

# Multiplierless In-filter Computing for tinyML Platforms

Abhishek Ramdas Nair\*, *Member, IEEE*, Pallab Kumar Nath\*, *Member, IEEE*, Shantanu Chakrabarty, *Senior Member, IEEE*, and Chetan Singh Thakur, *Senior Member, IEEE*

**Abstract**—Wildlife conservation using continuous monitoring of environmental factors and biomedical classification, which generate a vast amount of sensor data, is a challenge due to limited bandwidth in the case of remote monitoring. It becomes critical to have classification where data is generated, and only classified data is used for monitoring. We present a novel multiplierless framework for in-filter acoustic classification using Margin Propagation (MP) approximation used in low-power edge devices deployable in remote areas with limited connectivity. The entire design of this classification framework is based on template-based kernel machine, which include feature extraction and inference, and uses basic primitives like addition/subtraction, shift, and comparator operations, for hardware implementation. Unlike full precision training methods for traditional classification, we use MP-based approximation for training, including backpropagation mitigating approximation errors. The proposed framework is general enough for acoustic classification. However, we demonstrate the hardware friendliness of this framework by implementing a parallel Finite Impulse Response (FIR) filter bank in a kernel machine classifier optimized for a Field Programmable Gate Array (FPGA). The FIR filter acts as the feature extractor and non-linear kernel for the kernel machine implemented using MP approximation and a downsampling method to reduce the order of the filters. The FPGA implementation on Spartan 7 shows that the MP-approximated in-filter kernel machine is more efficient than traditional classification frameworks with just less than 1K slices.

**Index Terms**—IoT, FPGA, Filtering, Edge Computing.

## I. INTRODUCTION

ONE of the biggest challenges in biomedical classification is capturing data from different biosensors and providing interpretable information to improve diagnosis [1] [2]. On the other hand, in the case of wildlife conservation, identifying and localizing the threatened species and providing supportive or corrective measures to enable restoration is a challenge [3]. Emerging technologies in edge computing devices like low-power wireless sensor networks are currently being used in agriculture [4] and healthcare, [5] in combination with Machine Learning (ML) techniques, known as tinyML. Most

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).

Pallab Kumar Nath was with Department of Electronic Systems Engineering, Indian Institute of Science and currently associated with Department of Electronics Communication Engineering, Pandit Deendayal Energy University, Gandhinagar email: (pallab.nath@sot.pdpu.ac.in)

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

\* Both Abhishek Ramdas Nair and Pallab Kumar Nath have contributed equally to this paper.

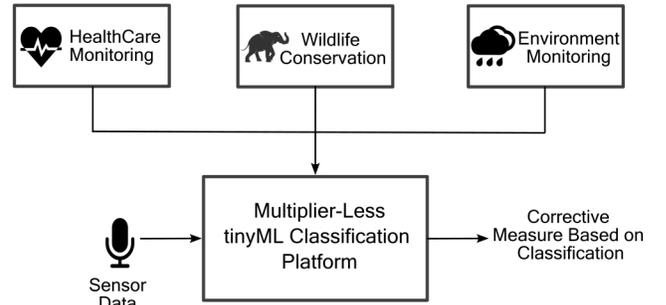


Fig. 1: Ecological Conservation and Corrective System.

of the edge-based sensor data are time-series, and it has been proven that such data can be efficiently used for tinyML classification [6]. This type of classification can be applied to healthcare with Electrocardiogram (ECG), Electroencephalogram (EEG), Electromyography (EMG), and other time-series biomedical sensor data [2].

Similar systems have been in place for ecological [7] and marine monitoring [8]. Acoustic based classification using wireless biosensors have been proven highly efficient in case of ecological applications [9] [10] as shown in Fig.1. These sensors may generate a high amount of data, but the relevant training data will be sparse, like in the case of rare or near-extinct species detection [11]. Hence, classification at the sensor node becomes even more critical as large data transmission over the network will require higher bandwidth. Despite performing well for a high volume of data, Deep Neural Networks (DNNs) do not generalize well in IoT applications, as training data is rare [12]. Moreover, training a DNN requires high-powered systems to generate appropriate learned parameters. IoT-based edge devices typically have limited resources and power. Hence, training on edge is highly improbable. However, IoT systems can have inference executed on devices using quantized and pruned versions of the neural networks [13]. Hence, such wireless sensors must perform classification at the edge and be tunable while having the lowest possible area and power.

Machine learning techniques like Support Vector Machines (SVMs), K-Nearest Neighbour (KNN), and kernel machines have proven to be robust and interpretable for rare event classification [14] [15] [16]. However, these techniques have traditionally been computationally intensive for training and inference. As most of the computation is based on Matrix-Vector Multiplication (MVM) operation, replacing multipliers with more fundamental basic primitives like addition/subtraction

will enable to design energy-efficient classification framework [17]. Multipliers tend to produce a precision explosion. We can exploit the computational primitives and approximations inherent in digital units like counters, underflow/overflow, and additions/subtraction by replacing them. In literature, there have been ways to tackle precision explosion due to multiplication for multiply-accumulate operations like quantization [18] or representation in a new number system [19].

Traditionally, IoT-based machine learning and neural networks train offline with full precision and deploy the inference at lower precision fixed point [20]. Even with quantization-aware training, the backpropagation in training is done in full precision, and only the forward pass is quantized [21]. This may be an efficient training technique, but it still is expensive for re-training on the IoT platform. There have been instances where the gradients in backpropagation have been quantized [22] [23]. However, these systems fail to achieve convergence during backpropagation [18], or they work well only when training data is available in large numbers. Moreover, the front-end, like filters and feature extractors in most edge devices, is implemented at higher precision with only the classifier quantized.

This paper leverages the energy-efficient bird density detection tinyML system [24] which uses the in-filter computing [6] with template-based SVM architecture [25]. Here we apply the Margin Propagation (MP) principle [26] to this architecture to develop a multiplierless in-filter computing framework, which exploits the computing and nonlinear primitives in the feature extraction process. Our goal is to exploit the hardware efficiency due to MP approximation for designing multiplierless filtering and classification operations, including nonlinear transformations using the kernel functions, also used as a feature extractor for training and inference. The multiplierless MP-based kernel machine has been proven to provide energy-efficient classification [27]. In our design, we implement an FIR filter bank, used as a feature extractor and kernel function, arranged in a multi-rate frequency model [28] using a multiplierless approach based on MP. This model reduces the FIR filter order used in the filter bank and helps achieve a low computation footprint. We believe that our proposed framework has the following key advantages:

- End-to-end multiplierless framework for acoustic classification using only basic primitives like addition/subtraction, underflow/overflow, shift, and comparison operations.
- Feature extraction and kernel function are combined to form an efficient computational system.
- Scalable system with user-defined memory footprint based on IoT hardware constraints.
- Integrated training using MP-based approximation mitigates approximation errors introduced in filtering and classifier.
- Since our framework uses basic computational primitives (no multipliers), it enables the implementation to push for much higher clock frequency (in this case 166MHz).

We have implemented the inference framework on an FPGA as proof of concept IoT implementation. We have validated our

architecture on the environmental sound dataset [29], which showcases the capabilities of potential deployment to identify wildlife sounds or even sounds that may indicate possible poaching or timber smuggling.

The rest of this paper is organized as follows. Section II provides a brief discussion of related work, followed by section III, where we present the in-filter computation using an MP kernel machine. Section IV provides the FPGA implementation details. Section V provides results with an audio-based dataset for detection and surveillance applications. Section VI concludes this paper, provides some useful applications and discusses possible future work using this framework.

## II. RELATED WORKS

A framework to build automated animal recognition in the wild, aiming at an automated wildlife monitoring system, has been discussed in [30]. The authors propose to use camera traps to capture data which becomes increasingly difficult in low light conditions and requires regular maintenance with increased power consumption. Extensive studies have been done using animal sounds for the recognition and classification of species employing SVMs and Hidden Markov Models (HMM) [31]. However, hardware implementation of such systems is not energy efficient for wildlife deployment. An efficient hardware implementation of acoustic classification for biometric security access was realized in [32], where Mel-Frequency Cepstral Coefficients (MFCC) is used as a feature extractor and implemented on-chip along with an SVM classifier. The MFCC and the SVM kernel occupy a high amount of hardware. To improve the hardware efficiency, we can explore the method where we can combine the feature extraction and SVM kernel as a single function as described in [6]. This eliminates the use of separate feature extraction and improves the hardware efficiency in terms of power (8 mW) and area. The work in [24] shows efficient microcontroller implementation of the work described in [6] for classification in case of ecological applications.

We still have a scope to improve the system's efficiency by exploring approximate computing to replace traditional resource-heavy operations like multipliers with more basic operators like additions. In literature, we have seen many approximate techniques like Canonic Signed Digits (CSD) [33], logarithmic function using look-up tables [34] or powers of 2 approximation [35]. These systems do not offer a complete end-to-end multiplierless computation.

Traditionally, energy efficiency can be achieved using a quantization technique instead of approximate computation, like fixed-point precision [36]. However, such linear quantization techniques result in accuracy loss as the data is represented uniformly. To mitigate this error, adaptive quantization can be used as described in [37]. This technique uses a measurement model to estimate the correct quantization for all the parameters of the classification system. Implementation of such a system is not feasible in the case of edge devices as there is the overhead of estimating the quantization levels, compromising the device's energy efficiency. Representation of the entire framework in the bfloat number system can provide similar effects of adaptive quantization [38].

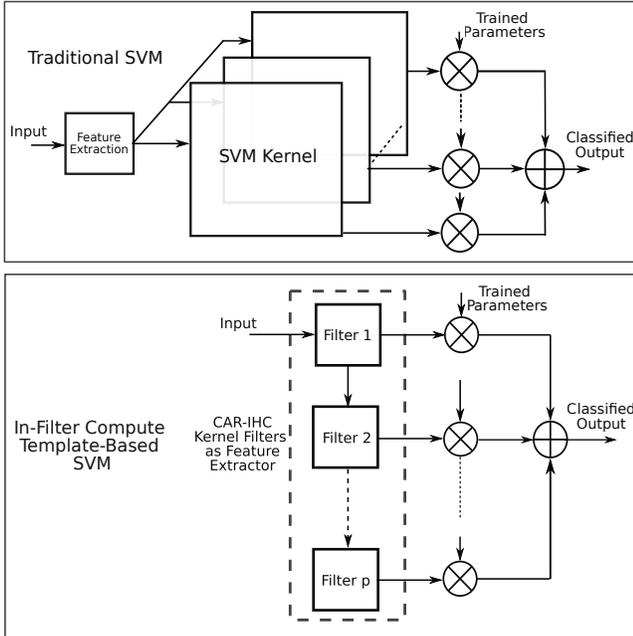


Fig. 2: Comparison of Traditional SVM and In-Filter Compute Template based SVM. The Template based SVM, using  $p$  filters, enables user-specified hardware constraints and uses feature extractor as the Kernel Function [6].

In [27], we see an end-to-end multiplierless system using the MP approximation technique for kernel machines. We extend the capabilities of this framework in our current work where we use the feature extraction used as kernel function, as used in [6] and [24], and implement this kernel in MP approximation using the MP principle in [27]. The resulting framework is a one-of-a-kind digital hardware implementation of a multiplierless acoustic classifier with a feature extractor used as a kernel.

### III. IN-FILTER COMPUTATION USING MARGIN PROPAGATION KERNEL MACHINE

In-filter computation described in [6] and [24], combines the feature extraction and non-linear SVM kernel into a single function [25] as opposed to a traditional SVM as shown in Fig.2. We leverage this principle, use an FIR filter as the kernel function, and implement this framework using MP-based approximation. MP-based kernel machine has proven to be an energy-efficient system for implementing a classification framework for edge devices [27].

#### A. Precision Explosion and MP as Adaptive Quantization

Consider a fully connected single hidden layer network. In the case of a fully connected network, for a 8-bit quantized inputs, the hidden layer output will generate 16-bit output from multiplication. For the 16-dimension input vector, the hidden layer will generate 20-bit output, and the final output layer will have 44-bit output. If we quantize this output to 8-bit, we may lose a substantial amount of accuracy. In contrast, the MP operation, which involves addition, would generate 18-bit at the output, which would result in lower accuracy

loss. Thus, MP avoids numerical precision explosion caused by conventional MAC operations. Moreover, the quantization error can be mitigated by having an online training system, which has been shown in [27].

#### B. Multiplierless Kernel Machine using MP

We develop a classification framework based on multiplierless kernel machine using the MP approximation [27]. Consider a vector  $\mathbf{x} \in \mathbb{R}^d$ , the decision function for kernel machines [39] is given as,

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{K} + \mathbf{b}. \quad (1)$$

Here,  $\mathbf{K}$  is a function of  $\mathbf{x}$ . Following the derivations in [27], we can rewrite eq.(1) in MP domain as,

$$f_{MP}(\mathbf{x}) = z^+ - z^-. \quad (2)$$

where,

$$z^+ = MP([\mathbf{w}^+ + \mathbf{K}^+, \mathbf{w}^- + \mathbf{K}^-, \mathbf{b}^+], \gamma_1). \quad (3)$$

$$z^- = MP([\mathbf{w}^+ + \mathbf{K}^-, \mathbf{w}^- + \mathbf{K}^+, \mathbf{b}^-], \gamma_1). \quad (4)$$

$\gamma_1$  is a hyper-parameter that is learned using gamma annealing. Here  $\mathbf{K}^+ = \mathbf{K}$  and  $\mathbf{K}^- = -\mathbf{K}$ .  $\mathbf{K}$  is the kernel which we derive using in-filter computation described in Section III-C. We normalize the values for  $z^+$  and  $z^-$  for better stability of the system using MP,

$$z = MP([z^+, z^-], \gamma_n). \quad (5)$$

Here,  $\gamma_n$  is the hyper-parameter used for normalization. In this case,  $\gamma_n = 1$ . The output of the system can be expressed in differential form,

$$p = p^+ - p^-. \quad (6)$$

Here,  $p \in \mathbb{R}$ ,  $p^+ + p^- = 1$  and  $p^+, p^- \geq 0$ . As  $z$  is the normalizing factor for  $z^+$  and  $z^-$ , we can estimate the output using reverse water filling algorithm [40], which is generated by the MP function for each class,

$$\begin{aligned} p^+ &= [z^+ - z]_+ \\ p^- &= [z^- - z]_+ \end{aligned} \quad (7)$$

As shown in the Fig.3, the kernel function forms a vector ( $p \times 1$ ) defined as  $\mathbf{K}$ . Using the principle of template based classification described in [25] and [6], we use the parallel FIR filterbank as the kernel as well as the feature extractor.

#### C. FIR Filter Bank as Kernel

Filter banks are commonly used for feature extraction in acoustic classification [41] [42]. We use FIR filter bank due to its stability and ease of implementation especially in approximate computing [43] [44]. Each filter in the filter bank has resonators with center frequencies based on the Greenwood function [45]. Fig.3 shows the detailed block architecture of the filter bank. The input  $x(n) \in \mathbf{x}$  is an acoustic instance sampled at 16 kHz frequency, i.e,  $N = 16000$ .  $\mathbf{B}_p$  denotes

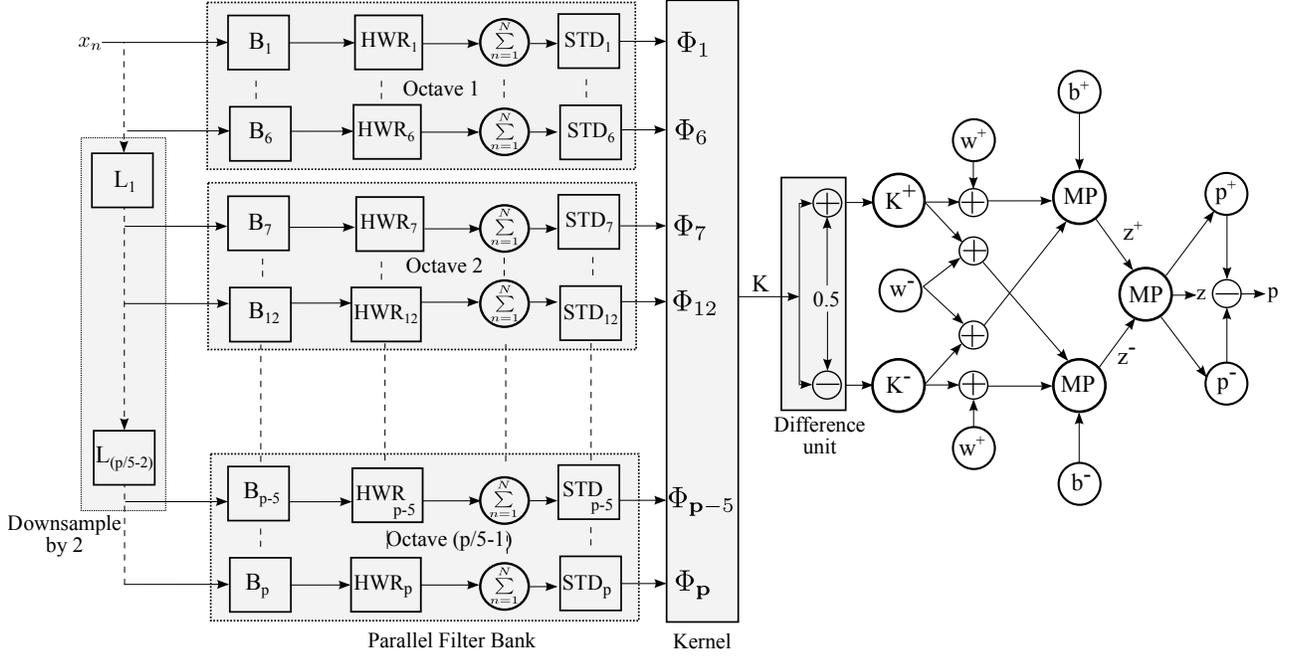


Fig. 3: FIR parallel filter bank framework for MP based classification for  $p \times 1$  Kernel. Here, the input  $x_n$  is provided to the parallel FIR filter bank to generate a  $p \times 1$  kernel. This kernel is used as input for a single layer classification network formed using MP modules. The parallel filter bank and the downsampling low pass filter blocks also use MP modules for computation.

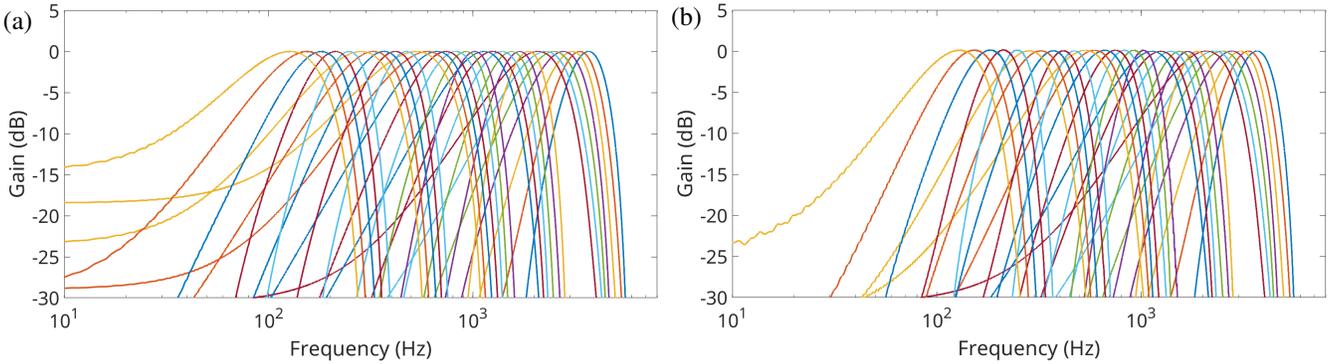


Fig. 4: FIR Filter Bank Output (Gain Response) for chirp signal (a) Without Downsampling (Filter Order ranges from 15 to 200) (b) With Downsampling (Filter Order fixed at 15)

the  $p^{th}$  bandpass FIR filter and  $p \in P$ , i.e.,  $P$  is the total number of filters in the filter bank.

$$\mathbf{B}_p(n) = \sum_{k=0}^{M-1} h_p(k)x(n-k). \quad (8)$$

Here,  $h_p$  is the filter coefficient based on  $p^{th}$  filter cut off frequency.  $M$  is the order of the filter. The output of the band pass filter is half wave rectified using HWR block, which is then accumulated over  $N$  samples and then standardized (STD block) to get the kernel function  $\Phi_p \in \mathbb{R}$ . The derivation for this kernel function is described in Appendix A.

The filter bank has been divided into multiple octaves with a bank size of 5 filters per octave. Octaves are defined based on the sampling frequencies in decreasing order. The cut-off frequencies is equally spaced within the octaves. The

coefficients ( $h_p(n)$ ) are precomputed and provided as inputs to each filter. We use the technique of downsampling input sampling frequency and segregating the cut-off frequencies into separate octaves, as shown in Fig.3. The cut-off frequencies are arranged in descending order which helps to reduce input sampling frequency. This is a proven efficient way of implementing a filter bank, as shown in [28]. The downsampling employs a low pass filter (L) used for anti-aliasing at the input for each octave. Downsampling of input ensures usage of lower order FIR filter to obtain the desired output. This is shown in Fig.4 using a comparison of the output of an FIR filter bank with and without downsampling for a size of 30 filters. We use a chirp signal as the input with increasing frequency and sampling rate at 16 kHz. We require a range of higher-order FIR filters to get the desired output in filter

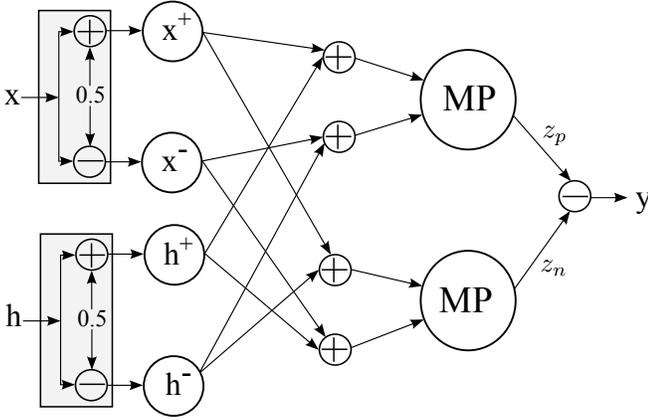


Fig. 5: Filtering operation using MP.

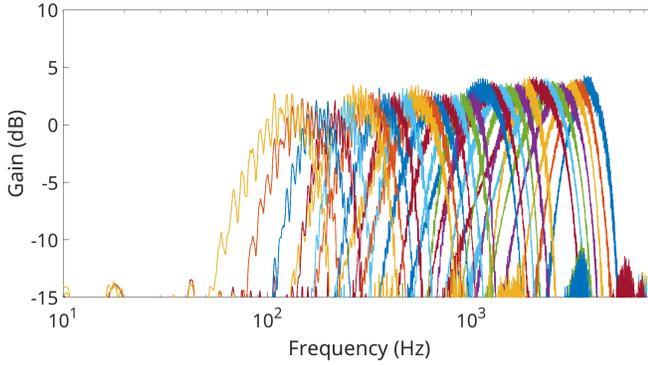


Fig. 6: MP Filter bank output (Gain Response) for chirp signal. This response shows distortion due to MP approximation errors.

banks without downsampling. The higher cut-off frequencies required a filter order of 15, and as the cut-off frequencies were reduced, the order of the filters increased, which resulted in the lowest 5 frequencies requiring 200 ordered filters. In contrast, we see that the same output can be obtained with just 15 ordered FIR filters by downsampling the input signal for each range of cut-off frequencies. In this case, the cut-off frequencies were segregated into 6 octaves, corresponding to the sampling frequencies in descending order. The additional low pass filter required the same order as the bandpass filter. Thus, this down-sampling technique provides an efficient way of implementing an FIR filter bank for low-powered devices.

#### D. Filtering operation in MP domain

We use two types of filtering operation in our filter bank, i.e., a low pass filter for downsampling and a bandpass filter. These filtering operations result in an inner product computation between the filter coefficients ( $h_p(n) \in \mathbf{h}$ ) and input samples ( $x(n)$ ) as per eq.(8). Following the derivations in [27], we can express this filtering operation in MP domain as below,

$$y = MP([\mathbf{h}^+ + \mathbf{x}^+, \mathbf{h}^- + \mathbf{x}^-], \gamma_f) - MP([\mathbf{h}^+ + \mathbf{x}^-, \mathbf{h}^- + \mathbf{x}^+], \gamma_f). \quad (9)$$

For this implementation, we have  $\mathbf{h}^+ = \mathbf{h}$ ,  $\mathbf{h}^- = -\mathbf{h}$ ,  $\mathbf{x}^+ = \mathbf{x}$  and  $\mathbf{x}^- = -\mathbf{x}$ .  $\gamma_f$  is the MP parameter for the filtering

operation. Fig.5 shows the MP implementation of the filtering operation for the low pass and the bandpass filters. Since the property of MP inherently exhibits low pass filtering, based on the reverse water filling algorithm described in [40], we require a lower-ordered low pass filter implementation in the case of the MP domain. We can see the frequency response of the filter bank in the MP domain in Fig.6.

We observe some amount of distortion in the gain response of the chirp signal output. This is due to the MP approximation of the inner product for filtering operation. The learning algorithm can mitigate this approximation error, where the weights will be adjusted, considering the approximation error. MP approximation technique minimizes the error rather than mitigating the approximation itself, improving the system's overall accuracy. This technique requires basic primitives like comparators, shift operators, counters, and adders to implement the system, making it hardware-friendly.

#### IV. FPGA IMPLEMENTATION

The high-level block diagram of the proposed design is shown in Fig 7. The target frequency of the proposed design is set to 50 MHz, and the input sampling rate is set to 16 KHz. The number of clock cycles available in between two samples are 3125. The architecture is designed such a way that processing of a new sample is completed within this time limit. Here, 3 MP modules (MP0-2) work simultaneously to meet the time constraints. The MP0 is used to implement 4 low pass (LP) filters and two MP modules (MP1 and MP2) are responsible for Band Pass (BP) filtering operation. The internal architecture and working principle of a MP module is described in [27]. The window size of LP filter is 6 and the samples are stored in a register bank of dimension 6-bit. In LP filter section, four register banks (LPRegBank) are used to store inputs for four LP filters. The selection of a particular register bank is done by the select lines  $sel0$  and  $sel1$ . The ROM (ROM0) is used to store coefficients for LP filters. The precision of the data path is set to 10 bits for the proposed design. Initially, the input samples ( $x(n)$ ) are stored in a register bank and fed to the MP0 for implementing LP filter  $L_1$ , and the output of  $L_1$  is down-sampled by 2 and passed to the corresponding parallel BP filter bank (for generating octave 2) (as discussed in Section. III-C). Here, MP0 implements 4 LP filters in time multiplexed fashion and generates desired outputs for Octave 1, 2, 3 and 4. The outputs are stored in corresponding register banks (RegBank1-4) using select signal  $sel1$ . The contents of the register banks (RegBank1-4) are used for parallel BP filtering operation and generates kernel function  $\Phi_5$  to  $\Phi_{30}$ .

One single MP module (MP1) is used repeatedly to generate outputs for octave 1. The window size of the BP filter is

TABLE I: FPGA Implementation Summary.

Device	Spartan 7 xc7s6cpga196-2			
Frequency		50 MHz		
Dynamic Power		17 mW		
Slices		903		
FFs	LUTs	DSP	BRAM	
2376	1503	0	0	



TABLE III: ESC-10 dataset classification accuracy results in percent. Number of filters for our work is fixed at 30. We used 8-bit fixed point for our design

Classes	Normal SVM			CARIHC SVM		MP In-Filter Compute			
	Floating Point			Floating Point		Floating Point		Fixed Point (8-bit)	
	SVs	Train	Test	Train	Test	Train	Test	Train	Test
<b>Dog (129/33)</b>	42	90	94	89	90	91	94	91	94
<b>Rain (119/40)</b>	44	86	90	89	87	90	90	88	88
<b>Sea_Waves (200/50)</b>	80	87	90	84	78	89	88	88	88
<b>Crying Baby (144/49)</b>	37	93	84	91	87	92	87	89	88
<b>Clock Tick (114/50)</b>	54	81	76	92	85	85	86	85	84
<b>Person Sneeze (101/44)</b>	49	85	75	87	80	86	80	85	80
<b>Helicopter (197/50)</b>	45	92	88	95	90	88	85	85	86
<b>Chainsaw (99/34)</b>	41	90	85	93	82	92	81	92	80
<b>Rooster (124/54)</b>	40	93	93	93	96	90	94	91	94
<b>Fire Crackling (152/66)</b>	46	93	83	89	87	89	92	90	88

TABLE IV: FSDD classification accuracy results in percent. Number of filters for our work is fixed at 30. We used 8-bit fixed point for our design

Classes	Normal SVM			CARIHC SVM		MP Kernel			
	Floating Point			Floating Point		Floating Point		Fixed Point	
	SVs	Train	Test	Train	Test	Train	Test	Train	Test
<b>Theo (761/254)</b>	107	96	96	93	91	92	93	92	92
<b>Nicolas(889/297)</b>	15	100	100	98	97	99	99	98	98

design, as this FPGA caters to edge applications. Table I shows the resource utilization and power consumption of our design. We were able to implement our design with usage of less than 1K of FPGA slices and the dynamic power consumption is limited to 17 mW for a 50 MHz operating frequency. Table II compares similar designs using varied edge datasets for resource utilization and power consumption. We see a better resource utilization of our design in comparison to these systems with lower power consumption in mW/MHz. The proposed study consumes almost the same amount of LUTs and 488 fewer FFs than the similar design presented in [6]. Due to multiplierless design, the proposed architecture does not consume any DSPs, whereas the design reported in [6] consumes 4 DSPs. We computed the number of LUTs required to replace 4 DSPs for fair comparisons. We have implemented  $4 \times 4$ , and  $8 \times 8$  signed multipliers (Baugh Wooley) in FPGA and found that they have consumed 19 and 72 (4 $\times$  more) LUTs, respectively. The design reported in [6] uses 4 signed multipliers and the dimensions are  $20 \times 12$ ,  $20 \times 12$ ,  $12 \times 12$ ,  $16 \times 8$  respectively. The approximation calculation shows that all 4 multipliers consume at least 890 LUTs. Hence the proposed multiplierless design can save at least 25% hardware resources (LUTs + FFs) compared to the design presented in [6]. The power consumption (mW/MHz) is almost same for the proposed design and the design presented in [6], as the design is small and only 4 multipliers are used. However, for the bigger network such as DNN our multiplierless approach would give significant benefit. The proposed design can achieve maximum operation frequency of 166 MHz which can be used to support more input sampling rate.

## V. RESULTS AND DISCUSSION

The framework's classification ability is showcased using the environmental sounds dataset. Identification of different environmental sounds shows the versatile nature of the framework that can be put to use in various acoustic applications. As

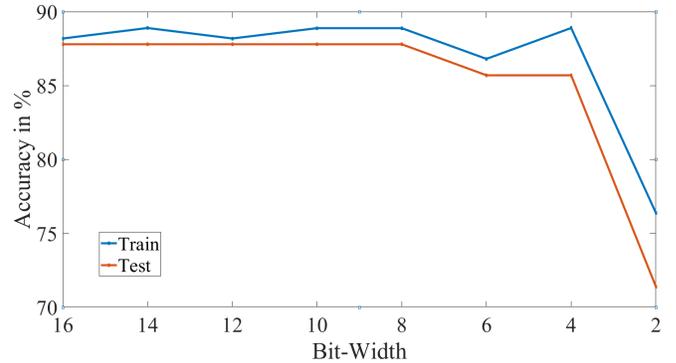


Fig. 8: Impact of bit-width on dataset accuracy for Crying Baby class from ESC-10.

wildlife conservation would result in rare data event detection, Environmental Sounds Classification (ESC-10) dataset [29] would be an ideal dataset to showcase the framework capabilities in case of ecological application. Also, we compared our system with [6] using ESC-10 and Free Speech Digit Dataset (FSDD), where we use speaker identification as the application.

ESC-10 dataset 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 a 1-second version of the same class, thus increasing the dataset's number of samples. Table III shows the results for this dataset, having classes like a dog bark, rain, sea waves, crying baby, clock ticking, person sneezing, helicopter, chainsaw, crawling rooster, and fire crackling. The classification uses one versus all methodology to identify the classes, where the data is balanced and randomly arranged for train and test sets (shown in brackets). We use the in-built MATLAB

library with default command lines for the traditional SVM. Here, the CARIHC SVM employs a completely different approach compared to standard SVM for arriving at the accuracy, which is detailed in [6], [25]. Since the dataset size is small, the accuracy values differ by a bigger margin for some classes between the traditional SVM and the other two SVM implementation, as small variations in positive or negative classification will lead to a greater impact on accuracy number. Similarly, we compare the identification of two speakers from the FSDD dataset. These results show that our framework takes advantage of template SVM methodology with a fixed number of templates and the MP approximation technique, delivering comparable results. We have also compared our system with similar systems, which is area efficient, as shown in Table II.

We use an 8-bit fixed point for implementing the hardware. We performed an empirical analysis of the dataset (using the crying baby class) with different bit widths. As shown in Fig.8, the training and testing accuracy remains stable till 8-bit and decreases sharply for bit width lower than 8-bit. We use Keras [51] implementation for training our system, as this software framework is robust and highly optimized. The FIR filter banks are quantized at 8-bit, and a custom layer for the MP function is written for the Keras framework. The optimization of the model is done using Tensorflow [52] libraries for quantization.

## VI. CONCLUSION

This paper presents a novel multiplierless framework for acoustic classification using an FIR filterbank as the feature extraction and kernel function stage simultaneously. This framework is entirely multiplierless since the FIR filter bank is implemented using MP approximation along with the inference logic. Furthermore, the framework is tunable to any time series data by tuning the filter parameters in the FIR filter bank. The framework's scalability is evident as the number of filters is user-defined and can be controlled to adhere to IoT system constraints. Being completely multiplierless makes the system highly efficient for deployment in battery-powered edge devices. Various time series data generated from different biosensors can be used in their raw format and for classification using this framework without additional preprocessing or hardware. A network of edge devices running our proposed classification framework can be used for continuous monitoring of wildlife species and detecting anomalies in case of poaching or timber smuggling. This framework can be extended to other biomedical applications using edge devices capable of healthcare monitoring with raw ECG, EMG, and EEG signals. Wearable IMU sensors with this framework can be used to detect anomalies in posture. To make our framework more energy efficient, we can fabricate this system into an Application Specific Integrated Chip.

## APPENDIX

### A. FIR Filter Kernel Derivation

The output of each band pass filter, as shown in Fig.3, forms the input to the HWR blocks in parallel.

$$HWR(q) = \max(0, q). \quad (10)$$

The half wave rectified output is summed over  $N$  samples of a single input, and this forms the input for the standardization ( $STD$ ) blocks in parallel. Let  $d_p(n) = HWR_p(B_p(n))$ ,

$$s_p = \sum_{n=1}^N d_p(n). \quad (11)$$

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

$$STD(S_{p,i}) = \frac{S_{p,i} - \mu_p}{\sigma_p}. \quad (12)$$

where  $\{s_p \in S_{p,i} | 1 \leq i \leq M\}$  with  $M$  as the number of training inputs,  $\mu_p = \text{mean}(S_{p,1}, S_{p,2}, \dots, S_{p,M})$  and

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

$$\Phi_p = STD(S_{p,i}). \quad (13)$$

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

The summation over  $N$  samples of the output of HWR is taken as per eq.(11) for each filter. Then standardization technique, commonly used in neural network optimizations [53], is applied across  $M$  training input samples as per eq.(12). Note that  $\mu_p$  and  $\sigma_p$  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  $N$  samples with each sample denoted as  $x(n)$ . It is then processed by a parallel FIR filter bank to estimate the kernel function  $\Phi_p$  with  $p$  as the filter stage out of  $P$  filters as per (13). The output is a  $P \times 1$  kernel vector ( $\mathbf{K}$ ), as shown in Fig.3.

## 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] I. Tobore, J. Li, L. Yuhang, Y. Al-Handarish, A. Kandwal, Z. Nie, L. Wang *et al.*, "Deep learning intervention for health care challenges: some biomedical domain considerations," *JMIR mHealth and uHealth*, vol. 7, no. 8, p. e11966, 2019.
- [2] B. Zhao, J. Mao, J. Zhao, H. Yang, and Y. Lian, "The role and challenges of body channel communication in wearable flexible electronics," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 283–296, 2020.
- [3] G. W. Witmer, "Wildlife population monitoring: some practical considerations," *Wildlife Research*, vol. 32, no. 3, pp. 259–263, 2005.
- [4] A. Zgank, "Iot-based bee swarm activity acoustic classification using deep neural networks," *Sensors*, vol. 21, no. 3, p. 676, 2021.
- [5] M. A. Quintana-Suárez, D. Sánchez-Rodríguez, I. Alonso-González, and J. B. Alonso-Hernández, "A low cost wireless acoustic sensor for ambient assisted living systems," *Applied Sciences*, vol. 7, no. 9, p. 877, 2017.
- [6] A. R. Nair, S. Chakrabarty, and C. S. Thakur, "In-filter computing for designing ultra-light acoustic pattern recognizers," *IEEE Internet of Things Journal*, 2021.

- [7] T. Popović, N. Latinović, A. Pevsić, vZ. Zevcević, B. Krstajić, and S. Djukanović, "Architecting an iot-enabled platform for precision agriculture and ecological monitoring: A case study," *Computers and electronics in agriculture*, vol. 140, pp. 255–265, 2017.
- [8] J. Luo, Y. Yang, Z. Wang, and Y. Chen, "Localization algorithm for underwater sensor network: A review," *IEEE Internet of Things Journal*, 2021.
- [9] A. De La Piedra, F. Benitez-Capistros, F. Dominguez, and A. Touhafi, "Wireless sensor networks for environmental research: A survey on limitations and challenges," in *Eurocon 2013*. IEEE, 2013, pp. 267–274.
- [10] D. Tuia, B. Kellenberger, S. Beery, B. R. Costelloe, S. Zuffi, B. Risse, A. Mathis, M. W. Mathis, F. van Langevelde, T. Burghardt *et al.*, "Perspectives in machine learning for wildlife conservation," *Nature communications*, vol. 13, no. 1, pp. 1–15, 2022.
- [11] Y. Shan, D. Paull, and R. McKay, "Machine learning of poorly predictable ecological data," *Ecological Modelling*, vol. 195, no. 1-2, pp. 129–138, 2006.
- [12] S. S. Heydari and G. Mountrakis, "Effect of classifier selection, reference sample size, reference class distribution and scene heterogeneity in per-pixel classification accuracy using 26 landsat sites," *Remote Sensing of Environment*, vol. 204, pp. 648–658, 2018.
- [13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
- [14] 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.
- [15] P. Nair and I. Kashyap, "Hybrid pre-processing technique for handling imbalanced data and detecting outliers for knn classifier," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE, 2019, pp. 460–464.
- [16] M. Maalouf and T. B. Trafalis, "Robust weighted kernel logistic regression in imbalanced and rare events data," *Computational Statistics & Data Analysis*, vol. 55, no. 1, pp. 168–183, 2011.
- [17] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.
- [18] Y. Guo, "A survey on methods and theories of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.
- [19] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1051–1056.
- [20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.
- [21] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," *arXiv preprint arXiv:2004.07320*, 2020.
- [22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.
- [23] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.
- [24] H. Sabbella, A. R. Nair, V. Gumme, S. Yadav, S. Chakrabarty, and C. S. Thakur, "An always-on tinyml acoustic classifier for ecological applications," in *ISCAS*, 2022.
- [25] P. Kumar, A. R. Nair, O. Chatterjee, T. Paul, A. Ghosh, S. Chakrabarty, and C. S. Thakur, "Neuromorphic in-memory computing framework using memristor cross-bar based support vector machines," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 311–314.
- [26] S. Chakrabarty and G. Cauwenberghs, "Margin propagation and forward decoding in analog vlsi," *A (A)*, vol. 100, p. 5, 2004.
- [27] A. R. Nair, P. K. Nath, S. Chakrabarty, and C. S. Thakur, "Multiplierless mp-kernel machine for energy-efficient edge devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.
- [28] R. K. Singh, Y. Xu, R. Wang, T. J. Hamilton, A. van Schaik, and S. L. Denham, "Car-lite: A multi-rate cochlea model on fpga," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.
- [29] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, 2015.
- [30] H. Nguyen, S. J. Maclagan, T. D. Nguyen, T. Nguyen, P. Flemons, K. Andrews, E. G. Ritchie, and D. Phung, "Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring," in *2017 IEEE international conference on data science and advanced Analytics (DSAA)*. IEEE, 2017, pp. 40–49.
- [31] F. Weninger and B. Schuller, "Audio recognition in the wild: Static and dynamic classification on a real-world database of animal vocalizations," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 337–340.
- [32] R. Ramos-Lara, M. López-García, E. Cantó-Navarro, and L. Puente-Rodríguez, "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.
- [33] B. Mandal, M. P. Sarma, K. K. Sarma, and N. Mastorakis, "Implementation of systolic array based svm classifier using multiplierless kernel," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014, pp. 35–39.
- [34] Z. Xue, J. Wei, and W. Guo, "A real-time naive bayes classifier accelerator on fpga," *IEEE Access*, vol. 8, pp. 40755–40766, 2020.
- [35] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li, "Deepshift: Towards multiplication-less neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2359–2368.
- [36] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International conference on machine learning*. PMLR, 2016, pp. 2849–2858.
- [37] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard, "Adaptive quantization for deep neural network," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [38] A. Hagiescu, M. Langhammer, B. Pasca, P. Colangelo, J. Thong, and N. Ilkhani, "Bfloat mlp training accelerator for fpgas," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2019, pp. 1–5.
- [39] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.
- [40] M. Gu, *Theory, Synthesis and Implementation of Current-mode CMOS Piecewise-linear Circuits Using Margin Propagation*. Michigan State University, Electrical Engineering, 2012.
- [41] H. B. Sailor, D. M. Agrawal, and H. A. Patil, "Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification," in *InterSpeech*, vol. 8, 2017, p. 9.
- [42] 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.
- [43] W.-S. Lu and A. Antoniou, "Design of digital filters and filter banks by optimization: A state of the art review," in *2000 10th European signal processing conference*. IEEE, 2000, pp. 1–4.
- [44] L. B. Soares, S. Bampi, and E. Costa, "Approximate adder synthesis for area-and energy-efficient fir filters in cmos vlsi," in *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2015, pp. 1–4.
- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] H. Khosravi and E. Kabir, "Introducing a very large dataset of hand-written farsi digits and a study on their varieties," *Pattern recognition letters*, vol. 28, no. 10, pp. 1133–1141, 2007.
- [50] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.
- [51] F. Chollet *et al.*, "Keras," 2016, <https://github.com/fchollet/keras>.
- [52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

- [53] 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.