In-filter Computing For Designing Ultra-light Acoustic Pattern Recognizers

Abhishek Ramdas Nair, *Member, IEEE,* Shantanu Chakrabartty, *Senior Member, IEEE,* and Chetan Singh Thakur, *Senior Member, IEEE*

***Abstract*—We present a novel in-filter computing framework that can be used for designing ultra-light acoustic classifiers for use in smart internet-of-things (IoTs). Unlike a conventional acoustic pattern recognizer, where the feature extraction and classification are designed independently, the proposed architecture integrates the convolution and nonlinear filtering operations directly into the kernels of a Support Vector Machine (SVM). The result of this integration is a template-based SVM whose memory and computational footprint (training and inference) is light enough to be implemented on an FPGA-based IoT platform. While the proposed in-filter computing framework is general enough, in this paper, we demonstrate this concept using a Cascade of Asymmetric Resonator with Inner Hair Cells (CAR-IHC) based acoustic feature extraction algorithm. The complete system has been optimized using time-multiplexing and parallel-pipeline techniques for a Xilinx Spartan 7 series Field Programmable Gate Array (FPGA). We show that the system can achieve robust classification performance on benchmark sound recognition tasks using only ~1.5k Look-Up Tables (LUTs) and ~2.8k Flip-Flops (FFs), a significant improvement over other approaches.**

***Index Terms*—SVM, Neuromorphic, IoT, FPGA, Cochlea, Edge Computing.**

I. Introduction

Internet-of-Things like unattended ground sensors [1] (UGS), intruder detection systems [2] [3], wild-life tracking [4] or, structural health monitoring systems [5] generally operate in remote locations. They have to be active at all times to ensure that it can detect events of interest. In most cases, the events of interest are infrequent or rare. As a result, most of these IoT systems use an embedded pattern classifier to relax data storage and wireless transmission requirements [6]. An example of such a system is illustrated in Fig. 1. Here the system wirelessly transmits alerts only when it detects signatures (acoustic or visual) about a target (for example a wild life species).

Based on this selective transmission, the IoT platform can conserve a significant battery power and hence prolong its operational life. However, the key challenge in designing such IoTs is that the integrated classifier needs to be robust and highly energy-efficient. While deep neural network (DNN) based classification systems can achieve very high accuracy [7] [8], there exist certain limitations when applying them to IoTs for rare-event detection. First, by the nature of the problem, the training data corresponding to the rare event is sparse and might not be suitable for DNNs. Even if it were possible to train a DNN for deployment, a compressed or a quantized variant of DNN, like the Binary Neural Networks (BNNs) [9], has to be used to optimize computational resources. Retraining BNNs on the IoT platform to account for data and hardware drifts is challenging due to quantization effects. Full-precision training is not possible due to limited computational resources. Also, if the parameters of the DNNs can be quantized, the input features cannot be significantly quantized without affecting the classification accuracy. K-Nearest Neighbour (KNN) [10] [11] and Support vector machines (SVMs) [12] [13], on the other hand, have been shown to generalize well with sparse training data [14] . However, SVM is more robust to outliers, and the convexity of SVM training ensures any recalibration is interpretable and stable. There have been many instances where SVMs have performed well as an acoustic classifier [15] [16]. In literature, several approaches have been proposed to reduce the computational and memory footprint of SVMs [17] [18] [19] [20]. However, in these platforms, computing features and classification are generally treated independently, both during training and inference.

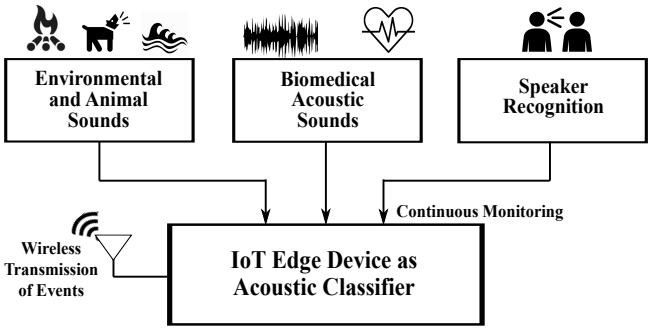

Fig. 1: Smart IoT architecture using a classifier to reduce wireless transmission bandwidth

In this paper, we present an in-filter computing framework that exploits the computing and nonlinear primitives in the feature extraction process to design ultra-light IoT acoustic classifiers. The approach is motivated by the fact that acoustic front-ends like the neuromorphic cochlea [21] can be designed to be highly computationally efficient using different degrees of linear and nonlinear transformations. Our goal is to systematically exploit and map these nonlinear transformations into the kernel functions used in SVMs, such

Abhishek Ramdas Nair and Chetan Singh Thakur are with the Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore, KA, 560012 INDIA e-mail: (abhisheknair@iisc.ac.in, csthakur@iisc.ac.in).

Shantanu Chakrabartty is with Department of Electrical and Systems Engineering, Washington University in St. Louis,USA, 63130 e-mail: (shantanu@wustl.edu).

that both classification and feature extraction are co-optimized for training and inference. This results in a template-based SVM [22] [23] architecture that has an ultra-low computational footprint for inference and training. This feature not only relaxes communication bandwidth requirements on the IoT system but also allows recalibration (retraining) to account for statistical drifts. The main advantage of using template-based SVM is the ability of the framework to use arbitrary functions without any restriction on its properties, like positive-definite kernels for traditional SVM. This allows us to use hardware-friendly mapping or functions that need not be specified in a closed-form, such as using an ordinary differential equation (ODE). This property is beneficial especially for hardware implementation, where the inherent nonlinearity of the device can be used as a kernel rather than engineering a specific nonlinearity. As a proof-of-concept, we have applied the in-filter computing framework using an acoustic feature extractor based on Cascade of Asymmetric Resonators with Inner Hair Cells (CAR-IHC) [21] [24] [25]. The CAR-IHC model exhibits inherent nonlinearity and hence performs well as a kernel for classification. We believe that our proposed framework has the following key advantages:

- A template-based SVM architecture that allows an arbitrary function to be used as a kernel, unlike a conventional SVM that requires a positive-definite kernel.
- Combining the feature extraction and SVM kernel into one function makes the system ultra-light and computationally efficient.
- The memory footprint of the proposed system is user-defined and can be specified based on the IoT hardware constraints.
- A novel fast training algorithm with reduced training complexity in terms of memory and computational complexity.
- A system that can scale without affecting significant hardware changes due to the time-multiplexing approach allows the framework to deploy for more complex tasks.

As proof of concept IoT implementation, we have implemented this inference framework on Xilinx Spartan 7 series FPGA [26], a low-cost and low-power FPGA. We have validated our architecture on various auditory datasets such as the environmental sound dataset [27] and speech-based dataset [28].

The rest of this paper is organized as follows. In Section II, a brief discussion of related work is provided, followed by section III, where we present the modified template-based SVM algorithm and explain the uniqueness of the formulation. In Section IV, we explain the novel training algorithm used for our framework. Section V provides the FPGA implementation details. Section VI provides results obtained with an audio based dataset for detection and surveillance applications. Section VII concludes this paper and provides some useful applications, and discusses possible future work using this framework.

II. Related Work

Hardware implementations of SVM using FPGAs have been successfully achieved over the years with high accuracy and the least possible area and power. Binary classifications or even multi-class classifications using a Modified One-against-all (M-OAA) approach for SVMs have been implemented on FPGAs [29] [30]. Since the kernel is one of the most important parts of the SVM algorithm, the kernel function consumes maximum resources in implementation. This is demonstrated in [17] with linear and nonlinear SVM implementations on FPGA. The authors show that nonlinear kernel implementations use more resources than linear kernels. However, at the same time, there is a drop in accuracy by more than 10% when using a linear kernel compared to a nonlinear kernel. The authors implement a kernel with parallel inputs enabling high operating frequency but at the cost of high resource utilization in terms of LUTs and DSPs. This shows that in order to get good classification, we require a nonlinear kernel, but at the same time, we need to achieve hardware efficiency for an ultra-light implementation.

Regarding acoustic feature extraction, acoustic signals require a certain amount of pre-processing to extract the salient features before it is used for classification. One such FPGA-based approach is detailed in [18]. The authors use (Discrete Wavelet Transforms) DWTs for feature extraction from a given audio signal. This DWT feature extraction forms the input to a standard SVM having a Radial Basis Function (RBF) kernel, which is nonlinear. This classification system is used for phoneme recognition using data from the TIMIT dataset. Due to hardware constraints and the complexity of the DWT algorithm, the authors chose to implement only the SVM classifier on the FPGA. The acoustic signals are pre-processed using a software implementation of the DWT algorithm and are provided as inputs to the SVM hardware. This implementation has the disadvantage of offline software feature extraction, making the hardware incapable of using unprocessed acoustic signals as inputs. At the same time, the SVM hardware implementation consumes a high number of FPGA resources in terms of LUTs and DSPs. Also, the weights and support vectors from the SVM training are stored in external ROMs. This makes the implementation impractical for a small IoT-based edge device.

Furthermore, time-series data need not always be a speech signal, and there may be cases where we may need to classify non-auditory time-series signals. In [19], authors use Mel-frequency cepstral coefficients (MFCC) technique to extract salient features from pulmonary sounds to detect wheezing using standard SVM classification. Here, MFCC, as well as SVM, was implemented on FPGA. This implementation provided an end-to-end solution on hardware that could classify between a normal and an abnormal pulmonary sound. In this implementation, MFCC itself is a resource-heavy algorithm, and additional hardware is required for the SVM classifier to be implemented. Also, ROMs store support vectors and weights along with additional registers to store MFCC coefficients. The MFCC coefficient calculations, which are being done on hardware, also contribute to high DSP usage. The authors have demonstrated their hardware capability using only a 6 kHz input sampling frequency, making the hardware limited in terms of the flexibility of signals that it can process. Hence, such a system cannot be used in an IoT edge device

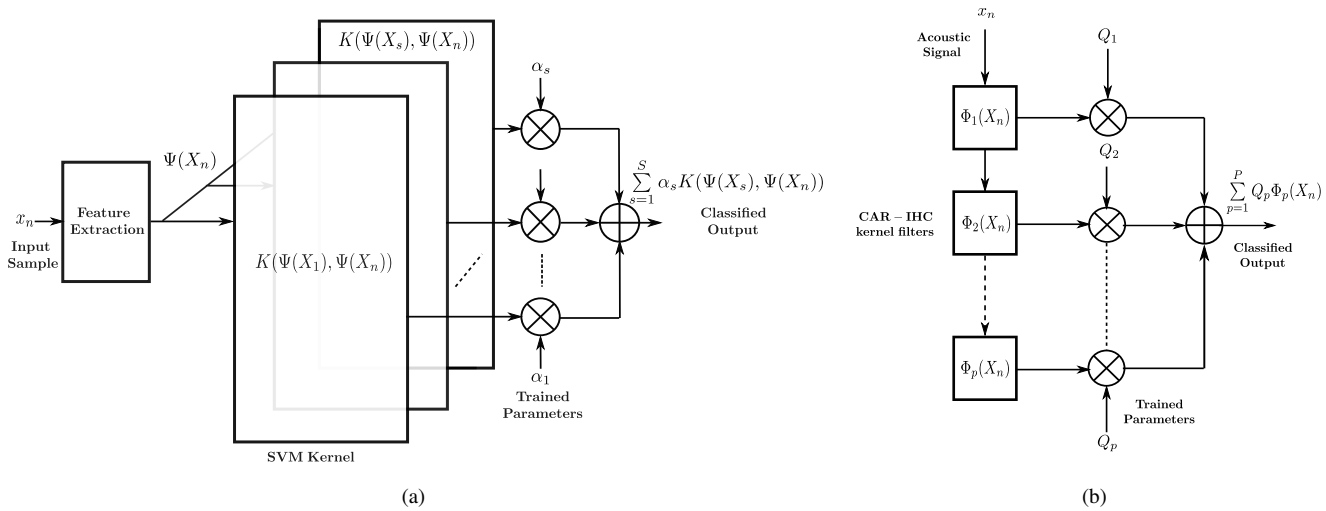

Fig. 2: Architecture of : (a) traditional SVM based acoustic classifier (b) proposed in-filter SVM framework

due to the high resource utilization and rigidity.

Another representative example of an IoT for acoustic classification is a speaker identification system used in security systems. One such system was realized on FPGAs in [20]. Similar to the implementation in [19], the authors implemented an SVM classifier with MFCC as the feature extractor on hardware. The input data was sampled at 8 kHz, making it resource-efficient, but at the same time, it was less flexible in terms of processing signals of higher sampling frequency. External SRAM was used to store the MFCC coefficients and training parameters. Despite having a slight improvement in terms of hardware efficiency compared to the previous implementation, this implementation lacked flexibility and still had a significant amount of resource usage, given the hardware constraints applicable for an IoT device.

Our framework addresses all the shortcomings of prior works by having a neuromorphic cochlea-based CAR-IHC kernel integrated inside a template-based SVM system. This kernel exhibits nonlinearity for better classification and, at the same time, inherently provides a robust feature extracting capability in order to get a good classification. This kernel has multiple tunable parameters which can be adjusted to get the best feature extraction depending on the application. The template-based SVM provides the flexibility of choosing the right number of templates as support vectors, which can be tuned as per the application. This avoids the additional requirement for the storage of support vectors. Flexibility, scalability, low resource usage, and low power make this framework ultra-light and ideal for IoT deployment for many applications.

III. Template-based SVM and In-filter Computing Formulation

Rooted in statistical learning theory, an SVM minimizes the structural risk by maximizing a classification margin over a set of training samples [31]. In the case of acoustic classification where the input is a time-series signal, one can define a data vector (for training and inference) at a time-instant n as $X_n \in \mathbb{R}^W$ constructed using a window $X_n = \{x_n, x_{n-1}, ..., x_{n-W+1}\}$ of previous W samples of the signal x_n. An SVM based binary classifier produces a decision label $y_n \in \{+1, -1\}$ corresponding to the data vector X_n according to

$$y_n = \text{sgn}(f(X_n)). \tag{1}$$

where $f : \mathbb{R}^W \to \mathbb{R}$ is given by

$$f(X_n) = \sum_{s=1}^{S} \alpha_s y_s K(X_s, X_n) + b. \tag{2}$$

Here $X_s \in \mathbb{R}^W$ is a subset of the training vector called support vectors with their a-priori known decision labels $y_s \in \{+1, -1\}$. $K : \mathbb{R}^W \times \mathbb{R}^W \to \mathbb{R}$ is a positive-definite kernel function that is also chosen a-priori and plays an important role in implementing nonlinear decision functions. $\alpha_s \in \mathbb{R}$ and b are training parameters, corresponding to the support vector x_s and is determined by solving a standard quadratic program based training procedure [31]. Note that the memory requirements to implement an SVM inference engine in hardware is proportional to the number of support vectors S, and hence in literature numerous techniques exist to reduce S using heuristic methods [32] [33].

For conventional SVM-based acoustic classifiers [34], as shown in Fig.2(a), the raw input signal is pre-processed by a feature extraction module or function $\Psi : \mathbb{R}^W \to \mathbb{R}^D$ before providing as an input to the SVM kernel. D is the feature dimension. The eq.(2) can be re-expressed as

$$f(X_n) = \sum_{s=1}^{S} \alpha_s y_s K(\Psi(X_s), \Psi(X_n)) + b. \tag{3}$$

In literature, the kernel function $K(.,.)$ and the feature extraction function $\Psi(.)$ are typically chosen independently. As a result, the memory footprint S of the SVM is determined by the complexity of the problem and the discriminative power of feature extraction. Note that a typical acoustic feature extraction function $\Psi(.)$, itself comprises several nonlinear

transformations that could directly be used as SVM kernels. However, for the SVM formulation to be valid, the nonlinear transformations must be mapped to a positive-definite kernel. In [22] we reported a mechanism to design SVMs using arbitrary template functions within a fixed memory footprint. The approach expressed the kernel in eq.(2), as an outer-product over P template functions $\Phi_p : \mathbb{R}^W \rightarrow \mathbb{R}$ as $K(X_s, X_n) = \sum_{p=1}^{P} \Phi_p(X_s)\Phi_p(X_n)$,. The template functions $\Phi_p(\cdot), p = 1, .., P$ then could represent P feature extraction modules. Following the derivations in [22], the SVM function $f(.)$ can be rewritten as:

$$f(X_n) = \sum_{p=1}^{P} Q_p \Phi_p(X_n) + b. \tag{4}$$

Here, $Q_p = \sum_{s=1}^{S} \alpha_s y_s \Phi_p(X_s)$ can be viewed as a consolidated training parameter that can be estimated using a reduced-complexity training procedure described in section IV. Note that the memory footprint of the reformulated template-SVM is determined by the number of template functions P, and each of the template functions $\Phi_p(\cdot)$ could be chosen arbitrarily. Here, $\Phi_p(\cdot)$ can be any function that can be used to express the input features in order to make a classification. This makes this framework flexible to implement various functions for classification. For example, in Fig.2(b), we illustrate how a cascade of filters a CAR-IHC feature extraction module could be used to implement $\Phi_p(\cdot)$, and described in the following section.

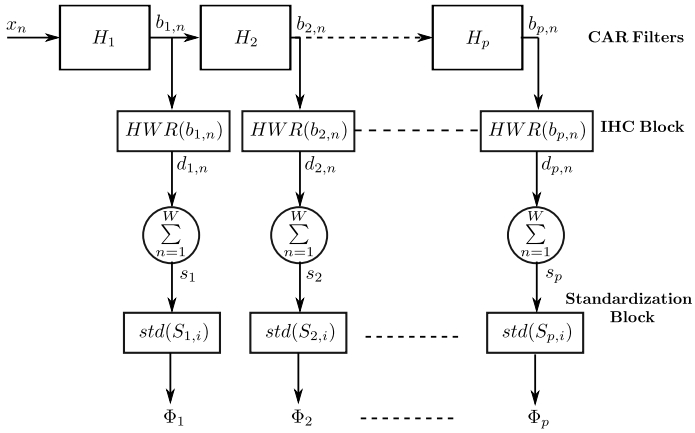

Fig. 3: Neuromorphic cochlea-based CAR-IHC model

A. CAR-IHC model as SVM Kernel

The biological cochlea is a nonlinear and causal system. This nonlinearity makes it ideal to use a cochlea model as an SVM kernel since it would give robust classification in higher dimensional space [35]. One such auditory filter model is the Cascade of Asymmetric Resonators with Fast-Acting Compression (CARFAC) model, which is a digital version of the cascade of pole-zero filter [21] [25] [36]. It consists of CAR block, which mimics Basilar Membrane (BM) functionality, IHC, Ganglion cells, and Outer Hair Cell (OHC). We use the CAR and IHC modules of this model for our kernel.

Given X_n with W samples and each sampled input data as x_n described previously. The system receives an audio sample at each sampling clock which is fed to the first CAR block H_1. There are P CAR blocks arranged in a cascaded manner as shown in Fig.3. The eq.(5) denotes the two pole two zero filter which mimics BM implemented as a CAR filter.

$$H_p = H(z) = g_p \left[\frac{z^2 + (-2a_{0,p} + k_p c_{0,p}) r_p z + r_p^2}{z^2 - 2a_{0,p} r_p z + r_p^2} \right]. \tag{5}$$

$a_{0,p}, c_{0,p}, k_p$ are the resonator filter coefficients for each filter, r_p is the pole-zero radius in the z-plane, and g_p is the DC gain factor. The CAR block transfer function $H(z)$ is derived in detail in Appendix. For the first CAR filter, the input is x_n, and due to the cascade nature of the CAR filters, the output of one filter becomes the input of the next stage filter. The output of each CAR filter is denoted by $b_{p,n}$ as shown in Fig.3, which forms the input to the IHC blocks in parallel. We use a simplified model of the IHC implemented using half wave rectifier (HWR), $HWR(\cdot) \in \mathbb{R}$, as per eq.(6).

$$HWR(q) = max(0, q). \tag{6}$$

From Fig.3, $q = b_{p,n}$ in eq.(6), which gives,

$$d_{p,n} = HWR(b_{p,n}). \tag{7}$$

The IHC generates output as per eq.(7). The IHC output is summed over W samples, and this forms the input for the standardization (std) blocks in parallel.

$$s_p = \sum_{n=1}^{W} d_{p,n}. \tag{8}$$

Here, $s_p \in \mathbb{R}$.

$$std(S_{p,i}) = \frac{S_{p,i} - \mu_p}{\sigma_p}. \tag{9}$$

where $\{s_p \in S_{p,i} | 1 \leq i \leq N\}$ with N as the training samples, $\mu_p = mean(S_{p,1}, S_{p,2}, .., S_{p,N})$ and

$$\sigma_p = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (S_{p,i} - \mu_p)^2}$$

$$\Phi_p = std(S_{p,i}). \tag{10}$$

Here, $\Phi_p \in \mathbb{R}$. The summation over W samples of the output of IHC is taken as per eq.(8) for each filter. Then standardization technique, commonly used in neural network optimizations [37], is applied across N training input samples as per eq.(9). Note that μ_p and σ_p are calculated only during training, and these vectors are passed as learned parameters to the inference engine. Therefore, an input signal vector X_n sampled at a sampling frequency f_s generates W samples with each sample denoted as x_n. It is then processed by a cascade parallel arrangement of neuromorphic cochlea-based CAR-IHC filters to estimate the kernel function Φ_p with p as the filter stage out of P filters as per (10). The output is a $P \times 1$ kernel vector, as shown in Fig.3. The classification

output is produced using eq.(4) employing this kernel vector, the output weight vector $Q \in \mathbb{R}^P$, and the bias $b \in \mathbb{R}$ obtained after the training process.

IV. Template-SVM Training

A conventional SVM training involves solving a quadratic optimization problem over a set of training data $(X_m, y_m), m = 1, .., M$, of size M [31]. The optimization can be expressed as:

$$\min_{\alpha_m} \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{M} \alpha_m \alpha_n y_m y_n K(X_n, X_m) - \sum_{m=1}^{M} \alpha_m. \quad (11)$$

$$\sum_{m=1}^{M} \alpha_m y_m = 0. \quad (12)$$

$$0 \leq \alpha_m \leq C. \quad (13)$$

Here, C is a hyper-parameter that is chosen through cross-validation and $n = 1, .., M$. Due to the quadratic nature of the optimization problem, the worst-case complexity of SVM training scales as $\mathcal{O}(M^2)$. In practice, the training complexity scales as $\mathcal{O}(MS)$, where S is the number of support vectors. However, the number of support vectors is unknown, so any SVM formulation has to accommodate the worst-case scenario.

In the template-based SVM, the kernel is expressed as an outer-product over a set of P templates as $K(X_n, X_m) = \sum_{p=1}^{P} \Phi_p(X_n)\Phi_p(X_m)$. Substituting in equation (11), the template-SVM training reduces to a lower complexity quadratic optimization problem as:

$$\min_{Q_p, \alpha_m} \frac{1}{2} \sum_{p=1}^{P} Q_p^2 - \sum_{m=1}^{M} \alpha_m. \quad (14)$$

$$s.t \;\; Q_p = \sum_{j=1}^{M} \alpha_j y_j \Phi_p(X_j). \quad (15)$$

$$\sum_{m=1}^{M} \alpha_m y_m = 0. \quad (16)$$

$$0 \leq \alpha_m \leq C. \quad (17)$$

Here, j is iterated over M training samples. Equations eq.(15), (16), (17) are the constraints imposed on eq.(14). The equation eq.(14) shows that the optimization complexity has been reduced to $\mathcal{O}(P + M)$ with P additional constraints that can be controlled based on the number of templates required for the application. This reduced complexity enables us to use this training algorithm to implement IoT devices, making them adaptive and deployable in dynamic environments. Thus, our framework is capable of online training. For the in-filter computing, the features or templates are computed as an input stream. Training the template SVM entails solving a simplified constrained quadratic problem. Thus, the architecture can be trained in an online manner. However, in a traditional SVM, the features would first need to be accumulated and fed to a kernel module. Note that there are several ways to efficiently solve the constrained optimization in eq.(14), including both batch and online variants. In the accompanying software for template-based SVM [23], we have used a growth-transform-based approach [38] to solve eq.(14).

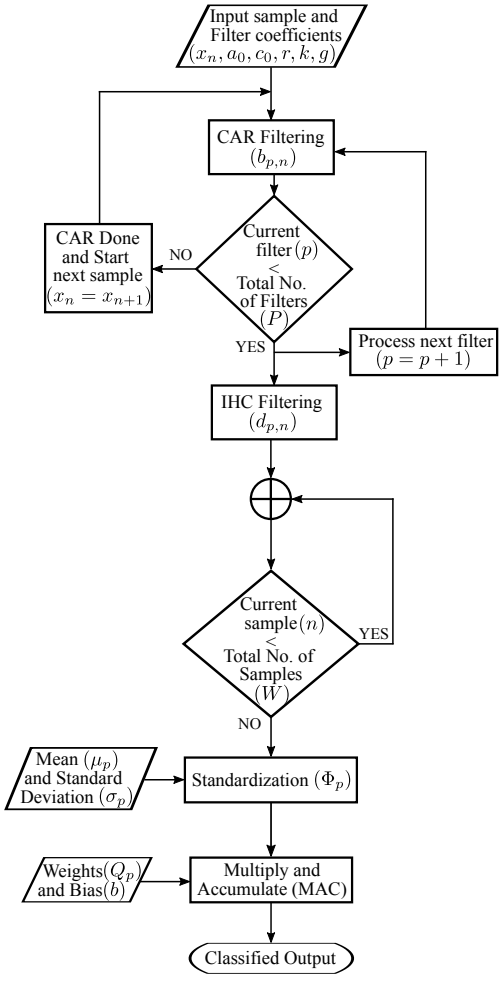

Fig. 4: High level hardware execution flow diagram of the framework.

V. FPGA Implementation of our SVM classifier

We demonstrate the efficiency of our in-filter computing architecture by implementing it on FPGA. This system is configurable as the in-filter parameters, and weight vectors are tunable based on the application. The weight vector and biases are trained offline, as mentioned in the previous sections.

Initially, we simulated this framework in floating-point in MATLAB software tool . In order to estimate the appropriate FPGA implementation, we simulated the model in fixed-point code. The CAR-IHC kernel is implemented in a 12-bit fixed-point code. The trained weights (Q_p) and bias (b) are stored as 8-bit, and the mean (μ_p) and standard deviation (σ_p) are stored as 12-bits. In our experiments, we analyzed that using 12-bits for inputs, filter coefficients and standardization parameters (mean and standard deviation) with 8-bits for weights and bias resulted in minimal accuracy degradation and reduced hardware resource utilization. We use pipelining to speed up the kernel execution in FPGA. The CAR-IHC kernel filters

are executed using the time-multiplexed technique where each filter uses the same hardware for generating output which makes the design small in area.

TABLE I: FPGA implementation summary.

FPGA Implementation Summary Spartan-7 xc7s6cpga196	
Clock Frequency	25 MHz
Audio Sampling Frequency	16 kHz
Number of Filters	30
Dynamic Power	8 mW
DSPs	4 (Total 10)
LUTs	1517 (Total 3750)
FFs	2864 (Total 7500)

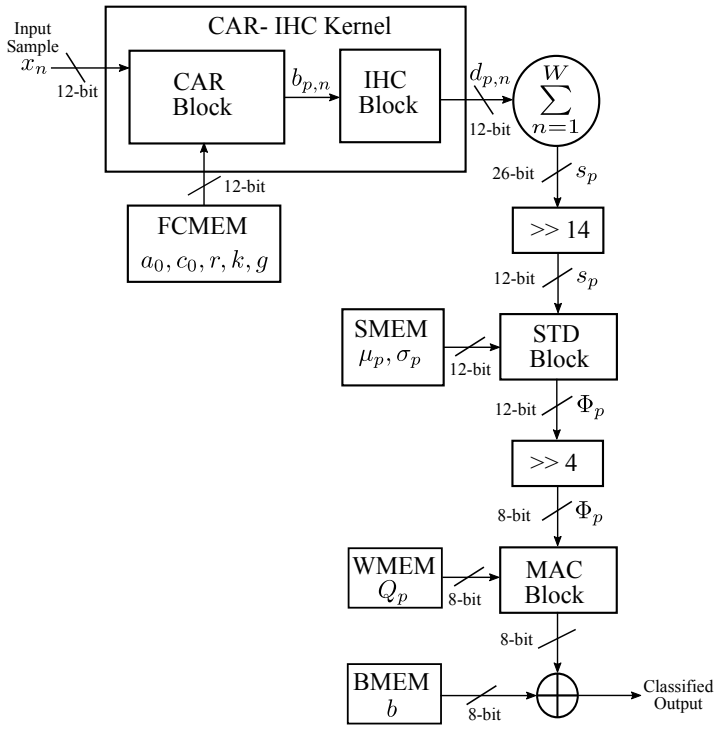

Fig. 5: Block diagram for FPGA implementation

Fig.4 shows the hardware execution flow, and Fig.5 shows the FPGA block architecture of the system. The input sample (x_n) sampled at f_s is provided as input to CAR block and filter coefficients (a_0, c_0, r, k, g), stored in the FCMEM memory block. The CAR block performs filtering as per eq.(5) followed by the IHC block, which performs half wave rectification as per eq.(6). The detailed implementation of the CAR-IHC block is shown in Fig.6. The time-multiplexed processing of each cascaded filter is determined by the $CAR\ Done$ select signal. If this signal is set, then the next sample (x_{n+1}) enters the CAR block. This signal is set only when the processing of all the filters is done. Hence, the same CAR block is used to process P filters using the stored filter coefficients for each filter (p). The $sel1$ and $sel2$ select lines control this filter coefficient flow. So the input to output delay is directly proportional to the number of cascaded blocks, P in this case, i.e., the number of filters used. The multipliers have been designed to operate in a pipelined manner, where multiplication of the coefficients will take multiple clock

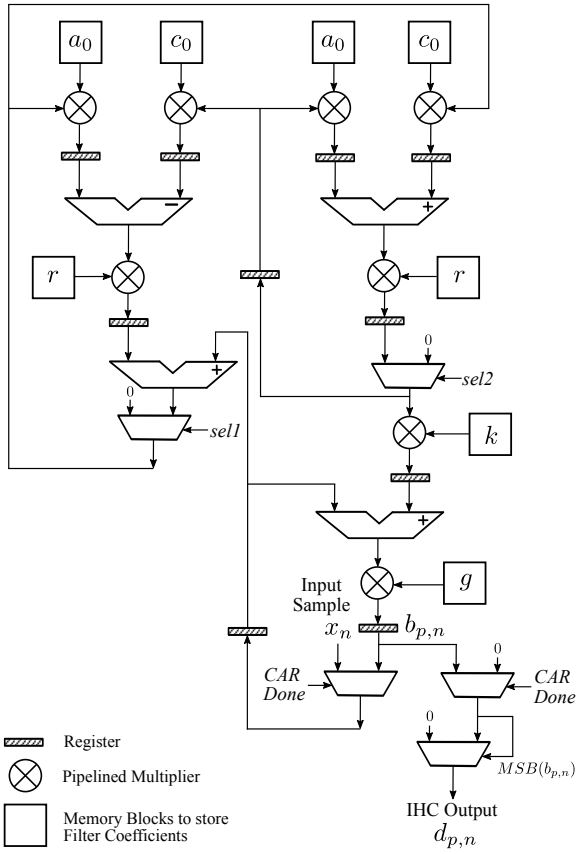

Fig. 6: CAR-IHC kernel hardware micro-architecture implementation. The filter coefficients (a_0, c_0, r, k, g) stored in block RAM and the input sampled at f_s are used as inputs to this block. This block is used in pipelined manner using select lines to enable the FSM for the design.

cycles for producing the output by reusing the hardware. The half wave rectification operation in the IHC block depends on the Most Significant Bit (MSB) select signal, determining the sign bit of CAR output ($b_{p,n}$). This results in discarding the values below zero to produce the rectified output.

The output of IHC block ($d_{p,n}$) is summed over the entire window of the input data, i.e., summation of W input samples across p filters. Standardization of the output (s_p) of summation is performed based on eq.(9) in the STD block. The size of s_p is 26-bit due to 16000 (f_s) additions, and the standardization parameters, i.e., mean (μ_p) and standard deviation (σ_p), are 12-bits, which are fetched from the SMEM memory block. We quantize s_p to 12-bit and further perform another level of quantization to 8-bit after the standardization operation. Heuristically, we found that using these multiple quantization levels achieves the lowest possible hardware resource utilization without impacting the classification accuracy. The output of standardization block (Φ_p) is now used to perform multiply-accumulate (MAC) operation with the learned weights (Q_p), which are also 8-bits, stored in the WMEM memory block. Finally, the classified output is obtained after the bias (b), stored in the BMEM memory block, is added to the output of the MAC block, as per eq.(4).

TABLE II: Comparison of architecture and resource utilization of related work.

Related Work	Mahmoodi, et al. [17]	Cutajar, et al. [18]	Boujelben, et. al. [19]	Ramos-Lara et al. [20]	This work
FPGA	Virtex4-xc4vsx35	Virtex-II XC2V3000	Artix-7 XC7A100T	Spartan 3 XCS2000	Spartan 7 xc7s6cpga196
Operating Frequency	151.286 MHz	42.012 MHz	101.74 MHz	50 MHz	25 MHz
Input Sampling Frequency	NA[1]	16 kHz	6 kHz	8 kHz	16 kHz
Flip Flop	11589	1576	17074	5351	2864
LUTs	9141	11943	16563	6785	1517
RAM (18 Kb)	99	NA[1]	4	NA[1]	0
DSP	81	64	87	21	4
Kernel type	Gaussian	RBF	Linear	RBF	CAR-IHC
Feature extraction	NA[1]	DWT (Offline)	MFCC (On-Chip)	MFCC (On-chip)	CAR-IHC (On-chip)

[1] These works did not report this entity for their designs.

Our system uses a 25 MHz system clock, and 16-bit input sampled at a 16 kHz audio sampling rate. We have used 30 filters in all the reported results here. However, it is parameterizable and can be changed based on the application requirements. Every input data sample takes about 300 clock cycles, i.e., 12μs, to be executed through the pipeline. We have an audio input sample being given to the system every 1562 clock cycles, i.e., 62.5μs. We have a buffer of around 1200 clock cycles, i.e., about 48μs, before the arrival of next sample, where the, system is idle. This shows that we can increase the sampling frequency to 80 kHz, i.e., sample at every 312 clock cycles without impacting the hardware architecture. On the other hand, for a sample of 16 kHz frequency, we can also increase the number of filters to up to 120 to use up the extra 48μs. The number of clock cycles required to execute a single audio sample increases linearly with the number of filters. There is an increase of about 2.5 % of area in overall hardware and 0.23 mW increase in power for every addition of filter. The increase in the number of filters may be required for complex auditory tasks. We can also reduce the operating frequency to as low as 400 kHz to reduce the power consumption of the system to below 1 mW. We can reduce it further to a few kHz if we reduce the input sampling frequency to a few Hz in other time-series data such as EEG/ECG signals. Hence, this shows that our system is highly flexible and scalable to suit any application in the time-series domain.

We use Xilinx Spartan series part xc7s6cpga196, a low-power FPGA manufactured on a 28 nm technology node.The Spartan series FPGA caters to edge computing and IoT platform systems, as the area footprint and power envelope are low. The dynamic power consumption for our system on FPGA is 8 mW. The logic design i.e., LUTs and registers, consumes around 2 mW of power, whereas the control signals take up 4 mW. DSPs consume 1 mw of power, and the clocks take up the remaining 1 mw of power. The weights and bias generated from the training procedure are quantized to 8-bit, and the CAR-IHC model uses 16-bit input samples to generate 16-bit output. This 16-bit kernel output data on accumulation over 16k samples increases to 30-bit, which is then reduced to 8-bit after a standardization and quantization operation. For this design, 16 kHz sampling rate with 30 filters uses 1517 LUTs and 2864 FFs summarized in Table I. This exhibits that the system can be implemented with minimum area and low power and hence suitable for IoT deployable edge devices.

The resource utilization comparison contrasts related work with our system, as shown in Table II. Our work has the advantage of being low in resource utilization compared to other works. Most of these systems use acoustic signals as input, with MFCC as the feature extractor and SVM as the classification algorithm. MFCC is a widely used feature extractor for acoustic signals since it extracts linearly separable features amongst most acoustic signals. Our framework uses a neuromorphic cochlea-based kernel that acts as a feature extractor as well as a nonlinear kernel for the SVM algorithm. This avoids the need for a separate feature extractor compared to other works. Our framework also does not require separate storage for support vectors, and at the same time, we have control over the number of weights that have to be stored based on the required application. Another advantage our system has over other systems is that it is highly tunable and is scalable for higher or lower sampling frequency input signals.

VI. Results and Discussion

We use datasets from two domains, namely speech and environmental sounds. Speech datasets prove the usability of our framework in security like voice-based access, where we can identify the speaker and provide biometric access. The environmental sounds showcase the framework's versatility which can be deployed for multiple sounds as the target for robust classification. We use MATLAB for software simulations and verification of the algorithm. The FPGA design implements the MATLAB code using fixed-point arithmetic.

Environmental Sounds Classification (ESC-10) dataset [27] consists of sound clips constructed from recordings publicly available through the Freesound project. It consists of 400 environmental recordings with 10 classes, i.e., 40 clips per class and 5 seconds per clip. Each class contains 40 wav format audio files. These clips had a lot of silence, so we trimmed the silence part and further trimmed the remaining clips into 1 second version belonging to the same class, thus increasing the dataset's number of samples. Table III shows the class labels, which depict the wide variety of data samples used. The classes include sounds from dog bark, rain, sea waves, crying baby, clock ticking, person sneezing, helicopter, chainsaw, crawing rooster and fire crackling. Here, the dataset was used to create balanced classes to identify one class versus other classes arranged randomly. The train and test accuracy values are shown with the train-to-test ratio mentioned in the bracket. One thing to note from these results is that with less amount of data too, our framework could classify the

TABLE III: ESC-10 dataset classification accuracy results in percent. The fixed point code consists of 16-bit inputs and 8-bit weights and biases. Number of filters for our work is fixed at 30.

Classes (Train/Test)	Traditional SVM			In-filter SVM (This Work)			
	Support Vectors	Floating pt.		Floating pt.		Fixed pt.	
		Train	Test	Train	Test	Train	Test
Dog (129/33)	51	88	87	89	90	89	87
Rain (119/40)	60	89	87	89	87	82	82
Sea_Waves (200/50)	113	86	82	84	78	80	74
Crying Baby (144/49)	58	91	83	91	87	93	85
Clock Tick (114/50)	86	91	86	92	88	92	85
Person Sneeze (101/44)	43	83	77	89	82	82	80
Helicopter (197/50)	48	94	88	96	90	95	85
Chainsaw (99/34)	39	92	85	93	85	93	82
Rooster (124/54)	36	92	94	93	96	93	96
Fire Crackling (152/66)	46	90	89	90	87	89	86

TABLE IV: FSDD classification accuracy results in percent. The fixed point code consists of 16-bit inputs and 8-bit weights and biases. Number of filters for our work is fixed at 30.

Classes (Train/Test)	Traditional SVM			In-filter SVM (This Work)			
	Support Vectors	Floating pt.		Floating pt.		Fixed pt.	
		Train	Test	Train	Test	Train	Test
Theo (761/254)	247	92	91	93	91	89	88
Nicolas (889/297)	197	98	98	98	97	97	94
Ywewelver (749/250)	196	92	90	94	91	89	88
Jackson (796/200)	35	99	99	99	99	99	98

sounds. We have compared our results with traditional SVM, which uses inputs after being pre-processed using the same CAR-IHC filters. For the traditional SVM, we use the in-built MATLAB library with default command lines. The number of support vectors for traditional SVMs is significantly higher than the number of filters used in our work, indicating that we can get comparable accuracy with lower hardware resources and can be used in low-powered devices. As the number of samples is low, we see lower accuracy for classes like clock tick and person sneeze. These classes have a lot of overlapping information with other classes, causing confusion.

The Free Spoken Digit Dataset (FSDD) [28] is an open dataset consisting of recordings of spoken digits in wav files at a sampling rate of 8 kHz. The recordings are trimmed so that they have near minimal silence at the beginning and ends. It consists of 4 speakers with 500 recordings per speaker, amounting to an overall of 2000 recordings. These recordings are English pronunciations of each digit from 0 to 9 by each speaker. We use our framework to identify the speaker based on the recordings. We create recordings of each speaker versus a random pool of remaining speakers. We can tune our system to each speaker and get a classification to identify whether our target speaker is speaking or not. Similar to ESC dataset results, FSDD results in Table IV also show that traditional SVM requires many more support vectors than the number of filters used in this work. As the number of support vectors is significantly higher for traditional SVMs, we see a slight reduction in accuracy for few classes in in-filter SVM. For the proposed in-filter SVM, the number of template vectors is determined by the fixed number of filters. The training algorithm tries to find the best possible solution within this fixed constraint. However, adhering to this constraint is one of the reasons for the reduction in accuracy. The other constraint with the proposed in-filter SVM approach is that the final solution is linear for the CAR-IHC (filter) function. Any nonlinear mapping is implemented only by the CAR-IHC function. Whereas in a standard SVM formulation that uses the CAR-IHC filter output as features, there is additional nonlinearity in the kernel mapping. Thus, the traditional SVM may be able to exploit this cross-filter nonlinearity to achieve better accuracy. FSDD classification showcases the capability of the framework to identify the right person, which can be used in giving access to a secure area or facility.

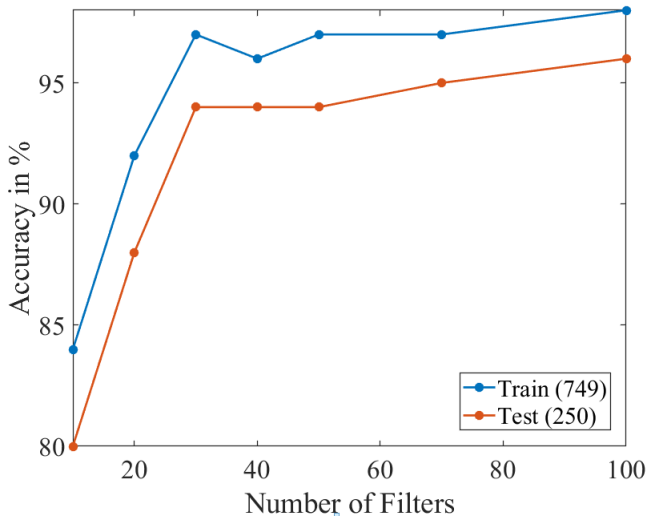

Fig. 7: Impact of increasing the number of filters on accuracy for FSDD (Yweweler) dataset.

We see from Tables III and IV that the number of support vectors (S) for the traditional SVM is always greater than the number of templates, i.e., filters (P) used for the proposed SVM ($P < S$). Traditional SVM has a computational complexity of $\mathcal{O}(S + MS)$, where $\mathcal{O}(MS)$ is the complexity of a

linear kernel. In contrast, the complexity of the proposed work is $\mathcal{O}(P)$. Thus, the computational complexity of traditional SVM increases with an increase in support vectors. We see from the results that the number of filters for in-filter SVM is less than the support vectors used in traditional SVM. As a case study, we take Yweweler class data from the FSDD dataset. The number of MAC operations required to classify this class in traditional SVM is 5096 compared to 30 MAC operations for in-filter SVM. We know that MAC operations consume maximum resources and, in turn, would increase the power consumption in any hardware design. Hence, our framework is efficient in comparison to an equivalent SVM hardware implementation.

We can tune the number of filters based on the application. The number of filters is determined by the trade-off between the hardware constraints (memory and speed) versus accuracy. Empirically, we were able to determine the optimum number of filters required for most datasets. Reducing the number of filters reduces the discriminatory information encoded by the features, and hence we observe a reduction in accuracy. We can tune the number of filters based on the application. As seen in the added Fig.7, increasing the number of filters beyond a specific value yields a marginal increase in accuracy. Thus, this marginal increase in accuracy would come at the cost of latency and increase in hardware resources, as explained in Section V. We chose 30 filters to satisfy the constraints of our implemented design. Hence, the same number of filters were used for the datasets. This shows that we can fix the number of filters based on the constraints and still obtain comparable results.

We performed an experiment to check the classification robustness of our framework. For this purpose, we added white Gaussian noise to the test input signals from the existing dataset and observed the accuracies across different Signal to Noise Ratios (SNRs). We used the MATLAB tool function $awgn$ to add the white Gaussian noise to the signals. Fig.8 shows the mean and variance plot of the test accuracy due to the addition of noise over 10 iterations. Here, we see that our framework is quite robust when we train the data with the added noise, and with no noise in training data, the test accuracy falls below 80 % as we reduce the SNR to below 25 dB.

TABLE V: List of possible domains where our framework can be implemented by tuning the frequency ranges of the CAR filters.

Time-series Data	**Typical Frequency Range (Hz)**	
	Low	**High**
Speech [39]	100	8k
Music [39]	40	18k
Accelerometers [40]	0.5	1.5k
ECG [41]	0.1	1k
EEG [42]	1	100
EMG [43]	24	400

In general, we can see from the dataset results that our work produces comparable results when the number of filters is close to the number of support vectors used in traditional SVM. This shows that we can choose the number of filters beforehand and arrive at an acceptable accuracy for the required application without relying on the algorithm to decide this hardware parameter. For each type of dataset, we need to tune the filter parameters for efficient classification. We also need to determine the number of filters used, i.e., the template vectors in SVM formulation based on multiple runs. This makes our framework highly flexible and tunable as per the application's needs. In all our experiments, we have used 30 filters. The fixed point code consists of 16-bit CAR-IHC kernel output generated using 16-bit input, and the weight and bias are limited to 8-bit values. From the results across these datasets, we see that our framework is good at identifying a person using speech. Also, the ESC-10 dataset results exhibit the framework's capability even to classify inanimate sounds that can be used in systems where such classifications can trigger a more fine-tuned action for corrective measures. Hence, by tuning the CAR filters to a certain frequency range, we can classify different time-series data as per Table V. Here, our framework can be configured for a wide range of frequencies, enabling it to use various sensors generating time-series data. This gives the flexibility of programming the framework for a specific application. Also, by determining the number of filters required for each type of classification, we can optimize the classification accuracy for any time-series data.

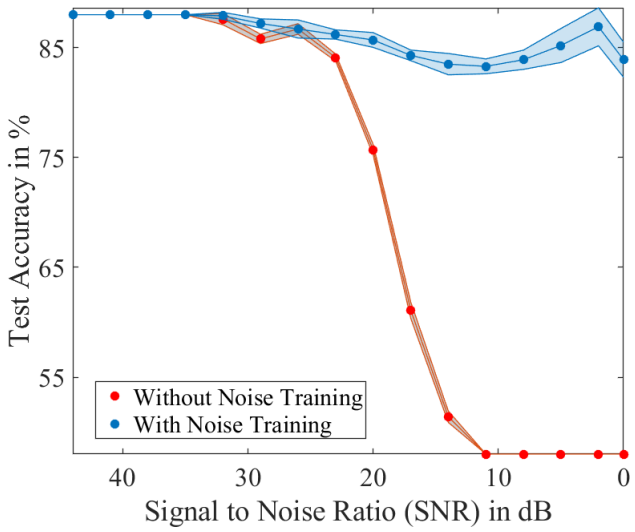

Fig. 8: Impact of noise on test accuracy for FSDD (Yweweler) dataset.

VII. Conclusion

In this paper, we have demonstrated our novel SVM-based acoustic classifier using the cochlea module as kernel and feature extraction stage simultaneously. The neuromorphic cochlea kernel of our unique algorithm does not require the kernel to be positive definite. This lack of restriction compared to traditional SVM enabled us to use cochlea-based CAR-IHC function as a kernel in our framework. Furthermore, the proposed system has the flexibility of handling different kinds of time-series data, as the kernel filter parameters can be tuned as per the frequency range of the input signal. This template-based SVM has a fixed number of templates in

contrast to varying support vectors in traditional SVMs. We can control the operating frequency by controlling the number of kernel filters, making it power efficient. This can be fine-tuned by matching the hardware constraints with the required application speed. Also, since the complexity of this novel SVM is low compared to traditional SVM, our framework is capable of performing online training. This flexibility and dynamic behavior make the framework ideal for implementing in IoT edge devices. In this paper, we have demonstrated the hardware efficiency of the in-filter computing framework on FPGA. However, this can be extended to create a custom hardware and used as a battery-powered edge device. In the future, we plan to deploy this framework in different environments as an edge classification device. We can use this algorithm on an embedded system like a microcontroller for greater flexibility in programming the device. Leveraging the reprogramability of our framework, we can build an IoT system that can be used to monitor various time-series data using a network of sensors placed at various locations for different applications. The proposed system can have several potential applications ranging from identifying animal behaviour pattern for ecologists using sensors placed in strategic locations in a forest area to healthcare data analysis using wearable sensors which provide time-series data like ECG or EEG data. Based on bird species sounds or any animal sound, we can track the presence of different rare species of wildlife in a particular environment over a period of time. In this case, we can remotely reprogram the hardware to detect different wildlife species as many of these species might not be active in a specific region for a particular season. Similarly, such systems can also be deployed for remote health care applications using signals like ECG/EEG or ultrasound for early disease detection [44] [45] [46] and for automation of industrial maintenance of machinery using various time-series data produced by mounted sensors. All these deployments lead to minimizing human intervention and reducing errors caused by logistics issues. Since our system can classify rare events with very low power consumption, we can deploy this system as an always-on system.

APPENDIX
CAR-IHC FILTER FORMULATION

A two pole two zero filter forms the asymmetric resonator whose transfer function is as below:

$$H(z) = \frac{Y}{X} = g[\frac{(z - z_{zero})(z - z^*_{zero})}{(z - z_{pole})(z - z^*_{pole})}]. \\ = g[\frac{z^2 + (-2a_0 + kc_0)rz + r^2}{z^2 - 2a_0rz + r^2}]. \quad (18)$$

The two pole coupled form has a pair of conjugate poles (z_{pole} and z^*_{pole}):

$$z_{pole}, z^*_{pole} = \frac{2a_0 \pm \sqrt{(2a_0r)^2 - 4r^2}}{2}. \quad (19) \\ = rcos(\theta_R) \pm irsin(\theta_R).$$

$$a_0 = cos(\theta_R). \quad (20)$$

where θ_R is the pole angle in the z plane. The conjugate zeros (z_{zero} and z^*_{zero}) are:

$$z_{zero}, z^*_{zero} = \frac{-(-2a_0 + kc_0)r}{2}.\pm\frac{\sqrt{((-2a_0 + kc_0)r)^2 - 4r^2}}{2}. \\ = rcos(\theta_Z) \pm irsin(\theta_Z).$$

$$a_0 - \frac{kc_0}{2} = cos(\theta_Z). \quad (21)$$

where θ_Z is the zero angle in the z plane. The zero radius is the same as the pole radius, r. The condition for complex zeros becomes relevant for high-frequency channels, where $cos(\theta_R) < 0$:

$$a_0 - \frac{kc_0}{2} > -1. \quad (22)$$

$$k < \frac{2 + 2a_0}{c_0}. \quad (23)$$

Here, r can be used to move the zeros and the poles simultaneously while k is fixed. k determines the distance between the poles and the zeros. The frequency of zeros are kept slightly higher than the poles. If we increase the value of k, the poles and zeros grow further apart, giving a slow roll off at higher frequencies. On the other hand, if the value of k is decreased to a low value, the poles and zeros grow closer, giving rise to sharp roll off making it asymmetric. This sharp roll off

TABLE VI: List of Symbols and Acronyms.

Symbols and Acronyms	Description
$\text{sgn}()$	Sign function
$\sqrt{}$	Square root function
$mean()$	Arithmetic mean function for a series
$\min_x f(x)$	Find x that minimizes the function $f(x)$
$s.t.$	Such that
$\mathcal{O}()$	Big O notation for complexity
FPGA	Field Programmable Gate Array
SVM	Support Vector Machine
CAR-IHC	Cascade of Asymmetric Resonator with Inner Hair Cells
LUT	Look-Up Table
FF	Flip-Flop
UGS	Unattended Ground Sensor
DNN	Deep Neural Network
BNN	Binary Neural Network
KNN	K-Nearest Neighbour
M-OAA	Modified One-Against-All
DWT	Discrete Wavelet Transform
RBF	Radial Basis Function
DSP	Digital Signal Processing
ROM	Read Only Memory
MFCC	Mel-frequency cepstral coefficient
SRAM	Synchronous Random Access Memory
ODE	Ordinary Differential Equation
CARFAC	Cascade of Asymmetric Resonators with Fast-Acting Compression
OHC	Outer Hair Cell
BM	Basilar Membrane
HWR	Half Wave Rectifier
MSB	Most Significant Bit
MAC	Multiply-Accumulate
EEG	Electroencephalography
ECG	Electrocardiography
ESC-10	Environmental Sound Clips-10
FSDD	Free Spoken Digit Dataset
EMG	Electromyography
SNR	Signal to Noise Ratio

is similar to the characteristic exhibited by auditory filtering. This property also enhances selection of frequencies. In order to keep the pole frequency half octave below zero frequency, k is kept sames as c_0.

To get unity gain at DC, we can solve for g as follows:

$$g = \frac{1 - 2a_0 r + r^2}{1 - (2a_0 - kc_0)r + r^2}. \tag{24}$$

The zero-crossing times of the filters impulse response does not change with respect to time, even when we change r.

$$r = 1 - damping \times \frac{2\pi f}{f_s}. \tag{25}$$

where $damping$ controls the damping factor, f is defined in eq.(26) and f_s is the sampling frequency. r keeps the damping away from zero and also makes the damping bounded. Changing r means varying the poles and the zeros of the filter. This satisfies the biologically observed condition where variation in stimulus level does not vary the impulse response zero crossings [25]. For each cascade stage, the initial values for zeros and poles are set. The Greenwood map function [47] is used to choose equidistant poles of the two pole two zero resonator. These are placed along the normalized length of the cochlea.

$$f = 165.4(10^{2.1x} - 1). \tag{26}$$

where, f is the frequency of the pole and x is the normalized position along the cochlea, varying from 0 at the apex of the BM, to 1 at the basal end.

ACKNOWLEDGMENT

This research was supported in part by (i) INSPIRE faculty fellowship (DST/INSPIRE/04/2016/000216), SPARC grant (SPARC/2018-2019/P606/SL) from Ministry of Human Resource Development and IMPRINT Grant IMP/2018/000550 from the Department of Science and Technology, India. The authors would like to acknowledge the joint Memorandum of Understanding (MoU) between Indian Institute of Science, Bangalore and Washington University in St. Louis for supporting this research activity.

REFERENCES

[1] G. L. Goodman, "Detection and classification for unattended ground sensors," in *1999 Information, Decision and Control. Data and Information Fusion Symposium, Signal Processing and Communications Symposium and Decision and Control Symposium. Proceedings (Cat. No. 99EX251)*. IEEE, 1999, pp. 419–424.

[2] J. D. Hoyt and H. Wechsler, "Detection of human speech in structured noise," in *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2. IEEE, 1994, pp. II–237.

[3] R. A. Valley, "Method and apparatus for detecting the presence of human voice signals in audio signals," Oct. 10 1995, US Patent 5,457,769.

[4] Y. Zhong, E. Dutkiewicz, Y. Yang, X. Zhu, Z. Zhou, and T. Jiang, "Internet of mission-critical things: human and animal classificationa device-free sensing approach," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3369–3377, 2017.

[5] S. J. Brown, "Multi-user remote health monitoring system," Aug. 8 2000, US Patent 6,101,478.

[6] S. Chakrabartty and G. Cauwenberghs, "Sub-microwatt analog vlsi trainable pattern classifier," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 5, pp. 1169–1179, 2007.

[7] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[8] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[9] X. Xiang, Y. Qian, and K. Yu, "Binary deep neural networks for speech recognition," *Proc. Interspeech 2017*, pp. 533–537, 2017.

[10] H. Yang, S. Liang, J. Ni, H. Li, and X. S. Shen, "Secure and efficient knn classification for industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10 945–10 954, 2020.

[11] Y. T. Quek, W. L. Woo, and L. Thillainathan, "Iot load classification and anomaly warning in elv dc picogrids using hierarchical extended k-nearest neighbors," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 863–873, 2019.

[12] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281–288, 2008.

[13] A. Palaniappan, R. Bhargavi, and V. Vaidehi, "Abnormal human activity recognition using svm based approach," in *2012 International Conference on Recent Trends in Information Technology*. IEEE, 2012, pp. 97–102.

[14] R. Palaniappan, K. Sundaraj, and S. Sundaraj, "A comparative study of the svm and k-nn machine learning algorithms for the diagnosis of respiratory pathologies using pulmonary acoustic signals," *BMC bioinformatics*, vol. 15, no. 1, pp. 1–8, 2014.

[15] A. Temko and C. Nadeu, "Classification of acoustic events using svm-based clustering schemes," *Pattern Recognition*, vol. 39, no. 4, pp. 682–694, 2006.

[16] X. Yue, Y. Liu, J. Wang, H. Song, and H. Cao, "Software defined radio and wireless acoustic networking for amateur drone surveillance," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 90–97, 2018.

[17] D. Mahmoodi, A. Soleimani, H. Khosravi, M. Taghizadeh *et al.*, "Fpga simulation of linear and nonlinear support vector machine," *Journal of Software Engineering and Applications*, vol. 4, no. 05, p. 320, 2011.

[18] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Hardware-based support vector machine for phoneme classification," in *Eurocon 2013*. IEEE, 2013, pp. 1701–1708.

[19] O. Boujelben and M. Bahoura, "Efficient fpga-based architecture of an automatic wheeze detector using a combination of mfcc and svm algorithms," *Journal of Systems Architecture*, vol. 88, pp. 54–64, 2018.

[20] R. Ramos-Lara, M. López-García, E. Cantó-Navarro, and L. Puente-Rodriguez, "Svm speaker verification system based on a low-cost fpga," in *2009 International Conference on Field Programmable Logic and Applications*. IEEE, 2009, pp. 582–586.

[21] 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.

[22] P. Kumar, A. R. Nair, O. Chatterjee, T. Paul, A. Ghosh, S. Chakrabartty, and C. S. Thakur, "Neuromorphic in-memory computing framework using memtransistor cross-bar based support vector machines," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 311–314.

[23] C. Shantanu, 2018, https://github.com/aimlab-wustl/GiniSVMMicro.

[24] R. F. Lyon, "Filter cascades as analogs of the cochlea," in *Neuromorphic systems engineering*. Springer, 1998, pp. 3–18.

[25] ——, *Human and machine hearing*. Cambridge University Press, 2017.

[26] Xilinx, "Xilinx spartan 7," 2018, https://www.xilinx.com/support/documentation/product-briefs/spartan-7-product-brief.pdf.

[27] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, 2015.

[28] Z. Jackson, "Free spoken digits dataset," 2016, https://github.com/Jakobovski/free-spoken-digit-dataset.

[29] M. Papadonikolakis and C.-S. Bouganis, "A novel fpga-based svm classifier," in *2010 International Conference on Field-Programmable Technology*. IEEE, 2010, pp. 283–286.

[30] S. Afifi, H. GholamHosseini, and R. Sinha, "Fpga implementations of svm classifiers: A review," *SN Computer Science*, vol. 1, pp. 1–17, 2020.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[32] T. Habib, J. Inglada, G. Mercier, and J. Chanussot, "Support vector reduction in svm algorithm for abrupt change detection in remote

sensing," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 3, pp. 606–610, 2009.
[33] W. Wang and Z. Xu, "A heuristic training for support vector regression," *Neurocomputing*, vol. 61, pp. 259–275, 2004.
[34] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003.
[35] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.
[36] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, "Fpga implementation of the car model of the cochlea," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 1853–1856.
[37] M. Shanker, M. Y. Hu, and M. S. Hung, "Effect of data standardization on neural network training," *Omega*, vol. 24, no. 4, pp. 385–397, 1996.
[38] A. Gangopadhyay, O. Chatterjee, and S. Chakrabartty, "Extended polynomial growth transforms for design and training of generalized support vector machines," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1961–1974, 2017.
[39] W. B. Snow, "Audible frequency ranges of music, speech and noise," *The Bell System Technical Journal*, vol. 10, no. 4, pp. 616–627, 1931.
[40] J. H. Migueles, C. Cadenas-Sanchez, U. Ekelund, C. D. Nyström, J. Mora-Gonzalez, M. Löf, I. Labayen, J. R. Ruiz, and F. B. Ortega, "Accelerometer data collection and processing criteria to assess physical activity and other outcomes: a systematic review and practical considerations," *Sports medicine*, vol. 47, no. 9, pp. 1821–1845, 2017.
[41] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ecg analysis," *Biomedical Signal Processing and Control*, vol. 43, pp. 216–235, 2018.
[42] D. P. Subha, P. K. Joseph, R. Acharya, and C. M. Lim, "Eeg signal analysis: a survey," *Journal of medical systems*, vol. 34, no. 2, pp. 195–212, 2010.
[43] P. V. Komi and P. Tesch, "Emg frequency spectrum, muscle structure, and fatigue during dynamic contractions in man," *European journal of applied physiology and occupational physiology*, vol. 42, no. 1, pp. 41–50, 1979.
[44] B. E. Dogan, G. L. Menezes, R. S. Butler, E. I. Neuschler, R. Aitchison, P. T. Lavin, F. L. Tucker, S. R. Grobmyer, P. M. Otto, and A. T. Stavros, "Optoacoustic imaging and gray-scale us features of breast cancers: correlation with molecular subtypes," *Radiology*, vol. 292, no. 3, pp. 564–572, 2019.
[45] H.-J. Kang, J. M. Lee, J. H. Yoon, K. Lee, H. Kim, and J. K. Han, "Contrast-enhanced us with sulfur hexafluoride and perfluorobutane for the diagnosis of hepatocellular carcinoma in individuals with high risk," *Radiology*, vol. 297, no. 1, pp. 108–116, 2020.
[46] Q. Yu, S. Huang, Z. Wu, J. Zheng, X. Chen, and L. Nie, "Label-free visualization of early cancer hepatic micrometastasis and intraoperative image-guided surgery by photoacoustic imaging," *Journal of Nuclear Medicine*, vol. 61, no. 7, pp. 1079–1085, 2020.
[47] D. D. Greenwood, "A cochlear frequency position function for several species 29 years later," *The Journal of the Acoustical Society of America*, vol. 87, no. 6, pp. 2592–2605, 1990.