Event Based Time-Vectors for auditory features extraction: a neuromorphic approach for low power audio recognition

Marco Rasetto
Department of Bioengineering and Center for Neural Basis of Cognition
University of Pittsburgh, Carnegie Mellon University
Pittsburgh, USA
marcorasetto@pitt.edu

Juan P. Dominguez-Morales
Robotics and Technology of Computers Lab
Universidad de Sevilla
Seville, Spain

Angel Jimenez-Fernandez
Robotics and Technology of Computers Lab
Universidad de Sevilla
Seville, Spain

Ryad Benosman
Department of Bioengineering
University of Pittsburgh
Pittsburgh, USA
benosman@pitt.edu

December 15, 2021

Abstract

In recent years tremendous efforts have been done to advance the state of the art for Natural Language Processing (NLP) and audio recognition. However, these efforts often translated in increased power consumption and memory requirements for bigger and more complex models. These solutions falls short of the constraints of IoT devices which need low power, low memory efficient computation, and therefore they fail to meet the growing demand of efficient edge computing. Neuromorphic systems have proved to be excellent candidates for low-power low-latency computation in a multitude of applications. For this reason we present a neuromorphic architecture, capable of unsupervised auditory feature recognition. We then validate the network on a subset of Google's Speech Commands dataset.

***K*eywords** Event based · Neuromorphic · Natural language processing

1 Introduction

With ever growing markets of the Internet-Of-Things [1] [2], portable and implantable medical devices[3][4], and mobile robotics[5], there is an increasing need of low power AI algorithms and systems able to operate with low latency and low memory requirements. One perfect example of this trend is the field of natural language processing for virtual assistants, where end users interact with generally low power devices that, after recognizing a wakeword ("Hey google"), require constant connection to servers in order to respond to commands and run more complex machine learning models [6]. We argue that a solution to this problem of efficiency could be found in Neuromorphic systems. Neuromorphic engineering is a concept coined by Carver Mead in the late eighties [7], it refers to the design and implementation of hardware and software to mimic the basic principles of biological nervous systems. In this approach, information is represented and transmitted by means of asynchronous pulses or action potentials emitted by artificial neurons separated in building blocks or different event-based processing layers. This architecture presents a clear advantage over traditional computation methods in terms of power consumption and real-time capabilities [8][9][10].

In order to fully appreciate the power efficiency of these systems, however, it is necessary to design their architecture from the ground up, starting from sensors. In the case of neuromorphic auditory systems, the sensor would be a bio-inspired cochlea. Bio-inspired cochleas are sensors which are inspired in how the biological human ear behaves, decomposing the input audio signal into different streams of events representing different frequency bands. Different software and hardware implementations of bio-inspired cochleas can be found in the literature, with all of them having a similar architecture: first, an input stage is used to receive the stimuli; then, a set of band-pass filters (following either a cascade or a parallel topology) is used to process the information; and, finally, these streams are outputted. Some examples of state-of-the-art event-based cochleas are [11, 12] (analog) and [13, 14] (digital). Additional processing steps can be applied to the output of these sensors for further natural language processing, sound recognition [15] and sound source localization tasks [16], among others. In this paper we propose a novel neuromorphic architecture for sound recognition feature extraction, designed to be compatible with a digital cochlea designed by Jimenez et al [14]. The proposed network is designed to work on a per event basis, learning patterns of relative timings of activation generated by the cochlea in a unsupervised manner. When complemented with a classifier, the network can be used for phoneme/word recognition, making it an important first stage for designing full Neuromorphic systems for low power, low latency NLP. Whereas similar solutions were proposed on visual neuromorphic architectures [17] [18], to our knowledge, this is the first time that this approach has ever been used for auditory pattern recognition.

2 Materials and Methods

2.1 Dataset

In this work, Google's Speech Commands Dataset [19] was used. This dataset consists of 65000 one-second long utterances of 30 short words, by thousands of different people. In our case, the "on" (3846 samples) and "off" (3746 samples) classes were used to train and test the proposed system.

Each sample was fed to a 32-channel monaural Neuromorphic Auditory Sensor (NAS) and then saved as an AEDAT file (consisting of addresses and timestamps of the spikes produced) using an USBAERmini2 board. Therefore, a total of 7592 AEDAT files with the spiking information decomposed in 32 different frequency bands were obtained.

2.2 Neuromorphic Auditory Sensor

A NAS [14] is a digital cochlea implementation that is based in Lyon's cascaded model [20] of the biological cochlea and also on spike signals processing (SSP) techniques [21, 22]. NAS decomposes the input audio obtained from an analog-to-digital audio converter with I2S interface in its frequency components, and outputs a stream of events using the Address-Event Representation (AER)[23].

After the input sound signal is digitized using the audio converter, the information is written sample-by-sample in a synthetic spike generator (SSG) [24], which provides a spike stream with a frequency that is proportional to the amplitude of the input audio signal. Then, the spike stream passes through a cascaded bank of spike-based band-pass filters that perform the frequency decomposition. The output of each of the filters is connected to an AER-monitor, which assigns a unique address to the spikes following the AER representation. It is important to mention that for every filter (also called channel), two different unique addresses are generated: one of them for the spikes corresponding to the positive part of the signal and the other one for the negative part. Finally, the spikes are propagated through an asynchronous AER bus to the output, being able to interface with other neuromorphic platforms such as SpiNNaker[25], Field-Programmable Gate Arrays (FPGAs) or other AER-based boards.

In this work, a 32-channels (64 different addresses) monaural NAS was implemented in an AER-Node board, which has a Spartan-6 FPGA. A block diagram of the NAS implementation can be seen in Fig. 1. The 32 channels of the implemented NAS are distributed along the whole range of the human hearing (20 Hz to 20 kHz, approximately [26]). The output of the NAS is connected to an USBAERmini2 board [27] that assigns a timestamp to the spikes and allows to log the spiking information on the computer by using either jAER[1] or Matlab as AER-DATA (.aedat) files. As the negative addresses are considered redundant for the scenario of auditory data classification, they are discarded before being relayed to the proposed Neuromorphic Network.

[1]http://jaerproject.org (accessed December 15, 2021)

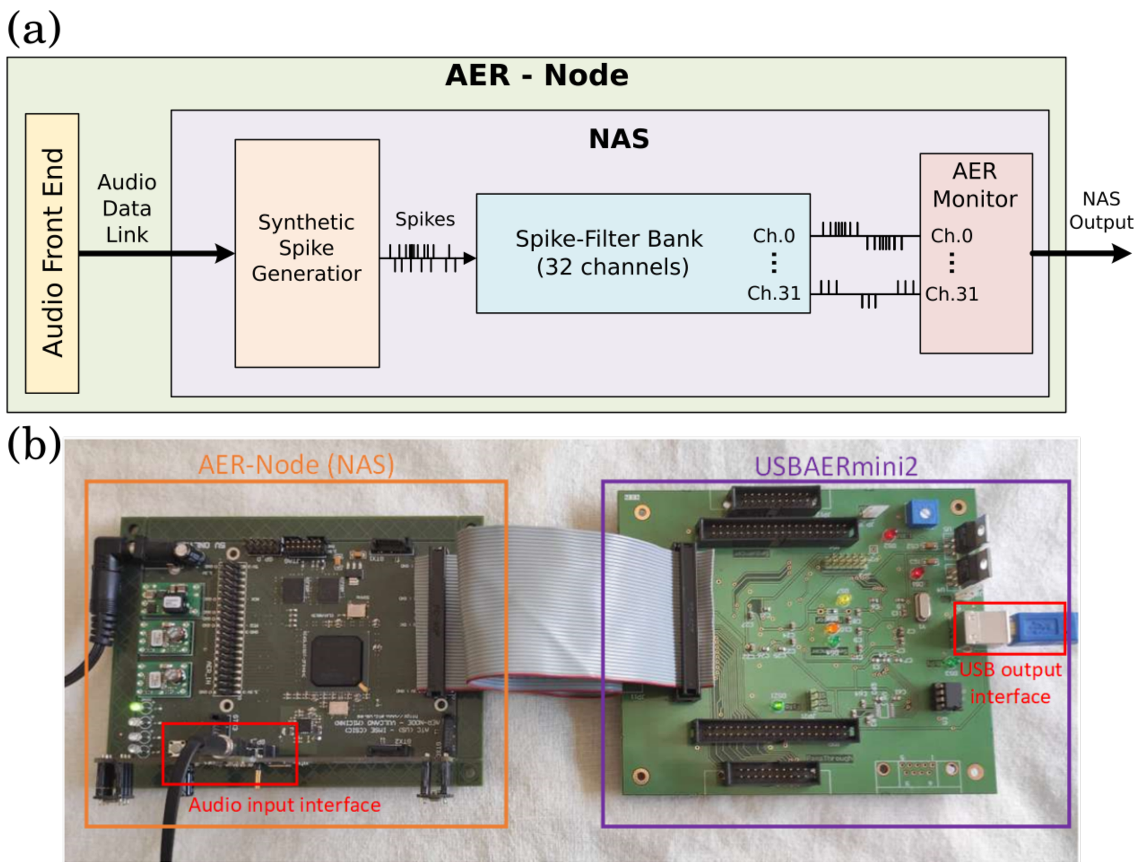

Figure 1: (a) Block diagram of the NAS implementation on an AER-Node board.(b) A picture of the AER-Node board, and a USBAERmini2, used to sample and save the NAS Output on a PC

2.3 Network Architecture

Neuromorphic cochleas, like biological counterparts, decompose auditory stimuli in a series of fundamental frequencies and encode the sound pressure with spikes. In the particular case of the NAS, sound pressure is first encoded as rate, transformed in spiking data, and then filtered in different channels using Spike-Filter Banks as shown in Figure 1. This process causes the spiking rate of higher frequency channels to be proportionally higher than lower frequency channels [14]. This difference in spiking rate is hard to treat when designing neuromorphic algorithms that deal with precise timings of activation, as they often expect the same integration time scale to be shared among neuron populations or units of the network. Our network (Fig. 2) tackles this problem by allowing spiking information to be integrated at two separate stages: a Local Features Layer and a Cross Features Layer. In the Local layer we extract pattern of activity of the NAS of individual channels, and we build some descriptors called Local Time-Vectors that will be normalized to deal with the spiking rate difference. These descriptors are what is used by the Local Features Layer to learn and recognize spiking patterns from single channels. The output of the Local Features Layer is then pooled across channels using another descriptor called Cross Time-Vector that is then fed to the Cross Feature Layer, with the purpose of classifying patterns of the whole Neuromorphic Cochlea. The Cross Features Layer output is then used to build Histograms of Activity that can be then used as input for a classifier with the purpose of recognizing the input stimuli to the whole network. In the following paragraphs we will explain the details of Time-Vector generation and how individual layers of the network work.

2.4 Local features layer

In this section we define the first layer of the architecture, aimed to extract temporal patterns of single channels of the NAS. We first define the concept of Local Time-Vector, and then we proceed to explain how we can learn recurring patterns and use this descriptor to generate a novel stream of events that will be processed by the second layer.

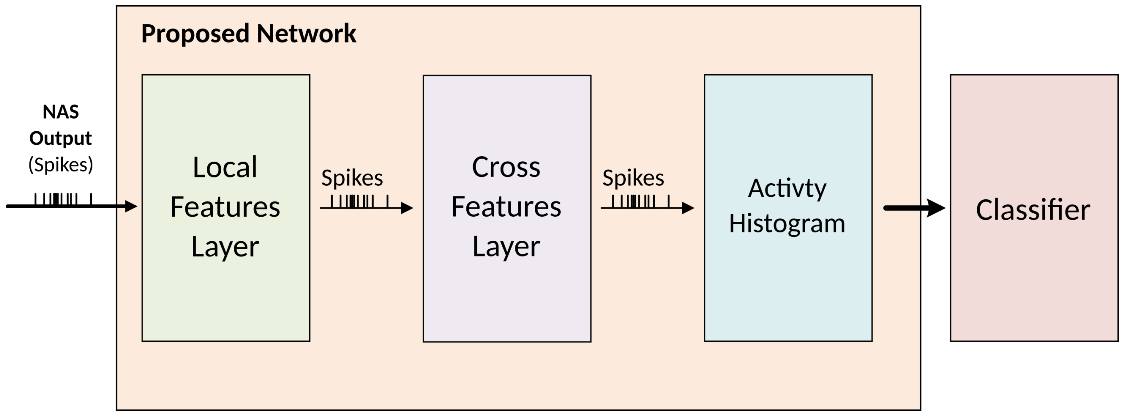

Figure 2: Block diagram of the proposed event based network. NAS output Spikes are fed to the Local Features Layer extracting temporal patterns of activity of single channels. Its output is then fed to the Cross Feature Layer that learns to extract patterns of activity across different channels. The Output of the Cross Feature Layers is then used to generate histograms of activity that can finally be classified in order to recognize the input stimuli.

2.4.1 Local Time-Vectors:

A time vector is a descriptor similar to Time-Surfaces [17], in which the temporal neighborhood of a certain event, is interpolated with an exponential decay. Similarly to synaptic computation, this representation is useful to extract the temporal relations between subsequent events, and characterize patterns of activity by their relative spike timings. In neuromorphic computation an event is a precisely timed digital signal, carrying temporal information along other information, like pixel coordinates (as for artificial retinas like ATIS [28] or DAVIS [29]) channel index (as for the NAS [14]). Therefore, events produced by a NAS can be described by the tuple:

$$ev_i = [t_i, ch_i] \qquad i \in \mathbb{N} \tag{1}$$

Where t_i is the timestamp of a single i event and ch_i represents the index of the frequency tuned channel producing it. ($ch_i \in [0, 31] \subset \mathbb{N}$) From it, we can define an ordered set T_{ch_i} of timestamps occurring before the event ev_i generating from the same channel:

$$T_{ch_i} = \{t_j | t_j <= t_i, t_{j-1} < t_j, ch_j = ch_i\} \tag{2}$$

and then we can select the last n timestamps that we will use to generate a Local Time Vector of length n,

$$TL_i = \{t_k | t_k \in T_{ch_i}, k = \{j, j-1, j-2...j-n-1\}\} \tag{3}$$

so that applying an exponential decay with a time coefficient τ_{ch_i},

$$SL_i = e^{\frac{-(t_i - TL_i)}{\tau_{ch_i}}} \tag{4}$$

we create a Local Time Vector SL_i for the reference event ev_i Where τ_{ch_i} is the time coefficient selected for that channel, and describes the temporal precision or sensitivity to the patterns produced by the NAS. The process to generate the Local time surfaces is summarized by the Figure 3, which shows the process of generating a Time-Vector with $n = 4$ spikes.

2.4.2 Temporal coefficients:

The NAS spiking output presents a difference in spiking rate, where lower frequency channels produce less spikes than higher ones, for the same sound magnitude or sound pressure. This relation is problematic for neuromorphic algorithms, as information is expected to be encoded in the relative timings of spikes. In the specific case of Local Time Vectors, the same pattern or sound pressure, will produce different Time-Vectors depending on the frequency of the channel, artificially increasing the complexity of the auditory feature encoding.

For this reason, when we create Local Time-Vectors for the Local Features Layer, we define a different τ_{ch_i} for each individual channels, so that the Time-Vectors will be normalized across different frequencies (Fig. 4) Because τ_{ch_i} is dependent on the channel index of the NAS but also serves as a fundamental parameter for the Local Features Layer we break it down in two terms:

$$\tau_{ch_i} = m_{ch_i} * \tau_{local} \tag{5}$$

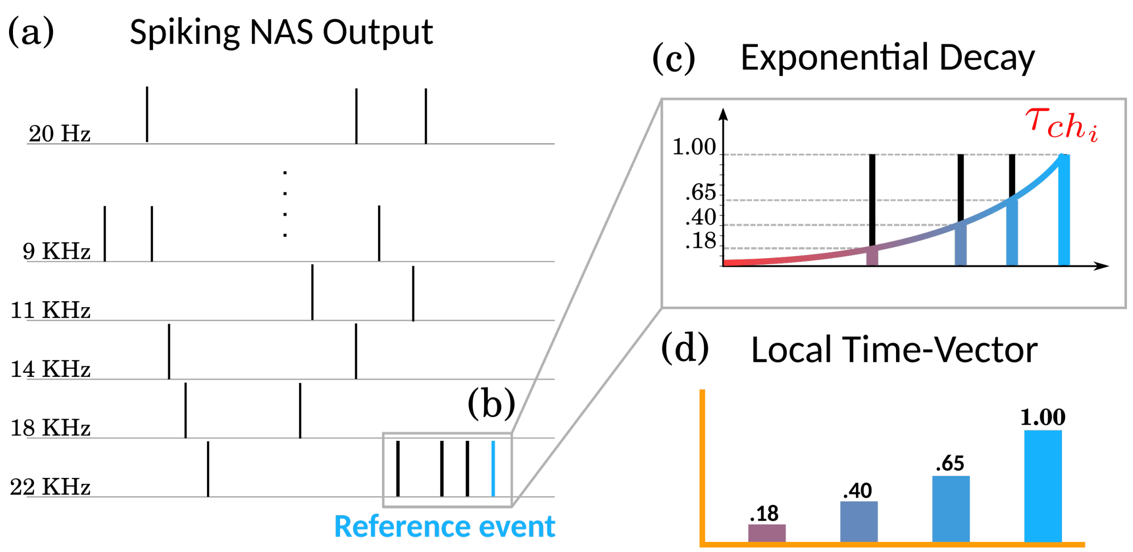

Figure 3: Example of a Local Time-Vector generation with n=4. Every time a new spike is produced by the NAS (a) we select the n-1 last events produced by the same channel ch_i of the reference event (b). We interpolate an exponential decay with a time constant τ_{ch_i} (c) selected for the channel ch_i and we define the Local Time vector as a vector containing the values obtained from (c).

Where τ_{local} is the parameter used to control the overall temporal precision of the first layer, and m_{ch_i} is a multiplicative factor that results from linear interpolation between the highest spiking frequency of channel 0 ($9 \cdot 10^4$ spikes/s) and the lowest frequency channel 32 ($2 \cdot 10^4$ spikes/s) [14].

$$m_{ch_i} = 2 + ch_i * (7/31) \tag{6}$$

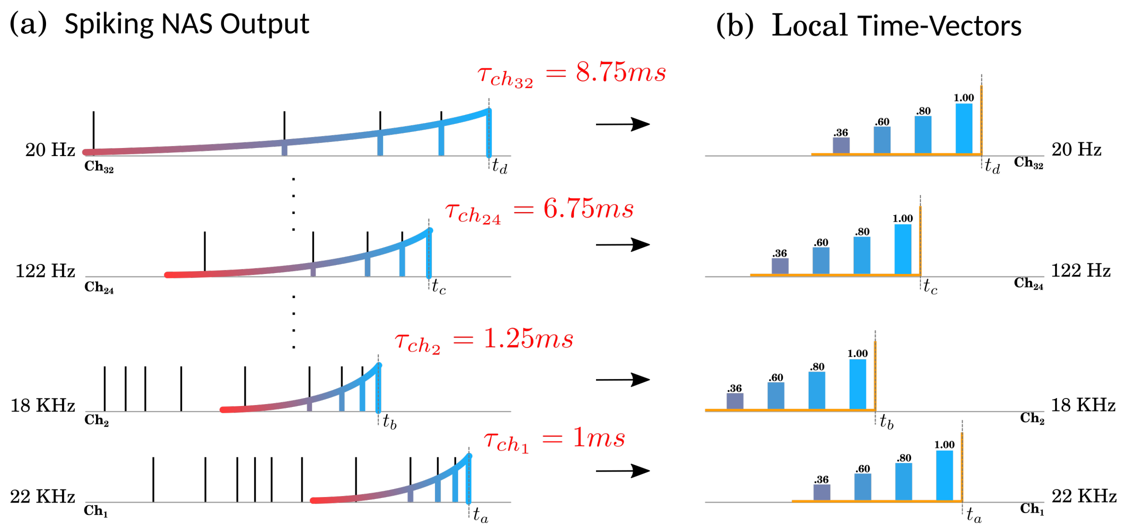

Figure 4: This figure shows the temporal scaling used to normalize the spike information coming from different channels. As different channels have different spiking rate for the same frequency amplitude (a), we select individual τ_{ch_i} so that the corresponding Local-Time vector will be the same.

2.4.3 Learning and inferring local features

In order to learn a useful representation of the Local Time-Vector generated by the NAS, we then cluster a training set as shown in Figure 5a. To perform the clustering, we decided to use a faster and less memory intensive K-Means implementation of the Scikit-Learn python library [30]: Mini Batch K-means [31]. Although, other clustering algorithms are compatible with this type of network [17], [18]. The centroids of the clustering algorithm are then considered features representing the multitude of different Local Time-Vectors generated by the NAS (Fig. 5c). Every time that a new event is generated (Fig. 5c), a Local Time-Vector is built and assigned to the closest center (Fig. 5d); the result of this assignment is a new event with the same t_i and ch_i of the input one, but with an added descriptor called local feature lft_i (Fig. 5e), which stores the center index of the closest cluster:

$$ev_i = [t_i, ch_i, lft_i] \qquad i \in \mathbb{N} \tag{7}$$

Where $lft_i \in [0, lk] \subset \mathbb{N}$, and lk is the selected number of clusters for the local layer.

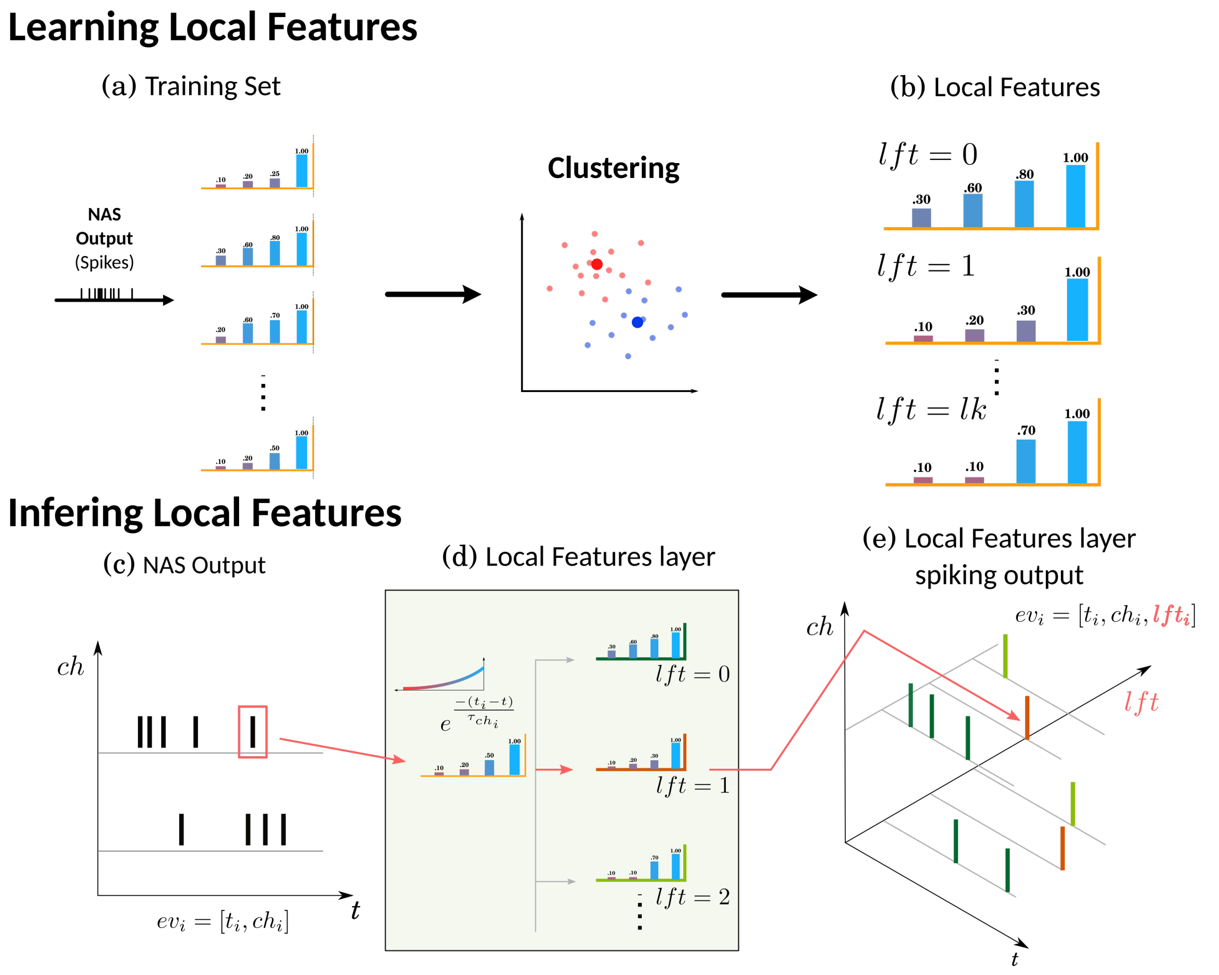

Figure 5: In this figure we show the two modalities in which the Local Feature Layer can operate (Learning and Infering). During learning, the layer clusters a Training set of Local Time-Vectors (a) and extract a set of lk features, were lk is the selected number of clusters for the local layer. During Infering, any incoming event (c) is classified by the closest cluster (d), and its class index represented by a new descriptor lft_i (e).

2.5 Cross features layer

The second layer is used to extract temporal patterns of local features across channels. The main idea is that once we characterized the temporal evolution of single channels we are now ready to pool information from multiple frequencies. Since Local-Time vectors were extracting patterns from single channels, we now need to define a new descriptor that we call Cross Time-Vector, to identify patterns across channels. Similarly as we did with Local Time-vectors, we will then show how we learn and infer Cross Time-vectors in this layer.

2.5.1 Cross Time-Vectors:

The Time-Surfaces used here have a different dimensionality than the Time-Vector presented previously, as events are now characterized by channel index ch_i and local feature lft_i. For this reason, we now define the temporal neighborhood of each new event ev_i by this equation:

$$TC_i = \max_{j \leq i} \{t_j \forall (n, m) : ch_j = n, lft_j = m\} \tag{8}$$

Composed by all the last events before t_i produced by every channel and local features. So that the resulting Cross Time-Surfaces is:

$$SC_i = e^{\frac{-(t_i - TC_i)}{\tau_{cr}}} \tag{9}$$

Where τ_{cr} is the temporal coefficient used to tune Cross Time-Surfaces to extract information at the temporal scale of interest. A graphical depiction of this process is summarised in Figure 6.

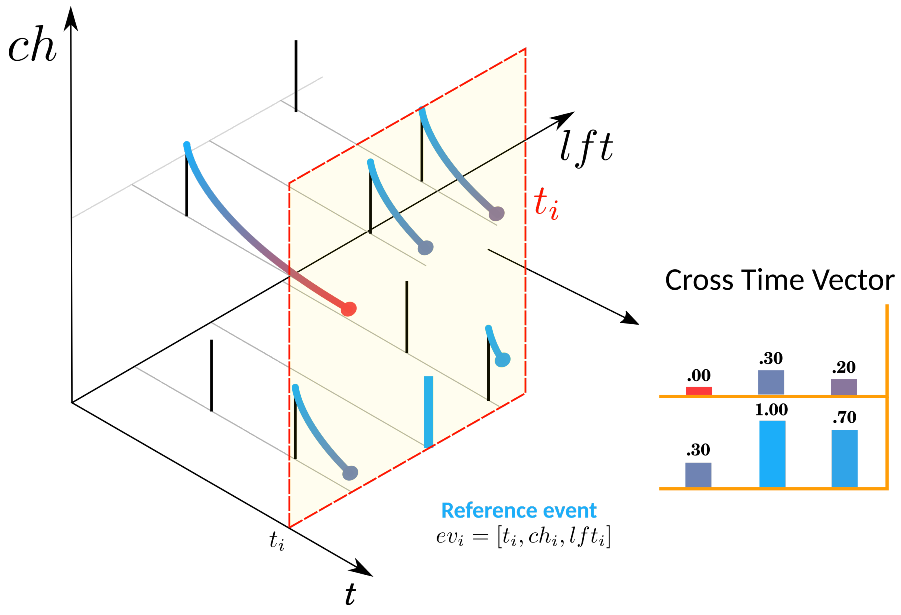

Figure 6: Example of Cross Time-Vectors generation. Each time the second layer receive a new event, the algorithm pools the last event of each channel and local feature. It then calculates the exponential decay for each event at the reference time t_i (yellow surface) and then it composes a Cross Time-Vector containing all the decay values for all the combinations of channel/local feature.

2.5.2 Learning and inferring cross features

Similarly for the Local features layer we use the same Kmeans algorithm [31] to learn a set of clusters of length from a training set of cross time-vectors. When inferring, every incoming event will generate a cross time-vector and will be assigned to the closest cluster generating a new event:

$$ev_i = [t_i, cft_i] \qquad i \in \mathbb{N} \tag{10}$$

Where $cft_i \in [0, ck] \subset \mathbb{N}$ is the index of the matching cross-feature, and ck is the total number of clusters for the cross layer.

2.6 Activity Histogram

One way to classify the output of event based networks is to build histograms of activity [17]. For a single audio recording or time window, we can build an histogram $\mathcal{H}$ by counting the number of events generated by the network per cross feature cft. We can then apply different classifiers on activity histograms in order to recognize spoken words, phonemes or sounds. In Figure 7 we show the network response to a spoken word ("off").

2.7 Classifier

In this paper we test the network with three different classifiers: Euclidean distance, Normalised Euclidean distance, and a Multi-Layer-Perceptron (MLP) model. In the first case, we compute the output histograms of the training set of

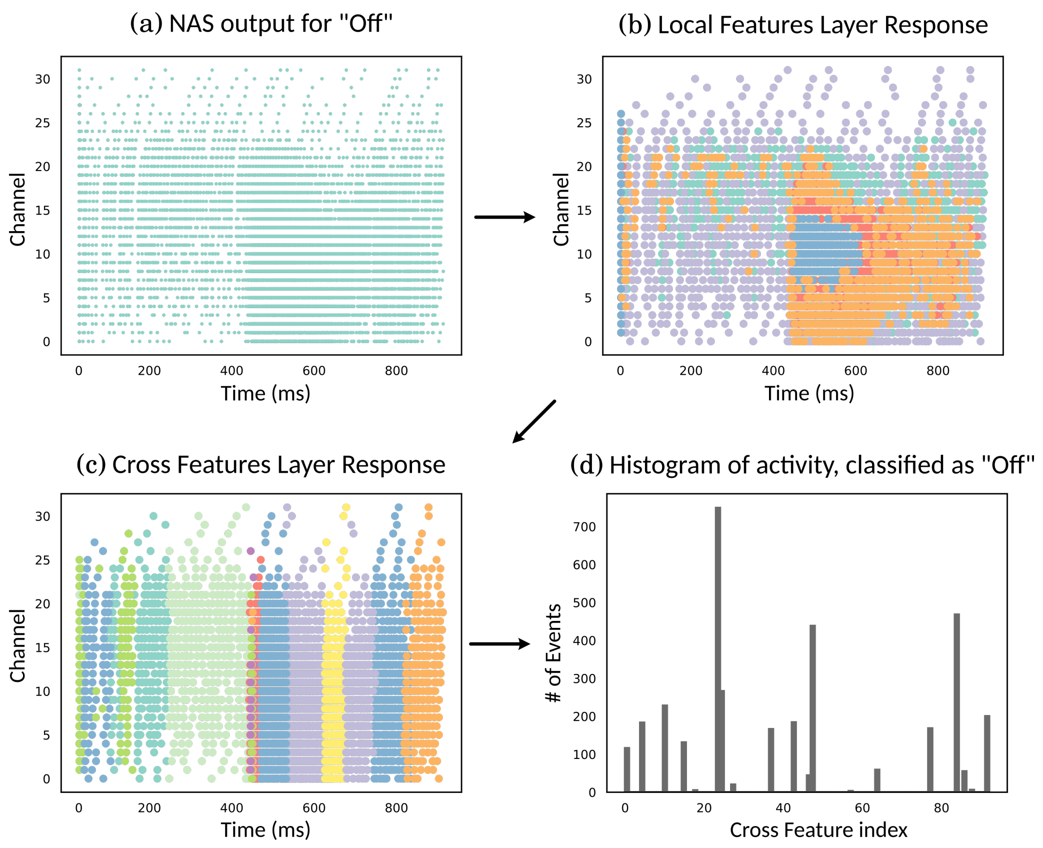

Figure 7: The response of the network to a recording of the spoken word ”Off”. The NAS spiking activity (a) is classified by the Local Feature Layer which response has been color-coded by the index of the corresponding local feature (b). Each event is then classified by Cross Feature by the Cross Feature layer (c), and finally the network response is collected by the Histogram of activity that shows the total number of events per Cross Feature.

the dataset, and then we average them by class, creating a set of ”prototypes” $\mathcal{R}$. Then for every new recording we can generate a new histogram and calculate its euclidean distance to the prototype:

$$d(\mathcal{H}_{recording}, \mathcal{R}_{class}) = \|\mathcal{H}_{recording} - \mathcal{R}_{class}\| \tag{11}$$

We then select the class with the minor distance to the recording. In the case of Normalized Euclidean distance we divide the histogram of each recording by the total number of events generated, also creating a new set of normalized prototypes $\mathcal{NR}$:

$$dn(\mathcal{H}_{recording}, \mathcal{NR}_{class}) = \|\frac{\mathcal{H}_{recording}}{card(\mathcal{H}_{recording})} - \mathcal{NR}_{class}\| \tag{12}$$

where card indicates the number of events of the recording. Finally we also use an MLP classifier to classify the histograms of training and test sets.

3 Results

We test the network on the yes/no words we recorded from the Google’s Speech Command dataset [19]. We randomly select 3740 ”yes” recordings and 3740 ”no” recordings. We then use a random sample of 70% for training set and 30% as testing set, while keeping the same number of samples for both classes. After testing different combinations of parameters we set $t_{local} = 1ms$, $n = 5$, and $lk = 6$ for the Local features layer, and $t_{cr} = 200ms$, and $ck = 96$ for the Cross features layer. Finally for the MLP classifier we select a network with a single hidden layer of 30 ”RelU” neurons. The result of this setup are reported in Table 1.

Classifiers	Train set accuracy	Test set accuracy
Eucl Histogram	74.66%	74.42%
Norm Eucl Histogram	75.06%	73.80%
MLP	83.41%	82.55%

Table 1: Results of the "yes"/"no" classification test.

4 Conclusions

We presented a neuromorphic network capable of unsupervised auditory features extraction, compatible with a digital neuromorphic cochlea [14]. We then coupled the network with three classifiers and tested it on a subset of the Google's Speech Commands dataset. The preliminary results showed that network was able to recognize simple commands for the Google's Speech Commands dataset with good accuracy, even when paired with weak classifiers, suggesting that the set of Local and Cross features extracted were informative for the task of word recognition. In order to completely evaluate the network we still need to expand the result section with the rest of Google's Speech Commands dataset and test the network with other datasets as well. Nevertheless, the compactness of the architecture, the small number of features per layer ($lk = 6$, $ck = 96$) and the simplicity of the classifiers make this network a good candidate for edge-computing sound recognition/NLP devices.

References

[1] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, oct 2021.

[2] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[3] A. Market, "Global industry analysis, size, share, growth, trends, and forecast 2017–2025," *URL: https://www. transparencymarketresearch. com/logistics-market. html (Last accessed: 16.01. 2018)*, 2018.

[4] P. Kumari, L. Mathew, and P. Syal, "Increasing trend of wearables and multimodal interface for human activity monitoring: A review," *Biosensors and Bioelectronics*, vol. 90, pp. 298–307, 2017.

[5] J. Harrop, "Mobile robotics expected to fuel industry growth in robotics market," *Control Engineering*, vol. 62, no. 2, pp. 20–21, 2016.

[6] J. Chen and X. Ran, "Deep learning with edge computing: A review.," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[7] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[8] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[9] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[10] J. Zhu, T. Zhang, Y. Yang, and R. Huang, "A comprehensive review on emerging artificial neuromorphic devices," *Applied Physics Reviews*, vol. 7, no. 1, p. 011312, 2020.

[11] M. Yang, C.-H. Chien, T. Delbruck, and S.-C. Liu, "A 0.5 V 55μW 64$\times$2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, 2016.

[12] V. Chan, S.-C. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[13] Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. M. Wang, and A. van Schaik, "A fpga implementation of the car-fac cochlear model," *Frontiers in neuroscience*, vol. 12, p. 198, 2018.

[14] A. Jiménez-Fernández, E. Cerezuela-Escudero, L. Miró-Amarante, M. J. Domínguez-Morales, F. de Asís Gómez-Rodríguez, A. Linares-Barranco, and G. Jiménez-Moreno, "A binaural neuromorphic auditory sensor for FPGA: A spike signal processing approach," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 4, pp. 804–818, 2016.

[15] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

[16] T. Schoepe, D. Gutierrez-Galan, J. P. Dominguez-Morales, A. Jimenez-Fernandez, A. Linares-Barranco, and E. Chicca, "Neuromorphic sensory integration for combining sound source localization and collision avoidance," in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, IEEE, 2019.

[17] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: a hierarchy of event-based time-surfaces for pattern recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1346–1359, 2016.

[18] G. Haessig and R. Benosman, "A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors," in *Micro-and Nanotechnology Sensors, Systems, and Applications X*, vol. 10639, p. 106391U, International Society for Optics and Photonics, 2018.

[19] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[20] R. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7, pp. 1282–1285, IEEE, 1982.

[21] A. Jimenez-Fernandez, A. Linares-Barranco, R. Paz-Vicente, G. Jiménez, and A. Civit, "Building blocks for spikes signals processing," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2010.

[22] A. Jimenez-Fernandez, G. Jimenez-Moreno, A. Linares-Barranco, M. J. Dominguez-Morales, R. Paz-Vicente, and A. Civit-Balcells, "A neuro-inspired spike-based pid motor controller for multi-motor robots with low cost fpgas," *Sensors*, vol. 12, no. 4, pp. 3831–3856, 2012.

[23] "The Adress-Event Representation communication protocol."

[24] R. Paz-Vicente, A. Linares-Barranco, A. Jimenez-Fernandez, G. Jimenez-Moreno, and A. Civit-Balcells, "Synthetic retina for AER systems development," in *2009 IEEE/ACS International Conference on Computer Systems and Applications*, pp. 907–912, IEEE, 2009.

[25] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[26] S. Rosen and P. Howell, *Signals and systems for speech and hearing*, vol. 29. Brill, 2011.

[27] R. Berner, T. Delbruck, A. Civit-Balcells, and A. Linares-Barranco, "A 5 Meps $100 USB2.0 address-event monitor-sequencer interface," in *2007 IEEE International Symposium on Circuits and Systems*, pp. 2451–2454, IEEE, 2007.

[28] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *2008 IEEE International Symposium on Circuits and Systems*, pp. 2130–2133, IEEE, 2008.

[29] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, 2010.