A Reservoir-based Convolutional Spiking Neural Network for Gesture Recognition from DVS Input

Arun M. George*, Dighanchal Banerjee†, Sounak Dey‡, Arijit Mukherjee§, Balamurali P¶
TCS Research & Innovation, Kolkata, India
Email: *arunm.george@tcs.com, †dighanchal.b@tcs.com, ‡sounak.d@tcs.com,
§mukherjee.arijit@tcs.com, ¶balamurali.p@tcs.com

***Abstract*—Mammalian neural circuits respond to different sensory stimuli by firing spikes at particular times. Closely mimicking this phenomenon, the evolving 3rd generation neural networks, known as *Spiking Neural Networks (SNNs)*, are found to be capable of memorizing and learning from the spatio-temporal spike patterns. This makes SNN applicable in identification of human actions and gestures, especially in the robotics domain. The paradigm is also suited for Neuromorphic Systems leading to less energy intensive applications. In this work, we present a novel spiking neural network constituting multiple convolutional layers and a reservoir layer to extract spatial and temporal features respectively from human gesture videos captured with DVS camera. We achieved more than 95% Top-3 accuracy on IBM DVS dataset and we claim that the performance of our network is better in terms of accuracy vs. learning parameters ratio when compared to other networks.**

I. Introduction

In the era of artificial intelligence, robots and drones have slowly gained prominence across domestic and industrial environments. *Assistive Robotics* is often viewed as the future of elderly-care in the domestic space, whereas, robots and drones are already being used industrially in multiple domains such as retail, manufacturing, healthcare, disaster management etc. Presently however, robots and humans do not share the same workspace - a situation which is likely to change drastically in a not-too-distant future. As a consequence, interaction between humans and robots is a real need, and has led to the evolution of a new research area on *Human Robot Interaction (HRI)*.

The problems in this area mainly centre around learning and identification of gesture/speech/intention of human co-workers along with the classical problems of learning and identification of objects/obstacles within a dynamic environment. The current state of the art vision-based solutions using artificial neural networks (including deep neural networks) [1], [2] are highly accurate and widely used. However, these require a large volume of training data for successful learning and accurate inferencing, and are hence both time and computation intensive. They are also bounded by the conventional von Neumann architectures leading to a data transfer bottleneck between memory and processing units and related power consumption issues as *robots and drones are constrained by their battery life*.

Lately, an alternative non von Neumann paradigm known as *Neuromorphic Computing* [3], [4] has evolved where computation is performed on processors architected as connections of millions of neurons and synapses, closely mimicking mammalian brains. Computation resembles sensory information processing in mammals with data represented as inherently sparse spike trains which capture environmental changes as events. Highly parallel computations are performed in neurons and the processed information are stored at synapses resulting in collocation of computation and memory. The learning method is based on the precise timing of two different layers of neurons spiking activity signifying how correlated they are. As a result, the Spiking Neural Networks (SNN) [5], the 3rd generation of neural networks which form the basis of neuromorphic systems, are capable of learning temporal dynamics of an event efficiently. The capability of faster online learning from less data, real time inferencing and optimal energy usage make these networks more suited for power-constrained devices like robots and drones.

In this paper, we have designed and implemented a novel spiking neural network using the idea of convolution and reservoir computing in order to classify human hand gestures. This SNN takes event-based input from a dynamic vision sensor (DVS) [6] camera, thus making the input data sparse and in form of spike trains. The network first learns the spatial features of the gesture by performing convolution over input data and then learns the temporal features by using a reservoir layer of spiking neurons. It has been trained on the IBM DVS gesture data set [7] and we have achieved Top-1 and Top-3 accuracy as 65% and 95% respectively over all 11 gesture classes. The achieved overall accuracy is 89% over 8 spatially non-overlapping classes. We have also observed that - *(i)* the network uses lesser number of parameters for learning, and *(ii)* the ratio of test accuracy to the number of parameters is high compared to existing state of the art.

The paper is organised as follows: Section II and III provide an overview of the state of the art with respect to gesture recognition and a background of SNN respectively. Section IV explains the components and functionalities of our proposed SNN model, followed by the implementation details & experimental results in Section V. We conclude with an outline of future works in Section VI.

II. Related works

A. Gesture recognition using DNN

A large body of work for identifying human gestures using state of the art deep neural networks (DNN) exists. The work

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

done by Tran et al. [1] is a significant one where the network tries to learn the spatio-temporal activity from a video using a deep 3D convolutional network (ConvNet). Simonyan et al. [8] produced better results by training a temporal ConvNet on optical flow instead of raw video frames. For classifying the action sequences, Baccouche et al. [9] proposed a method using Recurrent Neural Network (RNN) architecture with one hidden layer of long-short term memory (LSTM) [10] cells to learn dynamics of the spatial features extracted by a convolutional neural network (CNN). Shi et al. [11] used a 3D-ConvNet to capture 3D features of gestures and attached an LSTM network to capture the temporal patterns therein. Later works [2], [12] showed improvements by fusing different streams of features along with above techniques. Despite having high accuracy, these models are not the most efficient solutions as learning methods and inference frameworks of the conventional deep networks require huge amount of training data and are typically compute and energy intensive. Recently, developments have been made in DNN based frameworks to recognize gestures from a DVS data on mobile phones. Maro et al. [13] used online clustering-based method for learning the gestures and did not exploit the learning capabilities and processing architecture of SNN. Amir et al. [7] proposed a CNN architecture running on IBM True-North neuromorphic chip [14] to classify human gestures in real time - however, although the system utilizes benefits of neuromorphic chip, the training was not performed on True-North.

B. Gesture recognition using SNN

There are a number of research works on designing spiking neural networks capable of performing classification tasks. Diehl et al. [15] proposed a mechanism using *Spike Time Dependent Plasticity(STDP)* [16], *lateral inhibition and homeostasis* to recognize digits using MNIST dataset. Lee et al. [17] and Delbruck et al. [18] have proposed a supervised learning mechanism mimicking *back-propagation* of conventional ANNs that can efficiently learn to recognize the digits with higher precision. However, since SNN seems potent for learning temporal features, researchers have devoted efforts to capture the temporal dynamics within a video - for which, primarily, two methodologies are followedx.

The first one uses feed-forward layers of spiking neurons and a variant of back-propagation by defining an error function between desired and spiking activity. Shrestha et al. proposed *Spike LAYer Error Reassignment(SLAYER)* [19], a learning rule for back-propagating error to the previous SNN layers. This addresses the problem of approximating the derivative of the spike function that inherently brings in the question of biological plausibility. A recent prototype [20], [21] by the present authors jointly with BrainChip [22] shows a variant of the feed-forward approach.

The second method, based on the idea of *reservoir computing*, was independently researched by two research groups in the previous decade. These two models are Echo State Networks (ESN) [23] and Liquid State Machines (LSM) [24], with ESNs focussing on rate based neurons and LSMs on spiking neurons. ESN proposes driving a large recurrently connected network of artificial neurons with an input signal to produce a nonlinear response in the network nodes and obtain the desired output signal by training all these responses. On the other hand, in LSM, the temporal features of the input are extracted by a recurrently connected network of spiking neurons, called the *"liquid"*, the output of which is trained to produce certain desired activity based on some learning rule.

Over the years, both have evolved considerably and have been used in many temporal prediction problems, most of which are focussed on forecasting time series. A simplistic time series prediction on *Mackey-Glass System(MGS)* dataset by Zhang et al. [25] is one of the first notable works in this domain. Bellec et al. [26] further advanced it by incorporating the idea of LSTM into spiking domain and applying it on speech and digit recognition. By adding adaptive threshold for the neurons, they have balanced an excitation-inhibition ratio among the reservoir neurons, thus stabilising the network. They have also increased the computing and learning capability of the network by training with Back Propagation Through Time (BPTT) combined with a rewiring algorithm. They also proposed a Learning-to-Learn (L2L) scheme, which transfers prior knowledge to learn new tasks very quickly. However, these ideas remaine unexplored for learning spatial and temporal features together.

For gesture/action recognition, i.e. for classifying temporal and spatial features of a video simultaneously, works of Panda et al. [27] and Soures et al. [28] are most recent and noteworthy. In [27] the authors have applied a *Driven/Autonomous* approach for reservoir creation that can learn video activity with limited examples. The Driven Network is first trained with Recursive Least Square(RLS) [29] rule to provide a desired signal for the Autonomous model that learns to produce the desired activity from a supervised form of STDP [30]. The authors used frame difference and bounding box selection mechanism for encoding the input data but for real time purpose it is inherently compute intensive and resultant data is not sparse. While implementing it, we observed that driven/autonomous models are good at modeling temporal dependencies of a single-dimensional known time signal but lacks the potential to learn spatio-temporal features together as required for action recognition. Also, supervised STDP suffers from *catastrophic forgetting*. It is true that the network learns to recognize the action class very quickly, but it forgets them even faster.

On the other hand, Soures et al. [28] proposed a deep architecture of recurrently connected spiking neural network, each layer followed by an unsupervised winner-take-all (WTA) layer. This network is capable of understanding the interplay between the dynamic and high-dimensional information captured by the reservoir layer and encoding that to a low-dimensional representation by WTA layers. An *attention based neural mechanism* is used here to selectively process information in the reservoir layers. This work is beneficial with respect to memory and computational resources; however, the use of ANN-based spatial feature extraction dependent on pre-trained large models (like that of ResNet [31]) makes it unsuitable for

neuromorphic chips.

Before we describe our network architecture, a brief description of SNN and its learning mechanism, as described in section below, will be helpful.

III. SNN and its Learning Mechanism

In mammalian brains, billions of biological neurons are connected in a complex network via synapses. Stability is maintained via mutual excitation and inhibition processes unless some external stimuli interrupts the system. On receiving a stimulus, the membrane potential of the corresponding neuron increases and if a threshold is reached, a *spike* or *voltage surge* is generated which is carried forward to the next neuron. Spikes can be generated in bursts, in repeated fashion or as one time event depending upon the stimuli and nature of receiving neuron. Synapse, conductance of the cell body and other chemical interactions play important role in the subsequent processing of the spike(s). Rate at which these spikes occur carry information about the stimuli and the temporal relation (i.e. before or after) of spike-response between pre- and post-synaptic neurons make their synaptic bond stronger or weaker. In Hebbian learning, this is called *"fire together, wire together."*

SNNs, unlike classical ANNs, use such bio-plausible spiking models of neurons in order to remain closer to the mammalian brains compared to ANNs. Spiking neurons are energy efficient due to the inherent sparsity and asynchronous communication in form of spikes [32]–[34]. But SNNs need the input data to be converted into spike format before processing, unlike ANNs which can handle continuous valued input. SNNs are considered as the third generation of neural networks with formally proven computational capabilities comparable to that of regular ANNs [35].

A. Spiking Neuron Model

There are few mathematical models of spiking neurons having different levels of complexity and granularity - most detailed and complex being the Hodgkin-Huxley model [36] and most simplified (also, mostly used) being the Leaky Integrate and Fire (LIF) model [37]. In LIF, the membrane potential V, at any point in time, can be described by the differential equation 1. We use the same neuron model in our network.

$$\tau \frac{dV}{dt} = (V_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V) \quad (1)$$

The resting potential, V_{rest} is a point attractor towards which the membrane potential tends to evolve. Hence, without any input from other pre-synaptic neurons, membrane potential will remain at V_{rest}. Similarly, E_{exc} and E_{inh} represent the equilibrium potentials for excitatory and inhibitory synapses. Synapses are modelled as conductance values, namely, g_e, the excitatory conductance, and g_i, the inhibitory conductance. Excitatory pre-synaptic neurons increase the membrane potential, whereas, inhibitory pre-synaptic neurons tend to decrease it. When the membrane potential crosses a threshold, V_{thresh}, a spike is generated by the neuron. When a pre-synaptic neuron spikes, the conductance of the synapse is increased in magnitude. The dynamics of excitatory and inhibitory conductance are modelled as per equations 2 and 3 respectively.

$$\tau_e \frac{dg_e}{dt} = -g_e \quad (2)$$

$$\tau_i \frac{dg_i}{dt} = -g_i \quad (3)$$

B. Learning Rule

A spike event is mathematically modelled using the *Dirac Delta function*[1] at the time of spike. Due to the non-differentiability of this function, gradient based learning algorithms cannot be applied for SNN. In case of SNN, positive temporal correlation between pre- and post-synaptic spiking neurons strengthen the synaptic bond and that actually translates into learning (and in turn memory). This is completely unsupervised and local in nature. This type of learning is expressed via Spike Time Dependent Plasticity (STDP) [16]. STDP is a modification of classical Hebbian learning rule [38] improved with temporal asymmetry. A spiking neuron with STDP has been proved to be able to learn a linear dynamical system with minimum least square error [39]. For each synapse, we have a pre-synaptic trace x_{pre} which keeps track of the activity of the pre-synaptic neuron and a post-synaptic trace x_{post}, tracking activity of the post-synaptic neuron. Each trace decays exponentially with time as shown in the equations 4 and 5 with synaptic trace decay constants τ_{pre} and τ_{post}.

$$\tau_{pre} \frac{dx_{pre}}{dt} = -x_{pre} \quad (4)$$

$$\tau_{post} \frac{dx_{post}}{dt} = -x_{post} \quad (5)$$

When a spike occurs at pre- or post-synaptic neurons, it's trace is increased in magnitude by a constant value a. For each pre-synaptic firing, the synaptic weight is depressed with a value proportional to the post-synaptic trace and for each post-synaptic firing, it is potentiated with a value proportional to the pre-synaptic trace. The learning process of an arbitrary synapse is shown in Figure 1.

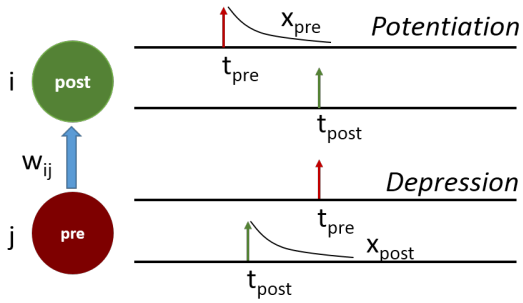


Fig. 1: Synaptic depression and potentiation in STDP

[1] https://en.wikipedia.org/wiki/Dirac_delta_function

The complete learning rule can be described as per equations 6 and 7.

$$\Delta w_{dep} = \eta_{dep}(x_{post} \times s_{pre}) \tag{6}$$

$$\Delta w_{pot} = \eta_{pot}(x_{pre} \times s_{post}) \tag{7}$$

s_{pre} and s_{post} represent spike of the pre- and post-synaptic neurons. In practice, equations 6 and 7 are vector equations where s_{pre} and s_{post} denote the spike vectors of a population of neurons and $\times$ is an outer product operation.

C. Lateral Inhibition and Homeostasis

Lateral Inhibition or Winner-Take-All [40], [41] is a bio-plausible non-linear computation mechanism employed in neural networks. It is mainly used in neural networks to enhance competition between neurons. In lateral inhibition, the first excited neuron which produces a spike, tries to stimulate other inhibitory neurons or directly inhibits one or more of them. In a learning scenario, a pattern which is being learnt will excite one or more neurons, which in turn will try to deactivate other neurons with the help of lateral inhibition, preventing them from learning the same pattern. In SNN world, this mechanism helps multiple neurons to compete and learn different things. We utilize a softer form of Lateral Inhibition like that of k-WTA, which is proven to be computationally less power intensive than a hard Lateral Inhibition [42] and leads to better shared feature selectivity in cortical pyramidal cells [43].

Homeostasis is the process of maintaining a steady internal state, widely observed in many biological systems like maintaining body temperature, blood sugar, pH levels etc. In this context, homeostasis of neuronal firing rate is meant to prevent the dominating effect of any particular neuron. We employ a rate homeostasis similar to that used in Diehl et al. [15], where threshold of neuronal firing is adapted so that continuous firing by the same neuron is discouraged. Our membrane threshold, V_{thresh} is a combination of a static threshold value, $V_{thresh-static}$ and a dynamic memory based component, θ which increases with each firing by a constant value and decays exponentially with time. The complete spiking mechanism is described by equations:

$$S(t) = \begin{cases} 1, & V(t) > V_{thresh} \\ 0, & V(t) \geq V_{thresh} \end{cases} \tag{8}$$

$$V_{thresh} = V_{thresh-static} + \theta(t) \tag{9}$$

$$\theta(t+1) = \begin{cases} \theta(t) + C, & S(t) = 1 \\ \theta(t), & S(t) = 0 \end{cases} \tag{10}$$

$$\tau_\theta \frac{d\theta}{dt} = -\theta \tag{11}$$

In the following section, we describe the network that we have designed and implemented for gesture recognition.

IV. Network Architecture

The proposed SNN architecture (refer Figure 2) consists of three main components: *(i)* the data pre-processing layer, *(ii)* the Convolutional Spiking layer (CSNN), and, *(iii)* the reservoir layer.

The data pre-processing layer performs compression and encoding on DVS data in order to render the computation faster, while the CSNN layer, whose design and action is inspired from CNN, contains multiple spiking layers which extract the spatial features from the input data. The feature extraction mechanism is hierarchical in nature with each layer concentrating on features of higher orders than the previous one. The reservoir layer helps in extracting temporal aspects of features from the convolutional layer. The convolutional features of a layer along with its reservoir features form an enriched feature set, which is then passed to a classifier for final recognition of the gestures. The figure shows multiple connected CSNN layers with one of them enlarged on the right side to describe the detailed composition. A description of each component is given below.

A. Data pre-processing layer

Data captured by DVS is usually stored in Address Event Representation (AER) format [44]. Any luminosity change (*event*) at a pixel is captured and represented as a quadruple [x, y, t, p] where (x, y) is the pixel co-ordinate, t is the timestamp of the event, and p signifies polarity of change in luminosity. In this work, we have used the IBM DVS dataset [2] (details given in Section V-A). This AER dataset contains events captured at very high temporal resolution ($\sim 10^9$). Though by definition, each of these records is a separate temporal event, it is found that a number of these can be safely considered redundant for our classification task, and we can compress the dataset to a lower time resolution scale without losing spatial information. Few experiments led us to a scaling down factor of 10^4 to derive a safe workable dataset. Within the evenly distributed time window of length 10000 between each A_i and A_{i+1} (in the original dataset), if multiple events or spikes occurr at a particular pixel, we consider it as a single spike occurring at x_i when the time axis is scaled down by a factor of 10^4 (in the derived dataset). This compression approach is illustrated in Figure 3a. The useability of this technique is emphasised in Figures 3b and 3c, with the former showing a snapshot of a set of events (AER records) in higher time resolution scale (as in the original dataset) and the latter showing the compressed version. Evidently, no significant spatial information are lost by incorporating the compression mechanism; instead it leads to faster processing time and less data handling. The dataset is captured with a resolution of 128×128 and after compression, the AER records are converted to multiple *spike-frames* of size 128×128, with each pixel having values 1 or 0 only.

B. Convolutional Spiking Layer (CSNN)

The Convolutional Spiking layer comprises of *class-wise filter blocks* along with a lateral inhibition based competition

[2]http://research.ibm.com/dvsgesture/

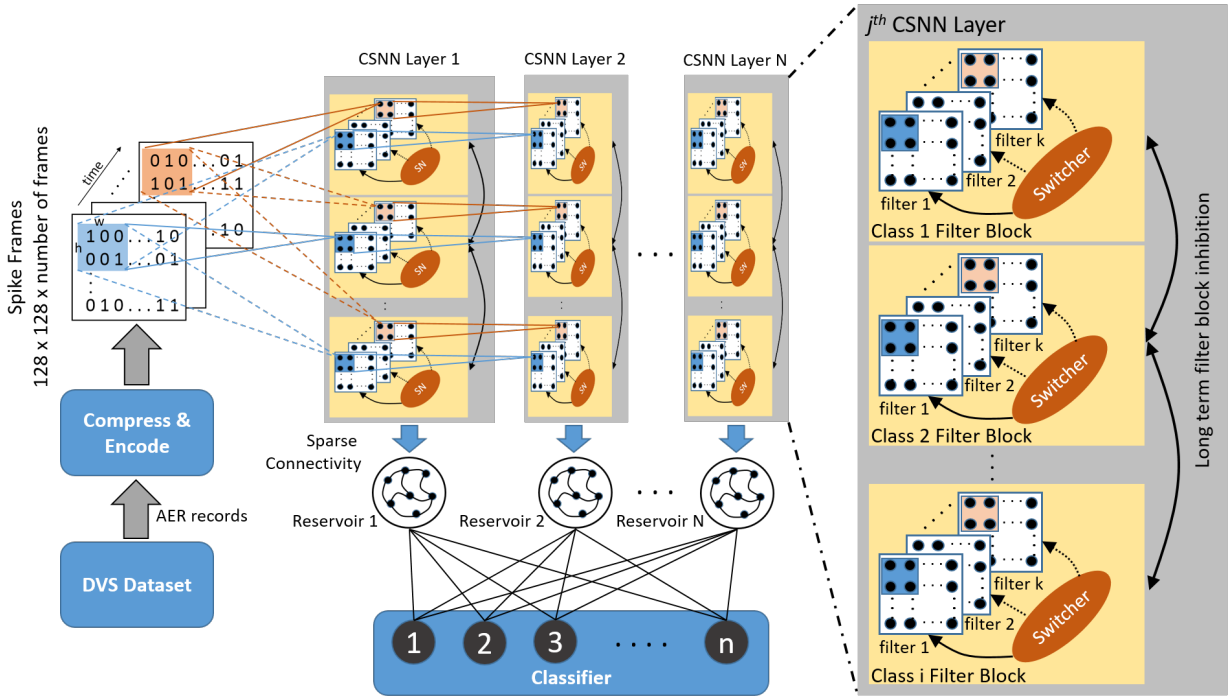
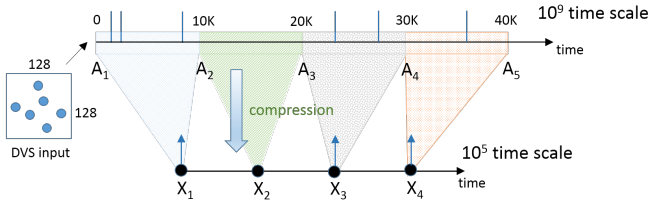
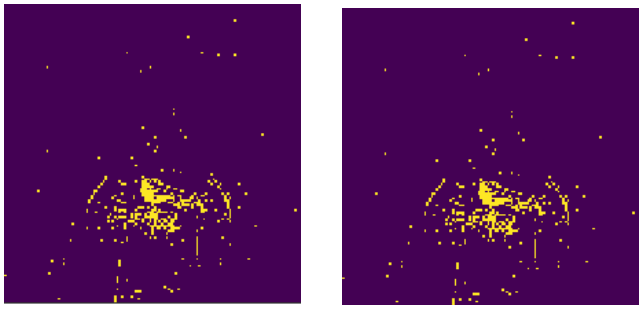


Fig. 2: Architecture of the CSNN & reservoir-based Network



(a) Compression mechanism



(b) Before compression

(c) After compression

Fig. 3: DVS data preprocessing.

mechanism between the filter blocks. The output from the data pre-processing module acts as the input to this layer in form of spike-frames. There can be multiple such convolutional layers in the architecture. Connectivity from input to the convolutional layer as well as from one convolutional layer to the next is same as that of a traditional CNN. These class-wise filter blocks contain multiple filters inside them and that helps in capturing spatially co-located patterns within the spike-frames of a single gesture class.

As shown in Figure 2, each spike frame of a gesture is connected to convolutional layers via a sliding window based connectivity. For this purpose, initially a sliding window of the size $w \times h$ pixels is chosen from within the input spike frame. Each input pixel belonging to this sliding window is connected to a single neuron of a filter block in the first convolutional layer. For the second neuron in the same filter block, the initial window is slided horizontally by a *stride* and each pixel in the new shifted window is connected to this second neuron. The process is repeated horizontally and then vertically till the entire spike-frame is connected to all the neurons in this filter. As more spike-frames come over time, similar sliding mechanism connects those frames with same neurons in the same filter. Once the first convolutional layer is connected to the input, the second and other consecutive layers can be connected to previous convolutional layers in the same fashion. The number of convolutional layers is a design choice, to be set according to the complexity of the spatial features present in the dataset.

As mentioned, *class-wise filter blocks* capture spatially co-located patterns within the spike-frames of a single gesture class. Instead of learning a 3D filter (spatial and temporal) from consecutive spike-frames, the proposed approach uses a switching mechanism. Among all filters of the same filter block, only one is activated at a time. The selection is made by a special LIF neuron node, called the *switcher* node, at periodic intervals. By applying inhibition, the switcher node forces all other filters to remain inactive for 'some' time steps, except the next filter in the block. This duration of inactivation depends entirely on the strength of inhibition, a configurable parameter, applied by the switcher node. After the period of inactivity,

all filters start to compete again and the one which spikes the most is considered as the winner, exactly as in the WTA mechanism mentioned in Section III-C. Based on the *decay time constant* of the switcher node, this process will repeat during the training time of convolutional filters. The switcher node ensures that all filters get a chance during the training phase and that spatially collocated but temporally separable features appear on different filters without getting mixed up. This is essential for spiking neurons to learn overlapping features. *This type of forced inhibition and selection mechanism is applicable to spiking neurons only to make sure that some of them learn a particular feature; unlike ANNs, where neurons are selected to learn by global error reducing mechanism like backpropagation.*

Another level of long-term inhibition exists between the class-wise filter blocks which ensures that during training phase, only one of the filter blocks is active at a given point of time for a given gesture video. This also ensures that multiple filter blocks are not trying to learn the same redundant pattern. Lateral inhibition among filter blocks allows them to compete for classes. Weights in the filter blocks are initialised randomly and one of the blocks wins for the first time for a particular gesture class. This win ensures that it will spike maximally only for that particular class during the later part of training. Once a filter block wins due to maximum initial spiking, an inhibition signal of higher strength is sent to other filter blocks preventing them form being activated.

There are two distinct benefits of this filter-block-wise inhibition mechanism:

(i) Since all the filter blocks are not active at a given time, the number of active convolutional neurons of a CSNN layer during training time for each gesture is reduced.

(ii) It allows us to set different periodicity (i.e. different decay time constant) for switcher nodes of different filter blocks according to its associated gesture. Switching periodicity is dependent on the total duration of gesture and different spatial patterns present therein. If multiple repetitive patterns occur within a short duration, switching periodicity for that particular filter block can be set to a small value. For example, right hand wave gesture repeats patterns more frequently than right hand clockwise rotation; hence the optimal switching periodicity for right hand wave is set at 10ms (by parameter tuning) whereas that for right hand clockwise rotation is 62ms.

During testing time, both long term inhibition between filter block as well as switching of filters within a block are removed as they are useful during training only.

C. Reservoir layer

The last component in the architecture is a reservoir layer where spiking activity of each CSNN layer (i.e. the spatial features present within the gesture) is forwarded to one reservoir. The temporal components of the gestures are captured explicitly by the spiking behaviour of the recurrently connected sparse reservoir.

The reservoir is a recurrent neural network where instead of training the recurrent connections, a population of neurons with cyclic connections and random sparse connectivity is constructed. At the core, reservoir is a form of Liquid State Machine (LSM). The output spikes of the reservoir captures the higher dimensional embedding of the spatio-temporal features present in the data. The output from the reservoir is connected to a readout layer of neurons and the synaptic weights between reservoir and readout are modified to get the desired classification.

Connectivity from convolutional filter blocks to the reservoir is considered as a sparse connection with random weights. Amount of sparsity is a hyper-parameter and can be tuned for optimal performance. The weight matrix of the reservoir is constructed as a random sample from uniform distribution with a specific density (in our case, 0.2). Reservoir weight matrix is constructed ensuring that it's spectral radius, i.e largest absolute eigen-value is less than unity. This helps in keeping the dynamics of the reservoir from falling into chaotic regimes. Neurons in the reservoir act as inhibitory as well as excitatory. The ratio of excitatory to inhibitory neurons is kept as 4:1 for our purpose.

V. Datasets, Implementation and Results

A. Datasets

The network described in Section IV was trained and tested on IBM DVS Gesture data set. The data set consists of a set of 11 hand and arm gestures of 29 people under three different lighting conditions captured using a DVS128 camera. Each *trial* consists of one person performing all 11 gestures sequentially in front of a static background. The gesture set includes hand waving (both arms), arm rotations (both arms, clockwise and counter-clockwise), arm rolling, hand clapping, playing guitar or drums in the air and some "other gestures" performed by that particular person. The three lighting conditions are combinations of natural light, fluorescent and LED lights, which were selected to control the effect of shadows and flicker noise from fluorescent lights on the DVS128. Each gesture lasts for about 6 seconds. To evaluate classifier performance, out of total 29 persons' data, 23 were marked as the training set and the remaining 6 were kept for testing.

B. Implementation

The network was implemented using *BindsNet v0.2.4* [45], a GPU based open source SNN simulator written in Python. The simulator itself being in a development stage had some bugs which have been handled. It was used primarily because it supports parallel computing unlike other available simulators like Brian, Nest etc. We also observed that BindsNet needs more memory footprint for simulating SNN networks compared to ANN frameworks (like TensorFlow) for ANN models of similar size as a result of which the networks may run at a lesser speed at times.

Table I summarizes the parameters used for implementing the neuron model and learning mechanism as described in

Section III. Most of these parameters are consistent with the values of their biological counterparts. The threshold voltage of neurons $V_{thresh-static}$ (as in Eqn. 9) is set at a comparatively high value to reduce spikes due to random noise. Also, the learning rate parameters η_{dep} & η_{pot} (as in Eqns. 6 & 7) are set in a manner such that features are best learnt in CSNN layer (as explained in Section V-C). Value of E_{exc} & E_{inh} (as in Eqn. 1) are kept same as V_{rest}. Also value of τ_e and τ_i (as in Eqns. 2 & 3) have been kept same as that of τ.

TABLE I: Parameters for neuron model & learning mechanism

Parameter	Value	Parameter	Value
$V_{thresh-static}$	-15.0 mV	V_{rest}	-65.0 mV
τ	100 ms	τ_θ	10^7 ms
τ_{pre}	20 ms	τ_{post}	20 ms
η_{dep}	10^{-3}	η_{pot}	10^{-2}

C. Results & Discussion

Spatial patterns learnt by corresponding neurons on different filter blocks are visualized in Figure 4. Each square block shows a 20×20 pattern learnt by a randomly chosen single neuron from a filter block. Three rows are shown in the figure, which represent learning by three such neurons from three different spatial positions of a filter block. Each column represent the pattern learnt by the corresponding neuron for a given class and there are 11 such columns. The patterns learnt are sparse and varied according to their spatial positions.

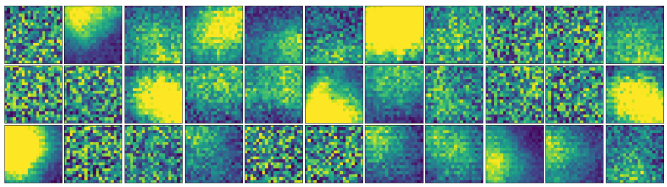
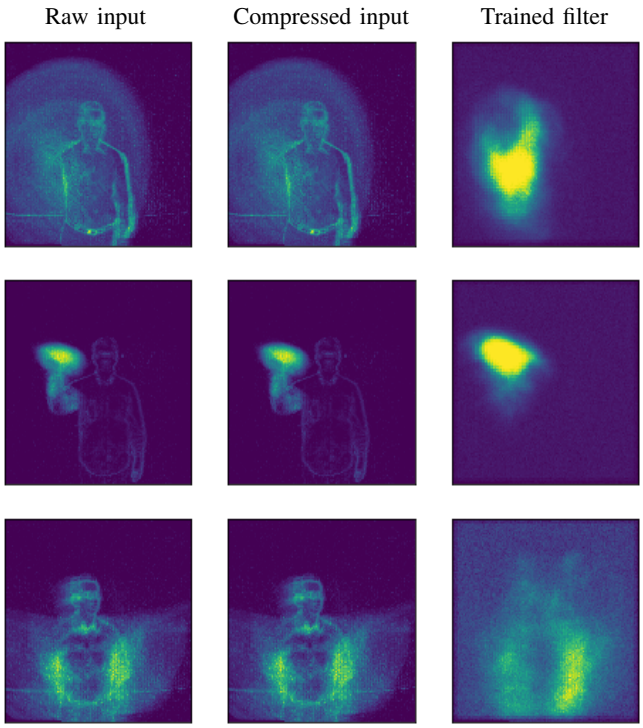


Fig. 4: Neuronal activities in CSNN filters

Table II shows the comparison between input, compressed input and the accumulated feature map of a filter layer of a filter block over the entire simulation time. First two rows show spatially overlapping classes namely *right hand clockwise rotation* and *right hand wave* while the third row represent one example from *other gesture* class. In the learnt feature map, it is seen that only key regions where the hand motion occurs are being learnt and less significant details like the features of the person who does the gesture are treated as noise and are not learnt.

Various experiments were conducted to determine the effectiveness of our approach with different hyper-parameters and settings. The results are summarized in Table III. All the experiments were conducted with a single convolutional spiking layer connected to a reservoir. In Experiment 1, a convolutional window size of 20×20 is taken, which is large enough to cover the activity regions in each frame of the

TABLE II: Visualisation of Data and feature learning



gesture video. Further, we used only a single filter per filter block (i.e. total 11 filters), without any switching mechanism. The resultant accuracy is 59% on the test data.

A separate run containing 8 out of 11 classes was executed. The 8 classes were chosen such that they do not contain any purely temporally separable classes (i.e. *right hand clockwise rotation* and *left hand clockwise rotation* are included but their counter clockwise versions are excluded). This increased the accuracy from 59% to 88% indicating that the network performs well while classifying gestures which are spatially separable.

For Experiment 3, we increased the number of filters in the convolutional filter blocks to 2 (i.e. total 22 filters), introducing the switching mechanism. Accuracy on the entire test data increased from 59% to 63% as the switching mechanism is beneficial for spatially overlapping classes where two competing filters are trying to learn two spatially collocated gestures. At the same time, accuracy in identification of the 8-class set reduced from 88% to 86% because these classes being distinctly spatially separable, benefit more if there are no competing filters. Experiments 2 and 4 are repetitions of experiment 1 and 3 respectively but with a smaller convolutional window (6×6), increasing the no. of learnable parameters. The best accuracy obtained is 65% on the entire test data and 89% on the spatially separable data. It is to be noted that the reservoir size i.e. the number of neurons inside the reservoir is 8000 in case of Experiment 1 & 3 while it is 12000 for Experiment 2 & 4. As window size is decresed from 20×20 to 6×6, thereby increasing the CSNN filter size from 13×13 to 62×62, the number of neurons in reservoir had to be increased for proper extraction of spatio-temporal

TABLE III: Experimental Results

Exp. no.	No. of filters	Window size (h,w)	Stride	Reservoir size	Training accuracy	All class test accuracy	8 class test accuracy	Total no. of parameters	Active params at each time step	Efficiency ratio	Top 3 accuracy
E1	11	(20, 20)	9	8000	72.8	59.2	88.3	0.8316M	0.1556M	0.711	95.51
E2	11	(6, 6)	2	12000	82.5	58.7	88.1	1.6542M	0.2703M	0.354	94.68
E3	22	(20, 20)	9	8000	73.8	63.2	86.5	1.5752M	0.2232M	0.401	90.03
E4	22	(6, 6)	2	12000	81.6	65.0	89.5	3.1764M	0.4087M	0.205	95.12

§The accuracy figures shown are in %-age, and the number of parameters are in millions

features from this input.

One interesting outcome of our experiments is the achievement of above-90% value for Top-3 accuracy for all cases, with best Top-3 accuracy being 95% in cases of Experiment 1 & 4. Clearly, there is a gap between overall accuracy (65%) and Top-3 accuracy (95%) and that can be attributed to the sensitivity of SNN to learnt weights. As the gesture videos have different durations, it may happen that one particular class is learnt more by the network thereby increasing corresponding weights in the corresponding filter blocks more than that of similar spatially overlapping classes. For example, the average duration of the *right hand counter clockwise* videos is lesser than that of *right hand clockwise* one. Hence, the latter gets more training time than the former and corresponding weights in the filterblock are slightly higher on its side. As a result, during testing, whenever the former class video appears, it makes the filter block corresponding to the second class to spike slightly higher than its own, resulting in wrong classification and thereby reducing overall accuracy from 95% to 65%.

The best state of the art result on IBM DVS Gesture dataset is 91% on a temporal cascade filters combined with 16-layered CNN architecture [7]. One surprising result we observed is that, even though Top-1 accuracy is lower than that of ANN based approaches, our model triumphs with respect to two metrices:

- number of parameters to be learned during training - this is directly related to the memory complexity of the runtime network and corresponding time and power consumption.
- ratio of test accuracy and number of parameters (in millions) - this is an indicator of the efficiency of the SNN network as it tells us how much accuracy we are able to achieve per parameter that is learnt.

These metrices are important as state of the art deep learning models, despite providing higher accuracy, are larger in size and thus have an increasing *number of parameters* and decreasing *accuracy per parameter ratio*. A standard ResNet-50 [31] Feature extraction architecture has nearly 20 Million parameters, with other state of the art networks nearing this value. Our network in all test configurations were under 4 Million parameters with max accuracy observed at 3.1 Million. Top-1 Accuracy of Resnet 50 on imagenet classification is around 79.26, making its accuracy to parameter ratio 0.0398 where as in all our experiments, lowest we have encountered is 0.2, roughly 5 times better. Accuracy to parameter ratio worsens for really large networks like VGG-16 and Resnet 152.

VI. Conclusion

In this paper we have discussed a novel spiking neural network to classify hand gestures using IBM DVS dataset. It learns both spatial and temporal features of the gestures and is capable of classifying those with standard accuracy.

As a continuation of this work, we will be experimenting further with layers, parameter values and different connection patterns in order to enhance the accuracy for spatially overlapping classes. We also plan to replace the logistic regression classifier in the last layer of the architecture with an SNN based classifier. Finally, we intend to run the network on available neuromorphic platforms like IBM TrueNorth [14], Intel Loihi [46], Brainchip Akida [22] etc. and replace the input DVS data set with live feed from DVS camera so that the network can learn and classify in real time.

References

[1] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[2] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4724–4733.

[3] S. Furber, "Large-scale neuromorphic computing systems," *Journal of neural engineering*, vol. 13, no. 5, p. 051001, 2016.

[4] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electronics*, vol. 1, no. 1, p. 22, 2018.

[5] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications." *Acta neurobiologiae experimentalis*, vol. 71, no. 4, pp. 409–433, 2011.

[6] "The dynamic vision sensor," https://inivation.com/dvs/, iniVation, 2018.

[7] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[8] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[9] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *International workshop on human behavior understanding*. Springer, 2011, pp. 29–39.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 802–810.

[12] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1933–1941.

[13] J.-M. Maro and R. Benosman, "Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities," *arXiv preprint arXiv:1811.07802*, 2018.

[14] "Ibm truenorth," https://www.ibm.com/blogs/research/tag/truenorth/, IBM.

[15] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[16] H. Markram, W. Gerstner, and P. J. Sjstrm, "Spike-timing-dependent plasticity: A comprehensive overview," *Frontiers in Synaptic Neuroscience*, vol. 4, p. 2, 2012.

[17] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in neuroscience*, vol. 12, 2018.

[18] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[19] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[20] S. Dey, A. Mukherjee, G. Bzard, and D. McLelland, "Demo: Human gesture recognition using spiking input on akida neuromorphic platform," *Neural Information Processing Systems (NeurIPS)*, 2019.

[21] "Human gesture recognition using spiking input on akida neuromorphic platform," https://ir.brainchipinc.com/press-releases/detail/90/brainchip-and-tata-consultancy-services-tcs-jointly, BrainChip Inc., and Tata Consultancy Services, 2019.

[22] "Brainchip unveils the akida development environment," https://www.brainchipinc.com/news-media/press-releases/detail/61/brainchip-unveils-the-akida-development-environment, Brainchip Inc.

[23] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[24] W. Maass, T. Natschlger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[25] A. Zhang, W. Zhu, and M. Liu, "Self-organizing reservoir computing based on spiking-timing dependent plasticity and intrinsic plasticity mechanisms," in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 6189–6193.

[26] 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 *Advances in Neural Information Processing Systems*, 2018, pp. 787–797.

[27] P. Panda and N. Srinivasa, "Learning to recognize actions from limited training examples using a recurrent spiking neural model," *Frontiers in neuroscience*, vol. 12, p. 126, 2018.

[28] N. Soures and D. Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," *Frontiers in neuroscience*, vol. 13, p. 686, 2019.

[29] D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, 2009.

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

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[32] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[33] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-mevents/s 22-pj/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, 2014, pp. 675–678.

[34] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[35] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

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

[37] L. Lapicque, "Recherches quantitatives sur lexcitation electrique des nerfs traite comme une polarization," *J. Physiol. Pathol. Gen.*, vol. 9, pp. 620–635, 1907.

[38] H. D.0, *The Organization of Behavior*. Wiley and Sons, 1949.

[39] R. E. Suri, "A computational framework for cortical learning," *Biological Cybernetics*, vol. 90, no. 9, pp. 400–409, 2004.

[40] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies in Applied Mathematics*, vol. 52, no. 3, pp. 213–257, 1973.

[41] M. Oster, R. Douglas, and S.-C. Liu, "Computation with spikes in a winner-take-all network," *Neural Computation*, vol. 21, no. 9, pp. 2437–2465, 2009.

[42] W. Maass, "On the computational power of winner-take-all," *Neural Computation*, vol. 12, no. 11, pp. 2519–2535, 2000.

[43] Z. Jonke, R. Legenstein, S. Habenschuss, and W. Maass, "Feedback inhibition shapes emergent computational properties of cortical microcircuit motifs," *Journal of Neuroscience*, vol. 37, no. 35, pp. 8511–8523, 2017.

[44] L. Camunas-Mesa, A. Acosta-Jimenez, T. Serrano-Gotarredona, and B. Linares-Barranco, "A digital pixel cell for address event representation image convolution processing," in *Bioengineered and Bioinspired Systems II*, R. A. Carmona and G. Linan-Cembrano, Eds., vol. 5839, International Society for Optics and Photonics. SPIE, 2005, pp. 160 – 171.

[45] H. Hazan, D. J. Saunders, H. Khan, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, "Bindsnet: A machine learning-oriented spiking neural networks library in python," *Frontiers in neuroinformatics*, vol. 12, p. 89, 2018.

[46] "Neuromorphic computing," https://www.intel.in/content/www/in/en/research/neuromorphic-computing.html, Intel Inc.