ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

RN-Net: Reservoir Nodes-Enabled Neuromorphic Vision Sensing Network

*Sangmnin Yoo, Eric Yeu-Jer Lee, Ziyu Wang, Xinxin Wang, and Wei D. Lu**

Neuromorphic computing systems promise high energy efficiency and low latency. In particular, when integrated with neuromorphic sensors, they can be used to produce intelligent systems for a broad range of applications. An event-based camera is such a neuromorphic sensor, inspired by the sparse and asynchronous spike representation of the biological visual system. However, processing the event data requires either using expensive feature descriptors to transform spikes into frames, or using spiking neural networks (SNNs) that are expensive to train. In this work, a neural network architecture is proposed, reservoir nodes-enabled neuromorphic vision sensing network (RN-Net), based on dynamic temporal encoding by on-sensor reservoirs and simple deep neural network (DNN) blocks. The reservoir nodes enable efficient temporal processing of asynchronous events by leveraging the native dynamics of the node devices, while the DNN blocks enable spatial feature processing. Combining these blocks in a hierarchical structure, the RN-Net offers efficient processing for both local and global spatiotemporal features. RN-Net executes dynamic vision tasks created by event-based cameras at the highest accuracy reported to date at one order of magnitude smaller network size. The use of simple DNN and standard backpropagation-based training rules further reduces implementation and training costs.

1. Introduction

Event-based cameras are neuromorphic vision sensors that produce visual signals as asynchronous spikes.[1] An event camera produces a spike when and only when a momentary pixel intensity difference exceeds a threshold, which can offer better energy efficiency and latency when compared with conventional cameras that produce data at a constant frame rate even when the scene is stationary during video recording.

Various visual tasks have been implemented using event-based cameras. Earlier datasets were generated by reproducing conventional image recognition tasks such as MNIST,[2] CIFAR-10,[3] and Caltech 101,[2] where stationary images were placed in front of the camera and moved around to create pixel intensity differences along time. Objects in the tasks are dynamic in their locations with their shape fixed over time, which requires networks to be capable of classifying objects regardless of their positional change. Beyond the positional dynamic movement, behaviorally more dynamic datasets, such as DVS128 Gesture,[4] N-CARS,[5] and DVS Lip[6] were later created by the event camera to fully utilize the camera's strength where the temporal evolutions of both the object's shape and movement are critical.

To process either the positionally dynamic datasets or the behaviorally and morphologically more dynamic datasets, many efforts have been made based on deep neural networks (DNNs) consisting of convolution and fully connected layers or spiking neural networks (SNNs).[4–24] However, DNNs require additional recurrent units which increase memory and training costs to process temporal information embedded in the datasets due to the lack of temporal processing capability and diminish the merits of asynchronous operation of the event-based camera due to its synchronized processing. SNNs allow temporal processing, however the necessity of backpropagation through time (BPTT) for proper training increases the training cost.[25] As the solution, feature descriptors such as time surface (TS) were proposed.[5,17,26] It is cost-effective and powerful in that it encodes only the latest temporal information and can be integrated with DNN structures, enabling gradient-based training. However, its latest information-dominant encoding fashion discards useful information prior to the last spike, which may be essential for features that evolve over a longer history.

In this article, we introduce a neural network architecture, reservoir nodes-enabled neuromorphic vision sensing network (RN-Net), that leverages simple reservoir layers and DNN blocks for temporal and spatial processing, respectively. Two reservoir layers are employed, with the one at the front encoding the temporally local information on the sensor by directly receiving asynchronous events (spikes) and the one at the back encoding the temporally global information without the expensive recurrent units, respectively, both leveraging native dynamics of reservoir

S. Yoo, E. Y. J. Lee, Z. Wang, X. Wang, W. D. Lu
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor 48109, MI, USA
E-mail: wluee@umich.edu

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202400265.

DOI: 10.1002/aisy.202400265

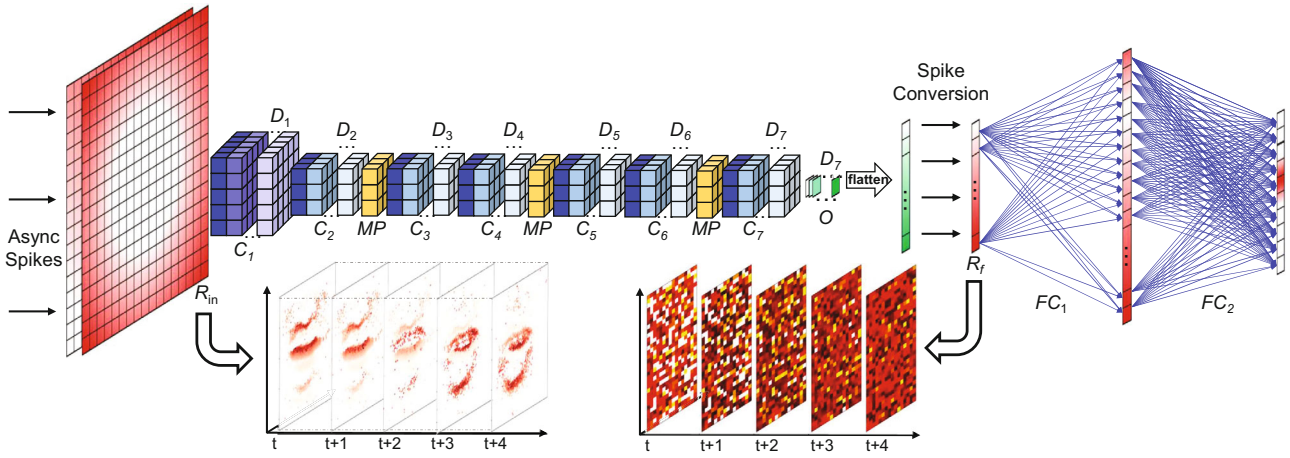

Figure 1. RN-Net structure. R_{in} and R_f are reservoir layers for local and global temporal feature encoding, respectively. Bottom left: outputs from R_{in} from a representative input in the DVS Lip dataset. Bottom right: outputs from R_f. Outputs from R_f are reshaped in 2D for better visualization. Deeper color in R_{in}, R_f and output layers of convolution (Conv) blocks and fully-connected (FC_{1-2}) layers represents a higher analog value. Hidden layers within Conv blocks are not presented. The DNN structure for DVS Lip dataset is representatively illustrated. C_N, D_N, MP, FC_N represent N-th Conv layer, kernel depth of N-th Conv layer, max pooling layer and N-th FC layer, respectively.

nodes (RNs). The DNN blocks find spatial information in data temporally processed by the reservoirs, resulting in spatiotemporal output in the end.

Specifically, the nature of short-term memory (STM) memristors implementing RNs allows RNs to work as native feature descriptions of temporal information in all prior spikes produced by the event camera or hidden layers, without dedicated memory and complementary metal-oxide-semiconductor (CMOS) logic circuits usually required by recurrent units or complex feature descriptor algorithms.[27–30] RNs in the network can be thought of as analogous to cells in a retina which directly sense and encode raw and asynchronous visual inputs, and transmit them to the brain.[31] Besides, the better encoding capability of RNs beyond the latest spike than TS that focuses only on the latest one enables RN-Net to perform better with a much smaller network size than others. (**Figure 1**) An example of the proposed RN-Net is shown in Figure 1.

The main contributions of this work can be summarized below: 1) We implement a neural network with multiple reservoirs and DNN blocks that process temporal and spatial information embedded in asynchronous event streams generated by event-based cameras, respectively. 2) On-sensor RNs based on STM memristors offer richer temporal spike encoding at a lower cost, which makes DNN blocks simpler, leading to more efficient operation and training. 3) RN-Net performs CIFAR10-DVS, N-Caltech 101, DVS128 Gesture, and N-CARS tasks at the highest accuracy reported to date and DVS Lip task at one of the highest accuracy at one order of magnitude lower network size than other networks with similar capacity.

2. Backgrounds

2.1. Event-Based Dataset

In the early years, conventional static vision tasks were reproduced by the event-based camera by moving images in front of the camera to make changes in pixel intensity.[2,3] Beyond the conversion, dedicated datasets have been created to maximize the strengths of event-based cameras.[4,6,32–35] Among them, we use CIFAR10-DVS, N-Caltech 101, DVS128 Gesture, N-CARS, and DVS Lip datasets in this work. The CIFAR10-DVS and N-Caltech 101 are object detection tasks, The DVS128 Gesture dataset is for human action recognition, the N-CARS is for car classification in a real-world setting, and the DVS Lip dataset is for word recognition based on lip motions of speaking participants. Data of the first two datasets are dynamic in object's location over time, while the rest of the datasets include temporally dynamic data in both shape and movement. The details of the datasets are summarized in **Table 1**. We split datasets that are not originally divided into train/test data 9:1.

2.2. Related Works

There have been various attempts to better execute the event-based tasks, utilizing DNNs consisting of convolution and fully connected layers, and SNNs have been demonstrated.[4–24] Given that DNNs are specialized in processing static spatial data, recurrent units, such as long short-term memory (LSTM),[36,37] gated recurrent unit (GRU)[38] and bidirectional gated recurrent unit (BiGRU)[6,18–20] are typically required to process temporal information hidden in the sequence (global) of momentary (local) features. The momentary features are processed synchronously in temporally local frames which are created either by simple accumulation of spikes within a prefixed time range[22] or using input representation algorithms like graph construction,[24,39] 3D point cloud,[21] event frame,[40] event spike tensor,[23] and voxel grid.[6,41] Synchronized processing requires storing and analyzing a large number of events as a pre-processing step, which diminishes the advantages of asynchronous and sparse spike generation features of event-based cameras. Recurrent units require storing multiple state data for each node, causing increased training costs.

Table 1. Details of datasets used for RN-Net demonstration.

Dataset	Train	Test	Category
CIFAR10-DVS	9000	1000	10 (Objects)
N-Caltech 101	7839	870	101 (Objects)
DVS128 Gesture	1077	264	11 (Actions)
N-CARS	15422	8607	2 (Car/Background)
DVS Lip	14896	4975	100 (Words)

SNNs store temporal information in the neuron dynamics using models such as leaky integrate-and-fire (LIF) neurons[9,13] and can be trained using gradient-based approaches such as BPTT.[25] To address the non-differentiability of spikes, surrogate approaches that replace spikes with neuron membrane potential have been developed.[42] However, BPTT requires backpropagation through both the network layers and time, which technically makes the network n times larger when unfolded in time, where n is the number of timesteps going back in time. As a result, BPTT is expensive to train. Other works have employed bio-inspired local learning rules, such as spiking timing dependent plasticity, for less training cost.[43] Although these methods result in significantly improved efficiency, the local learning rule is generally worse than BPTT in training quality.

To use spikes directly, feature descriptors such as TS[5,17,26] were proposed. TS stores only the last spike event for every pixel and converts the time to the last spike information into an analog value by resetting the amplitude of each node to 1 every time it receives a spike and then relaxes following a decay function:

$$S(t) = e^{-\frac{t-T(t)}{\tau}} \tag{1}$$

where $S(t)$ is an analog vector representing a node on the time surface, t is the current time, $T(t)$ is the time information of the last spike received by the node, and τ is a pre-defined time constant.[26,44] The TS conversion allows the encoded data in the analog surface to be processed with DNN and trained using gradient-based training while reducing memory and computing costs compared to other feature descriptors, since only the last spike instant needs to be recorded. However, since TS only stores the last spike, it cannot handle spatiotemporal features whose correlation is beyond its temporal neighbors, which is common in real-world problems. Additionally, storing the last spike timing information for each pixel (node), and calculating the analog state based on Equation (1) still incurs substantial costs. More advanced approaches such as Leaky Surface were subsequently developed to encode the temporal information beyond the last spike.[45] However, expensive pre-processing is still required, for example, to store the previous node state and time elapsed from the latest spike, and to calculate the new state for every pixel at every time instant.

2.3. Reservoir Nodes

We note that in a reservoir computing (RC) system, the reservoir maintains short-term memory and performs nonlinear transformation (encoding) of the temporal input data into the reservoir states, represented by the states of the reservoir nodes (RNs). RC systems have been efficiently implemented in hardware using devices such as memristors for vision (MNIST handwritten digits recognition), speech (NIST TI46), and Time-series forecasting (Mackey–Glass time series) tasks.[27,28] In these systems, the reservoir nodes encode the spikes' spatiotemporal information naturally following the internal device dynamics, without any external memory or arithmetic and logic units (ALUs). By leveraging the internal device and circuit dynamics to process temporal data, these implementations have shown excellent energy efficiency and performance.[27,28,46–48]

Generally, due to the STM property, RNs will be affected more strongly by the near history events and weakly by far history events, with the extent of the non-linearity determined by the internal RN time constant.[30,49] Inspired by this principle, we hypothesize that RNs can be directly used to encode temporal spike data similar to what TS aims to accomplish, but at a lower cost and better encoding capability beyond the last spike since RNs *non-linearly* accumulate all prior incoming spike information.

3. Methods

3.1. Event Encoding with Reservoir Nodes

A reservoir node can be implemented using only a single STM memristor, making hardware implementation very light weight.[27,28] In a STM memristor, the node state (i.e., device conductance) is natively excited by the incoming spikes and relaxes in between the spikes,[30,49] as described by the following equation:

$$G_t = P_c * (G_{\max} - G_{t-1}) * \delta_{\text{spk}}(t) + G_{t-1} * e^{-\frac{1}{\tau}} \tag{2}$$

where G_t is the node state at time t, P_c is a potentiation factor, $G_{\max}$ is the upper bound of the node state, τ is the characteristic relaxation time constant, and $\delta_{\text{spk}}(t)$ is a delta function representing a spiking event:

$$\delta_{\text{spk}}(t) = \begin{cases} 0, & \text{when no spike.} \\ 1, & \text{when receiving spike.} \end{cases} \tag{3}$$

$\delta_{\text{spk}}(t)$ can represent incoming events from an event-based camera or spikes from preceding layers within the network.

Figure 2 shows the comparison of the RN approach implemented with a single STM memristor based on Equation (2) versus the TS approach introduced in Section 2.2. Both approaches convert the spiking patterns into an analog state that can be processed by subsequent DNN blocks. Compared to TS encoding which resets the state to 1 after each spike, the RN implementation allows longer-term history to be represented in the state in a non-linear fashion. For example, at $t = 50$, the TS output is identical to that at $t = 30$, since both values were reset to 1, 10 time steps earlier (at $t = 20$ and 40, respectively). However, this representation missed the differences in the two temporal sequences (e.g., an additional preceding spike at $t = 15$). In contrast, the RN implementation clearly differentiates the two cases as all inputs before the current time are accumulated non-linearly. Combined

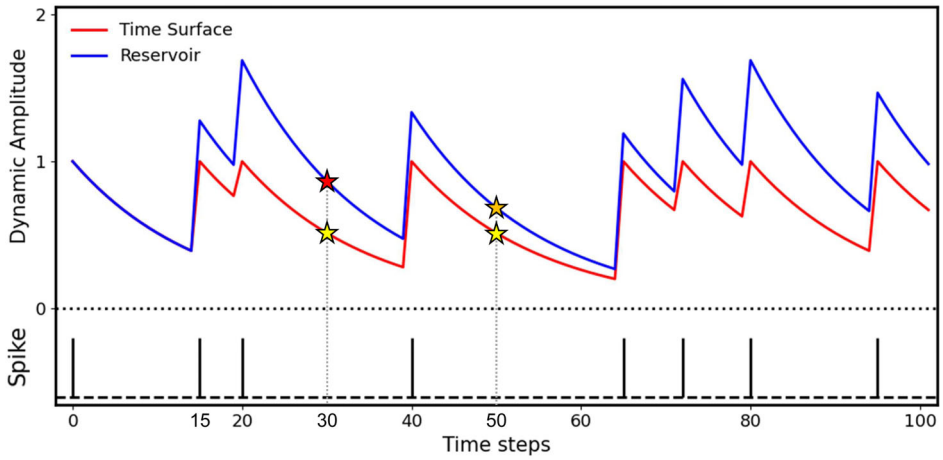

Figure 2. Dynamics of a reservoir node (blue) and a time surface node (red) under identical spikes (black) shown below.

with the hardware efficiency, for example, Equation (2) is natively implemented in a single device owing to internal device physics (ionic and electronic dynamic processes)[27,28] without the need of additional dedicated memory (to store t and $T(t)$) and ALUs (to execute Equation (1)), we believe the RN-Net to be a suitable solution for asynchronous, real-time processing of event data.

The proposed RN-Net operates by directly taking asynchronous spikes as they are generated. The memristor-based RNs located in the first part of RN-Net, each of which is connected to a pixel, autonomously transform the temporal spikes into analog values following Equation (2) on the sensor in real-time, analogous to visual inputs encoded by cells in a retina.[31] The states of the RNs are retrieved only when the network operates a forward pass, as shown in Figure 1. In the process, the temporal events from a pixel output are encoded as the state of the corresponding RN node in the R_{in} layer, and the spatial information is preserved in the R_{in} layer since each RN node is independently connected to a corresponding pixel. As a result, the states of all nodes in the R_{in} layer encode the spatiotemporal information of the inputs. The values of the R_{in} node states can then be read out and processed using conventional DNN blocks. Notably, owing to the discrete memristor dedicated to a pixel, RN-Net is free of parasitic resistance-capacitance effects of the crossbar format that is common for memristor applications. In the second half of RN-Net, after the convolution blocks and the spikes conversion layer, the spatiotemporal features are again embedded in the spikes. Instead of using the number of spikes to perform classification in the subsequent FC layers, we chose to use another RN layer (R_f layer in Figure 1) to encode the spatiotemporal features discovered by the first half of the network. At the end of a video input clip, the states of R_f capture the long-term temporal information and are supplied to the FC layers for classification.

Figure 3 shows examples of five temporally consecutive states of the input layer (R_{in}) RNs, responding to asynchronous events from two representative datasets. The states are obtained at a constant time interval (i.e., 30 ms). Frequent input spikes lead to stronger RN states due to the excitation term (the first term of Equation (2)), while RN states natively relax for inputs that are

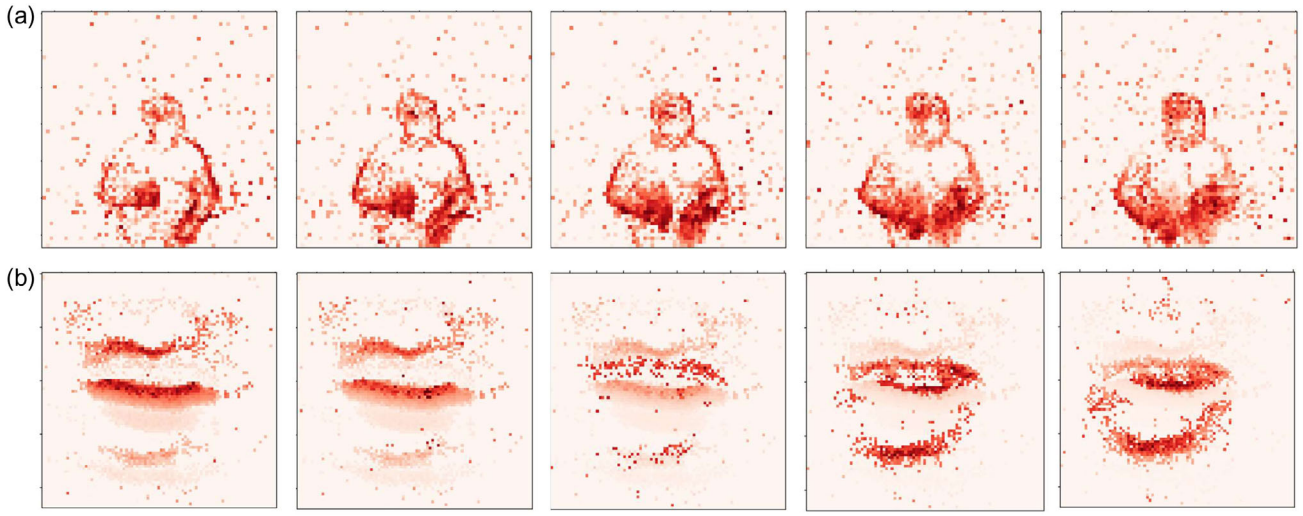

Figure 3. Temporally consecutive states of the input reservoir nodes, responding to asynchronous events in the a) DVS128 Gesture and b) DVS Lip datasets. Each state is retrieved at a constant time interval of 30 ms. Deeper red represents a higher amplitude value of an RN state.

temporally too far from the current time instant due to the internal decay term (the second term of Equation (2)). These properties allow RNs to capture temporal features of inputs independently and the R_{in} layer possesses spatial features when the RNs are gathered, as shown in Figure 3. For example, in Figure 3b, both the speaking motion and movement (top to bottom) of the lips can be captured by the RNs' states in the R_{in} layer. Temporally fast movements are reflected in a single moment (e.g., the 3rd plot in Figure 3b), while temporally slow movements are reflected in multiple moments captured at the different time instants (e.g., the first two plots).

3.2. Model Architecture

The RN-Net is formed sequentially by an input RN layer (R_{in}) as a feature descriptor for temporally local encoding, convolution (Conv) layers (C_N) for spatial processing, a spike conversion (SC) layer, another RN layer (R_f) for temporally global encoding, and multi-layer perceptron (MLP) (F_N) for classification. Figure 1 illustrates the overall architecture of the proposed network for the DVS Lip dataset.

The R_{in} layer has as many RNs as the output dimension of the event-based camera. For example of the DVS128 Gesture dataset with $2 \times 128 \times 128$ pixels, where 2 represents the polarity of the events, 32768 ($2 \times 128 \times 128$) independent nodes are used to form the R_{in} layer. Each RN in the R_{in} layer processes input spikes asynchronously from a corresponding pixel in the event camera in real time without any preprocessing, following Equation (2). Examples of the outputs generated by R_{in} are shown in Figure 3.

Depending on the application, different temporal resolutions can be chosen by adjusting P_c and τ of Equation (2) and the interval of R_{in} state acquisitions. A shorter time constant τ and more frequent acquisitions (shorter interval) produce temporally finer outputs, while a longer time constant and less frequent acquisitions (longer interval) produce spatially more detailed outputs owing to more input spikes and the slower internal decay.[6] Accordingly, a shorter acquisition interval results in more frequent activations for the following layers. The convolution layers (C_1–C_7 with depths D_1–D_7 in Figure 1) then process the encoded spatiotemporal features in R_{in} at a constant time interval (i.e., 30 ms). Similar to Conv blocks in conventional DNNs, the output O from the Conv layers reflects the spatial features existing in the R_{in} states, and in this case, the spatiotemporal features embedded in the spike stream within the time interval.

Output O is then flattened and sent to the spike conversion SC layer. We use a predetermined threshold to convert the outputs O from the Conv layers into spikes in the SC layer. Spikes generated after the SC layer are shown in **Figure 4**. Thus, the spikes after SC represent the local spatiotemporal features captured by the R_{in} layer and the Conv layers, and become much sparser compared with the original spiking inputs.

The spikes from the SC layer are then supplied to the RNs in the R_f layer for global spatiotemporal feature encoding and subsequent classification in MLP. Similar to the RNs in the R_{in} layer, RNs (e.g. also implemented with STM memristors) in R_f naturally potentiates/relaxes with the presence/absence of spikes from the SC layer. The state of R_f thus represents the historical (global) spatiotemporal features by encoding the processed local spatiotemperal features over time, and is then used by the subsequent fully-connected (FC) layers of MLP (FC_1–FC_2 in Figure 1) to perform final classification functions. Similar to the RNs in the R_{in} layer, RN states in R_f are retrieved at a certain time interval (i.e., 0.3 s, which is longer than that used in R_{in}) over the whole video clip and fed to the FC layers.[50] The final decision is made based on accumulated potentials through multiple FC feedforwards. (i.e., 5)

3.3. Training Method

We train RN-Net with standard backpropagation, using the output potentials calculated by FC_2 at the end of data presentation. Unlike BPTT that calculates error and gradient across each timestep,[25] error and gradient of RN-Net is calculated only once for one input training video, leading to lower training cost. To address the non-differentiability arising from spike generation in the SC layer, we adopted surrogate gradient for the SC layer. Since the surrogate is used only in one layer of the network, we expect the error introduced by this approach to be lower when compared with SNN approaches which require surrogates at all layers.

We also applied several data augmentation techniques, such as random/center crop, horizontal flip and Gaussian noise to minimize overfitting effects. In the case of random/center crop, we referred to existing works.[6,19] During training, we center-cropped the original input dimension (128×128) to 96×96

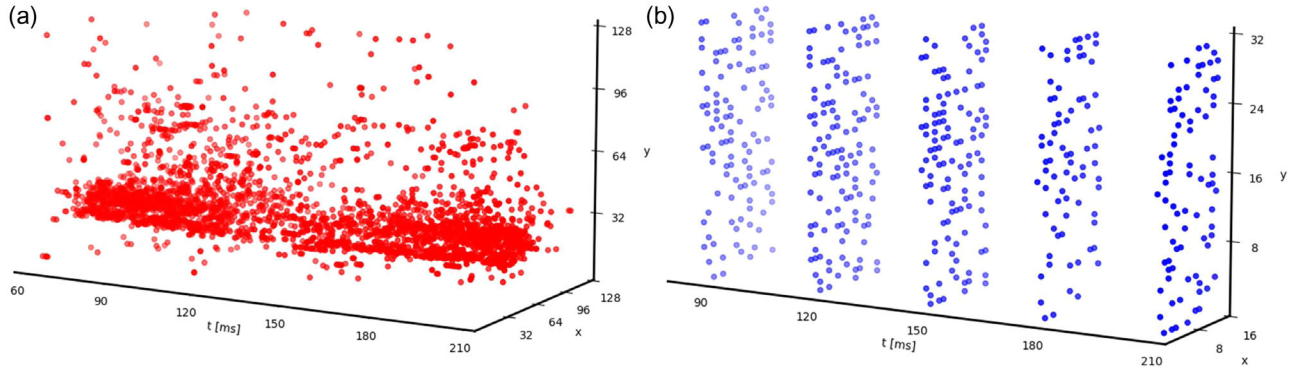

Figure 4. a) Visualization of asynchronous input spikes and b) spikes generated after the spike conversion (SC) layer.

and then random cropped them to 76×76, while the inputs are directly center-cropped to 76×76 in the testing phase. Horizontal flip with a probability of 0.5 and Gaussian noise with standard deviation of $5e^{-4}$ were used during training for the same purpose. These techniques helped generalize the input data and reduce overfitting during training. These techniques were used only during training of RN-Net for the DVS Lip.

4. Experimental Section

4.1. Experiment Setup

The DVS128 Gesture dataset consists of repetitions of the same motion of a participant along the clip, while lip motions in the DVS Lip dataset are not repetitive, as shown in Figure 3. As a result, a fraction of a clip should be sufficient to classify the action in DVS128 Gesture, while the whole clip is necessary for DVS Lip classification. Based on this understanding, we only used the first 1.5 s for all videos in the DVS128 Gesture dataset during training and inference, corresponding to 10% of the longest video (15.5 s). For CIFAR10-DVS, N-Caltech 101, and N-CARS datasets, the first 0.6 s, 0.3 s, and 90 ms are used, respectively when the videos are longer than the lengths for the same purpose. The DVS Lip dataset has variable input lengths due to the irregular length of the words and the unique speaker's traits, with most video clips (99%) having lengths between 0.75 and 1.5 s. To make the data size regular across the dataset, we chose to keep the captured R_{in} outputs at 50 by simply applying null (no spike) for clips shorter than 1.5 s and clipping videos longer than 1.5 s during DVS Lip training and inference.

For all the datasets, the same potentiation factor P_c (0.5) and time constant τ (60 ms) in Equation (2) are used for the RNs in R_{in}. The values are referred to prior works.[27,28] The states of R_{in} are captured every 30 ms and sent to the subsequent Conv layers. We note that we used one set of state retrieval frequency and time constant chosen for all the datasets to demonstrate the potential of RN-Net, although tuning the time constants and feeding data at different intervals for a specific task can further enhance the performance.[27]

Due to the different input dimensions and the different number of categories in the datasets, the convolution layers and FC layers in RN-Net are configured accordingly. The detailed network configurations for the datasets are summarized in **Table 2** and **3**.

The output of a DNN block consists of a convolution and a pooling layer (if exists), following:

$$O_N = f(BN(MP(C(I_N)))) \tag{4}$$

where O_N is the output of N-th layer, $f(x)$ is a ReLU activation function, $BN(x)$ is a batch normalization function, $MP(x)/C(x)$ is a maxpooling/convolution operation, and I_N is input of the N-th layer.[51]

After the last convolution layer, the outputs are converted to spikes for subsequent temporal feature encoding at R_f.[52] The SC layer generates spikes by comparing the analogue values in O (Figure 1) with a pre-fixed threshold value, set as 0.3.

RNs in the R_f layer encode the global spatiotemporal features. P_c and τ for the RNs in R_f are set as 0.3, 2 and 0.1, 2 s for DVS Lip and other datasets, respectively. A longer τ is used in R_f than in R_{in} to process the longer temporal correlations. The RN states in the R_f layer are retrieved every 0.3 s for both DVS128 Gesture and DVS Lip, every 0.12 s for CIFAR10-DVS, every 60 ms for N-Caltech 101, and every 30 ms for N-CARS (corresponding to

Table 2. Network configuration of RN-Net for DVS Lip dataset. The last column represents the output dimension of the network for datasets other than N-Caltech 101. The 6th convolution layer and the following maxpooling layer are exclusively applied for N-Caltech 101 due to its larger input dimension (240×180) than the others (128×128).

DVS128 Gesture				
Layer	Kernel Size	Out Channel	Pad/Stride	Output Dim
R_{in}	–	2	–	128×128
MaxPool	2	2	0/2	64×64
Conv1	3	64	0/1	62×62
MaxPool	3	64	0/2	30×30
Conv2	3	128	1/1	30×30
MaxPool	3	128	0/2	14×14
Conv3	3	256	0/1	12×12
Conv4	3	512	1/1	12×12
MaxPool	3	512	0/2	5×5
Conv5	3	512	0/1	3×3
MaxPool	3	512	0/1	1×1
Conv6	3	512	0/1	–
MaxPool	3	512	0/1	–
R_f	–	512	–	–
FC1	–	512	–	–
FC2	–	11	–	–

Table 3. Network configuration of RN-Net for DVS Lip dataset.

DVS Lip				
Layer	Kernel Size	Out Channel	Pad/Stride	Output Dim
R_{in}	–	2	–	76×76
Conv1	5	64	0/2	36×36
Conv2	3	128	1/1	36×36
MaxPool	3	128	0/2	17×17
Conv3	3	128	1/1	17×17
Conv4	3	256	1/1	17×17
MaxPool	3	256	0/2	8×8
Conv5	3	256	1/1	8×8
Conv6	3	512	1/1	8×8
MaxPool	3	512	0/2	3×3
Conv7	3	512	0/1	1×1
R_f	–	512	–	–
FC1	–	512	–	–
FC2	–	100	–	–

5 R_f acquisitions for all other than N-CARS (3)). In the case of clips shorter than a fixed video length, we let RNs in R_f continue relaxing after the last meaningful input without padding the input data with new spikes (e.g., through repeating the clip). These relaxed states are still captured following the normal schedule and fed to the classification layers. Examples of RN states at different time instants in DVS Lip task are shown in **Figure 5**, along with the spikes they receive from the SC layer. For better visualization, the data are reshaped in 2D format. As shown in the figure, even without any new inputs (e.g., after 0.9 s for Data2), the RN states do not immediately decay to zero due to the slow decay term in Equation (2), and the evolution of the RNs still represents useful information. This approach also simplifies training and inference processes and allows the system to handle inputs with irregular lengths.

MLP consists of two FC layers, an activation function, and a batch normalization layer, following:

$$O_F = FC(f(BN(FC(I)))) \tag{5}$$

where O_F is the output of the block, $f(x)$ is a ReLU activation function, $BN(x)$ is a batch normalization function, $FC(x)$ is an FC layer, and I is the input to the FC layer. In RN-Net, the output of MLP is used as the final output.

The Pytorch framework[53] was used for all the experiments with methods described earlier. SoftMax function was chosen to calculate the probability of an output neuron in the final output, and the neuron with the largest probability was selected as the classification result. During training, the cost was derived by categorical cross-entropy function.[54] ATan was selected as a surrogate gradient calculation for the SC layer during training.[52] We also used Adam optimizer[55] with an initial learning rate between $1e^{-2}$ and $1e^{-5}$ and a weight decay between 0 and $1e^{-3}$ as the model optimizer and ReduceLROnPlateau with a factor of 0.9, patience between 1 and 3, threshold between $1e^{-4}$ and $1e^{-6}$, minimum learning rate between $1e^{-4}$ and $1e^{-7}$ as the learning scheduler, and selected a set of hyper-parameters based on the best performances. Batch sizes of 32 and 150 epochs were used. All experiments are performed on an Intel Xeon Gold 6226R and an NVIDIA A40.

4.2. Experimental Results

RN-Net shows excellent performance (classification accuracy) on all the datasets. Comparisons of classification accuracy and model size (the number of parameters) with existing networks are shown in **Figure 6** and **Table 4** and **5**. The accuracies of networks listed in Table 5 were obtained from,[6] and the parameter size of ResNet variants are obtained from.[56] RN-Net outperforms all existing networks on CIFAR10-DVS, N-Caltech 101, and N-CARS tasks with much smaller network size, while only using the first 10 percent of the longest clip on DVS128 Gesture. We attribute the improved performance of RN-Net even at a smaller network size to the RN layers' powerful capability to effectively encode temporal information hidden in the event streams. The spatiotemporal features captured at the R_{in} and R_f layers allow more efficient processing by the subsequent DNN blocks and FC blocks, respectively. For DVS Lip, RN-Net achieves top 2 accuracy, only behind MSTP[6] which is a multi-branch network with different input channels generating frames in different temporal granularity, and employs Voxel Grid

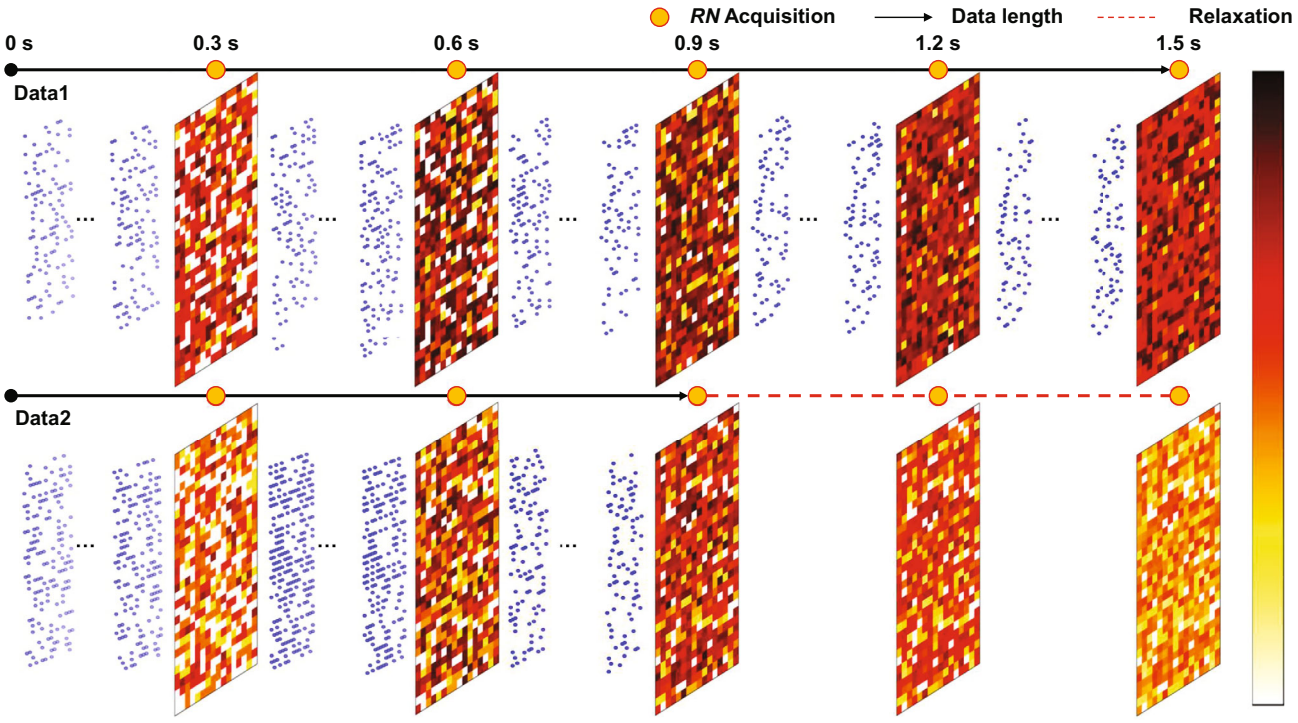

Figure 5. Visualization of outputs from R_f over the whole 1.5 s clip, along with spikes generated from the spike conversion layer. For Data2 whose input video length is only 0.9 s, the RN states will continue to relax and still used for classification.

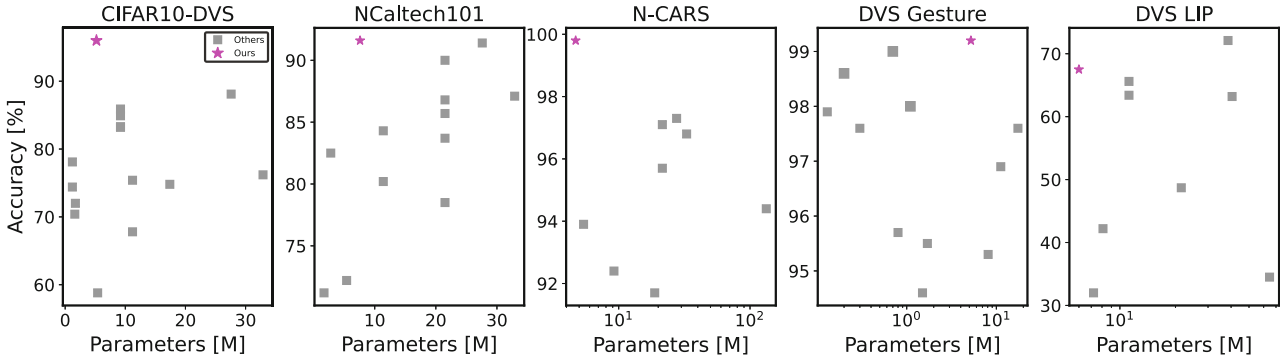

Figure 6. RN-Net on CIFAR10-DVS, N-Caltech101, N-CARS, and DVS Lip compared with existing works in classification accuracy and network size.

Table 4. Performance and Comparison of RN-Net on CIFAR10-DVS, N-Caltech 101, DVS128 Gesture, and N-CARS datasets. Data of RN-Net are in bold.

Model	Backbone	Params [M]	CIFAR10-DVS [%]	N-Caltech 101 [%]	DVS Gesture [%]	N-CARS [%]
HATS[5]	Linear SVM	–	52.4	64.2	–	90.2
Gao et al.[59]	FLNN	5.4	58.8	72.2	–	93.9
tdBN[12]	ResNet-19	11.2	67.8	–	96.9	–
SlideGCN[60]	GCN	–	68.0	76.1	–	93.1
LIAF-Net[13]	–	1.6	70.4	–	97.6	–
TA-SNN[9]	–	1.7	72.0	–	98.6	–
SEW-ResNet[10]	Wide-7B-Net	1.2	74.4	–	97.9	–
PLIF[11]	–	17.4	74.8	–	97.6	–
MVF-Net[61]	ResNet-18 + 34	32.9	76.2	87.1	–	96.8
TCJA-TET-SNN[7]	–	9.2	83.3	82.5	99.0	–
PSN[62]	VGG11	9.2	85.9	–	–	–
MST[63]	Swin-T	27.6	88.1	91.4	–	97.3
N-ImageNet[64]	ResNet-34	21.5	–	86.8	–	–
She et al.[65]	–	1.7	–	71.2	98.0	–
MatrixLSTM+E2VID[66]	ResNet-34	21.5	–	85.7	–	95.7
ACE-BET[67]	ResNet-34	21.5	–	90.0	–	97.1
Event Clouds[21]	PointNet	8.1	–	–	95.3	–
Asynet[68]	VGG13	133	–	–	–	94.4
Ours	**RN-Net**	**5.2**	**96.0**	**91.6**	**99.2**	**99.8**

Table 5. Comparison of existing models and RN-Net on Lip reading dataset including DVS Lip. The accuracy values are from.[6] Data of RN-Net are in bold.

Model	Input	ACC [%]	Params [M]	Backbone	Preprocess	Local encoding	Global encoding
DFTN[18]	Video	63.2	40.5	ResNet-18	–	DFN	BiGRU
Feng et al.[19]	Video	63.4	11.2	ResNet-18	–	–	BiGRU
Martinez et al.[20]	Video	65.5	11.2	ResNet-18	Greyscaling	–	Multi-scale TCN
Event Clouds[21]	Event	42.2	8.1	PointNet	Random Sampling	3D point cloud	3D point cloud
EV-Gait-3DGraph[24]	Event	32.0	7.2	–	OctreeGrid Filtering	3D-Graph	N/A
EV-Gait-IMG[22]	Event	34.5	64.6	–	Event Noise Cancelling	Image-like	N/A
EST[23]	Event	48.7	21.5	ResNet-34	Normalized Time Stamp	MLP (Grid)	N/A
MSTP[6]	Event	72.1	38.5	ResNet-18	Multiple Time-Scaling	Voxel Grid	BiGRU
Ours	**Event**	**67.5**	**6.0**	**–**	**–**	**Reservoir**	**Reservoir**

as the feature descriptor and using larger BiGRU and ResNet-18 network architectures. By comparison, RN-Net uses a single branch and a much lighter DNN network, without expensive feature description, preprocessing, and recurrent units that can lead to significant hardware and latency overheads.

Interestingly, we found that all prior works that showed success for the DVS Lip task are based on DNNs using optimized input representation techniques, and SNN-based works are more successful on the others. This observation seems to suggest that conventional SNNs work well for tasks with temporally coarse and repetitive features, but suffer from tasks where both temporally fine-grained features and global temporal features are critical, such as DVS Lip. We hypothesize that the use of RNs with different time constants to encode temporal features and integrated DNN blocks to capture spatial features in RN-Net allow it to better process spatio-temporal features at both local and global scales. Similar concepts, when implemented in SNNs, may help SNN performance for similar tasks.

4.3. Ablation Study

The effects of the RN in R_{in} and R_f layers on model performance are investigated in an ablation study by replacing each RN layer with a TS or temporal average pooling (TAP, only for R_f) layer without modifying the rest of the network. (**Table 6**) In both tasks, the case with both RN layers shows the best performance, while the case with TS at both places shows the worst performance. However, replacing R_f with TS degrades the performance more than that of R_f for the DVS128 Gesture task, while the DVS Lip task shows the opposite tendency. We attribute this difference to different characteristics of the datasets. Data in DVS128 Gesture are repetitive along the clip, which can tolerate less precise input encoding because there are opportunities to capture the lost information at a different time instant. In contrast, in DVS Lip information at different time instants is unique, which makes input encoding more critical for the network performance. Interestingly, although the use of RNs achieves the best results for both datasets, the use of TAP in the R_f layer achieves better results than the use of TS for the DVS Lip task, while the use of TS is better for the DVS128 Gesture task. We speculate that this is caused by the characteristics of TS, which completely discards far history that is critical for the DVS Lip dataset. In contrast, the repetitive inputs in DVS128 Gesture alleviate this problem of TS implementation, where its capability to encode local temporal information outperforms TAP. As a result, this study illustrates that the RNs' capability to capture both local and global temporal dynamics allows networks based on them (i.e., RN-Net) to achieve higher accuracy for both tasks.

Table 6. Performance comparison of different R_{in}/R_f layer configurations for DVS128 Gesture and DVS Lip tasks with the original RN-Net (bold).

Local encoding R_{in}	Global encoding R_f	Gesture [%]	Lip [%]
Time Surface	Time Surface	96.2	44.5
Time Surface	Reservoir Node	97.7	49.7
Reservoir Node	Time Surface	97.0	61.0
Reservoir Node	**Reservoir Node**	**99.2**	**67.5**
Reservoir Node	Temporal AvgPool	96.6	62.5

4.4. Power Estimation

The power efficiency largely depends on implementation methods of RNs, using either a memristor or a resistor-capacitor (RC) unit.[27,28,48] In the following sections, the power of a proposed hardware system running RN-Net for DVS128 Gesture and DVS Lip is representatively estimated based on an existing memristor technology and reported values.[49]

4.4.1. Spike Encoding

One physical RN device (i.e., a memristor) is assigned to one pixel of the event-based camera. Every time the pixel senses an intensity difference larger than the pre-fixed value, it produces a spike (event), which is input to the dedicated RN. Then, the RN natively encodes temporal information of the spikes following the Equation (2).

The total spike encoding energy consumed by RNs in the R_{in} layer during a clip can be calculated following:

$$E_{\text{encoding,in}} = \sum_{n=1}^{N} {V_{\text{pulse}}}^2 * G(x_n, y_n, p_n) * t_{\text{pulse}} \tag{6}$$

where E_{encoding} is the total energy consumed in the R_{in} layer, N is the total number of input spikes in the video clip, V_{pulse} is the (fixed) amplitude of the input spike, $G(x_n,y_n,p_n)$ is the conductance of the RN device assigned to the pixel located at x_n, y_n, with polarity p_n, and t_{pulse} is the (fixed) time duration of the spike.

Similarly, in R_f, an RN device is assigned to a neuron in the output layer of the convolution blocks. The total spike encoding energy of R_f during a clip is calculated as:

$$E_{\text{encoding},f} = \sum_{i=1}^{N_f} {V_{\text{pulse}}}^2 * G(N_i) * t_{\text{pulse}} \tag{7}$$

where $G(N_i)$ is the conductance of the RN device assigned with the i-th neuron in the output layer of convolution blocks along a clip. N_f is the total number of spikes from the output layer during the state retrieval window.

Considering the minimum time interval (1 μs) of events for typical event-based cameras,[4,6] we set t_{pulse} to 1μs for both R_{in} and R_f. V_{pulse} and G_{max} are set to 1.5 V and 100 μS, according to.[49]

We simulated the average total energy consumption of R_{in} and R_f for a typical clip in DVS128 Gesture and DVS Lip tasks. Using the total energy and the total number of spikes, the average energy consumption per spike was also derived. The results are shown in **Table 7**.

4.4.2. State Retrieval

The states of RNs in R_{in} and R_f are retrieved for the forward-pass every 30 and 300 ms, respectively. The total retrieval energy of R_{in} and R_f is calculated as:

Table 7. Total encoding energy and the energy per spike for spike encoding in R_{in} and R_f, for a typical DVS128 Gesture and DVS Lip video clip.

Dataset	R_{in}/R_f	Total Energy [μJ]	Energy/Spike [pJ]
DVS128 Gesture	R_{in}	15.0	156.2
	R_f	1.1	123.3
DVS Lip	R_{in}	0.3	22.8
	R_f	0.7	185.7

$$E_{\text{retrieval,in}} = \sum_{p=0}^{1}\sum_{y=1}^{128}\sum_{x=1}^{128} {V_{\text{read}}}^2 * G(x,y,p) * t_{\text{read}} \tag{8}$$

$$E_{\text{retrieval},f} = \sum_{i=1}^{D_7} {V_{\text{read}}}^2 * G(N_i) * t_{\text{read}} \tag{9}$$

where $E_{\text{encoding,in}}$ and $E_{\text{encoding},f}$ are the energy consumption for the state retrieval of R_{in} and R_f, respectively, V_{read} is the amplitude of the read pulse, $G(x,y,p)$ is the conductance of a node located at x_n,y_n, with polarity p_n in R_{in}, $G(N_i)$ is the conductance of a node assigned to the i-th node in R_f, and t_{read} is the duration of the read pulse.

Different from spike encoding, the read operations do not induce conductance change. For state retrieval, the V_{read} amplitude was set to 0.5 V according to,[49] and pulse duration was set as 1 μs for both R_{in} and R_f layers.

Based on the energy per read, the number of read retrievals and the number of nodes in both reservoir layers, the average total energy of RN state retrieval during the whole video clip, energy per state retrieval, and energy per node were calculated. The values are presented in **Table 8**.

4.4.3. Data Conversion

The SC layer does not require analog-digital conversion because the SC layer takes digital values from the convolution blocks and outputs a binary value using a threshold value within the digital domain. Analog-digital converters (ADCs) are however needed at the state retrieval process, which converts the retrieved analog values into digital values for the following digital DNN blocks. To be conservative, 8-bit ADCs between the RN output and the digital DNN blocks were considered to provide sufficient precision during conversion.

For the DVS128 Gesture dataset, the number of RNs in the R_{in} layer is 8192 ($2 \times 64 \times 64$) and the number of retrievals is 50. Thus, the total number of ADC operations is 409600 during 1.5 s running time. It requires at least a 273KSPS (kilo samples per second) ADC. For the R_f layer, 512 RNs are necessary, and the states are retrieved five times during 1.5 s running time. It requires 2560 ADC operations during a video, corresponding to a 1.7KSPS ADC. To perform conservation estimate, we chose the power of a commercial 8 bit-ADC (ADC7040 ultra-low power SAR ADC from TI), which consumes 171 μW for 274.7KSPS of the task.

For the DVS Lip dataset, 11552 ($2 \times 76 \times 76$) RNs are in the R_{in} layer and the total number of ADC operations is 577600 (11552×50) during 1.5 s running time, which demands at least a 385KSPS ADC. For the R_f layer, the same ADC used for DVS128 gesture dataset is needed to support the same structure. If we utilize the same commercialized ADC, 171 μW are additionally consumed for DVS Lip dataset. Furthermore, we expect that the power consumption will become less with less precision ADCs (e.g., 6-bit or 4-bit) which are commonly used in neuromorphic circuits.

4.4.4. Overall System Power Estimation

The total energy of the RN-Net hardware system can be calculated by summing up the energy used by the R_{in}/R_f layers and ADCs calculated above, plus the energy used for the DNN blocks to run a clip for DVS128 Gesture and DVS Lip. To calculate the energy of the DNN blocks, we derived the number of MAC operations for running a video clip in RN-Net, according to Table 3. The number of MAC operations during a video clip was calculated based on the number of MAC operations in the convolution and fully-connected layers, multiplied by the number of forward passes (i.e., 50) through these layers during the clip. We use published, conservative TOPS/W value (i.e., 2 TOPS/W from the Google Edge TPU[57]) to estimate the energy used by the DNN blocks since these blocks can be implemented in any digital accelerator. By adding all the energy costs, the total energy consumed by RN-Net per video clip, and the average power of RN-Net for DVS128 Gesture and DVS Lip tasks are estimated at 10.3 and 11.6 mW, respectively, and more details are presented in **Table 9**.

Table 8. Total retrieval energy during a DVS128 Gesture and DVS Lip video clip, and the energy per state retrieval and per node, for R_{in} and R_f RN layers.

Dataset	R_{in}/R_f	Total Energy [nJ]	Energy/ Retrieval [nJ]	Energy/ Node [pJ]
DVS128 Gesture	R_{in}	2234.9	44.7	1.4
	R_f	24.6	4.9	9.6
DVS Lip	R_{in}	178.7	3.6	0.3
	R_f	33.4	6.7	13.0

Table 9. Total energy and average power of the RN-Net system, along with the total number of MAC operations in the DNN blocks when running DVS128 Gesture and DVS Lip tasks.

Dataset	Convolution/ Forward-pass [MOPS]	Fully-connected [MOPS]	Whole Network [GOPS]	Energy [mJ]	Power [mW]
Gesture	608.7	2.7	30.4	15.4	10.3
Lip	686.3	2.7	34.3	17.3	11.6

5. Conclusion

In this work, we propose a hybrid reservoir/DNN network, reservoir nodes-enabled neuromorphic vision sensing network (RN-Net), for asynchronous event-based vision processing. The use of dynamic reservoir nodes enables efficient spatiotemporal encoding of both local and global features at different hierarchies in real time and eliminates external memory and logic/recurrent units. RN-Net achieves the highest classification accuracy on CIFAR10-DVS, N-Caltech 101, DVS128 Gesture, and N-CARS and one of the highest DVS Lip classification accuracy with a much lighter network structure. The reservoir nodes can be efficiently implemented with STM memristors, taking advantage of internal device physics to perform signal processing. The low hardware and training costs of RN-Net make it an attractive option for event-camera applications. Additionally, new devices, such as optical neural transistors, can potentially both sense optical inputs and process temporal information in one device,[58] allowing better power-efficiency in vision processing following approaches discussed here.

Acknowledgements

This work was supported in part by the Semiconductor Research Corporation (SRC) and Defense Advanced Research Projects Agency (DARPA) through the Applications Driving Architectures (ADA) Research Center and in part by the National Science Foundation under Grants CCF-1900675 and ECCS-1915550.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

event-based camera, memristor, neuromorphic, reservoir computing, SNN

Received: April 5, 2024
Published online:

[1] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, D. Scaramuzza, *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 154.
[2] G. Orchard, A. Jayawant, G. K. Cohen, N. Thakor, *Front. Neurosci.* **2015**, *9*, 00437.
[3] H. Li, H. Liu, X. Ji, G. Li, L. Shi, *Front. Neurosci.* **2017**, *11*, 00309.
[4] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, D. Modha, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Piscataway, NJ **2017**.
[5] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, R. Benosman, in *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2018**.
[6] G. Tan, Y. Wang, H. Han, Y. Cao, F. Wu, Z. J. Zha, in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2022**, 2022–June 20062.
[7] R.-J. Zhu, Q. Zhao, T. Zhang, H. Deng, Y. Duan, M. Zhang, L.-J. Deng, *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, 3377717.
[8] S. Deng, Y. Li, S. Zhang, S. Gu, in *Inter. Conf. on Learning Representations (ICLR)*, IEEE, Piscataway, NJ **2022**, pp. 1–17.
[9] M. Yao, H. Gao, G. Zhao, D. Wang, Y. Lin, Z. Yang, G. Li, in *2021 IEEE/CVF International Conf. on Computer Vision (ICCV)* Montreal, QC, Canada **2021**, pp. 10201–10210.
[10] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, Y. Tian, *Adv. Neural Inf. Process. Syst.* **2021**, *25*, 21056.
[11] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, Y. Tian, *Proc. IEEE International Conf. Comput. Vis.* **2021**, *1*, 2641.
[12] H. Zheng, Y. Wu, L. Deng, Y. Hu, G. Li, in *35th AAAI Conf. on Artificial Intelligence, AAAI 2021*, virtual, 02, **2021**, Vol. *12B*, p. 11062.
[13] Z. Wu, H. Zhang, Y. Lin, G. Li, M. Wang, Y. Tang, *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6249.
[14] J. Kaiser, H. Mostafa, E. Neftci, Synaptic plasticity dynamics for deep continuous local learning (DECOLLE), **2020**, https://www.frontiersin.org/articles/10.3389/fnins.2020.00424.
[15] A. Kugele, T. Pfeil, M. Pfeiffer, E. Chicca, Efficient processing of spatio-temporal data streams with spiking neural networks, **2020**, https://www.frontiersin.org/articles/10.3389/fnins.2020.00439.
[16] S. B. Shrestha, G. Orchard, *Advances in Neural Information Processing Systems* **2018**, 2018-December 1412.
[17] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, R. B. Benosman, *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1346.
[18] J. Xiao, S. Yang, Y. Zhang, S. Shan, X. Chen, in *Proc. - 2020 15th IEEE Inter. Conf. on Automatic Face and Gesture Recognition, FG 2020*, IEEE, Piscataway, NJ **2020**, pp. 364–370.
[19] D. Feng, S. Yang, S. Shan, X. Chen, in *2021 IEEE Inter. conf. on Multimedia & Expo Workshops (ICMEW)*, Shenzhen, China, **2020**, pp. 1–2.
[20] B. Martinez, P. Ma, S. Petridis, M. Pantic, in *ICASSP, IEEE Inter. Conf. on Acoustics, Speech and Signal Processing – Proc.*, IEEE, Piscataway, NJ **2020**, 2020-May 6319.
[21] Q. Wang, Y. Zhang, J. Yuan, Y. Lu, in *Proc. - 2019 IEEE Winter Conf. on Applications of Computer Vision, WACV 2019*, IEEE, Piscataway, NJ **2019**, pp. 1826–1835.
[22] Y. Wang, B. Du, Y. Shen, K. Wu, G. Zhao, J. Sun, H. Wen, in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2019**, 2019-June 6351.
[23] D. Gehrig, A. Loquercio, K. Derpanis, D. Scaramuzza, in *Proc. of the IEEE Inter. Conf. on Computer Vision*, IEEE, Piscataway, NJ **2019**, 2019-October 5632.
[24] Y. Wang, X. Zhang, Y. Shen, B. Du, G. Zhao, L. Cui, H. Wen, *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3436.
[25] P. Werbos, Backpropagation through time: What it does and how to do it, **1990**, http://ieeexplore.ieee.org/document/58337/?reload=true.
[26] A. Grimaldi, V. Boutin, S. Hoi Ieng, R. Benosman, A robust event-driven approach to always-on object recognition, *TechRxiv*, July **2023**.
[27] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, W. D. Lu, *Nat. Commun.* **2017**, *8*, 1.
[28] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, W. D. Lu, *Nat. Electron.* **2019**, *2*, 480.
[29] J. Moon, Y. Wu, W. D. Lu, *Neuromorphic Comput. Eng.* **2021**, *1*, 014006.
[30] T. Chang, S. Hyun Jo, W. Lu, *ACS Nano* **2011**, *5*, 7669.

[31] R. H. Masland, *Neuron* **2012**, *76*, 266.
[32] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, Y. Andreopoulos, in *2019 IEEE/CVF International Confernce on Computer Vision (ICCV)*, Seoul, Korea (South) **2019**, pp. 491–501.
[33] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, D. Scaramuzza, *Int. J. Robotics Res.* **2017**, *36*, 142.
[34] M. Gehrig, W. Aarents, D. Gehrig, D. Scaramuzza, *IEEE Robotics and Automation Letters*, **2021**, *6*, 4947.
[35] T. Serrano-Gotarredona, B. Linares-Barranco, *Front. Neurosci.* **2015**, *9*, 00481.
[36] S. Hochreiter, J. Schmidhuber, *Neural Comput.* **1997**, *9*, 1735.
[37] A. Graves, ArXiv, **2013**, 1–43, https://doi.org/10.48550/arXiv.1308.0850.
[38] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, in *2014 Conference on Neural Information Processing Systems Deep Learning and Representation Learning Workshop*, Montreal, QC, Canada, December **2014**, pp. 1–9.
[39] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, Y. Andreopoulos, in *Proc. the IEEE Inter. Conf. on Computer Vision*, IEEE, Piscataway, NJ **2019**, 2019-Octob 491.
[40] H. Rebecq, T. Horstschaefer, D. Scaramuzza, in *British Machine Vision Conf. 2017, BMVC 2017*, London, UK, September **2017**.
[41] A. Z. Zhu, L. Yuan, K. Chaney, K. Daniilidis, in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, IEEE, Piscataway, NJ **2019**, 2019-June 1694.
[42] E. O. Neftci, H. Mostafa, F. Zenke, *IEEE Signal Processing Magazine*, **2019**, *36*, 51.
[43] C. Lee, P. Panda, G. Srinivasan, K. Roy, *Front. Neurosci.* **2018**, *12*, 00435.
[44] R. Tapiador-Morales, J.-M. Maro, A. Jimenez-Fernandez, G. Jimenez-Moreno, R. Benosman, A. Linares-Barranco, *Sensors* **2020**, *20*, 3404.
[45] M. Cannici, M. Ciccone, A. Romanoni, M. Matteucci, in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, IEEE, Piscataway, NJ **2019**, 2019-June 1656.
[46] J. Li, C. Zhao, K. Hamedani, Y. Yi, in *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA **2017**, pp. 3439–3446, ISBN 2161-4407.
[47] M. Farronato, P. Mannocci, M. Melegari, S. Ricci, C. M. Compagnoni, D. Ielmini, *Adv. Mater.* **2022**, 2205381, https://doi.org/10.1002/adma.202205381.
[48] H. Fang, B. Taylor, Z. Li, Z. Mei, H. H. Li, Q. Qiu, in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, **2021**, pp. 361–366, ISBN 0738-100X.
[49] C. Du, W. Ma, T. Chang, P. Sheridan, W. D. Lu, *Adv. Funct. Mater.* **2015**, *25*, 4290.
[50] M. L. Alomar, M. C. Soriano, M. Escalona-morán, V. Canals, I. Fischer, C. R. Mirasso, J. L. Rosselló, *IEEE Trans. Circuits Syst. II: Express Br.* **2015**, *62*, 977.
[51] S. Ioffe, C. Szegedy, in *32nd Inter. Conf. on Machine Learning, ICML 2015, Lille*, France, July **2015**, Vol. *1*, p. 448.
[52] J. K. Eshraghian, X. Wang, M. Bennamoun, M. Ward, W. D. Lu, G. Lenz, *Proceedings of the IEEE*, **2023**, *111*, pp. 1016.
[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Vol. *32*. Curran Associates, Inc., **2019**, pp. 8024–8035, https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.
[54] Z. Zhang, M. R. Sabuncu, *Advances in Neural Information Processing Systems* **2018**, 2018-Decem 8778.
[55] D. P. Kingma, J. L. Ba, in *3rd Inter. Conf. on Learning Representations, ICLR 2015 – Conf. Track Proc.*, San Diego, CA, USA, May, **2015**, pp. 1–15.
[56] M. C. Leong, D. K. Prasad, Y. T. Lee, F. Lin, *Appl. Sci.* **2020**, *10*, 557.
[57] Coral, Edge tpu performance benchmarks, **2020**, https://coral.ai/docs/edgetpu/benchmarks/, (accessed May 20, 2023).
[58] Y. Ni, L. Yang, J. Feng, J. Liu, L. Sun, W. Xu, *SmartMat* **2023**, *4*, e1154.
[59] S. Gao, G. Guo, H. Huang, X. Cheng, C. L. P. Chen, *IEEE Access* **2020**, *8*, 45974.
[60] Y. Li, H. Zhou, B. Yang, Y. Zhang, Z. Cui, H. Bao, G. Zhang, in *Proc. of the IEEE/CVF Inter. Conf. on Computer Vision (ICCV)*, IEEE, Piscataway, NJ **2021**, pp. 934–943.
[61] Y. Deng, H. Chen, Y. Li, *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 8275.
[62] W. Fang, Z. Yu, Z. Zhou, D. Chen, Y. Chen, Z. Ma, T. Masquelier, Y. Tian, *Advances in Neural Information Processing Systems*, **2023**, *36*, 53674.
[63] Z. Wang, Y. Fang, J. Cao, Q. Zhang, Z. Wang, R. Xu, in *Proc. of the IEEE/CVF Inter. Conf. on Computer Vision (ICCV)*, IEEE, Piscataway, NJ **2023**, pp. 1761–1771.
[64] J. Kim, J. Bae, G. Park, D. Zhang, Y. M. Kim, in *Proc. of the IEEE/CVF Inter. Conf. on Computer Vision (ICCV)*, IEEE, Piscataway, NJ **2021**, pp. 2146–2156.
[65] X. She, S. Dash, S. Mukhopadhyay, in *Inter. Conf. on Learning Representations*, **2022**, https://openreview.net/forum?id=bp-LJ4y_XC.
[66] M. Cannici, M. Ciccone, A. Romanoni, M. Matteucci, *Computer Vision – ECCV 2020* (Eds: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm) Springer International Publishing, Cham, **2020**, pp. 136–152, ISBN 978-3-030-58565-5.
[67] C. Liu, X. Qi, E. Y. Lam, N. Wong, *IEEE Access* **2022**, *10*, 55638.
[68] N. Messikommer, D. Gehrig, A. Loquercio, D. Scaramuzza, **2020**, http://rpg.ifi.uzh.ch/docs/ECCV20_Messikommer.pdf.