Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach

Juan P. Dominguez-Morales[1], Qian Liu[2], Robert James[2], Daniel Gutierrez-Galan[1], Angel Jimenez-Fernandez[1], Simon Davidson[2], and Steve Furber[2]
[1]Robotics and Computer Technology Lab.
University of Seville, Seville, Spain
Email: jpdominguez@atc.us.es
[2]Advanced Processor Technologies Group, School of Computer Science.
University of Manchester. Manchester, UK

***Abstract*—Speech recognition has become an important task to improve the human-machine interface. Taking into account the limitations of current automatic speech recognition systems, like non-real time cloud-based solutions or power demand, recent interest for neural networks and bio-inspired systems has motivated the implementation of new techniques.**

Among them, a combination of spiking neural networks and neuromorphic auditory sensors offer an alternative to carry out the human-like speech processing task. In this approach, a spiking convolutional neural network model was implemented, in which the weights of connections were calculated by training a convolutional neural network with specific activation functions, using firing rate-based static images with the spiking information obtained from a neuromorphic cochlea.

The system was trained and tested with a large dataset that contains "left" and "right" speech commands, achieving 89.90% accuracy. A novel spiking neural network model has been proposed to adapt the network that has been trained with static images to a non-static processing approach, making it possible to classify audio signals and time series in real time.

***Index Terms*—speech recognition, audio processing, Spiking Neural Networks, Convolutional Neural Networks, neuromorphic hardware, deep learning.**

I. Introduction

Voice commands are commonly used in multiple personal virtual assistants [1], like Cortana in Microsoft Windows, or Siri in iOS. Users are able to control their personal computers or mobile phones by using natural language sentences, like "Remind me to call Robert in the afternoon", or more directly, "Call Robert". This kind of assistants are based on a field of Artificial Intelligence (AI) called Natural Language Processing (NLP) to identify what the user is saying [2], [3]. The audio is processed and analyzed using Digital Signal Processing (DSP) techniques, such as speech processing [4].

Speech recognition is the interdisciplinary sub-field of speech processing, in which spoken sentences are recognized and translated to text (or other data representation) using specific methodologies. Typically, these methods identify each spoken word in isolation, applying several processing steps to obtain features that are then mapped to a specific word [5].

In recent years, the application of Artificial Neural Network (ANN) to this field has become commonplace. Notably, the combination of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) has led to significant progress in developing human-machine interface, as in [6], [7], [8], [9]. Recently, the Google WaveNet system [10] demonstrated significantly improved comprehension of entire conversations as well as being able to generate human-like speech from text, based on a CNN trained on raw audio voice characterization.

Training CNNs is a relatively easy task. There exist several frameworks and training mechanisms to achieve this. The most used training algorithm (for ANN and CNN training) is the well-known Levenberg-Marquardt back-propagation algorithm [11]. In contrast, there is no established standard training algorithm for Spiking Neural Networks.

Spike-Time-Dependant Plasticity (STDP) is a biological process that is able to adjust the strength (weights) of the connections between neurons based on the relative timing of a particular neuron's output and input spiking activity. This process has been implemented in several simulators and hardware platforms, including SpiNNaker [12], and has become one of the most ubiquitous approaches for training spike-based networks especially for unsupervised learning [13]. STDP has proved to be very useful and robust for static input signals like images [14], [15], but it is more difficult to apply when it comes to processing time-varying signals such as audio samples.

As an alternative to STDP, the weights of the connections between neurons in a network could be set by hand or based on particular statistical algorithms. This approach was taken into account in papers like [16], in which the authors set the weights using two different firing-rate based normalizations for classifying between eight different pure tones. This option is complex because it generally needs several trial-and-error loops in order to find the best weight configuration, which can take a long time. Also, this way of setting the weights of the connections is too task specific and lacks the generality and

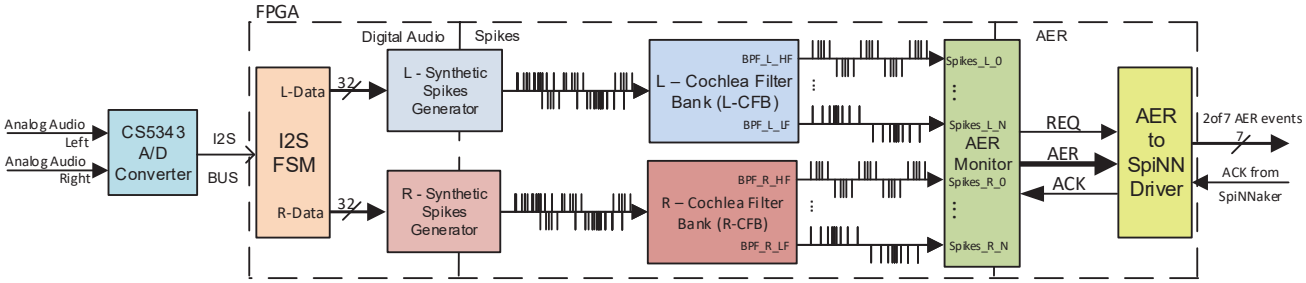

Fig. 1: Neuromorphic Auditory Sensor (NAS) block diagram.

biological plausibility of STDP.

Due to the increasing interest in SNNs, numerous works have tried to develop new frameworks or methods to automatically train SNN models. The first approach is to develop new STDP-based algorithms, as in [17], who used the force firing technique for incremental learning, making it possible to learn new patterns continually in real-time using an unsupervised learning procedure. Also, in [18], the authors used a new learning rule, named fatiguing STDP, which combines the long-term STDP dynamics with a mechanism of short-term synaptic fatigue dynamics.

There are many other bio-inspired techniques for training neural models, such as the use of evolutionary algorithms [19] to adjust the weights of the network.

In recent time, the difference in classification error between deep SNNs and deep ANNs has diminished significantly [20]. These exciting results suggest that, if trained appropriately, an SNN can be used for machine learning inference without introducing penalties in data classification accuracy. Using a deep SNN instead of a deep ANN alternative can provide a machine learning system with power saving and input noise tolerance benefits [21].

Additionally, such deep SNNs can be trained on input data generated from a neuromorphic spiking sensor device, unlocking the potential for a real-time inference system on a spiking neuromorphic platform [22]. We believe that only when these are combined the true strengths of a fully spike-based processing system will be apparent.

The aforementioned developments in deep SNNs show accurate classification of static input data (images) using a deep convolutional SNN. We show in this work that we are able to train a similarly structured network on time series input data from a Neuromorphic Auditory Sensor (NAS) [23] produced from a range of sound inputs. By using a technique of generating a training dataset consisting of many overlapping 'snapshots' of the NAS output and a 'time-buffering' input to the SNN, we are able to produce a robust inference on time varying spiking inputs.

The rest of the paper is structured as follows: section II presents an overview of the system architecture and the speech commands database that was used in this work along with how the train and test datasets were generated. Then, section III describes the whole framework that was used to train and simulate the SNN with the audio samples dataset that was obtained from the previous section. Then, section IV describes the results of both the training and the simulation. Up to this point, this setup is used for training and testing the system with static inputs (audio samples are converted into images), so in section V we propose a novel SNN architecture to use the network that was previously trained with static data in real time using a live input from a neuromorphic cochlea. Finally, the conclusions of this work are presented in section VI.

II. System overview, dataset acquisition and preprocessing of the information

In audio processing, a Digital Signal Processor is usually used to carry out large audio processing tasks, due to the computational capabilities of these devices. The neuromorphic approach uses bio-inspired devices that mimic the behavior of biological senses, reproducing with greater fidelity the individual steps by which the ear and the auditory cortex interact to process aural information.

In recent years, several researchers have developed theoretical cochlea models, using either analog or digital circuits to implement their models. As a result, many neuromorphic hardware platforms have appeared and are being used in research projects. There exist several models of analog [24] [25] [26] and digital cochleae [27] [28] [29] [30].

In this work we use a Neuromorphic Auditory Sensor [23] (NAS), which is a digital cochlea implementation. It is a FPGA-based sensor, in which all processing modules are spike-based. As it is implemented on a reconfigurable platform, this sensor's configuration parameters are flexible and can be adapted to any application.

This kind of sensors mimic how the biological cochlea processes audio signals. The cochlea is able to decompose the input audio signal into different frequency bands (also called channels). This decomposition is carried out by a series of cascade-connected stages that subtract the information from consecutive spike-based low-pass filters' output spikes in order to reject out-of-band frequencies, obtaining a response equivalent to that of a bandpass filter [23]. The entire NAS architecture is shown in Fig. 1. A flow of spikes coded as AER (Address-Event Representation) [31] events is obtained in the output, which can be either sent to the SpiNNaker board

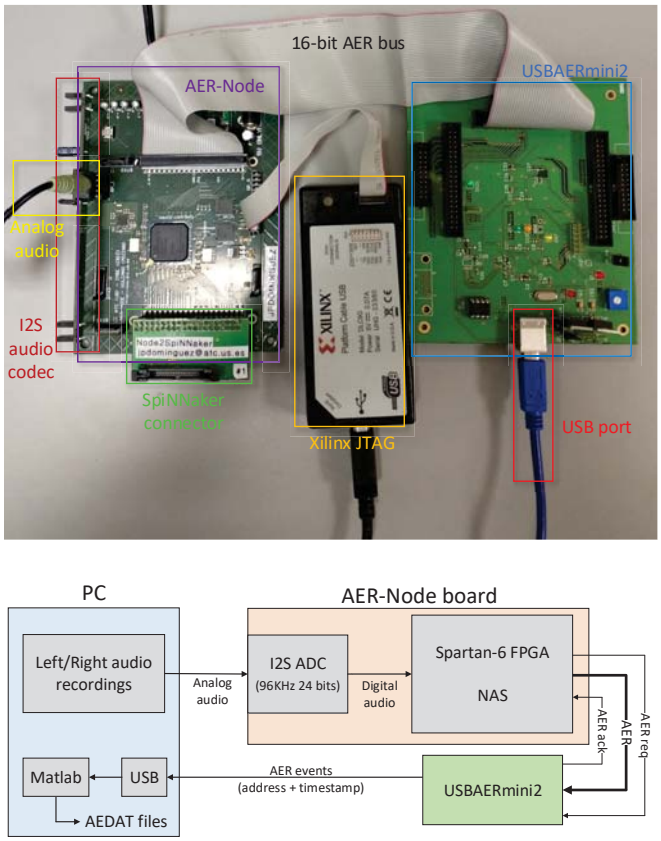

Fig. 2: Picture (top) and block diagram (bottom) of the hardware setup for the dataset generation.

through the AER-SpiNNaker interface module [32] or to the computer using a USBAERmini2 board [33].

A 32-channel mono-aural NAS was used in this work, employing this neuromorphic approach in a speech recognition task where spoken commands corresponding to the words ”left” and ”right” are classified. The Speech Command dataset, which consists of 65000 one-second long utterances of 30 short words, was used in this work. This data was collected by Google and released under a Creative Commons BY 4.0 license. Only the ”left” and ”right” voice commands of the dataset (a total of 4720 audio files from thousands of different speakers) were used in this work, since one of the final goals of the COFNET project is to drive a robot by using only these two voice commands.

Each of the audio samples were sent to the audio input of an AER-Node platform [34], which consists of a Spartan-6 FPGA in which a 32-channel mono-aural NAS is programmed. With this sensor the audio signal is decomposed into frequency bands and then packetized using the Address-Event Representation protocol (AER). An USBAERmini2 board receives this information and sends it to the computer through the USB port. A MATLAB script is used to collect the AER packets that are received through the serial port and to store them into AEDAT[1] files (one file is generated per audio sample), which is a common format used for storing this kind of information. The hardware setup used for generating the dataset is shown in Fig. 2.

These AEDAT files were then converted to sonogram images using NAVIS’s algorithms [35] in order to train a CNN. To do this, a bin width of 20 ms was selected in order to calculate the firing rate for each of the NAS’ channels in every bin. This was done by counting the number of spikes fired in that portion of time and dividing that value by 20 ms (see Algorithm 1), which is the length that was selected for this work. Fig. 3 shows images from both the ”left” and the ”right” classes after this process was carried out. In order to make the training of the network more robust to a real scenario, in which the core information of the audio could be presented not only in the center of the image but in any position of it, an overlapping shifting window was used, generating several images for each audio sample with the information centered in different timestamps.

Algorithm 1 Sonogram calculation

```
bin_width = 20 ms
sonogram = zeros(max(in_addr), max(in_timeStamp)/bin_width)
for i=1:max(in_addr) do
    sonogram(in_addr(i), in_tStamp(i)/bin_width)++
end for
sonogram = sonogram/bin_width
```

A total of 141726 images were generated in this process, 121565 of which were used to train the network and the remaining 20161 images to test it and obtain the accuracy ratio of the system.

III. Off-line SNN training and SNN construction

The general off-line SNN training method proposed by Liu et. al. [36] is based on two novel activation functions. One is Noisy Softplus (NSP) [20], which closely mimics the LIF firing activity driven by current influx with different noise levels. The other, Parametric Activation Function (PAF), maps abstract numerical numbers of activation functions to specific physical units of a spiking neuron. Thus, the combination provides an equivalent representation of a spiking LIF neuron with abstract activation functions of ANNs. PAF allows using more generalized activation functions (e.g., ReLU instead of NSP) to model a LIF neuron once its parameters are fitted by NSP. Therefore, the weights of a SNN can be trained off-line on an equivalent ANN exactly the same way as conventional ANNs (e.g., using Backpropagation and Stochastic Gradient Descent), but using PAFs. The simple steps can be described as follows: firstly, estimate the parameter of PAFs; then, train an equivalent ANN using the PAF version of the activation functions (e.g., PAF-ReLU); finally, transfer the trained weights back to the SNN without further transformation.

The off-line SNN training tool is published in Github[2]. It is comprised of two main parts: the Matlab code for ANN

[1]https://inilabs.com/support/software/fileformat
[2]https://github.com/qian-liu/off_line_SNN

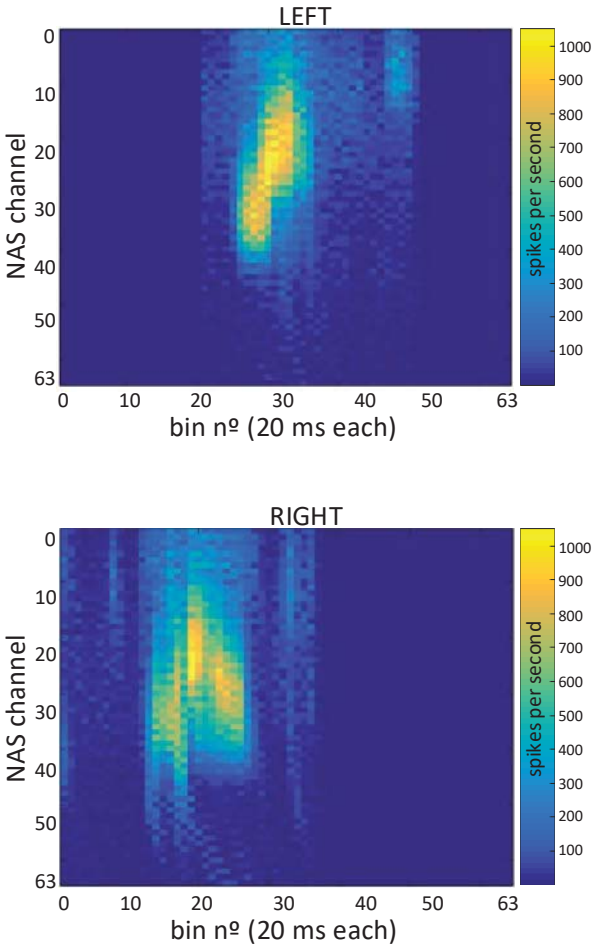

Fig. 3: Sonogram images corresponding to one "Left" (top) and one "Right" (bottom) audio samples from the Speech Command dataset after obtaining their spiking information from the NAS.

training and the Python code for reading trained weights and translating into PyNN language. The Matlab code is based on an ANN training tool called DeepLearnToolbox[3], and we implemented the two activation functions described above. It is worth noting that the NSP and its derivative takes two variables as inputs: the mean of the noisy current x and its variance σ. Therefore, the computation of both the forward and backward paths are doubled and the state to be stored is also doubled in size. The PAF is easily implemented by multiplying the parameter p of the original activation function: $p \times f(x)$.

The Python code reads the network architecture of an ANN layer-by-layer, and constructs equivalent populations of LIF neurons accordingly. It then takes the layer-wise weights of the ANN and translates them to the connection list between populations of LIF neurons. After the building-up phase, the testing code (which is simulated in NEST) generates Poisson Spike trains based on parameter configurations and the intensity of pixels of an input image; then, it feeds the network with the spike trains and records the output spike trains on the classification layer; finally, it analyses the results where the highest firing rates determine the class to which an image is assigned. The overall performance on the whole testing dataset is then compared with the ANN testing result.

[3]https://github.com/rasmusbergpalm/DeepLearnToolbox

IV. Results

A 5C-3P-3C-2P Spiking Convolutional Neural Network (5x5 kernel-size convolutional layer followed by a 2x2 pooling layer, another 3x3 convolutional layer followed by a 2x2 pooling layer, and then a fully connected layer) was trained in Matlab with rate-based sonograms (See Fig. 3) that contained the firing rate information obtained from a NAS using "left" and "right" speech commands from a well-known open database that was presented in section II. The CNN architecture is shown in Fig. 4.

An accuracy result of 92.21% was achieved when training the CNN in Matlab for 30 epochs, at a learning rate value of 0.1 and a synaptic time constant of 0.005 ms, using the ReLU activation function on the fully connected layer. After this, the network was fine-tuned for one more epoch with the Noisy Softplus activation function that was described in section III, starting off with the weights of the connections that were obtained from the previous step (using ReLU as the activation function).

After the fine-tuning process, the performance of the network was almost the same, obtaining 90.80% accuracy. As was explained in previous sections, the trained weights were tweaked (fine-tuned) in this process, resulting on a slightly lower accuracy value in this case (less than 2% decrease), but improving the performance when translating from the ANN in Matlab to a SNN in pyNN (NEST).

The weights obtained from the ANN training and fine-tuning in Matlab were then saved and used to test a SNN. The SNN was built in pyNN for the NEST simulator based on the architecture of the ANN that was trained in the previous step. The network was tested using 20161 samples, achieving 89.90% accuracy (the confusion matrix is shown in Fig. 5). As can be observed, the result obtained in the "left"/"right" classification in the SNN simulation that was run on NEST is almost the same as the one that was obtained when training the ANN in Matlab, meaning that, with this process, the authors have found a proper way to train audio signals (or time series) without compromising the accuracy of the network.

Tests were carried out using the NEST simulator and also deploying the whole SNN model in a 48-chip SpiNNaker hardware platform. In future works, the authors would be focusing on making use of the NAS-SpiNNaker live connection [37] to test the speech commands recognition using a real-time input from a microphone connected to the NAS. The next section will describe the SNN architecture for testing this approach in real-time.

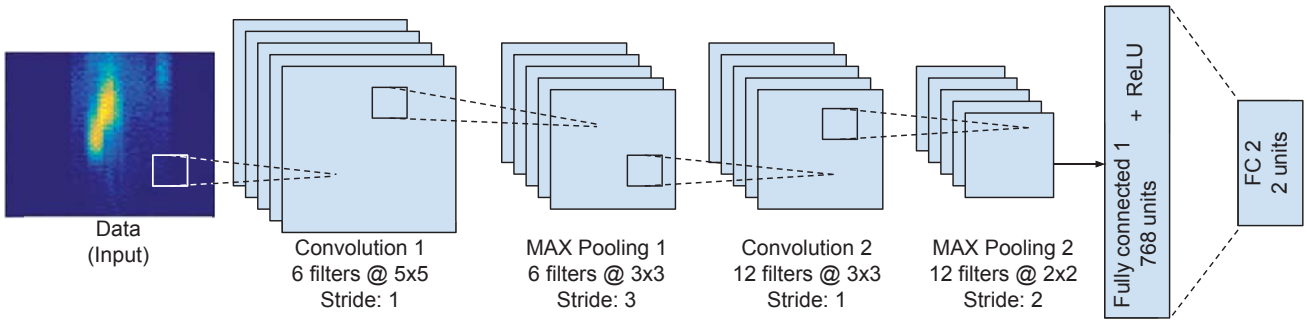

Fig. 4: CNN architecture used for training the ”left”/”right” commands in Matlab.

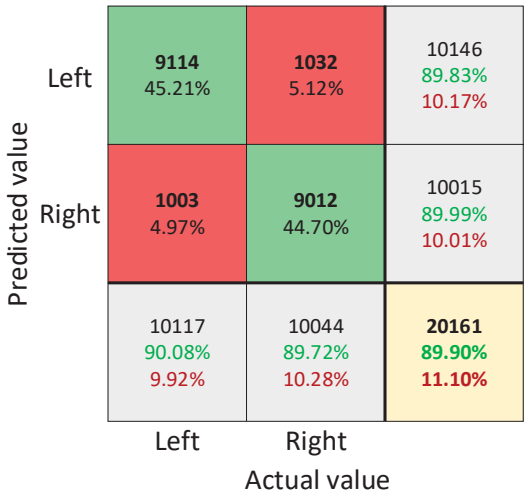

Fig. 5: Confusion matrix of the SNN test using 20161 samples (10117 ”left” and 10044 ”right” samples).

V. SNN ARCHITECTURE FOR AUDIO SAMPLES CLASSIFICATION IN REAL TIME

The accuracy result of the system achieved when using the method described in section III proves that this mechanism could be used to classify audio samples like the ones used in this work or even more complex ones as long as they can be converted into images.

This is a completely offline approach, which means that audio samples are not being inputted in real time. These samples have to be already recorded and converted into spikes in order to classify them. The point on using the NAS is that, besides of processing the sound information in a bio-inspired way, it is able to provide a real time output with the audio signal decomposed into frequency bands (32 bands or cochlea channels in this case) and already converted into spikes, as the biological cochlea would do.

Even when not making use of the real-time capabilities of this neuromorphic sensor, using it could be useful for tasks in which the classification does not need to be done in a short period of time. In [38], Dominguez-Morales et al. use a NAS to process heart sounds recordings and classify whether it is a healthy person or a pathological patient in order to help cardiologists in the auscultation process. Applications like this do not require an immediate output from the classifier, meaning that the sound could be recorded and analyzed later.

However, in tasks like robot navigation with speech commands, recognizing the command and acting on the motors of the robot are actions that need to be done as soon as possible. Otherwise, the navigation would not feel fluid and natural. One of the main goals of the COFNET project (TEC2016-77785-P) is to drive a 4-wheel SUMMIT XL robot from Robotnik using the fusion of the neuromorphic information coming from a neuromorphic retina (Dynamic Vision Sensor) and from a NAS (using speech commands). To accomplish this while using the same training approach considered in this work, the authors propose the SNN architecture shown in Fig. 6.

This architecture takes into account that the input data has been trained using a deep Spiking Convolutional Neural Network (SCNN) as it was a static input (image). That is, the image is converted from a matrix (two-dimensional array) to a single dimension array by flattening the matrix (e.g. a 28x28 MNIST image is converted to an array of 768 elements). The whole trained SCNN is presented in the figure in a cloud shape. To adapt the trained model in order to use real-time input from the NAS, a new layer of spiking populations has to be added.

These populations act as a layer between the NAS and the trained SCNN and its goal is to adapt the spiking information that comes out of the NAS in real time in order to serve as input to the network. This is done by having 64 populations (due to the fact that the network is trained with 64 20 ms-bins images) of 64 neurons each (two neurons per NAS channel) that are connected like a daisy chain, with delayed one-to-one connections between every two. The delay that is set for these connections is 20 ms, because of the bin width used. NAS’s output is connected to the first of these populations, propagating the same spiking information to the next one after 20 ms. Then, each of the 64 populations of this layer is connected to the previously trained SCNN with no delay. This way, as soon as the speech command is sent to the NAS, the populations between the NAS and the SCNN will start to

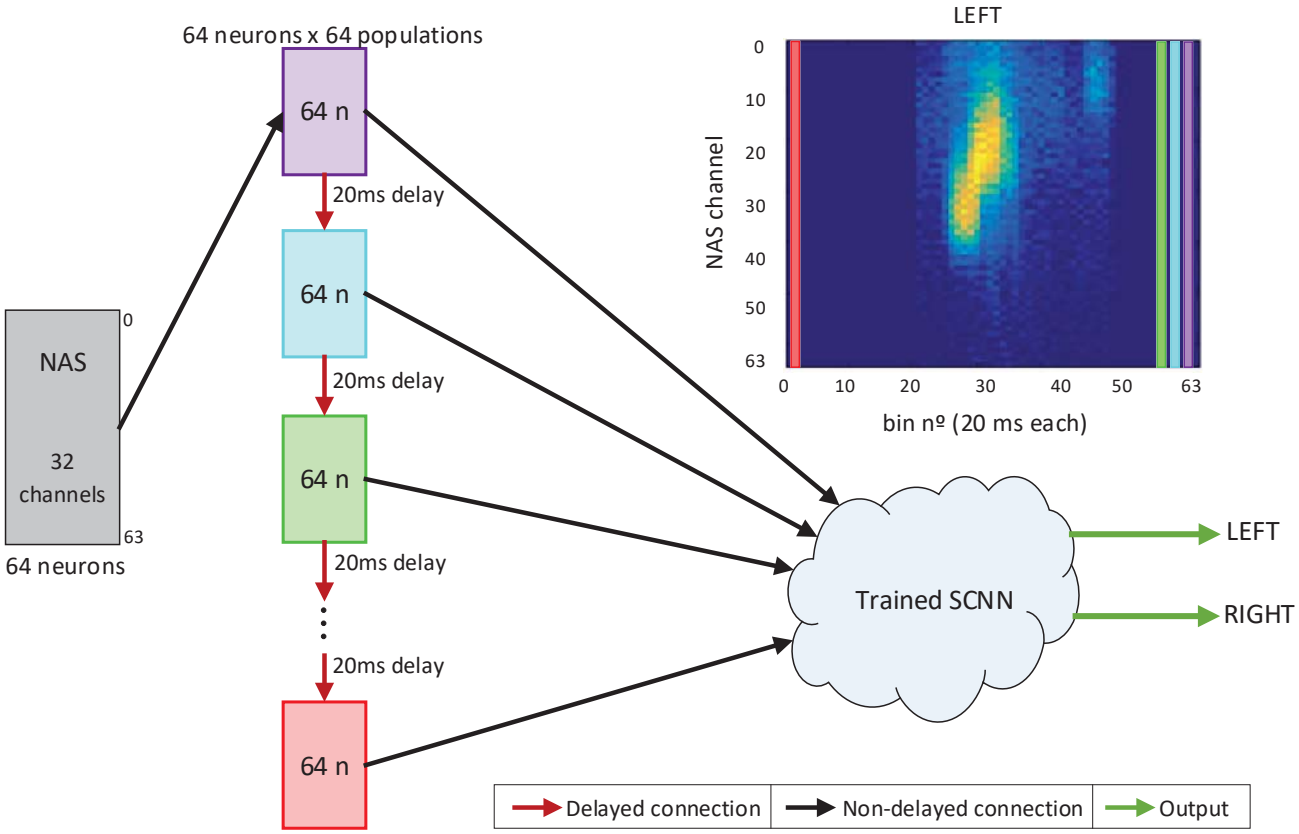

Fig. 6: Real-time NAS audio input SCNN scenario with a buffering layer consisting of a set of delayed populations.

propagate the information. This "time-buffered" architecture allows to run real-time experiments for speech recognition and audio samples classification with a previous step of training the network with static audio images, which is a novelty in the neuromorphic engineering field. Having several images for the same speech command with the information shifted and centered in a different bin allows not only the training to be more robust but also the real-time test to take less time to start providing the correct result. That is, spikes from the NAS do not need to propagate through many populations to be classified correctly since the network was trained to recognize that the important information of the speech command could also appear in the first bins (which correspond to the first populations) instead of just in the middle section of the sonogram.

VI. Conclusions

In this work, the authors have presented a novel mechanism for training time series offline and testing them later in real time in a Spiking Convolutional Neural Network with the information obtained from the live output of a Neuromorphic Auditory Sensor.

The results obtained in this work prove that almost the same accuracy results (1% less in this case) can be achieved when testing a deep Spiking Neural Network using the weights obtained from a previously trained Convolutional Neural Network with spike-rate based images, which is a novelty for time-dependent signals like audio signals.

A database with 4720 "left" and "right" speech commands from the Speech Commands Dataset was used to generate 141726 sonogram images with the spiking information obtained from a neuromorphic cochlea (NAS). These images were later used for training and testing the system, achieving an accuracy result of 89.90% when simulating and deploying the network in the SpiNNaker hardware platform.

The authors have also presented a novel SNN architecture for audio samples classification in real time using the output from a neuromorphic sensor as input to the network and a buffering layer with delayed populations that adapts the information from a real-time domain to a static domain, in which the SNN is trained for. This approach could also be used for processing time series or time-dependent signals with SNNs in real time.

Acknowledgment

This work is supported by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P). The work of Juan P. Dominguez-Morales was supported by a Formación de Personal Universitario Scholarship from the Spanish Ministry of

Education, Culture and Sport. Juan P. Dominguez-Morales and Daniel Gutierrez-Galan would like to thank Simon Davidson, Steve Furber and the whole APT group for their kindness and constant help during their stay in Manchester.

REFERENCES

[1] R. S. Cooper, J. F. McElroy, W. Rolandi, D. Sanders, R. M. Ulmer, and E. Peebles, "Personal virtual assistant," Jun. 29 2004, uS Patent 6,757,362.

[2] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[3] G. Gazdar and C. S. Mellish, *Natural language processing in Lisp: An introduction to computational linguistics*. Addison-Wesley Wokingham, England, 1989.

[4] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR Upper Saddle River, 2001, vol. 95.

[5] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[6] G. Williams and S. Renals, "Confidence measures for hybrid HMM/ANN speech recognition." 1997.

[7] P. McGuire, J. Fritsch, J. J. Steil, F. Rothling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, "Multi-modal human-machine communication for instructing robot grasping tasks," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1082–1088.

[8] O. Russakovsky, L.-J. Li, and L. Fei-Fei, "Best of both worlds: human-machine collaboration for object annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2121–2131.

[9] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[10] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[12] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[13] P. U. Diehl and M. Cook, "Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 4288–4295.

[14] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, p. 4, 2015.

[15] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep neural networks for object recognition," *arXiv preprint arXiv:1611.01421*, 2016.

[16] J. P. Dominguez-Morales, A. Jimenez-Fernandez, A. Rios-Navarro, E. Cerezuela-Escudero, D. Gutierrez-Galan, M. J. Dominguez-Morales, and G. Jimenez-Moreno, "Multilayer spiking neural network for audio samples classification using SpiNNaker," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 45–53.

[17] Z. Hu, T. Wang, and X. Hu, "An STDP-based supervised learning algorithm for spiking neural networks," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 92–100.

[18] T. Moraitis, A. Sebastian, I. Boybat, M. Le Gallo, T. Tuma, and E. Eleftheriou, "Fatiguing STDP: Learning from spike-timing codes in the presence of rate codes," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1823–1830.

[19] N. Pavlidis, O. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 4. IEEE, 2005, pp. 2190–2194.

[20] Q. Liu and S. Furber, "Noisy Softplus: A biology inspired activation function," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 405–412.

[21] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreo, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in Neuroscience*, vol. 6, p. 32, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00032

[22] F. Perez-Peña, J. A. Leñero-Bardallo, A. Linares-Barranco, and E. Chicca, "Towards bioinspired close-loop local motor control: A simulated approach supporting neuromorphic implementations," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[23] 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, 2017.

[24] S.-C. Liu, A. Van Schaik, B. A. Mincti, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 2027–2030.

[25] B. Wen and K. Boahen, "A silicon cochlea with active coupling," *IEEE transactions on biomedical circuits and systems*, vol. 3, no. 6, pp. 444–455, 2009.

[26] T. J. Hamilton, C. Jin, A. van Schaik, and J. Tapson, "An active 2-D silicon cochlea," *IEEE Transactions on biomedical circuits and systems*, vol. 2, no. 1, pp. 30–43, 2008.

[27] C. D. Summerfield and R. F. Lyon, "ASIC implementation of the Lyon cochlea model," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 5. IEEE, 1992, pp. 673–676.

[28] A. Mishra and A. E. Hubbard, "A cochlear filter implemented with a field-programmable gate array," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 54–60, 2002.

[29] M.-P. Leong, C. T. Jin, and P. H. Leong, "An FPGA-based electronic cochlea," *EURASIP Journal on Applied Signal Processing*, vol. 2003, pp. 629–638, 2003.

[30] I. Gambin, I. Grech, O. Casha, E. Gatt, and J. Micallef, "Digital cochlea model implementation using Xilinx XC3S500E spartan-3E FPGA," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 946–949.

[31] The Adress-Event Representation communication protocol. [Online]. Available: https://www.ini.uzh.ch/ amw/scx/std002.pdf

[32] L. Plana, J. Heathcote, J. Pepper, S. Davidson, J. Garside, S. Temple, and S. Furber, "spI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems."

[33] 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*. IEEE, 2007, pp. 2451–2454.

[34] T. Iakymchuk, A. Rosado, T. Serrano-Gotarredona, B. Linares-Barranco, A. Jimenez-Fernandez, A. Linares-Barranco, and G. Jimenez-Moreno, "An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 1556–1559.

[35] J. P. Dominguez-Morales, A. Jimenez-Fernandez, M. Dominguez-Morales, and G. Jimenez-Moreno, "NAVIS: Neuromorphic Auditory VISualizer tool," *Neurocomputing*, vol. 237, pp. 418–422, 2017.

[36] Q. Liu, Y. Chen, and S. Furber, "Noisy Softplus: an activation function that enables SNNs to be trained as ANNs," *arXiv preprint arXiv:1706.03609*, 2017.

[37] J. P. Dominguez-Morales, A. Rios-Navarro, D. Gutierrez-Galan, R. Tapiador-Morales, A. Jimenez-Fernandez, E. Cerezuela-Escudero, M. Dominguez-Morales, and A. Linares-Barranco, "Live demonstration-multilayer spiking neural network for audio samples classification using SpiNNaker," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–1.

[38] J. P. Dominguez-Morales, A. F. Jimenez-Fernandez, M. J. Dominguez-Morales, and G. Jimenez-Moreno, "Deep neural networks for the recognition and classification of heart murmurs using neuromorphic au-

ditory sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 24–34, Feb 2018.