frontiers in Neuroscience

ORIGINAL RESEARCH
published: 09 February 2018
doi: 10.3389/fnins.2018.00023

Feature Representations for Neuromorphic Audio Spike Streams

Jithendar Anumula, Daniel Neil†, Tobi Delbruck and Shih-Chii Liu*

Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland

Event-driven neuromorphic spiking sensors such as the silicon retina and the silicon cochlea encode the external sensory stimuli as asynchronous streams of spikes across different channels or pixels. Combining state-of-art deep neural networks with the asynchronous outputs of these sensors has produced encouraging results on some datasets but remains challenging. While the lack of effective spiking networks to process the spike streams is one reason, the other reason is that the pre-processing methods required to convert the spike streams to frame-based features needed for the deep networks still require further investigation. This work investigates the effectiveness of synchronous and asynchronous frame-based features generated using spike count and constant event binning in combination with the use of a recurrent neural network for solving a classification task using N-TIDIGITS18 dataset. This spike-based dataset consists of recordings from the Dynamic Audio Sensor, a spiking silicon cochlea sensor, in response to the TIDIGITS audio dataset. We also propose a new pre-processing method which applies an exponential kernel on the output cochlea spikes so that the interspike timing information is better preserved. The results from the N-TIDIGITS18 dataset show that the exponential features perform better than the spike count features, with over 91% accuracy on the digit classification task. This accuracy corresponds to an improvement of at least 2.5% over the use of spike count features, establishing a new state of the art for this dataset.

Keywords: dynamic audio sensor, spike feature generation, exponential kernels, recurrent neural network, audio word classification

OPEN ACCESS

Edited by:
Jeffrey L. Krichmar,
University of California, Irvine,
United States

Reviewed by:
Thomas Nowotny,
University of Sussex, United Kingdom
Garrick Orchard,
National University of Singapore,
Singapore

****Correspondence:***
Jithendar Anumula
anumula@ini.uzh.ch

†Present Address:
Daniel Neil,
BenevolentAI, New York, NY,
United States

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *07 September 2017*
Accepted: *11 January 2018*
Published: *09 February 2018*

Citation:
Anumula J, Neil D, Delbruck T and Liu S-C (2018) Feature Representations for Neuromorphic Audio Spike Streams. Front. Neurosci. 12:23. doi: 10.3389/fnins.2018.00023

1. INTRODUCTION

The event processing methods for the asynchronous spikes of event-based sensors such as the Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008; Berner et al., 2013; Posch et al., 2014; Yang et al., 2015) and the Dynamic Audio Sensor (DAS) (Liu et al., 2014; Yang et al., 2016) fall roughly into two categories: either by the use of neural network methods or machine learning algorithms. These methods have been primarily developed for event-based vision sensors and with the availability of DVS datasets (Orchard et al., 2015; Serrano-Gotarredona and Linares-Barranco, 2015; Barranco et al., 2016), performances of these methods can be compared.

In recent years, the field of deep learning has seen major developments leading to networks that achieve state-of-art performance on complex tasks such as speech recognition and visual object recognition (Schmidhuber, 2014; LeCun et al., 2015). With event-based sensors finding increasing relevance in event-driven artificial sensory or cognitive systems, there has been a new effort in interfacing these sensors with these powerful machine learning networks. However, deep learning frameworks typically use frame-based data. To interface the output of the event-based sensors to the

deep network, there are two alternative methods. The first method is to present the spikes to spiking deep networks as has been reported (Farabet et al., 2012; Pérez-Carrasco et al., 2013; Zhao et al., 2015; Esser et al., 2016; Amir et al., 2017). By using conversion methods that convert pre-trained standard deep networks into equivalent-accurate spiking networks (Diehl et al., 2015; Rueckauer et al., 2017) or by using the training methods from deep learning on networks that capture the underlying parameters of the spiking neuron (O'Connor et al., 2013; Stromatias et al., 2015), we are starting to see spiking deep networks that can be competitive with the standard deep networks.

Another method is to create either synchronous or asynchronous feature frames from the spikes before presentation to the time-stepped deep networks. This method has seen success in the field of neuromorphic vision primarily, as pre-processing methods produce frames from event-driven sensor data to use as inputs to deep networks for classification tasks (Moeys et al., 2016; Neil and Liu, 2016; Lungu et al., 2017). Although these pre-processing methods are outperformed on standard classification tasks by the methods using the traditional frame based sensors, they can help reduce computation by using the data driven nature of the sensors and processing the networks only when the sensor produces events.

This work aims to methodically examine existing and novel spike pre-processing methods for processing the output of the DAS for use with deep networks and machine learning algorithms, in particular for real-time applications. We consider two existing feature extraction methods that generate feature frames using spike counts within a fixed time bin and constant spike count (event) bins respectively. We also propose a new pre-processing method that generates feature frames by applying an exponential kernel to each event. We compare the performances of the different pre-processing methods by combining them with deep learning recurrent neural networks which include gated units (Chung et al., 2014; Neil et al., 2016) and testing the networks on two audio classification tasks (isolated recordings and connected streams) using a recorded audio spike dataset called N-TIDIGITS18. This dataset consists of spike recordings from a Dynamic Audio Sensor in response to the TIDIGITS (Leonard and Doddington, 1993) audio dataset.

2. METHODS

This section presents a description of the hardware cochlea sensors, details the feature generation methods, including the proposed exponential feature generation method and briefly describes the deep network architectures used in this study.

2.1. Dynamic Audio Sensor

The Dynamic Audio Sensor is a binaural silicon cochlea system, with each ear connected to a set of 64 bandpass filters whose center frequencies are logarithmically distributed from approximately 50 Hz to 20 kHz. The events are then asynchronously generated from each of the filters. A silicon cochlea sensor using half wave rectification for the generation of events is the CochleaAMS1b (Chan et al., 2007) and the CochleaAMS1c (Liu et al., 2014), while a cochlea sensor using asynchronous delta modulation for the generation of events is the CochLP (Yang et al., 2016). The CochleaAMS1c sensor is an improved design of the CochleaAMS1b. Each channel of the CochleaAMS1b and CochleaAMS1c has four neurons and each neuron implements a different threshold level for spike generation. In many of the experiments, only the events from a single neuron of one ear are used. An example output for the CochleaAMS1c is shown in **Figure 1**. The methods evaluated in this work were carried out on recordings from the CochleaAMS1b and CochleaAMS1c, while they will be evaluated on CochLP in the future.

2.2. Feature Extraction Methods

The event data from the cochlea sensors can be converted to frame-based features through multiple methods. One commonly used feature type is the Spike Count (SC) feature (Zai et al., 2015; Anumula et al., 2017), that is generated by the creation of a histogram across the frequency channels of the events within a time window. In the case of the DAS, the feature vector for each time frame is, at maximum, a 64-length vector where each element consists of the number of events in that frequency channel. The two main variants of SC features are time-binned and event-binned features. Their formulation is described below.

2.2.1. Raw Spikes

An audio event stream can be mathematically represented as

$$e_i = \left[t_i, f_i\right], i \in \mathbb{N} \tag{1}$$

where e_i is the ith event from the frequency channel f_i in the event stream at time t_i. The f_i can range between 1 and N_c where N_c is the number of frequency channels in the sensor. Also note that the events are time ordered, i.e., for $i < j$, $t_i \leq t_j$. These raw spike information can be processed directly as a sequence by the

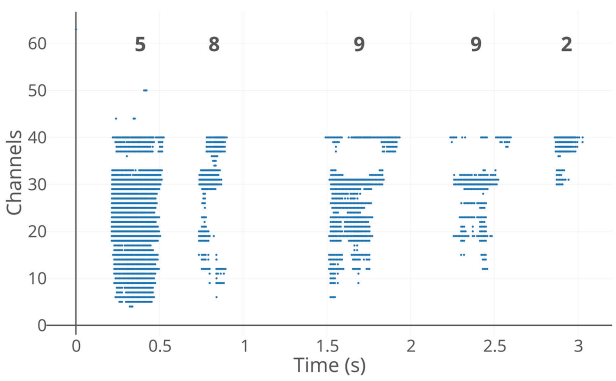

FIGURE 1 | CochleaAMS1c spike output example. The y-axis indicates the 64 frequency channels of the sensor with lower frequency channels at the top. The spikes are in response to the spoken digit sequence "5-8-9-9-2" from the speaker "IM" in the TIDIGITS dataset. The five digits in the sequence can be clearly seen to be apart with significant gaps between them in the encoded sample above. This example also demonstrates the data driven nature of the sensor where it outputs events only when there's a stimulus in the environment.

recurrent networks. Such a method is not usually feasible though because of the inability of the standard recurrent networks to process longer sequences, but they can be efficiently processed through the Phased LSTM, a recently introduced gated recurrent network architecture (Neil et al., 2016).

2.2.2. Time-Binned Spike Count Features

For the generation of time binned Spike Count features, the frame duration for generating the feature is of fixed time length. Time-binned SC features have been used for the speaker identification task using spike recordings generated from the TIMIT dataset (Liu et al., 2010; Li et al., 2012), the YOHO dataset (Chakrabartty and Liu, 2010), and real-world DAS recordings (Anumula et al., 2017).

The time-binned SC features F^{tb} for a time window length of T_l are defined as follows:

$$F_j^{tb}\left(f\right) = \mathbf{card}(\{e_i \mid T_l \cdot (j-1) \leq t_i < T_l \cdot j,\, f_i = f\}) \quad (2)$$

where F_j^{tb} is the jth frame of the features, **card**() is the cardinality of a set, $\cdot$ is the standard multiplication operator, and f is the position of the frequency channel.

Figure 2 shows how the time-binned SC features are generated from the spikes.

2.2.3. Event-Binned Spike Count Features

Event-binned SC features consist of frames in which there are a fixed number of events. Unlike time-binned spike count features, event binning is a data driven approach and eliminates the need for input normalization. These features have been used for both the DVS and the DAS. In the robot predator-prey scenario in Moeys et al. (2016), the DVS retina data is integrated into 36 × 36 frames as 2D histograms obtained by integrating 5,000 events in 200 possible gray level values. Since the DVS frames are sparse, active DVS frame pixels accumulate about 50 events. Constant-event frames from the spiking TIMIT dataset have also been used together with a Support Vector Machine Classifier in a speaker identification task (Li et al., 2012).

The event-binned spike count features F^{eb} are defined as follows. The jth frame is given by

$$F_j^{eb}\left(f\right) = \mathbf{card}(\{e_i \mid E \cdot (j-1) \leq i < E \cdot j,\, f_i = f\}) \quad (3)$$

where **card**() is the cardinality of a set, $\cdot$ is the standard multiplication operation, f is the position of the frequency channel and E is the number of events binned into a single frame.

Figure 3 shows how the event-binned spike count features are generated from the spikes.

2.2.4. Comparison of Time Binning and Event Binning

Although both methods capture the distribution of the events across the frequency channels, there is a difference between the features generated from these methods. The main difference is that the time window used for time binning is of constant length, while the time window of the event-binned features are of varying lengths. The lengths depend on the input event rate over time. This can be seen in the examples of time-binned and event-binned SC features for a single word as shown in **Figure 4** and for a sentence as shown in **Figure 5**. In **Figure 5**, it can be seen that the information about silences in the sentence is temporally smeared in the event-binned features. This property is not desirable as it could be a disadvantage when trying to extract information that depend on the silence periods within the

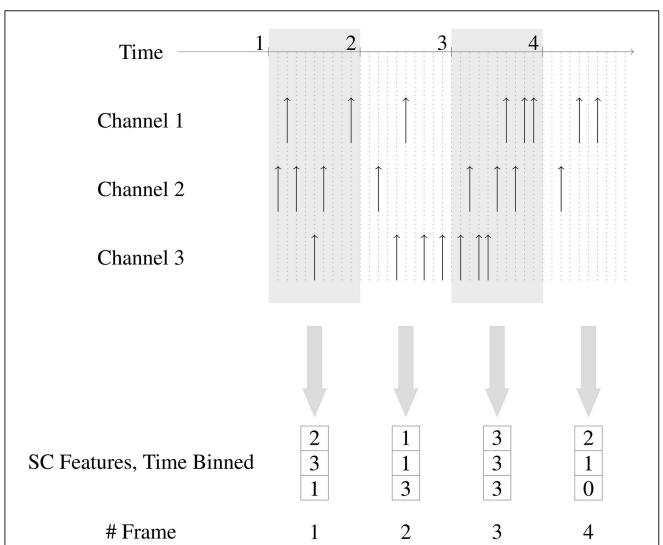

FIGURE 2 | Generation of time-binned Spike Count features. Three channels are shown in this example. The fixed length time windows used for binning the events are non overlapping and of unit time length. In frame 2 , there is 1 event in channel 1, 1 event in channel 2 and 3 events in channel 3, and hence the corresponding feature is (1, 1, 3).

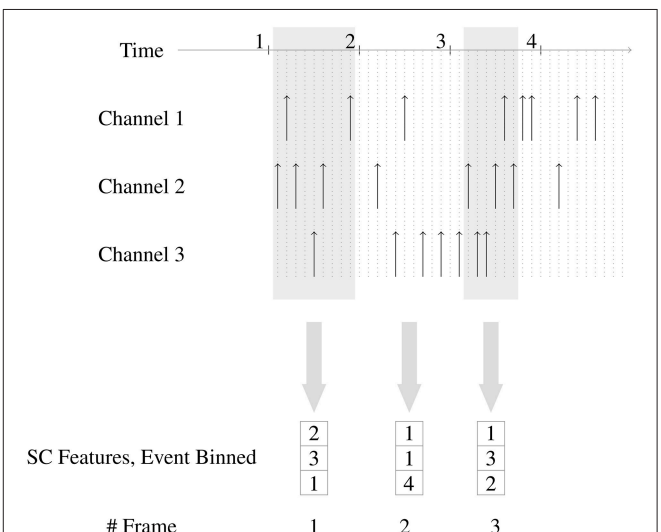

FIGURE 3 | Generation of event-binned Spike Count features. Three channels are shown in this example. Every time window frame used for binning the events has 6 events and there is no overlap of events between consecutive time window frames. In the second time frame, the 6 events are distributed as 1, 1, and 4 across channels 1, 2, and 3, respectively, hence the corresponding feature is (1, 1, 4).

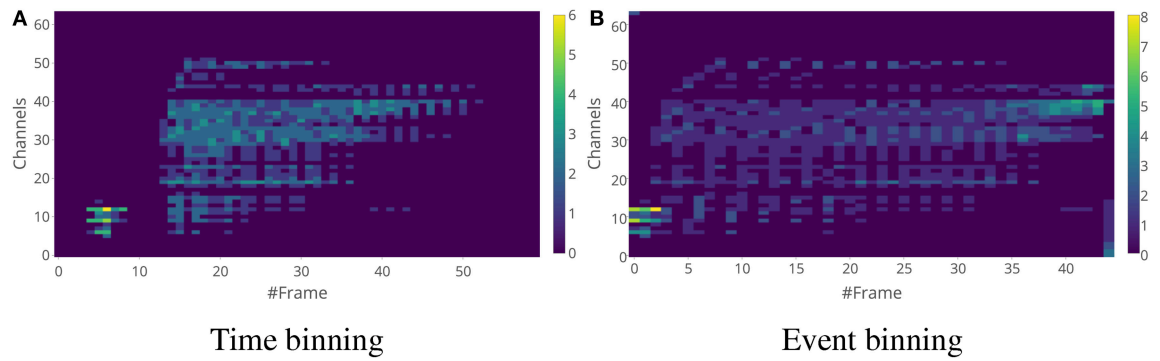

FIGURE 4 | Spike Count features for a digit sample "2". The time window length for time binning in **(A)** is 5 ms and the number of events in a single frame for event binning in **(B)** is 25. There does not seem to be a clear advantage of choosing event binning over time binning when it comes to individual digits. Note that event binning for this example produces fewer frames compared to the time binning.

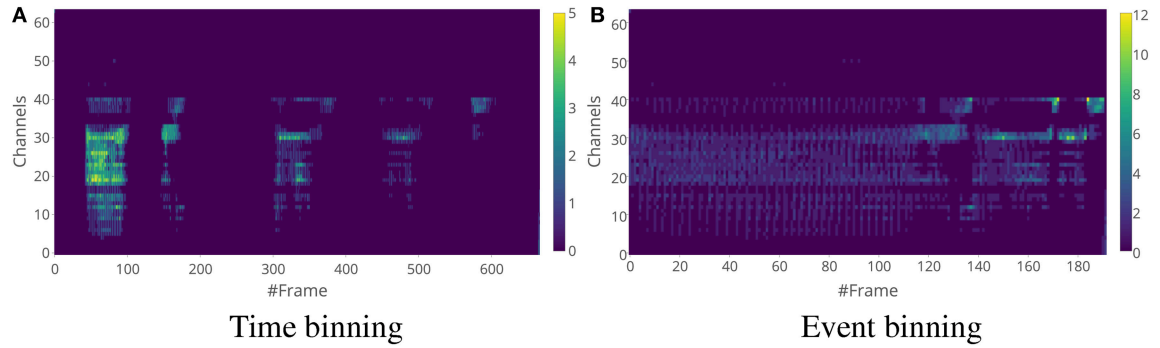

FIGURE 5 | Spike Count features for a digit sequence "5-8-9-9-2". The time window length for time binning in **(A)** is 5 ms and the number of events in a single frame for event binning in **(B)** is 25. The event binning method does not completely encode the timing information in the sample. Also, the silence periods between the digits is absent in the event-binned features.

sentences, unless silence segmentation is done before generating the features.

2.2.5. Data-Driven Time-Binned Spike Count Features

Further, a data-driven time-binning method is introduced and employed in this work. In contrast to the previous time-binned SC features described in section 2.2.2, a feature frame is not processed if no spikes occurred within the corresponding time bin. In addition, this method specifically uses a brief time-bin length. This allows fewer inputs compared to time-binned spike counts (as a fixed-size vector is either presented or skipped), and far fewer inputs to be presented to the network compared to sequentially presenting raw events while maintaining much of the time resolution. Here, using a short time-bin length allows a high degree of spike time accuracy to be maintained, as individual spikes have correct timestamps discretized to the bin length. These data-driven time-binned SC features F^d can be defined as

$$F_j^d = F_i^{tb}, \text{where } i \text{ is such that } \mathbf{max}\left(F_i^{tb}\right) > 0 \text{ and } \mathbf{card}\left(\{k \mid k \leq i, \mathbf{max}\left(F_k^{tb}\right) > 0\}\right) = j \quad (4)$$

2.2.6. Exponential Features

Finally, we introduce a real-valued feature representation that is more amenable to training deep neural networks. This feature is created by convolving each spike with an exponential kernel, that captures the timing information carried by the spikes and has been used in various models, for e.g., Abdollahi and Liu (2011) and Lagorce et al. (2015, 2016). Exponentials are frequently used in neuronal models such as the exponential integrate-and-fire model (Brette and Gerstner, 2005). Although other kernels such as the Gaussian kernels used in the analysis of neuronal firing patterns (Szűcs, 1998) can also be used, we restrict our study here to exponential kernels because they can be applied easier to create real-time features. The resulting output after the convolution is sometimes treated as a real-valued time surface as described in Lagorce et al. (2016). These exponential features have also been used in classification tasks such as image classification (Tapson et al., 2013; Cohen et al., 2016). We first describe the creation of the exponential features and then the binning methods used on these features.

For an audio event stream defined as in Equation (1), the exponential feature F_i^e for an event e_i is constructed by first

defining a time context $\mathcal{T}_i$ for the event. The time context is an N_c dimensional vector where N_c is the number of frequency channels in the audio sensor and is defined as

$$\mathcal{T}_i(f) = \max_{j \le i} \{t_j \mid f_j = f\} \tag{5}$$

where f is the position of a frequency channel. The exponential feature for an event is then defined as

$$F_i^e(f) = e^{-(t_i - \mathcal{T}_i(f))/\tau} \tag{6}$$

An illustration describing the generation of the exponential features for the events is shown in **Figure 6**.

Once these exponential features are created, the events are binned into time window frames either through time binning or event binning like in the SC features, and the average of the exponential features for the events in the time window frame is used as the exponential feature for the frame. For the rest of the paper, we use the term "exponential features" to mean exponential features for a frame. Examples of time binning and event binning exponential features for a single word are shown in **Figure 7** and for a sentence are shown in **Figure 8**.

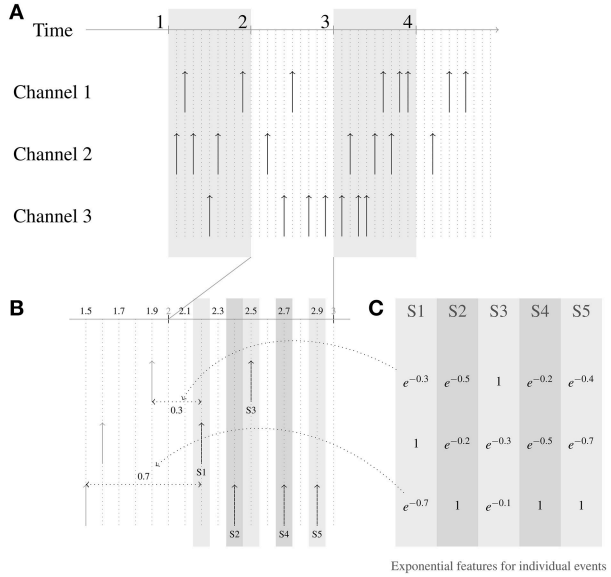

FIGURE 6 | Generation of exponential features for events. Three channels are shown in this example. The time constant parameter t used for generating the features is 1 time unit. The events streams are shown in **(A)**, the zoomed-in picture of the events in the second frame are shown in **(B)**, and the exponential features for this frame is shown in **(C)**. Consider the event at time $t = 2.2$, labeled S1. In channel 1, the closest event in time to the current event occurred 0.3 time units before, and thus the corresponding feature value for the channel 1 in the exponential feature vector for event S1 is $e^{-(0.3/1)}$. Similarly for channel 3, the closest event in time to the current event occurred 0.7 time units before, and thus the corresponding entry for channel 3 in the exponential feature for S1 is $e^{-(0.7/1)}$. For channel 2, since the current event is at channel 2, the exponential feature value at channel 2 is $e^{-(0/1)=1}$.

For a real-time implementation, the exponential features are computed recursively as follows.

$$F_i^e(f) = \begin{cases} e^{-(t_i - t_{i-1})/\tau} F_{i-1}^e(f), & \text{if } f \neq f_i \\ 1, & \text{if } f = f_i \end{cases} \tag{7}$$

With F_0^e initialized to a zero vector, it can easily be seen that the above implementation corresponds to the definition in Equation (6).

2.3. Recurrent Neural Networks

Convolutional Neural Networks are typically used in vision classification tasks and have been successfully used together with the Dynamic Vision Sensor (Moeys et al., 2016). These networks have a feedforward architecture where the neurons in one layer only drive the neurons in the upper layers. However, recurrent neural networks (RNNs) in which neurons in one layer recurrently receive input from neurons in the same layer, are more generally used when the inputs consist of temporal sequences.

Given a sequence $x = (x_1, x_2, \ldots, x_T)$, the RNN layer updates its hidden state h_t with $t \in \{0, 1, 2, \ldots, T\}$, with h_0 being the initial state and $h_t = \phi(h_{t-1}, x_t)$, where ϕ is a non-linear function. Generally, the update function for the hidden state is of the form $h_t = \varphi(Uh_{t-1} + Wx_t)$, where U and W are connection matrices of appropriate sizes and φ is an activation function such as a logistic sigmoid or the hyperbolic tangent (Chung et al., 2014).

Training RNNs using gradient descent to learn long term time dependencies in the input is difficult because of the vanishing/exploding gradient problem (Bengio et al., 1994). In order to counter this problem, the Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) neuron model was proposed. This model has an activation function that is managed by different gates acting like a memory control for the neuron. The subsequently proposed Gated Recurrent Unit (GRU) (Cho et al., 2014) model performs well on similar tasks and has the advantage of using fewer parameters. In our experiments, we use both GRU and LSTM RNNs and the following sections introduce these models.

2.3.1. Long-Short Term Memory

The form of LSTM used in this work derives from Graves (2013):

$$i_t = \sigma_i(W_{xi}x_t + W_{hi}h_{t-1} + w_{ci} \odot c_{t-1} + b_i) \tag{8}$$
$$f_t = \sigma_f(W_{xf}x_t + W_{hf}h_{t-1} + w_{cf} \odot c_{t-1} + b_f) \tag{9}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{10}$$
$$o_t = \sigma_o(W_{xo}x_t + W_{ho}h_{t-1} + w_{co} \odot c_t + b_o) \tag{11}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{12}$$

The introduction of gating functions in Hochreiter and Schmidhuber (1997) differed from traditional RNNs, and allowed substantially easier training for recurrent networks. The gate activation vectors, i_t, f_t, o_t, represent the input, forget, and output gates respectively. Each neuron stores an internal cell activation

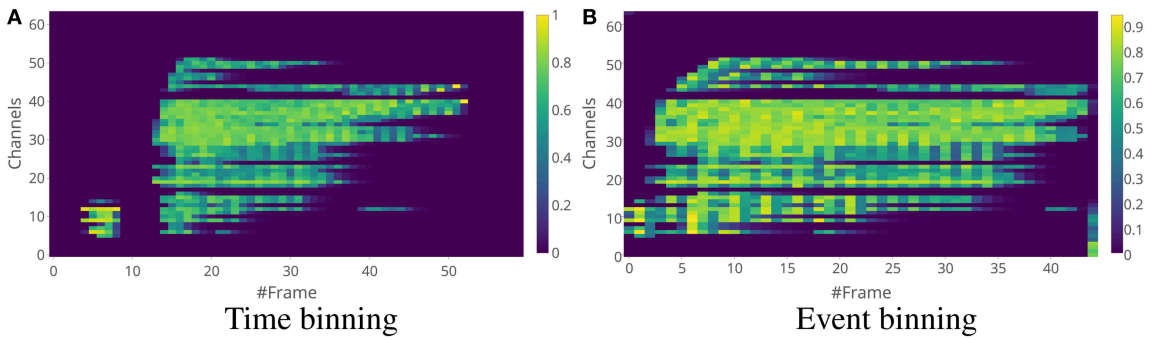

FIGURE 7 | Exponential feature examples for the same word as in **Figure 4**. The time window length for time binning in **(A)** is 5 ms and the number of events in a single frame for event binning in **(B)** is 25. One main difference between the spike count features and the exponential features is that the exponential feature values are in the range between 0 and 1, while the spike count feature values depend on the volume of the spikes in the time window.

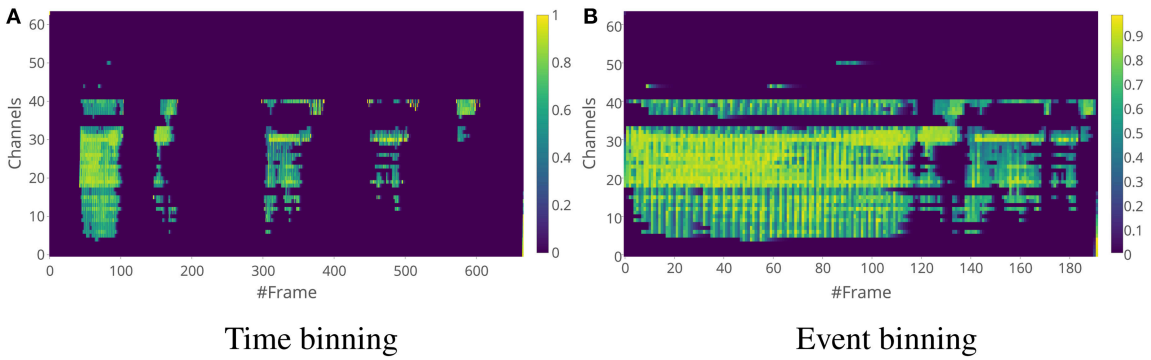

FIGURE 8 | Exponential feature examples for the same sequence as in **Figure 5**. The window length for time binning in **(A)** is 5 ms and the number of events in a single frame for event binning in **(B)** is 25.

vector c_t, while the input and hidden state vectors are x_t and h_t, respectively. A sigmoidal nonlinearity, $y = 1/(1+e^{-x})$, is applied to constrain the gates to lie between 0 and 1, and applied to the gates with σ_i, σ_f, and σ_o for the input, forget, and output gates. For these gates, each gate has a weight parameter for the input x and the hidden state h, resulting W_{xi} and W_{hi}, W_{xf} and W_{hf}, W_{xo} and W_{ho} for the input, forget, and hidden gates, respectively. Additionally, each gate has a bias b_i, b_f, and b_o for the input, forget, and output gates. The $\odot$ notation signifies an elementwise (Hadamard) product, implying that each cell state c_t is a linear interpolation between the previous cell state (controlled by f_t) and the new cell state (controlled by i_t). Finally, the cell state is transformed by the output gate o_t to produce a new hidden state h_t. Optionally, peephole connections, Gers and Schmidhuber (2000) w_{ci}, w_{cf}, and w_{co}, are commonly employed for the cell state c_t to further influence the input, forget, and output gates.

2.3.2. Gated Recurrent Units

Another commonly used gated architecture is the GRU architecture. The primary difference compared to LSTM is the removal of one gate, which results in faster training and execution time while achieving approximately the same accuracy in most tasks. The form employed in this work is the most common implementation from Chung et al. (2014):

$$r_t = \sigma_r(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{13}$$

$$u_t = \sigma_u(W_{xu}x_t + W_{hu}h_{t-1} + b_u) \tag{14}$$

$$c_t = \sigma_c(W_{xc}x_t + r_t \odot (W_{hc}h_{t-1}) + b_c) \tag{15}$$

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot c_t \tag{16}$$

Similar to the above, there are the gate states r_t and u_t, referred to as the reset and update gates, as well as a combination gate and state c_t called the candidate state. As above, each consists of the application of a matrix multiplication of a weight vector (W_{xr}, W_{xu}, W_{xc}) to the input (x_t), as well as another matrix multiplication of a weight vector (W_{hr}, W_{hu}, W_{xc}) to the previous hidden state (h_{t-1}), for the reset, update, and candidate gates respectively. For the two pure gates, these two terms are summed with a bias (b_r and b_u) and the logistic sigmoid nonlinearity $y = 1/(1 + e^{-x})$ is applied to constrain each gate to lie between 0 and 1. For the candidate state, the reset gate is applied elementwise to the previous hidden state, and the bias b_c is added before the candidate state is transformed nonlinearly using the same logistic sigmoid. Finally, the new hidden state h_t is the result

of a linear mixture of the update gate elementwise applied to the candidate state, and the previous hidden state controlled by the complement of the update gate.

2.3.3. Phased LSTM

The Phased LSTM model, which was introduced in Neil et al. (2016), equips the LSTM model with the ability to process irregularly-sampled continuous-time sequences through the application of a novel time gate k_t. This time gate, similar to other gates, produces a continuous value between 0 and 1 but is instead controlled by an external timing input. Each neuron has independent, learnable timing parameters that allow the neuron's time gate to execute a rhythmic wake-sleep cycle over time. When the time gate is open (close to 1), the neuron performs as a normal LSTM neuron does; when the time gate closes (close to 0), the neuron performs no updates until its next wake period. Other neurons, however, can still inspect a sleeping neuron's state. When continuous time sequences are applied, the timestamp of the event controls which subset of neurons are updated, and permits calculations based on the rhythmic wake-sleep cycle of the neurons in response timestamp itself.

Rigorously, the opening and closing of the gate is a periodic oscillation controlled by three parameters: a period τ that controls the duration, a shift s that applies a phase shift offset, and an on ratio r_{on} that determines the duration of the open period. The time (khronos) gate k_t can be calculated as:

$$\phi_{i,t} = \frac{(t - s_i) \bmod \tau_i}{\tau_i}, \quad k_{i,t} = \begin{cases} \dfrac{2\phi_{i,t}}{r_{on,i}}, & \text{if } \phi_{i,t} < \dfrac{1}{2} r_{on,i} \\ 2 - \dfrac{2\phi_{i,t}}{r_{on,i}}, & \text{if } \dfrac{1}{2} r_{on,i} < \phi_{i,t} < r_{on,i} \\ \alpha\phi_{i,t}, & \text{otherwise} \end{cases} \tag{17}$$

The neuron index i indicates which parameters are neuron-specific ($\phi_{i,t}$, $k_{i,t}$, s_i, τ_i, $r_{on,i}$) and which are global (t, α). Here, $\phi_{i,t}$ is introduced as an auxiliary variable to represent the percentage of the phase within the rhythmic cycle, ranging from 0 to 100%. There are three piecewise phases of the operation of the gate functionally represented in Equation (17): an open and rising phase (during the first half of r_{on}), an open and falling phase (during the second half of r_{on}) and an off phase. The linear slopes of the rising and falling phase have a constant gradient to preserve strong gradient information, in the same manner that allows ReLUs to train so well (LeCun et al., 2015). Further, note a leak is applied during the off phase with analogy to the leaky rectified linear unit (He et al., 2015) to permit the flow of gradient information even when the neuron is off. However, after training, the leak can be set to zero (i.e., $\alpha = 0$) and thus truly off, so no updates need be performed when the neuron is in the closed or sleep phase of the cycle. This continuous-time equation is defined at all time points t but requires no computation between sampled data points, allowing irregularly-spaced points in time to be effectively used within this framework as the neurons have an explicit model of time. The LSTM equations from above can then be rewritten to permit arbitrary time points j rather than timestep indices, using a proposed cell state $\widetilde{c}_j$ and proposed hidden state $\widetilde{h}_j$ controlled by the time gate k_j:

$$i_j = \sigma_i(W_{xi}x_j + W_{hi}h_{j-1} + w_{ci} \odot c_{j-1} + b_i) \tag{18}$$

$$f_j = \sigma_f(W_{xf}x_j + W_{hf}h_{j-1} + w_{cf} \odot c_{j-1} + b_f) \tag{19}$$

$$\widetilde{c}_j = f_j \odot c_{j-1} + i_j \odot \sigma_c(W_{xc}x_j + W_{hc}h_{j-1} + b_c),$$
$$c_j = k_j \odot \widetilde{c}_j + (1 - k_j) \odot c_{j-1} \tag{20}$$

$$o_j = \sigma_o(W_{xo}x_t + W_{ho} + w_{co} \odot \widetilde{c}_j + b_o) \tag{21}$$

$$\widetilde{h}_j = o_j \odot \sigma_h(\widetilde{c}_j),$$
$$h_j = k_j \odot \widetilde{h}_j + (1 - k_j) \odot h_{j-1} \tag{22}$$

The sparseness in time of computation (typically, with r_{on} set to 5%) allows this implementation to be far sparser than traditional gated implementations in computation while maintaining high performance. Furthermore, as timesteps are no longer required and the neuron has an explicit model of time, even raw spike events can be directly used with Phased LSTM. For further information, refer to the formulation of Phased LSTM in Neil et al. (2016) or one of its publicly-available implementations[1,2].

2.4. Datasets

This paper introduces the N-TIDIGITS18 dataset by playing the audio files from the TIDIGITS dataset to the CochleaAMS1b sensor. The dataset is publicly accessible at http://sensors.ini.uzh.ch/databases.html. The dataset includes both single digits and connected digit sequences, with a vocabulary consisting of 11 digits ("oh," "zero" and the digits "1" to "9"). Each digit sequence is of length 1–7 spoken digits. There is a total of 55 male and 56 female speakers in the training set with a total of 8,623 training samples, while the testing set has a total of 56 male and 53 female speakers with a total of 8,700 testing samples.

The entire dataset is used or a reduced version of the dataset is used where only the single digit samples are used for training and testing. In the single digits dataset, there are two samples for each of the 11 single digits from every speaker, with a total of 2,464 training samples and 2,486 testing samples. The N-TIDIGITS18 dataset with all the samples was used to train a sequence classification task while the digit samples were used to train a digit recognition task. For most of our training, unless specified, we only use events from one ear and one neuron.

2.5. Network Architectures and Training Criterion

2.5.1. GRU/LSTM Architectures

Two network models were trained separately for the digit recognition task and the sequence classification task. For the digit recognition task, the network consists of 2 GRU layers with 100 units each, followed by a fully connected layer of 100 units with a ReLU activation followed by a Softmax classification layer. For the sequence classification task, each

[1]https://github.com/dannyneil/public_plstm

[2]https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/PhasedLSTMCell

network consists of a fully connected layer of 100 units with SELU activation (Klambauer et al., 2017) followed by 2 LSTM layers of 100 units each followed by the final classification layer. The recently introduced SELU activation helps with regularization of the network by pushing the neuronal activations of the corresponding layer to zero mean and unit variance without the need for batch normalization. The SELU activation function was used over other activation functions mainly because the overall accuracy was significantly improved by using it.

The network for the digit recognition task was trained using a categorical cross entropy objective, while the network for the sequence classification task was trained using the Connectionist Temporal Classification (CTC) objective (Graves et al., 2006). For the CTC objective on sequence classification, the accuracy metrics used were the label error rate and the phrase error rate. For the label error rate, we first calculate the average edit distance (Levenshtein, 1966) between the correct label sequences and the corresponding predicted label sequences. The edit distance between two sequences is the minimum number of insertions, deletions and substitutions required to transform one into the other. The label error is then given by the ratio of the calculated average edit distance and the total number of labels in the correct label sequences. This metric is not a strict proper fraction for its value can go above 1. The phrase error rate is given by the ratio of the correctly predicted label sequences and the total number of label sequences.

All networks were trained on the Tensorflow framework (Abadi et al., 2015) using Adam optimizer with a learning rate of 0.001 over 200 epochs. All the presented accuracy numbers are based on at least three experimental runs. The network and simulation parameters are summarized in **Table 1**.

2.5.2. Phased LSTM Architecture

The single-event architecture was used on the raw input spikes. Because of the volume of input spikes and the difficulty in training extremely long sequences, only one neuron (out of four possible neurons) from one ear was used, resulting in sequences of approximately 4,000 spikes. Because a raw spike address and the corresponding spike time was used, an embedding layer of size 40 was used. As in Neil et al. (2016), a multi-resolution embedding layer downsamples the address by 1, 2, 4, and 16, and concatenates the 10-dimensional embedded feature from each result together. This allows learning features across multiple pitches (neuron addresses) as well as learning features particular to each pitch. After the embedding layer, two layers of 250 Phased LSTM neurons are included, with period $\tau \sim \exp(\mathcal{U}(0, 3))$

TABLE 1 | Summary of the different training parameters used in this study.

Network	Model architecture	Batch size	No.of epochs
GRU RNN	2x 100 GRU - 100 Dense (ReLU) - 10 Softmax	128	200
LSTM RNN	100 Dense (SELU) - 2x 100 LSTM - 10 Dense	128	200
Phased LSTM	2x 250 Phased LSTM - 10 Dense	16	50

Tha Adam optimizer with a learning of 0.001 was used for all the networks.

TABLE 2 | Summary of investigated models on N-TIDIGITS18 dataset.

Feature type	Sensor	Task	Classifier	Accuracy (%)
MFCC		**Digit**	**GRU RNN**	**97.90**
Binned frames (fixed bins/sample)*	AMS1b	Digit	SVM	95.08
Constant time bins**	AMS1b	Digit	CNN	87.65
Constant time bins**	AMS1b	Digit	GRU RNN	82.82
Single events (raw data)	AMS1b	Digit	Phased LSTM	87.75
Data-driven time-binned features	AMS1b	Digit	Phased LSTM	91.25[a]
Constant time bins	AMS1b	Digit	GRU RNN	86.4
Exponential features	**AMS1b**	**Digit**	**GRU RNN**	**90.9**
Constant time bins	AMS1c	Digit	GRU RNN	88.6
Exponential features	**AMS1c**	**Digit**	**GRU RNN**	**91.1**
Constant time bins	AMS1b	Sequence	LSTM RNN	86.1[b]
Exponential features	**AMS1b**	**Sequence**	**LSTM RNN**	**87.3[b]**

The MFCC features are extracted from the original TIDIGITS dataset.
[a]*Events from all neurons and both ears used in training.*
[b]*Label accuracy on sequences.*
**Abdollahi and Liu (2011).*
***Neil and Liu (2016).*

milliseconds (with $x \sim \mathcal{U}(a, b)$ implying a random draw of x from the uniform distribution between limits a and b), shifts $s \sim \mathcal{U}(0, 100)$ milliseconds, and the on ratio $r_{on} = 0.05$ resulting in 5% activity. The output of the second Phased LSTM layer is fully connected via a dense layer to the ten output classes.

For the N-TIDIGITS18 dataset, only 40% of 0.5 ms time bins (also timesteps) have data (with an average of 3.6 spikes per time bin), running at a 2.5× increase in speed over calculating every timestep. Furthermore, the number of bins are far fewer in number than the number of input spikes as would be the case with processing the raw input data. Compared to processing every spike sequentially in the full dataset (all neurons, all ears), there are now 30 times fewer timesteps, resulting in a dramatic decrease in training time when training on data-driven bins.

All Phased LSTM networks were trained on the Lasagne framework (Dieleman et al., 2015) using the Adam optimizer and a learning rate of 0.001 over 50 epochs.

3. RESULTS

We present the network accuracy results of the different pre–processing methods on the audio classification tasks based on the N-TIDIGITS18 dataset when these features are presented to the different recurrent models.

3.1. Comparison of Feature Representations

The performance of the pre–processed features are tested through two classification tasks. The first is a word recognition task on the single digit samples in the dataset, and the second is a sentence prediction task on the connected digit samples. The classifiers used for different tasks and their performances on the different feature types are shown in **Table 2**. The results in the table show that the networks using the exponential features consistently perform better than the spike count features across both the tasks. The Phased LSTM networks which were used to process either the raw event data or the data-driven bins outperform the spike count features, and produce similar accuracies as GRU RNNs with exponential features.

Although the Phased LSTM network takes a longer time to train because the input sequences of single spikes are longer, one advantage of this method over the other pre-processing methods is that there are no hyper parameters that need fine tuning such as the time window length parameter T_l used for binning or the tau parameter τ used in the exponential features.

The performance of the method using the binned frames on a Support Vector Machine (SVM) classifier is better than all the other methods but this method relies on access to the whole sample which is then converted into a fixed number of bins per sample and unfortunately cannot be used on a real-time recognition system.

3.2. Optimizing Parameters

As discussed in section 3.1, both spike count features and exponential features have a few hyperparameters that need fine tuning for optimal performance. The network hyperparameters were optimized once using a small validation split on the training data from the N-TIDIGITS18 dataset. The small validation dataset was created by using 10% of the training samples while the model was trained on the other 90% of the samples.

The variation of the error rates on the τ parameter and the T_l parameter for the exponential features in the sequence classification task are shown in **Figures 9A,B**, respectively. In **Figure 9A**, we can see that the error rate is very high for τ less than 2 ms, and then remains fairly steady for values of τ till 5 ms and then rises slowly as τ increases. The optimal value of τ is related to the mean inter-spike interval in the frequency channels. While for low values of τ the features do not properly encode the history of the events and thus do not perform well, while for increasing τ the exponential feature values saturate toward 1 and thus do not provide enough contrast among the features for the classification tasks to decode properly.

In **Figure 9B**, we can see that the error rates increase with larger T_l, because with a bigger T_l, the exponential features get smeared because of averaging over more events. But it should be considered that with increasing T_l, the sequences to be processed become shorter which makes the recurrent network training easier and efficient.

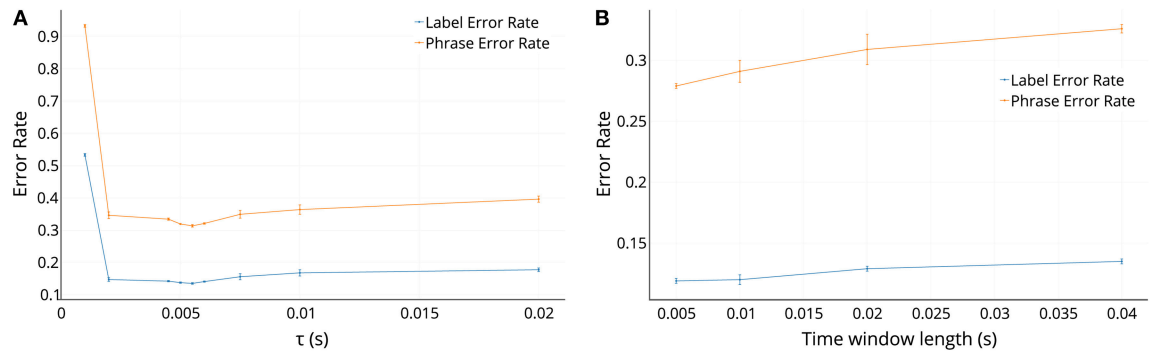

FIGURE 9 | The effect of τ and time window length parameters on the error rates in the sequence classification task in the case of exponential features. The window length for time binning in **(A)** is 5 ms and the value of τ used in **(B)** is 5.5 ms. Although using the optimal values gives lower error rates, using a larger time window length than the optimal value of 5 ms would help in having shorter sequences to process.

These plots suggest that the optimal τ is 5.5 ms and the optimal T_l is 5 ms, which were also the values used in the experiments for **Table 2**. Although the optimal value of T_l was 5 ms, using a parameter value of 40 ms would help save training time and number of computes required to process the network per unit time since there are fewer frames to process per unit time. This advantage comes though at the expense of a reduced accuracy of about 1.6% (88.1% for 5 ms and 86.5% for 40 ms) on the validation set.

3.3. Comparison of Processing Times for Feature Generation

To compare the processing times of conventional MFCC with the SC and exponential filters, we used a Raspberry Pi 3 Model B, with ARM Cortex A-53 processor. We used a feature frame rate of 100 Hz. We created random data for both the raw audio and the event data. For raw audio we used a sampling rate of 16 kHz with uniformly sampled data between $[-1, 1]$ at every sampling point. Since the observed average spike rate was about 3,400 Hz for the N-TIDIGITS18 dataset, for the computational test with event data, we generated Poisson spike trains with a total spike rate of 4,000 Hz. Across 100,000 runs, the average processing times per frame on the hardware were 5.79 ms for the MFCC features, 0.72 ms for the SC features and 2.2 ms for the exponential features. Thus the event-driven features are faster to generate by a factor of 2.2X for exponential features and 8X for SC features. This result is not surprising given the computational simplicity of the cochlea features afforded by the sensor preprocessing.

4. DISCUSSION

In this work, we performed a comparative study of the performance accuracy of a gated recurrent neural network that processes either the raw audio spikes or framed features extracted by different spike processing methods. We demonstrate the use of a recent LSTM model called the Phased LSTM which operates on raw audio spikes. We compared the performance accuracy of this model to that of the standard gated recurrent neural networks, the LSTM and the GRU networks, that processed framed features extracted by different spike processing methods.

The results show that it is possible to achieve a good performance through processing the raw events using the Phased LSTM model. This model, designed for use on long sequences, makes use of the inherent timing information present in the spikes. Although the training time is long because the model has to learn to process more timesteps, there are no meta-parameters to tune in the feature generation.

Alternatively, pre-processing the spikes to produce framed features is appealing because the input sequences to the recurrent networks will be shorter than the sequence of raw events. For both the single digit and digit sequence datasets, the network classification accuracy is higher by approximately 2.5% when using exponential features over spike count features. It should be noted that the results are obtained on the N-TIDIGITS18 dataset, a relatively small dataset. We will investigate in the future if the higher accuracy from using exponential features extend to larger datasets.

We hypothesize that the increased accuracy from exponential features is due to two reasons. First, interspike intervals in the spike streams carry information useful for the classification task and therefore exponential features are more desirable. Second, the encoded exponential feature values are real-valued and range between 0 and 1 while the spike count feature values are quantized in discrete quantities of 1. Having real-valued input features might help during training of the recurrent networks.

Even though the accuracy results from using the pre-processed audio spike frames were lower than the results obtained from using MFCC features, the focus of this work is to present improved methods for processing the outputs of event driven sensors in real-time applications. Our evaluation of the average processing time per frame on a Raspberry Pi shows that generation of the event-driven features is faster than that of MFCCs by a factor of 2–8 depending on the cochlea features used. We also aim towards an event-driven system where processing would be activated only if there are sufficient spikes from the sensor, e.g., the processing is inactivated during silent periods.

The results presented here serve as a baseline for future studies on algorithms that process spikes from spiking audio sensors. The pre-processing methods and the LSTM/GRU networks used in the work above are already implemented in real time (Anumula et al., 2017) using the jAER framework (Delbruck, 2008). The N-TIDIGITS18 dataset used in our experiments is publicly accessible at http://sensors.ini.uzh.ch/databases.html.

AUTHOR CONTRIBUTIONS

JA performed the RNN experiments and contributed to the writing, DN performed the Phased LSTM experiments and contributed to the writing, TD contributed to discussions on the feature extraction methods and assisted in the development of the hardware infrastructure of the cochlea boards, and S-CL contributed to the design of the experiments and the writing.

FUNDING

This work was partially supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 644732.

ACKNOWLEDGMENTS

We thank members of the INI Sensors group, particularly, Xiaoya Li for help with the development of the recording setup. We are also thankful to the reviewers for their valuable comments toward improving the manuscript.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Available online at: http://www.tensorflow.org/

Abdollahi, M., and Liu, S. C. (2011). "Speaker-independent isolated digit recognition using an aer silicon cochlea," in *2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (San Diego, CA), 269–272. doi: 10.1109/BioCAS.2011.6107779

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Honolulu, HI), 7243–7252. doi: 10.1109/CVPR.2017.781

Anumula, J., Neil, D., Li, X., Delbruck, T., and Liu, S.-C. (2017). "Live demonstration: event-driven real-time spoken digit recognition system," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)* (Baltimore, MD). doi: 10.1109/ISCAS.2017.8050394

Barranco, F., Fermuller, C., Aloimonos, Y., and Delbruck, T. (2016). A dataset for visual navigation with neuromorphic methods. *Front. Neurosci*. 10:49. doi: 10.3389/fnins.2016.00049

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.* 5, 157–166. doi: 10.1109/72.279181

Berner, R., Brandli, C., Yang, M., Liu, S. C., and Delbruck, T. (2013). "A 240 × 180 10mW 12μs latency sparse-output vision sensor for mobile applications," in *2013 Symposium on VLSI Circuits* (Kyoto), C186–C187.

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Chakrabartty, S., and Liu, S. C. (2010). "Exploiting spike-based dynamics in a silicon cochlea for speaker identification," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (Paris), 513–516. doi: 10.1109/ISCAS.2010.5537578

Chan, V., Liu, S. C., and van Schaik, A. (2007). AER EAR: a matched silicon cochlea pair with address event representation interface. *IEEE Trans. Circ. Syst. I Regul. Papers* 54, 48–59. doi: 10.1109/TCSI.2006.887979

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Cohen, G. K., Orchard, G., Ieng, S. H., Tapson, J., Benosman, R. B., and van Schaik, A. (2016). Skimming digits: neuromorphic classification of spike-encoded images. *Front. Neurosci.* 10:184. doi: 10.3389/fnins.2016.00184

Delbruck, T. (2008). "Frame-free dynamic digital vision," in *Proceedings of International Symposium on Secure-Life Electronics*, vol. 1 (Tokyo: University of Tokyo), 21–26.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S. C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney), 1–8. doi: 10.1109/IJCNN.2015.7280696

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., et al. (2015). *Lasagne: First Release*. doi: 10.5281/zenodo.27878

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., et al. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Natl. Acad. Sci. U.S.A.* 113, 11441–11446. doi: 10.1073/pnas.1604850113

Farabet, C., Paz, R., Pérez-Carrasco, J., Zamarreño-Ramos, C., Linares-Barranco, A., LeCun, Y., et al. (2012). Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing. *Front. Neurosci.* 6:32. doi: 10.3389/fnins.2012.00032

Gers, F. A., and Schmidhuber, J. (2000). "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, Vol. 3 (Como: IEEE), 189–194. doi: 10.1109/IJCNN.2000.861302

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning, ICML '06* (Pittsburg, CA; New York, NY: ACM), 369–376. doi: 10.1145/1143844.1143891

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)* (Santiago), 1026–1034. doi: 10.1109/ICCV.2015.123

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems 30*, eds I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Long Beach, CA: Curran Associates, Inc.), 972–981.

Lagorce, X., Ieng, S.-H., Clady, X., Pfeiffer, M., and Benosman, R. B. (2015). Spatiotemporal features for asynchronous event-based data. *Front. Neurosci.* 9:46. doi: 10.3389/fnins.2015.00046

Lagorce, X., Orchard, G., Gallupi, F., Shi, B. E., and Benosman, R. (2016). Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans. Patt. Anal. Mach. Intell.* 39, 1346–1359. doi: 10.1109/TPAMI.2016.2574707

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi: 10.1038/nature14539

Leonard, R. G., and Doddington, G. (1993). *Tidigits ldc93s10.* Philadelphia, PA: Linguistic Data Consortium.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Doklady* 10:707.

Li, C.-H., Delbrück, T., and Liu, S.-C. (2012). "Real-time speaker identification using the AEREAR2 event-based silicon cochlea," in *Proceedings of IEEE International Symposium on Circuits and Systems* (Seoul), 1159–1162.

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128 × 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Liu, S.-C., Mesgarani, N., Harris, J., and Hermansky, H. (2010). "The use of spike-based representations for hardware audition systems," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)* (Paris), 505–508.

Liu, S. C., van Schaik, A., Minch, B. A., and Delbruck, T. (2014). Asynchronous binaural spatial audition sensor with 2 × 64 × 4 channel output. *IEEE Trans. Biomed. Circ. Syst.* 8, 453–464. doi: 10.1109/TBCAS.2013.2281834

Lungu, I., Corradi, F., and Delbruck, T. (2017). "Live demonstration: convolutional neural network driven by dynamic vision sensor playing RoShamBo," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)* (Baltimore, MD). doi: 10.1109/ISCAS.2017.8050403

Moeys, D. P., Corradi, F., Kerr, E., Vance, P., Das, G., Neil, D., et al. (2016). "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)* (Krakow), 1–8. doi: 10.1109/EBCCSP.2016.7605233

Neil, D., and Liu, S. C. (2016). "Effective sensor fusion with event-based sensors and deep network architectures," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2282–2285.

Neil, D., Pfeiffer, M., and Liu, S.-C. (2016). "Phased LSTM: accelerating recurrent network training for long or event-based sequences," in *Advances in Neural Information Processing Systems* (Barcelona), 3882–3890.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking Deep Belief Network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Orchard, G., Jayawant, A., Cohen, G., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9:437. doi: 10.3389/fnins.2015.00437

Pérez-Carrasco, J. A., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., et al. (2013). Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE Trans. Patt. Anal. Mach. Intell.* 35, 2706–2719. doi: 10.1109/TPAMI.2013.71

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T. (2014). Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proc. IEEE* 102, 1470–1484. doi: 10.1109/JPROC.2014.2346153

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11:682. doi: 10.3389/fnins.2017.00682

Schmidhuber, J. (2014). Deep learning in neural networks: an overview. *CoRR* abs/1404.7828.

Serrano-Gotarredona, T., and Linares-Barranco, B. (2015). Poker-DVS and MNIST-DVS. Their history, how they were made, and other details. *Front. Neurosci.* 9:481. doi: 10.3389/fnins.2015.00481

Stromatias, E., Neil, D., Pfeiffer, M., Galluppi, F., Furber, S. B., and Liu, S.-C. (2015). Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Front. Neurosci.* 9:222. doi: 10.3389/fnins.2015.00222

Szűcs, A. (1998). Applications of the spike density function in analysis of neuronal firing patterns. *J. Neurosci. Methods* 81, 159–167. doi: 10.1016/S0165-0270(98)00033-8

Tapson, J., Cohen, G., Afshar, S., Stiefel, K., Buskila, Y., Wang, R., et al. (2013). Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. arXiv:1304.7118.

Yang, M., Chien, C. H., Delbruck, T., and Liu, S. C. (2016). "A 0.5V 55 μW 64×2-channel binaural silicon cochlea for event-driven stereo-audio sensing," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)* (San Francisco, CA), 388–389. doi: 10.1109/ISSCC.2016.7418070

Yang, M., Liu, S. C., and Delbruck, T. (2015). A Dynamic Vision Sensor with 1% temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding. *IEEE J. Solid State Circ.* 50, 2149–2160. doi: 10.1109/JSSC.2015.2425886

Zai, A., Bhargava, S., Mesgarani, N., and Liu, S.-C. (2015). Reconstruction of audio waveforms from spike trains of artificial cochlea models. *Front. Neurosci.* 9:347. doi: 10.3389/fnins.2015.00347

Zhao, B., Ding, R., Chen, S., Linares-Barranco, B., and Tang, H. (2015). Feedforward categorization on AER motion events using cortex-like features in a spiking neural network. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1963–1978. doi: 10.1109/TNNLS.2014.2362542

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.