscript.

CODE AVAILABILITY

The code for ODESA and experiments mentioned in this paper is available at https://github.com/yeshwanthravitheja/ODESA.py

APPENDIX

See Tables 4–7.

TABLE 4. Network parameters for IRIS dataset.

Parameter	Value
Layer 1: Number of neurons (N)	10
Layer 1: Time constant (τ)	0.6
Layer 1: Learning Rates (η, η_{thresh})	0.001
Layer 1: Threshold Drop (θ_{drop})	0.1
Output Layer: Number of neurons per class (k)	1
Output Layer: Number of Classes (N_c)	3
Output Layer: Time constant (τ)	0.9
Output Layer: Learning Rates (η, η_{thresh})	0.01
Output Layer: Threshold Drop (θ_{drop})	0.1

TABLE 5. Single layer network parameters for Oxford spike dataset.

Parameter	Value
Output Layer: Number of neurons per class (k)	$\sim$35
Output Layer: Number of Classes (N_c)	200
Output Layer: Time constant (τ)	0.0075
Output Layer: Learning Rates (η, η_{thresh})	0.1
Output Layer: Threshold Drop (θ_{drop})	0.001

TABLE 6. Network parameters for latency coded MNIST dataset.

Parameter	Value
Layer 1: Number of neurons (N)	6000
Layer 1: Time constant (τ)	1
Layer 1: Learning Rates (η, η_{thresh})	0.0001
Layer 1: Threshold Drop (θ_{drop})	0.01
Output Layer: Number of neurons per class (k)	10
Output Layer: Number of Classes (N_c)	10
Output Layer: Time constant (τ)	1
Output Layer: Learning Rates (η, η_{thresh})	0.001
Output Layer: Threshold Drop (θ_{drop})	0.01

TABLE 7. Network parameters for TIDIGITS dataset.

Parameter	Value
Layer 1: Number of neurons (N)	2000
Layer 1: Time constant (τ)	15
Layer 1: Learning Rates (η, η_{thresh})	0.00005
Layer 1: Threshold Drop (θ_{drop})	0.0005
Layer 2: Number of neurons (N)	4000
Layer 2: Time constant (τ)	30
Layer 2: Learning Rates (η, η_{thresh})	0.0005
Layer 2: Threshold Drop (θ_{drop})	0.001
Output Layer: Number of neurons per class (k)	1
Output Layer: Number of Classes (N_c)	11
Output Layer: Time constant (τ)	35
Output Layer: Learning Rates (η, η_{thresh})	0.005
Output Layer: Threshold Drop (θ_{drop})	0.015

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[2] D. G. Stork, “Is backpropagation biologically plausible?” in *Proc. Int. Joint Conf. Neural Netw.*, 1989, pp. 241–246.
[3] F. Crick, “The recent excitement about neural networks,” *Nature*, vol. 337, no. 6203, pp. 129–132, Jan. 1989.
[4] S. Grossberg, “Competitive learning: From interactive activation to adaptive resonance,” *Cognit. Sci.*, vol. 11, no. 1, pp. 49–50, 1987.
[5] S. M. Bohte, J. N. Kok, and H. L. Poutré, “Error-backpropagation in temporally encoded networks of spiking neurons,” *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, 2000.
[6] R. Gütig and H. Sompolinsky, “The tempotron: A neuron that learns spike timing–based decisions,” *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, 2006.
[7] R. V. Florian, “The chronotron: A neuron that learns to fire temporally precise spike patterns,” *PLoS ONE*, vol. 7, no. 8, Aug. 2012, Art. no. e40233.
[8] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks,” *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.
[9] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 795–805.
[10] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, “A solution to the learning dilemma for recurrent networks of spiking neurons,” *Nature Commun.*, vol. 11, no. 1, pp. 1–15, Dec. 2020.
[11] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018.
[12] D. Zipser and R. A. Andersen, “A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons,” *Nature*, vol. 331, no. 6158, p. 684, 1988.

[13] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," *Psychol. Rev.*, vol. 102, no. 3, p. 439, 1995.

[14] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Comput.*, vol. 33, no. 4, pp. 899–925, 2021.

[15] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, pp. 1–10, Dec. 2016.

[16] A. Tavanaei, T. Masquelier, and A. S. Maida, "Acquisition of visual features through probabilistic spike-timing-dependent plasticity," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 307–314.

[17] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated STDP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6178–6190, Dec. 2018.

[18] A. Vigneron and J. Martinet, "A critical survey of STDP in spiking neural networks for pattern recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

[19] F. Paredes-Valles, K. Y. W. Scheper, and G. C. H. E. D. Croon, "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2051–2064, Aug. 2020.

[20] R. Legenstein, C. Naeger, and W. Maass, "What can a neuron learn with spike-timing-dependent plasticity?" *Neural Comput.*, vol. 17, no. 11, pp. 2337–2382, Nov. 2005.

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

[22] B. Widrow and M. E. Hoff, "Adaptive switching circuits," Stanford Univ., Stanford Electron. Labs, Stanford, CA, USA, Tech. Rep. 1553-1, Jun. 1960.

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

[24] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 25, no. 2, pp. 473–509, Feb. 2013.

[25] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5394–5407, Nov. 2018.

[26] S. Schliebs and N. Kasabov, "Evolving spiking neural network—A survey," *Evolving Syst.*, vol. 4, no. 2, pp. 87–98, Jun. 2013.

[27] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Adaptive learning procedure for a network of spiking neurons and visual pattern recognition," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst.* Cham, Switzerland: Springer, 2006, pp. 1133–1142.

[28] J. L. Lobo, I. Laña, J. Del Ser, M. N. Bilbao, and N. Kasabov, "Evolving spiking neural networks for online learning over drifting data streams," *Neural Netw.*, vol. 108, pp. 1–19, 2018.

[29] N. K. Kasabov, "Evolving spiking neural networks," in *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence*. Cham, Switzerland: Springer, 2019, pp. 169–199.

[30] A. Belatreche, L. P. Maguire, M. Mcginnity, and Q. X. Wu, "Evolutionary design of spiking neural networks," *New Math. Natural Comput.*, vol. 2, no. 3, pp. 237–253, Nov. 2006.

[31] N. G. Pavlidis, D. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 4, Jul. 2005, pp. 2190–2194.

[32] R. A. Vazquez, "Training spiking neural models using cuckoo search algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2011, pp. 679–686.

[33] J. Wang, A. Belatreche, L. P. Maguire, and T. M. McGinnity, "SpikeTemp: An enhanced rank-order-based learning approach for spiking neural networks with adaptive structure," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 30–43, Jan. 2017.

[34] R. Gütig, "Spiking neurons can discover predictive features by aggregate-label learning," *Science*, vol. 351, no. 6277, Mar. 2016, Art. no. aab4113.

[35] Q. Yu, H. Li, and K. C. Tan, "Spike timing or rate? Neurons learn to make decisions for both through threshold-driven plasticity," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2178–2189, Jun. 2019.

[36] M. Zhang, J. Wu, Y. Chua, X. Luo, Z. Pan, D. Liu, and H. Li, "MPD-AL: An efficient membrane potential driven aggregate-label learning algorithm for spiking neurons," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 1327–1334.

[37] M. A. Farries and A. L. Fairhall, "Reinforcement learning with modulated spike timing–dependent synaptic plasticity," *J. Neurophysiol.*, vol. 98, no. 6, pp. 3648–3665, 2007.

[38] J. Brea, W. Senn, and J.-P. Pfister, "Matching recall and storage in sequence learning with spiking neural networks," *J. Neurosci.*, vol. 33, no. 23, pp. 9565–9575, Jun. 2013.

[39] J. Friedrich, R. Urbanczik, and W. Senn, "Spatio-temporal credit assignment in neuronal population learning," *PLoS Comput. Biol.*, vol. 7, no. 6, Jun. 2011, Art. no. e1002092.

[40] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Comput.*, vol. 19, no. 6, pp. 1468–1502, 2007.

[41] D. J. Rezende and W. Gerstner, "Stochastic variational learning in recurrent spiking networks," *Frontiers Comput. Neurosci.*, vol. 8, p. 38, Apr. 2014.

[42] N. Frémaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Frontiers Neural Circuits*, vol. 9, p. 85, Jan. 2016.

[43] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Jan. 2008.

[44] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1731–1740.

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

[46] S. Afshar, N. Ralph, Y. Xu, J. Tapson, A. V. Schaik, and G. Cohen, "Event-based feature extraction using adaptive selection thresholds," *Sensors*, vol. 20, no. 6, p. 1600, Mar. 2020.

[47] J. C. Tapson, G. K. Cohen, S. Afshar, K. M. Stiefel, Y. Buskila, T. J. Hamilton, and A. V. Schaik, "Synthesis of neural networks for spatio-temporal spike pattern recognition and processing," *Frontiers Neurosci.*, vol. 7, p. 153, Jan. 2013.

[48] R. B. Stein, "Some models of neuronal variability," *Biophys. J.*, vol. 7, no. 1, pp. 37–68, 1967.

[49] R. B. Stein, "A theoretical analysis of neuronal variability," *Biophys. J.*, vol. 5, no. 2, pp. 173–194, 1965.

[50] N. Fourcaud-Trocmé, D. Hansel, C. van Vreeswijk, and N. Brunel, "How spike generation mechanisms determine the neuronal response to fluctuating inputs," *J. Neurosci.*, vol. 23, no. 37, p. 11628–11640, 2003.

[51] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[52] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[53] N. Kasabov, "To spike or not to spike: A probabilistic spiking neuron model," *Neural Netw.*, vol. 23, no. 1, pp. 16–19, 2010.

[54] R. Jolivet, T. J. Lewis, and W. Gerstner, "Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy," *J. Neurophysiol.*, vol. 92, no. 2, pp. 959–976, Aug. 2004.

[55] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952.

[56] R. Jolivet, A. Rauch, H.-R. Lüscher, and W. Gerstner, "Predicting spike timing of neocortical pyramidal neurons by simple threshold models," *J. Comput. Neurosci.*, vol. 21, no. 1, pp. 35–49, Aug. 2006.

[57] S. M. Bohte, H. L. Poutré, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 426–435, Mar. 2002.

[58] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A spiking neural network training algorithm for classification problems," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[59] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection," *Integr. Comput.-Aided Eng.*, vol. 14, no. 3, pp. 187–212, 2007.

[60] N. Gueorguieva, I. Valova, and G. Georgiev, "Learning and data clustering with an RBF-based spiking neuron network," *J. Experim. Theor. Artif. Intell.*, vol. 18, no. 1, pp. 73–86, 2006.

[61] Q. Yu, H. Tang, K. C. Tan, and H. Yu, ''A brain-inspired spiking neural network model with temporal encoding and learning,'' *Neurocomputing*, vol. 138, pp. 3–13, Aug. 2014.
[62] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ''Gradient-based learning applied to document recognition,'' *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[63] P. U. Diehl and M. Cook, ''Unsupervised learning of digit recognition using spike-timing-dependent plasticity,'' *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.
[64] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, ''A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm,'' in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2011, pp. 1–4.
[65] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, ''Event-driven contrastive divergence for spiking neuromorphic systems,'' *Frontiers Neurosci.*, vol. 7, p. 272, Jan. 2013.
[66] S. Hussain, S.-C. Liu, and A. Basu, ''Improved margin multi-class classification using dendritic neurons with morphological learning,'' in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2640–2643.
[67] M. Beyeler, N. D. Dutt, and J. L. Krichmar, ''Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule,'' *Neural Netw.*, vol. 48, pp. 109–124, Dec. 2013.
[68] A. Patiño-Saucedo, H. Rostro-Gonzalez, T. Serrano-Gotarredona, and B. Linares-Barranco, ''Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the SpiNNaker neuromorphic platform,'' *Neural Netw.*, vol. 121, pp. 319–328, Jan. 2020.
[69] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, ''Feedforward categorization on AER motion events using cortex-like features in a spiking neural network,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 1963–1978, Sep. 2015.
[70] A. Tavanaei and A. Maida, ''BP-STDP: Approximating backpropagation using spike timing dependent plasticity,'' *Neurocomputing*, vol. 330, pp. 39–47, Feb. 2019.
[71] J. H. Lee, T. Delbruck, and M. Pfeiffer, ''Training deep spiking neural networks using backpropagation,'' *Frontiers Neurosci.*, vol. 10, p. 508, Aug. 2016.
[72] H. Mostafa, ''Supervised learning based on temporal coding in spiking neural networks,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2017.
[73] L. R. Iyer, Y. Chua, and H. Li, ''Is neuromorphic MNIST neuromorphic? Analyzing the discriminative power of neuromorphic datasets in the time domain,'' *Frontiers Neurosci.*, vol. 15, p. 297, Mar. 2021.
[74] R. Leonard, ''A database for speaker-independent digit recognition,'' in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 1984, pp. 328–331.
[75] M. Zhang, J. Wu, Y. Chua, X. Luo, Z. Pan, D. Liu, and H. Li, ''MPD-AL: An efficient membrane potential driven aggregate-label learning algorithm for spiking neurons,'' in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 1327–1334.
[76] R. Gütig and H. Sompolinsky, ''Time-warp–invariant neuronal processing,'' *PLoS Biol.*, vol. 7, no. 7, Jul. 2009, Art. no. e1000141.
[77] A. Shaban, S. S. Bezugam, and M. Suri, ''An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation,'' *Nature Commun.*, vol. 12, no. 1, pp. 1–11, Dec. 2021.
[78] W. Zhang and D. J. Linden, ''The other side of the engram: Experience-driven changes in neuronal intrinsic excitability,'' *Nature Rev. Neurosci.*, vol. 4, no. 11, p. 895, 2003.

YESHWANTH BETHI received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Bombay (IIT B), India, in 2016. He is currently pursuing the Ph.D. degree in event-based neural architectures with the International Centre for Neuromorphic Systems, Western Sydney University, Sydney, NSW, Australia. He was a Research Assistant with the Neuronics Laboratory, Indian Institute of Science (IISc), Bengaluru, India, from 2017 to 2019.

YING XU (Member, IEEE) received the M.Sc. degree in electrical engineering from Waseda University, Kitakyushu, Japan, in 2008, and the Ph.D. degree in neuromorphic engineering from The MARCS Institute, Western Sydney University. She is currently a Postdoctoral Research Fellow with the International Centre for Neuromorphic Systems, Western Sydney University. Her research interests include auditory neuromorphic systems, application-specified integrated circuit/field programmable gate array design, and machine learning.

GREGORY COHEN (Member, IEEE) received the B.Sc. (Eng.) degree in electrical and computer engineering, and the M.Sc. (Eng.) and B.Com. (Hons.) degrees in finance and portfolio management from the University of Cape Town, Cape Town, South Africa, in 2007, 2008, and 2010, respectively, and the joint Ph.D. degree in signal processing and neuromorphic engineering from Western Sydney University, Sydney Australia, and the University of Pierre and Marie Curie in Paris, France. Prior to returning to research from industry, he worked in several start-ups and established engineering and consulting firms, including worked as a Consulting Engineer in the field of large-scale HVAC, from 2007 to 2009, an Electronic Design Engineer, from 2009 to 2011, and an Expert Consultant for Kaiser Economic Development Practice, in 2012. He is currently an Associate Professor in neuromorphic systems with the International Centre for Neuromorphic Systems (ICNS), Western Sydney University, and the Program Lead for neuromorphic algorithms and space applications.

ANDRÉ VAN SCHAIK (Fellow, IEEE) received the M.Sc. degree in electrical engineering from the University of Twente, Enschede, The Netherlands, in 1990, and the Ph.D. degree in neuromorphic engineering from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1998. From 1991 to 1994, he was a Researcher with the Swiss Centre for Electronics and Microtechnology (CSEM), where he has been developing the first commercial neuromorphic chip—the optical motion detector used in Logitech trackballs, since 1994. In 1998, he was a Postdoctoral Research Fellow with the Department of Physiology, University of Sydney, and in 1999, he became a Senior Lecturer with the School of Electrical and Information Engineering and a Reader, in 2004. In 2011, he became a Full Professor at Western Sydney University, where he is currently a Pioneer of Neuromorphic Engineering and the Director of the International Centre for Neuromorphic Systems. He has authored more than 200 articles. He is an inventor of more than 35 patents. He has founded three technology start-ups. His research interests include neuromorphic engineering, encompassing neurophysiology, computational neuroscience, software and algorithm development, and electronic hardware design. He is a fellow of the IEEE for contributions to Neuromorphic Circuits and Systems.

SAEED AFSHAR (Member, IEEE) received the B.Eng. degree in electrical engineering from the University of New South Wales, Sydney, NSW, Australia, and the M.Eng. degree in electrical engineering from the Western Sydney University, Sydney. He is currently a Lecturer with the International Centre for Neuromorphic Systems, Western Sydney University. His research interests include event-based vision and audio processing for neuromorphic hardware.