Cognitive Computation
https://doi.org/10.1007/s12559-020-09798-2

Hardware-Optimized Reservoir Computing System for Edge Intelligence Applications

Alejandro Morán[1] · Vincent Canals[1,2] · Fabio Galan-Prado[1] · Christian F. Frasser[1] · Dhinakar Radhakrishnan[3] · Saeid Safavi[3] · Josep L. Rosselló[1,2]

Received: 7 March 2020 / Accepted: 30 November 2020

Abstract
Edge artificial intelligence or edge intelligence is an ever-growing research area due to the current popularization of the Internet of Things. Unfortunately, incorporation of artificial intelligence (AI) in smart devices operating at the edge is a challenging task due to the power-hungry characteristics of deep learning implementations, such as convolutional neural networks (CNNs). As a feasible alternative, reservoir computing (RC) has attracted a lot of attention in the field of machine learning due to its promising performance in a wide range of applications. In this work, we propose a simple hardware-optimized circuit design of RC systems presenting high energy-efficiency capacities that fulfill the low power requirements of edge intelligence applications. As a proof of concept, we used the proposed design for the implementation of a low-power audio event detection (AED) application in FPGA. The measurements and simulation results obtained show that the proposed approach may provide significant accuracy with the advantage of presenting ultra-low-power characteristics (the energy efficiency estimated is below the microjoule per inference). These results make the proposed system optimal for edge intelligence applications in which energy efficiency and accuracy are the key issues.

Keywords Artificial intelligence · Artificial neural networks · Neuromorphic circuits · Recurrent neural networks

Introduction

Development of efficient Internet of Things (IoT) systems [1] requires implementation of low-power machine learning (ML) methodologies in a single chip platform. ML applications often need to process large data sizes in a short time which demands significant parallelism and large chip areas (devices that are not yet available on edge devices) A sate of the art solution is to send the captured data to cloud servers, and wait for the server processed response [2]. This solution implies a lot of data transmission, which in turn results in network congestion in addition to server dependence; therefore, there is a growing demand for optimization of edge processing particularly in smart devices and IoT applications. However, research on edge intelligence (EI) is still not mature due to the area and power restrictions of edge nodes that hinder the implementation of traditional deep learning techniques, which consequently implies high computational power. Therefore, there is huge interest in the microelectronic industry to be able to efficiently implement accurate and energy-efficient EI chips.

Artificial neural networks (ANNs) arise as one of the main ML methodologies used to develop AI systems due to their capability of solving typical real-life problems such as image or sound recognition. The most popular ANN is the convolutional neural network (CNN) that consists in the use of a feed-forward neural network composed of different sequentially connected convolution and information reduction layers (as max-pooling or average-pooling). CNNs present state of the art performance when applied to image or sound recognition problems, at the price of the need to implement a large amount of multiply-and-accumulate (MAC) operations. This can result in severe

This article belongs to the Topical Collection: *Trends in Reservoir Computing*
Guest Editors: Claudio Gallicchio, Alessio Micheli, Simone Scardapane, Miguel C. Soriano

✉ Josep L. Rosselló
j.rossello@uib.es

[1] Universitat de les Illes Balears, Palma de Mallorca, 07122, Spain

[2] Balearic Islands Health Research Institute, Palma de Mallorca, 07010, Spain

[3] Endura Technologies, Greater San Diego Area, CA, USA

drawbacks in terms of latency, power, and energy consumption when considering CNNs as a feasible methodology for low-power EI applications. In this context, there are recent advances in low-power voice activity detection and keyword spotting implementations using relatively small CNNs [3–5], which incorporate spectrogram-based feature extraction methods similar to the one considered in this work.

Reservoir computing (RC) [6–13] is an attractive ANN training framework with relative computational simplicity due to its simple learning process and the use of fixed weights inside the ANN structure independent of the training process. RC systems are usually composed of a random recurrent neural network (RNN), which is referred to as the reservoir, and a set of inputs that are randomly connected to it. All internal and input connections to the reservoir are kept fixed, while the training process is normally carried out using ordinary least squares (OLS) over the states of the reservoir.

RC systems can be optimized to be implemented in hardware [14] by using a specific ring topology [15] so that each neuron is characterized to have a low fan-in that simplifies its hardware implementation. Along with this ring topology, the reservoir connectivity may also be optimized by selecting specific weights so that only simple shift-and-add operations are performed at each neuron of the reservoir instead of computationally expensive MAC operations at each neural connection. In this context, there are previous works regarding FPGA implementations focusing on the so-called single-node reservoir based on only one physical node [10] which can represent a ring topology by time division multiplexing with an input mask [16, 17] and specific nonlinearities with feasible electronic and optical implementations. Moreover, hardware implementations of RC have been previously applied to spoken digit recognition [17, 18]. In this context, our work differs from previous publications on FPGA implementations in two main aspects: the training method and the digital implementation. The first difference is that training is performed on a per-frame basis using log-mel energies as the input features. Also, the reservoir states are not stored in RAM since the implementation is register-based and fully parallel with a very simple nonlinearity at each node. This optimized RC model has demonstrated to provide good accuracy and energy efficiency characteristics when applied to time-series forecasting or equalization problems [14, 19, 20].

In this work, a feasible methodology for ultra-low-power AI applications is provided by using a hardware-optimized reservoir computing system. The circuitry is applied for an audio event detection problem, showing that it is a feasible alternative that may fit with the low power demands required for edge intelligence applications. Audio tagging or classification is potentially useful for filtering environmental data through multiple smart sensors. A clear example could be detecting danger if the system detects gun shots, people screaming, etc. For this reason, the system is evaluated using first a 2-class classification to determine whether there is a gun shot or not in a 4-s audio slice. Then a more complex 10-class task containing background and foreground audio is used to proof the proposed classifier: the Urban Sound 8K dataset [21].

The Optimized Hardware Reservoir

Reservoir Computing Principles

A reservoir computing (RC) system maps input data to a higher dimensional space, so that as the reservoir size increases, it is more likely to discriminate the input data. In RC systems, only the output layer, which is connected to the reservoir, is trained using OLS or cross-entropy loss minimization [22], while the rest of the reservoir connectivity remains fixed (see Fig. 1). The internal connectivity of the reservoir is normally sparse (typically 1% connectivity) [7], and nodes may be implemented using traditional artificial neural Networks (echo state networks) [23], spiking neurons (liquid state machines) [24–26], or cellular automata (ReCA systems) [27].

The reservoir implemented in this work is created using classical neurons, implementing a piece-wise linear function to a basic linear operation to its inputs. The output of each neuron at a given time is denoted as $x_i(t)$, where $i \in \{1, \ldots, N\}$ is the neuron index and N is the total number of neurons. External inputs are denoted as $u_j(t)$ for $j \in \{1, \ldots, M\}$ where M is the total number of instantaneous inputs. The connectivity matrix $\mathbf{R}$ is defined with dimensions $N \times (M + N)$, where the first M columns correspond to the external input weights, v_{ij}, whereas its last N columns correspond to the internal weights, r_{ij}.

$$\mathbf{R} = \begin{pmatrix} v_{11} & \cdots & v_{1M} & r_{11} & \cdots & r_{1N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NM} & r_{N1} & \cdots & r_{NN} \end{pmatrix} \tag{1}$$

We define the vector $\mathbf{z}(t)$ as the combination of the reservoir state vector $\mathbf{x}(t)$ and the external input vector $\mathbf{u}(t)$. Vector $\mathbf{x}(t)$ is composed of the N outputs evaluated at the discrete time step t at every neuron of the reservoir. Thus, vector $\mathbf{z}(t)$ has $M + N$ components. Its first M rows correspond to those of vector $\mathbf{u}$ and the remaining N rows to internal states $\mathbf{x}(t)$, so that $\mathbf{z}(t) = (u_1(t), u_2(t), \ldots, u_M(t), x_1(t), \ldots, x_N(t))^T$.

Reservoir dynamics is obtained by iterating the following expression:

$$\mathbf{x}(t+1) = f\left(\mathbf{R} \cdot \mathbf{z}(t)\right) \tag{2}$$

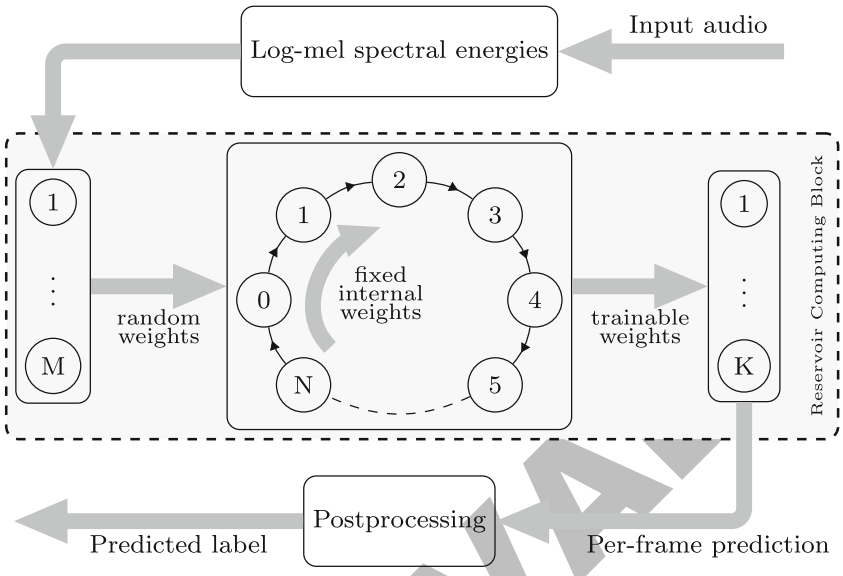

Fig. 1 Ring topology RC applied to log-mel spectral energies. The input audio is first pre-processed to obtain the log-mel features, so that linear combinations of these features are inputs to the reservoir, which in turn computes a higher dimensional nonlinear mapping in the time domain. Finally, the reservoir states are linearly combined to obtain a meaningful readout, which is post-processed to improve performance

where f is a nonlinear activation function such as the classical $\tanh(x)$ activation function or other forms of non-linearity. For low-cost hardware implementation, one may choose a piece-wise linear function activation, such as the simplification described in (3).

$$f(x) = \max(-1, \min(x, 1)) \tag{3}$$

The output of the RC system $\mathbf{y}(t) = (y_1(t), \ldots, y_K(t))^T$ is obtained as follows:

$$\mathbf{y}(t) = \mathbf{W} \cdot \mathbf{z}(t) + \mathbf{b} \tag{4}$$

where the bias $\mathbf{b}$ is a K-dimensional column vector and $\mathbf{W}$ is a $K \times (M+N)$-dimensional weight matrix. In this work both $\mathbf{W}$ and $\mathbf{b}$ are obtained using the OLS optimization method.

The Cyclic Reservoir

A ring topology or single cycle reservoir (SCR) [15] has been selected for a multidimensional audio event classification task. In particular, each audio signal is pre-processed in hardware by a feature extraction digital block, providing 64 ($M = 64$) 8-bit log-mel spectral features per frame used as inputs to the reservoir. For a compact hardware implementation, the chosen reservoir architecture is cyclic (see Fig. 1), where each neuron has two inputs: the external signal coming from one frequency channel and the output signal from the previous neuron in the ring. The connection weights of the neurons are fixed to either r for the internal inputs and to either $+v$, $-v$, or 0 (not connected) for the input-to-reservoir connections. To underline the random nature of external weights (which can be either positive, negative, or zero), the random parameters ξ_{ij} are introduced, which changes the sign of the external inputs. These parameters take the values $+1$, -1, or 0 randomly. The connectivity matrix is therefore given by:

$$\mathbf{R} = \begin{pmatrix} v\xi_{11} & \cdots & v\xi_{1M} & 0 & \cdots & \cdots & 0 & r \\ v\xi_{21} & \cdots & v\xi_{2M} & r & 0 & \cdots & \cdots & 0 \\ v\xi_{31} & \cdots & v\xi_{3M} & 0 & r & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ v\xi_{N1} & \cdots & v\xi_{NM} & 0 & \cdots & 0 & r & 0 \end{pmatrix} \tag{5}$$

where $\xi_{ij} \in \{-1, 0, +1\}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$.

Training on a Per-Frame Basis

Since the readout layer of the reservoir provides a real-time response, i.e., the output may change every time step, the supervised training method includes real-time labeling. Every single log-mel frame is therefore labeled with its corresponding value as depicted in Fig. 2. The figure illustrates the correspondence in time of a per-frame labeled sound. Although the label refers to the actual sound, it is assigned on every log-mel frame.

In order to assign a unique label to a given evaluation time period T_{eval}, a post-processing strategy is applied to the per-frame prediction, as indicated in Fig. 1. Although the OLS training method is performed on the per-frame labels, the model is validated including the next post-processing stage.

As is expressed in (4), the outputs $\mathbf{y}(t)$ are computed as a linear combination of the state vector $\mathbf{z}(t)$ that includes both inputs and neuron states. Each $y_j(t)$ time-varying output is bounded in the interval $[-1, +1]$ and is representing the suitability of the j^{th} category to be the correct one. For the estimation of the most likely category, we consider a certain time interval T_{eval}. The highest category that is over a given

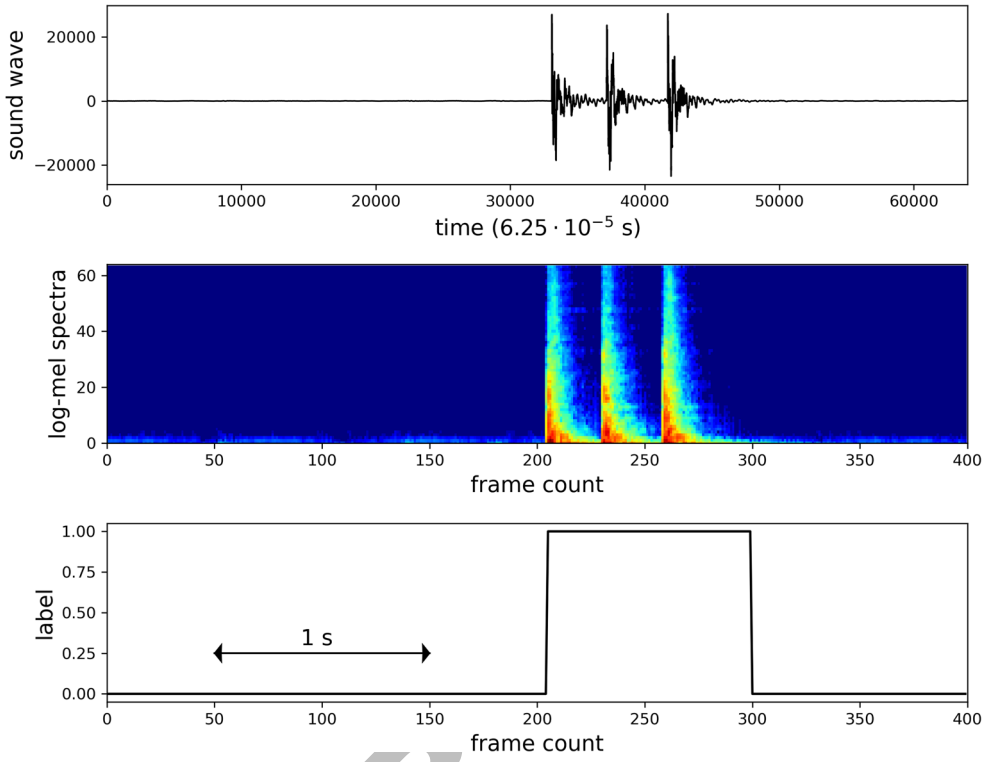

Fig. 2 Audio, features, and corresponding label assigned on a per-frame basis

threshold value (y_{th}) at each time step is added up to a generic score for each category so that:

$$S_j = \sum_{t_i=0}^{T_{eval}} Y_j(t_i) \tag{6}$$

where Boolean $Y_j(t) = 1$ in case $y_j(t_i) > y_{th}$ and $y_j(t_i) > y_k(t_i)$, $\forall\, j \neq k$, otherwise $Y_j(t_i) = 0$. Finally, for the estimation of the most likely category, a specific reduction factor D_j is defined for each one, so that the predicted label L is given by expression (7), where parameters D_j are selected for the best fit of the training database.

$$L = \arg\max_j \left\{ \frac{S_j}{D_j} \right\} \tag{7}$$

The Optimized Reservoir Architecture

Each neuron in the reservoir is implemented using digital blocks as shown in Fig. 3, where a non-linearity function is implemented, representing the activation function of expression (3). A linear transformation is performed to both inputs u_j and x_{i-1}, avoiding the use of multipliers due to the higher cost in terms of hardware resources with respect other functions as the addition or max function (see Table 1).

For the estimation of network configuration parameters (r and v), we adjust the best values to optimize both hardware and accuracy. In particular, we set $v = 1$ and $r = 1 - \frac{1}{2^4}$. Therefore, the $r \cdot x_{i-1}$ product is done using simple shift-and-add operations, with a lower cost in terms of logic gates (see Fig. 3). The post-processing (estimation of Y_j and S_j) could be performed using additional digital hardware or by a low-end (soft) microprocessor with a low cost if the number of categories is low compared to the reservoir size N. The main advantage of using a minimal soft microprocessor (e.g., Nios II [28] or MicroBlaze [29]) is versatility when compared to custom design-specific hardware, since it can be implemented in FPGA to control other hardware resources or perform small computations at the price of using additional resources and being less energy efficient due to e.g. instruction fetch and decode. In the case of a large reservoir, computationally intensive tasks would be done by the reservoir hardware, so that the energy and area devoted to the soft processor would be negligible. However, in the current implementation, both preprocessing and post-processing stages are performed offline. The implemented subsystem is the Reservoir Computing Block highlighted in Fig. 1. In addition, Table 2 reports the FPGA resources utilized to implement the ring-topology reservoir, which grow almost linearly with the number of nodes as expected.

Audio Event Detection Database

Gun-Shot Detection Database

A database composed of ambient and gunshots has been analyzed. Three different categories are defined ("silence",

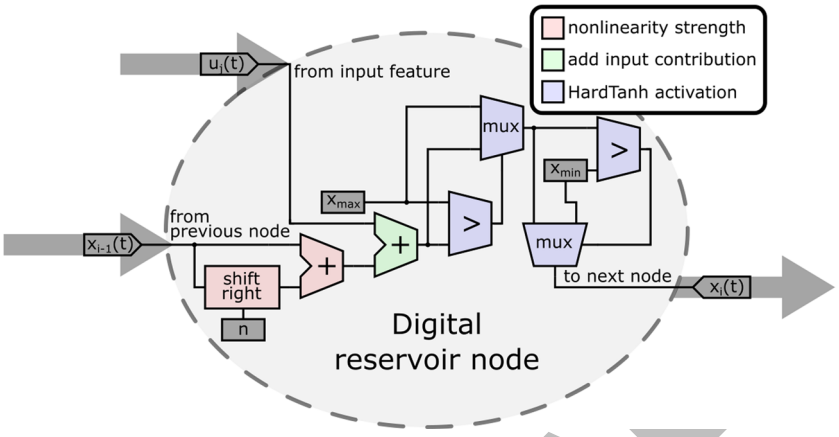

Fig. 3 Digital node hardware architecture for the ring topology reservoir

Table 1 Number of 8-bit hardware lookup tables required to perform an 8-bit and 16-bit binary operation

	Adder	Multiplier	Maximum
8-bit	8 LUTs	91 LUTs	15 LUTs
16-bit	16 LUTs	171 LUTs	26 LUTs
Ratio	1	11	1.7

Data obtained from Quartus Prime software for a Cyclone V FPGA device

Table 2 FPGA resource utilization for the 64-input reservoir subsystem

Nodes	ALMs	ALUTs	Registers
64	757.7	1769	512
192	2393.9	5041	1536
768	9368.8	18,945	6147
2048	26,555.9	111,322	16,391
4096	51,566.9	162,072	32,783
8192	101,588.7	263,571	65,566
12,288	151,610.5	365,070	98,349

No on-chip DSP or memory blocks were implemented to compile these designs. Data obtained from Quartus Prime software for a Cyclone V FPGA device. Note that hardware resources are given in both ALMs and their equivalent 4-bit ALUTs + Registers

Table 3 Gun-shot dataset, a total of 727 files are used combining both "ambient" and "gun-shot" situations

	Ambient	Gun-shot
Train	400	111
Test	113	103
Total	513	214

"ambient", and "gun-shot"). Both training and testing sets are described in Table 3 with a total of 727 different files between ambient and gunshots.

Multiclass Audio Database

The Urban Sound 8K database [21], containing ten different audio categories of urban sounds ("air conditioner," "car horn," "children playing," "dog bark," "drilling," "engine idling," "gun shot," "jackhammer," "siren," "street music"), is analyzed. It contains 8732 labeled audio files with a duration that is less than or equal to 4 s contaning both background and foreground audio samples. These files are preprocessed to obtain their corresponding log-mel spectrograms, which are used for training and testing following the 10-fold cross-validation experimental setup explained in [21]. Therefore, the results presented in this section are averaged over the 10 possible train or test set combinations.

Statistical Performance Metrics

The statistical performance metrics used for the gun-shot detection database are the next:

$$Overall\ Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

Table 4 List of relevant log-mel spectral feature extraction parameters

Parameter	Value
Audio sampling rate	16 kHz
Window size	512
Window overlap	352
Mel bins	64
Output bit width	8

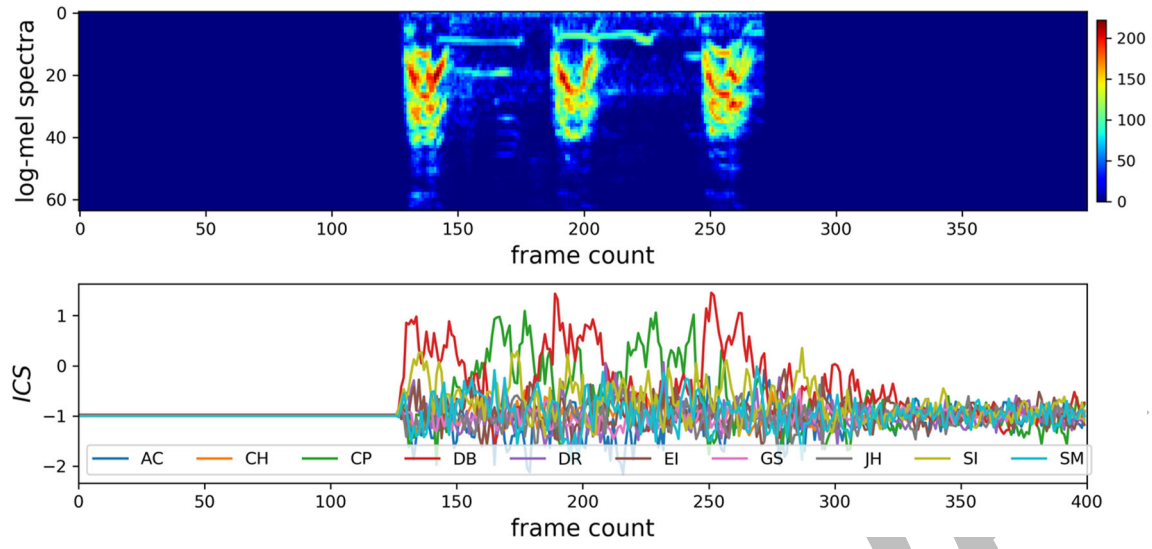

Fig. 4 Example evolution of the reservoir readout signals for an audio file with 3 foreground dog barks and background people speaking. The input log-mel energies (top) induce a behavior in the reservoir obtained from FPGA measurements (bottom) denoted as the Immediate Category Score (ICS) for each available class

$$True\ Negative\ Rate = \frac{TN}{TN + FP} \tag{8}$$

where the *Accuracy* is referred to the overall accuracy of the method (successes/total points), parameters *True Positive Rate* and *True Negative Rate* (also known as sensitivity and specificity, respectively) are related to how well the positives (and negatives) are differentiated respectively. Finally, TP, TN, FP, and FN in (8) are referring to true positives, true negatives, false positives, and false negatives respectively. These three results are estimated for different post-processing configurations in the "Results" Section.

In contrast, in the case of the Urban Sound database, we evaluate the mean top 1 and top 2 overall accuracy.

Results

This section reports inference performance metrics obtained from the proposed RC implementation applied to the input data described in "Audio Event Detection Database." In order to evaluate the model performance for the audio events, log-mel spectral features are obtained from a fixed point hardware simulation, relevant parameters are listed in Table 4 (see e.g. [30] for hardware implementation). This feature extraction block provides 64 8-bit features. The RC subsystem is implemented in an Intel Cyclone V FPGA and the post-processing stage is performed offline, i.e., the computation of (6) and (7) is done externally.

In all cases, input-to-reservoir binary weights ξ_{ij} correspond to the best known model sampled from a random uniform distribution. In Fig. 4, we show the reservoir evolution for a specific sound. The feature extraction block (top graph) along with the readout behavior extracted from FPGA measurements induced by an input log-mel spectrogram. There are two simultaneous sources: foreground dog barking and background people speaking. Although the dog bark (DB) class is correctly predicted, the background people speaking is being classified as children playing (CP), which is reasonable since it is the most similar class.

The results associated to the gun-shot detection database are shown in Tables 5 and 6. Firstly, the confusion matrix of a 192-node reservoir is shown in Table 5 while in Table 6 we illustrate the effect of varying the threshold y_{th} in the same reservoir model.

Table 5 Confusion matrix for an 8-bit 192-neuron reservoir

Prediction \ Reference	Gun-shot	Ambient
Gun-shot	85	0
Ambient	18	113

Post-processing parameters: $y_{th} = 1.11$, $D(silence, ambient, gunshot) = (64, 64, 1)$

Table 6 Performance metric values (in %) for an 8-bit 192-neuron reservoir with different post-processing parameters

	$y_{th} = 1.11$	$y_{th} = 1.0$	$y_{th} = 0.9$
Accuracy	91.6	96.3	94.9
Sensitivity	82.5	94.2	94.2
Specificity	100	98.2	95.6

The selected category factors are $\mathbf{D}(silence, ambient, gunshot) = (64, 64, 1)$

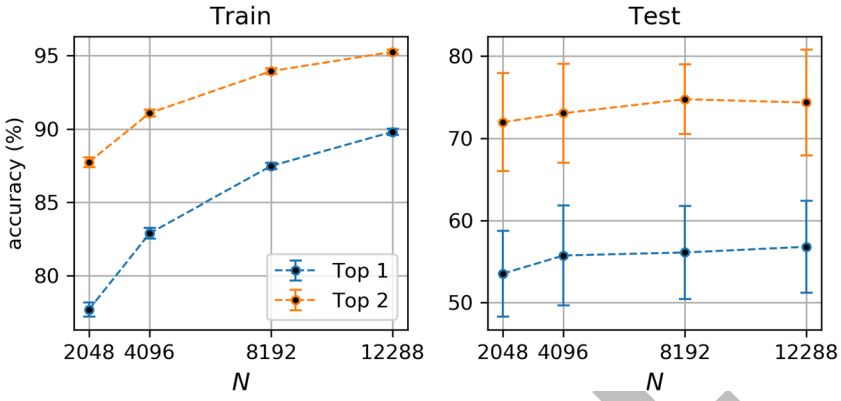

Fig. 5 Urban Sound train (left) and test (right) set mean metrics obtained from 10-fold cross-validation. Reduction factors have been selected so that they are proportional to the training set occurrences for each class (see text)

For the case of the multiclass Urban Sound database, mean top 1 and top 2 accuracies[1] averaged over the 10 different test set choices are shown in Fig. 5. These results have been obtained by randomly selecting 250,000 training frames and frame labels. Then, each reduction factor has been selected so that it is proportional to the number of occurrences of the corresponding class in the training set. For example, if class A has a training samples, then its reduction factor is $a/250,000$. The top 1 accuracy results shown in Fig. 5 imply a similar or higher accuracy than decision trees (J48) and k-NN ($k = 5$) methods applied to the same problem [21] and using the same slice duration of 4 s.

In Fig. 6, we analyze both the size and energy efficiency of the FPGA implementation. The number of parameters and MAC operations per inference needed is relatively low when compared to a CNN hardware implementation oriented to audio processing [31]. While the order of magnitude of MAC operations per inference is over 1000 million for the case of CNNs [31], the reservoir implementation needs between 10M and 50M MACs per inference (dashed blue line in Fig. 6). As can be appreciated, there is a trade-off between the reservoir size and energy efficiency with respect to the overall accuracy which is represented in Fig. 6 (in data points text and dotted orange lines). Nevertheless, there is a decrease of about 15% accuracy in the 12,288-node reservoir model when compared to different CNNs [32] without data augmentation.

With respect to the energy efficiency, there exists a relevant gap with respect to recently published on-chip implementations of sound recognition systems as the work in [33] related to a real-time bird sound recognition application. In their application, all computations are done by a Texas Instruments TMS320F2812 microcontroller and the classifier stage consumes 35 µ J per inference at 50-MHz clock frequency, which is about 40× less energy efficient than the proposed solution implemented in a Cyclone V FPGA for the case $N = 12,288$ depicted in Fig. 6.

Conclusions

This work is motivated by the increasing demand in edge intelligence devices, which avoid unnecessary data transfers to the cloud because of their ability to select the data/metadata that is relevant or worth capturing for post-processing in real time. In this regard, the proposed reservoir computing system (Fig. 1) is much smaller and requires much less MAC operations and parameters as compared to state of the art network architectures [32] (see Fig. 6), which is ultimately related to latency, power dissipation, and energy consumption. It is also shown that the proposed sound recognition hardware classifier is up to 40× more energy-efficient than a recently published sound recognition

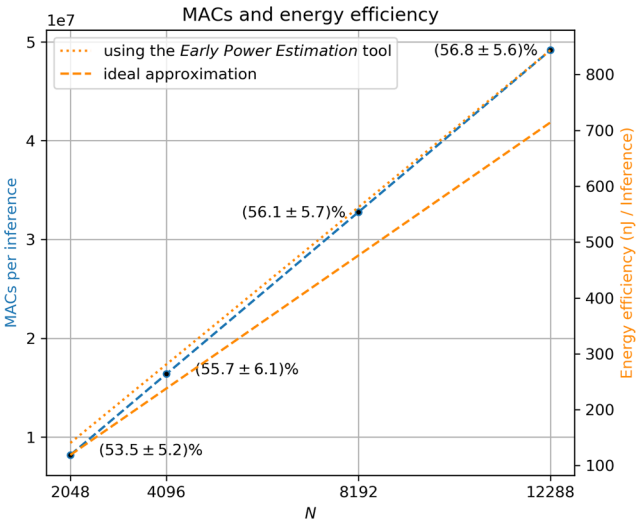

Fig. 6 Number of needed MACs per inference (left axis) and reservoir energy efficiency (right axis) with corresponding test accuracy for each data point (inner text) for different reservoir sizes (bottom axis). Ideal approximation for the energy efficiency corresponds to the energy consumed by n 64-node reservoirs, with $n = N/64$

[1]Top-N accuracy is computed by interpreting as correct those predictions for which the ground truth is one of the N most likely categories.

solution based on the use of a low-cost and low-power ARM Cortex-M4F microcontroller [33]. So that an attractive possibility would be to include RC dedicated hardware to similar system-on-chip architectures to potentially reduce energy consumption in certain tasks. Note that here RC has been applied to audio event detection but it is not limited to audio recognition. Similar systems have been applied to time series prediction [15] or image classification [34]. At the same time, in terms of error performance, traditional machine learning models as kNN or decision trees show similar or a lower accuracy for the specific multi-class audio event detection system.

Additionally, it is simple enough to be a candidate for a variety of battery-powered edge scenarios, e.g., always-on inference scenarios [5] or RC hardware acceleration or co-processing in system-on-chip architectures [35], which include mobile phones, smartwatches, or smart sensors. Potential use cases might include simple audio tagging or detection, monitoring of physiological data [36], and channel equalization [37]

At the algorithm level, we showed the cyclic reservoir is good enough to be used for temporal feature expansion on a per-frame basis. However, generalization in the Urban Sound 8K dataset is particularly difficult because background and foreground samples are mixed and some of them have noisy environments or additional background sources. Test set accuracy might be improved using e.g. noise reduction, data augmentation techniques [32, 38], or an alternative post-processing technique. To summarize, in this work, we have shown an ultra-low-power audio event detection system with an energy efficiency in the sub-$\mu J/Inf$ range that is obtained by optimizing the reservoir structure for building a totally parallelized ANN. These characteristics make the system optimal for edge intelligence applications.

Funding This work has been funded by a research grant from Endura Technologies and by the Ministerio de Ciencia e Innovación (MICINN/FEDER, UE), Spain, under project TEC2017-84877-R.

Compliance with Ethical Standards

Conflict of Interest The authors at the Balearic Islands University received a research grant from Endura Technologies (San Diego, CA, USA).

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Chen D, Cong J, Gurumani S, Wm Hwu, Rupnow K, Zhang Z. Platform choices and design demands for IoT platforms: cost, power, and performance tradeoffs. IET Cyber-Physical Systems: Theory & Applications. 2016;1(1):70–77.
2. O'Leary DE. Artificial intelligence and big data. IEEE Intelligent Systems. 2013;28(2):96–99.
3. Silva DA, Stuchi JA, Violato RPV, Cuozzo LGD. Exploring convolutional neural networks for voice activity detection. In: Cognitive technologies. Springer; 2017. p. 37–47.
4. Zhang Y, Suda N, Lai L, Chandra V. Hello edge: keyword spotting on microcontrollers. arXiv:171107128. 2017.
5. Liu B, Wang Z, Zhu W, Sun Y, Shen Z, Huang L, Li Y, Gong Y, Ge W. An ultra-low power always-on keyword spotting accelerator using quantized convolutional neural network and voltage-domain analog switching network-based approximate computing. IEEE Access. 2019;7:186456–186469.
6. Maass W, Natschläger T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Computation. 2002;14(11):2531–2560.
7. Jaeger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science. 2004;304(5667):78–80.
8. Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Computer Science Review. 2009;3(3):127–149.
9. Maass W. Liquid state machines: motivation, theory, and applications. In: Computability in context: computation and logic in the real world. World Scientific; 2011. p. 275–296.
10. Appeltant L, Soriano MC, Van der Sande G, Danckaert J, Massar S, Dambre J, Schrauwen B, Mirasso CR, Fischer I. Information processing using a single dynamical node as complex system. Nature Communications. 2011;2(1):1–6.
11. Scardapane S, Butcher J, Bianchi F, Malik Z. Advances in biologically inspired reservoir computing. Cognitive Computation. 2017;9(3):295–296.
12. Katumba A, Freiberger M, Bienstman P, Dambre J. A multiple-input strategy to efficient integrated photonic reservoir computing. Cognitive Computation. 2017;9(3):307–314.
13. Gallicchio C, Micheli A. Echo state property of deep reservoir computing networks. Cognitive Computation. 2017;9(3):337–350.
14. Alomar M, Skibinsky-Gitlin ES, Frasser CF, Canals V, Isern E, Roca M, Rosselló JL. Efficient parallel implementation of reservoir computing systems. Neural Comput & Applic: 32 1–15. 2017.
15. Rodan A, Tino P. Minimum complexity echo state network. IEEE Trans Neural Netw. 2010;22(1):131–144.
16. Alomar ML, Soriano MC, Escalona-Morán M, Canals V, Fischer I, Mirasso CR, Rosselló JL. Digital implementation of a single dynamical node reservoir computer. IEEE Transactions on Circuits and Systems II: Express Briefs. 2015;62(10):977–981.
17. Penkovsky B, Larger L, Brunner D. Efficient design of hardware-enabled reservoir computing in fpgas. Journal of Applied Physics. 2018;124(16):162101.
18. Larger L, Soriano MC, Brunner D, Appeltant L, Gutiérrez JM, Pesquera L, Mirasso CR, Fischer I. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Optics Express. 2012;20(3):3241–3249.
19. Skibinsky-Gitlin ES, Alomar ML, Isern E, Roca M, Canals V, Rossello JL. Reservoir computing hardware for time series forecasting. In: 2018 28th international symposium on power and timing modeling, optimization and simulation (PATMOS). IEEE; 2018. p. 133–139.
20. Skibinsky-Gitlin ES, Alomar ML, Canals V, Frasser CF, Isern E, Galán-Prado F, Morán A, Roca M, Rosselló JL. Fpga-based echo-state networks. In: International conference on time series and forecasting. Springer; 2018. p. 135–146.
21. Salamon J, Jacoby C, Bello JP. A dataset and taxonomy for urban sound research. In: Proceedings of the 22nd ACM international conference on multimedia; 2014. p. 1041–1044.

22. Costoya AM, Frasser CF, Roca M, Rossello JL. Energy-efficient pattern recognition hardware with elementary cellular automata. IEEE Transactions on Computers. 2019.
23. Jaeger H. The echo state approach to analysing and training recurrent neural networks. GMD Report. 2001.
24. Maass W, Natschläger T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. 2002;14(11):2531–2560.
25. Matsubara T, Torikai H, Hishiki T. A generalized rotate-and-fire digital spiking neuron model and its on-fpga learning. IEEE Transactions on Circuits and Systems II: Express Briefs. 2011;58(10):677–681.
26. Nouri M, Jalilian M, Hayati M, Abbott D. A digital neuromorphic realization of pair-based and triplet-based spike-timing-dependent synaptic plasticity. IEEE Transactions on Circuits and Systems II: Express Briefs. 2018;65(6):804–808.
27. Yilmaz O. Symbolic computation using cellular automata-based hyperdimensional computing. Neural Comput. 2015;27(12):2661–2692.
28. Corporation A. Nios ii processor reference handbook. 2008.
29. Xilinx I. Microblaze processor reference guide. reference manual 23. 2006.
30. Han W, Chan CF, Choy CS, Pun KP. An efficient mfcc extraction method in speech recognition. In: 2006 IEEE international symposium on circuits and systems. IEEE; 2006. p. 4–pp.
31. Meyer M, Cavigelli L, Thiele L. Efficient convolutional neural network for audio event detection. arXiv:170909888. 2017.
32. Salamon J, Bello JP. Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Processing Letters. 2017;24(3):279–283.
33. Kücüktopcu O, Masazade E, Ünsalan C, Varshney PK. A real-time bird sound recognition system using a low-cost microcontroller. Appl Acoust. 2019;148:194–201.
34. Schaetti N, Salomon M, Couturier R. Echo state networks-based reservoir computing for mnist handwritten digits recognition. In: 2016 IEEE Intl conference on computational science and engineering (CSE) and IEEE intl conference on embedded and ubiquitous computing (EUC) and 15th intl symposium on distributed computing and applications for business engineering (DCABES). IEEE; 2016. p. 484–491.
35. Saleh R, Wilton S, Mirabbasi S, Hu A, Greenstreet M, Lemieux G, Pande PP, Grecu C, Ivanov A. System-on-chip: reuse and integration. Proc IEEE. 2006;94(6):1050–1069.
36. Escalona-Morán MA, Soriano MC, Fischer I, Mirasso CR. Electrocardiogram classification using reservoir computing with logistic regression. IEEE Journal of Biomedical and health Informatics. 2014;19(3):892–898.
37. Patra JC, Pal RN. A functional link artificial neural network for adaptive channel equalization. Signal Process. 1995;43(2):181–195.
38. Inoue T, Vinayavekhin P, Wang S, Wood D, Greco N, Tachibana R. Domestic activities classification based on cnn using shuffling and mixing data augmentation. DCASE 2018 Challenge. 2018.