Multilayer Spiking Neural Network for Audio Samples Classification Using SpiNNaker

Juan Pedro Dominguez-Morales, Angel Jimenez-Fernandez, Antonio Rios-Navarro, Elena Cerezuela-Escudero, Daniel Gutierrez-Galan, Manuel J. Dominguez-Morales, and Gabriel Jimenez-Moreno

Robotic and Technology of Computers Lab,
Department of Architecture and Technology of Computers,
University of Seville, Seville, Spain
{jpdominguez,ajimenez,arios,ecerezuela,
dgutierrez,mdominguez,gaji}@atc.us.es
http://www.atc.us.es

Abstract. Audio classification has always been an interesting subject of research inside the neuromorphic engineering field. Tools like Nengo or Brian, and hardware platforms like the SpiNNaker board are rapidly increasing in popularity in the neuromorphic community due to the ease of modelling spiking neural networks with them. In this manuscript a multilayer spiking neural network for audio samples classification using SpiNNaker is presented. The network consists of different leaky integrate-and-fire neuron layers. The connections between them are trained using novel firing rate based algorithms and tested using sets of pure tones with frequencies that range from 130.813 to 1396.91 Hz. The hit rate percentage values are obtained after adding a random noise signal to the original pure tone signal. The results show very good classification results (above 85 % hit rate) for each class when the Signal-to-noise ratio is above 3 decibels, validating the robustness of the network configuration and the training step.

Keywords: SpiNNaker · Spiking neural network · Audio samples classification · Spikes · Neuromorphic auditory sensor · Address-Event Representation

1 Introduction

Neuromorphic engineering is a discipline that studies, designs and implements hardware and software with the aim of mimicking the way in which nervous systems work, focusing its main inspiration on how the brain solves complex problems easily. Nowadays, the neuromorphic community has a set of neuromorphic hardware tools available such as sensors [1, 2], learning circuits [3, 4], neuromorphic information filters and feature extractors [5, 6], robotic and motor controllers [7, 8]. In the field of neuromorphic sensors, diverse neuromorphic cochleae can be found [2, 9, 10]. These sensors are able to decompose the audio in frequency bands, and represent them as streams of short pulses, called spikes, using the Address-Event Representation (AER) [11] to interface with other neuromorphic layers. On the other hand, there are several software tools in the community for spiking neural networks (SNN) simulation, i.e. NENGO [12] and

BRIAN [13]; or jAER [14] for real-time visualization and software processing of AER streams captured from the hardware using specific interfaces [15]. Hardware platforms like the SpiNNaker board [16] allows to develop and implement complex SNN easily using a high-level programming language such as Python and the PyNN [17] library.

This manuscript presents a novel multilayer SNN architecture built in SpiNNaker which has been trained for audio samples classification using a firing rate based algorithm. To test the network behavior and robustness, a 64-channel binaural Neuromorphic Auditory Sensor (NAS) for FPGA [10] has been used together with an USB-AER interface [15] (Fig. 1) and the jAER software, allowing to produce different pure tones with frequencies varying from 130.813 Hz to 1396.91 Hz, record the NAS response storing the information in aedat files through jAER and use these files as input for the SNN that has been implemented in the SpiNNaker board.

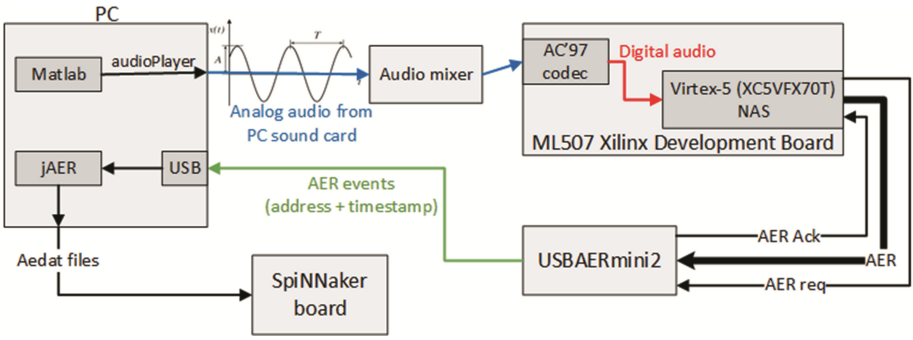

Fig. 1. Block diagram of the system

The paper is structured as follows: Sect. 2 presents the number of neurons, layers and connections of the SNN. Then, Sect. 3 describes the training algorithm used in every layer for the audio samples classification. Section 4 describes the test scenario, including information about the input files. Then, Sect. 5 presents the experimental results of the audio samples classification when using the inputs described in Sect. 4. Finally, Sect. 6 presents the conclusions of this work.

2 Hardware Setup

The standalone hardware used in this work consists of two main parts: the 64-channel NAS connected to the USB-AER interface for generating a spike stream for each audio sample, and the SpiNNaker for back-end computation and deployment of the SNN classifier.

2.1 Neuromorphic Auditory Sensor (NAS)

A Neuromorphic Auditory Sensor (NAS) is used as the input layer of our system. This sensor converts the incoming sound into a train of rate-coded spikes and processes them

using Spike Signal Processing (SSP) techniques for FPGA [5]. NAS is composed of a set of Spike Low-pass Filters (SLPF) implementing a cascade topology, where SLPF's correlative spike outputs are subtracted, performing a bank of equivalent Spikes Band-pass Filters (SBPF), and decomposing input audio spikes into spectral activity [10]. Finally, SBPF spikes are collected using an AER monitor, codifying each spike using the Address-Event Representation, and propagating AER events through a 16-bit parallel asynchronous AER port [11].

NAS designing is very flexible and fully customizable, allowing neuromorphic engineers to build application-specific NASs, with diverse features and number of channels. In this case, we have used a 64-channel binaural NAS, with a frequency response between 20 Hz and 22 kHz, and a dynamic range of +75 dB, synthesized for a Virtex-5 FPGA. Figure 1 shows a NAS implemented in a Xilinx development board, and a USB-AER mini2 board, that implements a bridge between AER systems and jAER in a PC (Fig. 2).

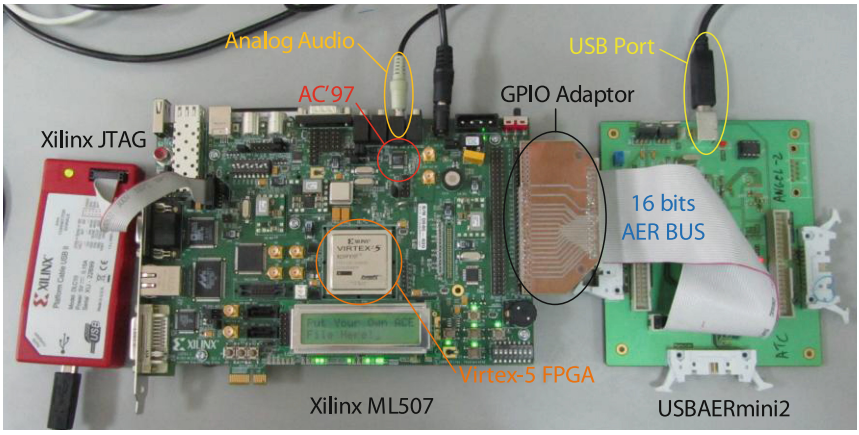

Fig. 2. 64-channel binaural NAS implemented in a Xilinx ML507 FPGA connected to an USB-AER mini2 board.

2.2 Spiking Neural Network Architecture (SpiNNaker)

SpiNNaker is a massively-parallel multi-core computing system designed for modelling very large spiking neural networks in real time. Each SpiNNaker chip comprises 18 general-purpose ARM968 cores, running at 200 MHz, communicating via packets carried by a custom interconnect fabric. Packets are transmitted and their transmission is brokered entirely by hardware, giving the overall engine an extremely high bisection bandwidth. The Advanced Processor Technologies Research Group (APT) [18] in Manchester are responsible for the system architecture and the design of the SpiNNaker chip itself.

In this work, a SpiNNaker 102 machine was used. The 102 machine, Fig. 3, is a 4-node circuit board and hence has 72 ARM processor cores, which are typically deployed as 64 application cores, 4 Monitor Processors and 4 spare cores. The 102 machine

requires a 5 V 1 A supply, and can be powered from some USB2 ports. The control and I/O interface is a single 100 Mbps Ethernet connection.

Fig. 3. SpiNNaker 102 machine.

3 Leaky Integrate-and-Fire Spiking Neural Network

The SpiNNaker platform allows to implement a specific spiking neuron model and use it in any SNN deployed on the board thanks to the PyNN package. Leaky Integrate-and-Fire (LIF) neurons have been used in a 3-layer SNN architecture for audio samples classification.

- **Input layer.** This layer receives the stream of AER events fired for the audio samples captured as aedat files through jAER. The number of input neurons is equal to the number of channels that the NAS has. As a 64-channel NAS (64 different AER addresses) was used in this work, the input layer consists of 64 LIF neurons.
- **Hidden layer.** The hidden layer has the same number of neurons as the desired number of classes to be classified in the output layer. As an example, this layer should consist of eight LIF neurons if eight different audio samples are expected to be classified.
- **Output layer.** As the previous layer, this also has as many neurons as output classes. The firing output of the neurons in this layer will determine the result of the classification.

Figure 4 shows the SNN architecture. Connections between layers are achieved using the FromListConnector method from PyNN, meaning that the source, destination and weight of the connection are specified manually. Using other connectors from this package will result on having the same weight in all the connections between consecutive layers, instead of a different value for each. In this architecture, each neuron in a layer is connected to every neuron in the next layer, and the weight value is obtained from the training step, which is described in Sect. 4. The threshold voltage of the neurons in the hidden layer is 15 mV, while this voltage is 10 mV in the neurons in the output layer.

Decay rate and refractory period are the same for both layers: 20 mV/ms and 2 ms, respectively.

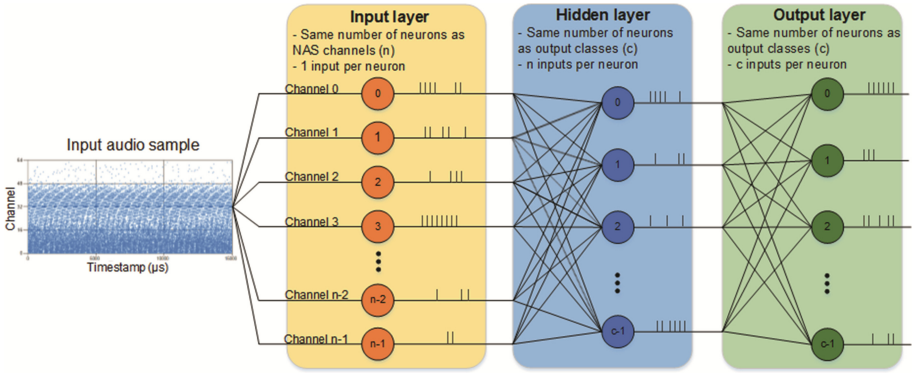

Fig. 4. SNN architecture using an audio sample aedat file as input.

4 Training Phase for Audio Classification

In the previous section, each of the three layers comprising the network were described. The training phase is performed offline and supervised. The main objective of this training is to obtain the weight values of the connections between the input and the hidden layer and between the hidden and the output layers for further audio samples classification. Therefore, two different training steps need to be done.

The weights of the first step of the training phase are obtained from the normalized spike firing activity for each NAS channel using a set of audio samples similar to those to be recognized (same amplitude, duration and frequencies). The firing rate for a specific channel ($FR_{channel\_i}$) is obtained by dividing the number of events produced in that channel by the NAS firing rate (FR_T), which is the number of events fired in the NAS in a particular time period.

$$FR_T = \left(\sum AERevents\right)/T_{sample} \tag{1}$$

$$FR_{channel\_i} = \left(\sum AERevents(i)\right)/FR_T \tag{2}$$

Figure 5 shows the normalized spike firing activity for a set of eight pure tones with frequencies that range from 130.813 Hz to 1396.91 Hz, logarithmically spaced.

The weights of the second step of the training phase are obtained from the firing output of each neuron in the hidden layer when using the set of audio samples as input after loading the weights calculated in the previous step into the connections between the input and the hidden layer. These firing outputs are normalized by dividing each of them by the maximum value. The results obtained are the weight values that will be used in the connections between the hidden and the output layer.

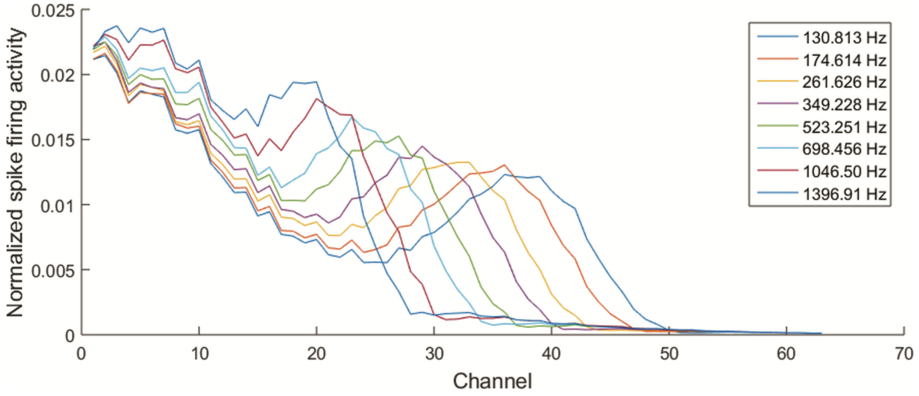

Fig. 5. Normalized spike firing activity for each NAS channel per audio sample.

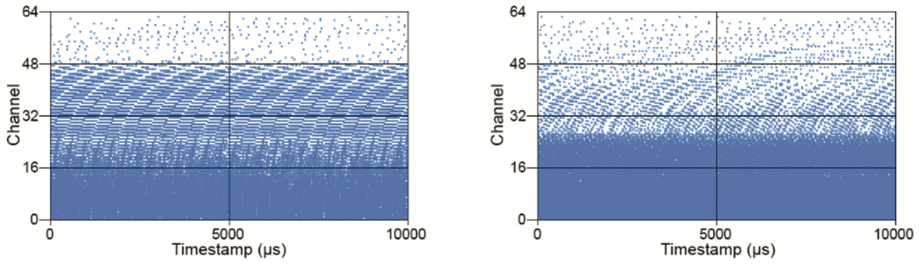

Fig. 6. First 10 ms cochleogram of the 130.813 Hz (left) and 1396.91 Hz (right) pure tones.

5 Test Scenario

In this work, the SNN architecture and training algorithm presented are tested using eight different audio samples. These output classes correspond to eight different pure tones with frequencies that range from 130.813 Hz to 1396.91 Hz, logarithmically spaced (130.813, 174.614, 261.626, 349.228, 523.251, 698.456, 1046.50 and 1396.91 Hz). These samples have a duration of 0.5 s and were generated using the audioplayer function from Matlab with a sampling rate of 48 KHz and a peak-to-peak voltage value of 1 V. After the signal is sent to the mixer, it propagates the sound to NAS input and sends an AER stream to the PC through the AER-USB interface. The jAER software running on the PC is able to capture this stream and save it as an aedat file. Figure 6 shows the cochleograms for the 130.813 Hz and the 1396.91 Hz pure tones after capturing them.

The first step of the training phase can be achieved by applying the equations presented in Sect. 4 to the set of eight aedat files corresponding to each pure tone. This will generate a CSV file containing the weights for the 64×8 connections between the input and the hidden layers of the SNN based on the firing rate of the spike streams for each audio sample. As described in the previous section, loading those weights into the corresponding connections and using the eight pure tones as input will result on a firing

output on the second layer neurons that will be used for training the connections between the second and the output layers of the SNN.

After the weights are set on these connections, new sets of the same pure tones (same frequencies) are recorded using different Signal-to-Noise Ratio (SNR) values and tested on the network, calculating the hit rate percentage for each class.

6 Experimental Results

Different pure tone sets with the same frequencies and properties (0.5 s and 0.5 V amplitude) as the ones used in this work were captured and used to test the network robustness and effectiveness. A 100 % hit rate was obtained for every class when the signal was a pure sine wave. Moreover, the network has also been tested by adding a noise signal consisting of random values to the pure tones original signals, obtaining audio samples with different SNR values (from 35.2 dB to 0 dB). The hit rate percentage for every class using the previous SNR values are listed in Table 1.

Table 1. Hit rate percentage of the audio samples classification SNN for different SNR values.

SNR (dB)	Pure tone frequency (Hz)							
	130.813	174.614	261.626	349.228	523.251	698.456	1046.5	1396.91
No noise	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
35.1993	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
21.3363	100 %	83 %	96 %	100 %	100 %	100 %	100 %	100 %
13.2273	100 %	81 %	92 %	100 %	100 %	100 %	100 %	96 %
7.4733	100 %	86 %	100 %	100 %	100 %	100 %	100 %	95 %
3.0103	74 %	90 %	100 %	98 %	100 %	100 %	100 %	98 %
2	93 %	88 %	20 %	32 %	16 %	92 %	32 %	97 %
1	10 %	5 %	0 %	0 %	0 %	88 %	26 %	94 %
0	0 %	0 %	0 %	0 %	0 %	76 %	22 %	91 %

The results show very high hit rate percentages when the SNR is above 3 dB. However, when the SNR falls below 3 dB and approaches zero dB (the amplitude of the pure tone is the same as the amplitude of the noise signal) the network is not able to classify every input signal as its corresponding class.

7 Conclusions

In this paper, a novel multilayer spiking neural network architecture for audio samples classification implemented in SpiNNaker has been presented. To achieve this goal, an optimized training phase for audio recognition has been described and specified in two different steps, which allow obtaining the weights for the connections between the input and the hidden layers and between the hidden and the output layers. The network was trained using eight pure tones with frequencies between 130.813 Hz and 1396.91 Hz and tested by adding a noise signal with SNR values between 35.1993 and 0 dB.

The hit rate values obtained after many tests confirm the robustness of the network and the training, which make it possible to classify every pure tone with a probability over 74 % even when the SNR value is 3 dB, obtaining almost a 100 % probability for every input when the SNR is above that value.

Finally, the SpiNNaker board has allowed to model and develop a leaky integrate-and-fire spiking neural network for this purpose in an easy, fast, user-friendly and efficient way, proving its potential, and promoting and facilitating the implementation of SNNs like these in real hardware platforms. The PyNN code used to test the SNN presented in this work is available at [19].

Acknowledgements. The authors would like to thank the APT Research Group of the University of Manchester for instructing us in the SpiNNaker. This work is supported by the Spanish government grant BIOSENSE (TEC2012-37868-C04-02) and by the excellence project from Andalusian Council MINERVA (P12-TIC-1300), both with support from the European Regional Development Fund.

References

1. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. IEEE J. Solid-State Circ. **43**, 566–576 (2008)
2. Chan, V., Liu, S.C., van Schaik, A.: AER EAR: a matched silicon cochlea pair with address event representation interface. IEEE Trans Circ. Syst. I **54**(1), 48–59 (2007)
3. Häfliger, P.: Adaptive WTA with an analog VLSI neuromorphic learning chip. IEEE Trans. Neural Netw. **18**, 551–572 (2007)
4. Indiveri, G., Chicca, E., Douglas, R.: A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. IEEE Trans. Neural Netw. **17**, 211–221 (2006)
5. Jiménez-Fernández, A., Jiménez-Moreno, G., Linares-Barranco, A., et al.: Building blocks for spikes signal processing. In: International Joint Conference on Neural Networks, IJCNN (2010)
6. Linares-Barranco, A., et al.: A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors. In: Proceedings of IEEE International Symposium on Circuits and Systems, pp. 2417–2420 (2015)
7. Linares-Barranco, A., Gomez-Rodriguez, F., Jimenez-Fernandez, A., et al.: Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. In: IEEE International Symposium on Circuits and Systems, pp. 1192–1195 (2007)
8. Jimenez-Fernandez, A., Jimenez-Moreno, G., Linares-Barranco, A., et al.: A neuro-inspired spike-based PID motor controller for multi-motor robots with low cost FPGAs. Sensors **12**, 3831–3856 (2012)
9. Hamilton, T.J., Jin, C., van Schaik, A., Tapson, J.: An active 2-D silicon cochlea. IEEE Trans. Biomed. Circ. Syst. **2**, 30–43 (2008)
10. Jimenez-Fernandez, A., Cerezuela-Escudero, E., Miro-Amarante, L., et al.: A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach. IEEE Trans. Neural Networks Learn. Syst. **1**(0) (2016)
11. Boahen, K.: Point-to-point connectivity between neuromorphic chips using address events. IEEE Trans. Circ. Syst II Analog Digit Sig. Process. **47**, 416–434 (2000)

12. Bekolay, T., et al.: Nengo: a Python tool for building large-scale functional brain models. Front Neuroinform. **7**, 48 (2014)
13. Goodman, D., Brette, R.: Brian: a simulator for spiking neural networks in python. Front Neuroinform. **2**, 5 (2008)
14. jAER Open Source Project. http://jaer.wiki.sourceforge.net
15. Berner, R., Delbruck, T., Civit-Balcells, A., Linares-Barranco, A.: A 5 Meps $100 USB2.0 address-event monitor-sequencer interface. IEEE International Symposium on Circuits and Systems (2007)
16. Painkras, E., et al.: SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. IEEE J. Solid-State Circ. **48**, 1943–1953 (2013)
17. Davison, A.P.: PyNN: a common interface for neuronal network simulators. Front Neuroinform. **2**, 11 (2008)
18. SpiNNaker Home Page. http://apt.cs.manchester.ac.uk/projects/SpiNNaker
19. Dominguez-Morales, J.P.: Multilayer spiking neural network for audio samples classification using Spinnaker Github page. https://github.com/jpdominguez/Multilayer-SNN-for-audio-samples-classification-using-SpiNNaker