2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)

2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID) | 979-8-3503-8440-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSID60093.2024.00038

Multiplierless In-filter Computing for tinyML Platforms

A.R. Nair*$, P.K. Nath#$, S. Chakrabartty†, C.S. Thakur*
{abhisheknair, csthakur}@iisc.ac.in, shantanu@wustl.edu, pallab.nath@sot.pdpu.ac.in
* Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore, India, 560012
† Department of Electrical and Systems Engineering, Washington University in St. Louis, USA, 63130
#Department of Electronics Communication Engineering, Pandit Deendayal Energy University, Gandhinagar
$ Both A. R. Nair and P. K. Nath have contributed equally to this paper.

***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. 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 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. 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]. On the other hand, in the case of wildlife conservation, identifying and localizing the threatened species is a challenge [2]. Emerging technologies in edge computing devices like low-power wireless sensor networks are currently being used in agriculture [3] and healthcare, [4] in combination with Machine Learning (ML) techniques, known as tinyML. Most of the edge-based sensor data are time-series, and it has been proven that such data can be efficiently used for tinyML classification [5]. This type of classification can be applied to healthcare with Electrocardiogram (ECG), Electroencephalogram (EEG), Electromyography (EMG), and other time-series biomedical sensor data [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 [6]. Hence, classification at the sensor node becomes even more critical as large data transmission over the network will require higher bandwidth.

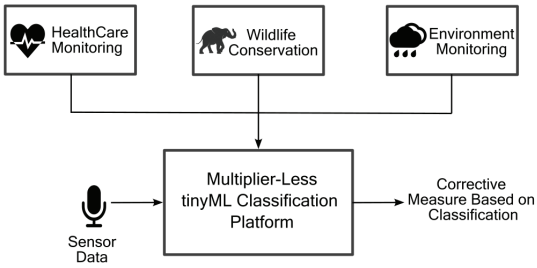

Fig. 1: Ecological Conservation and Corrective System.

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 [7]. Moreover, training a DNN requires high-powered systems to generate appropriate learned parameters. 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 [8]. 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 designing an energy-efficient classification framework. [9]. We can exploit the computational primitives and approximations inherent in digital units like counters, underflow/overflow, and additions/subtraction. In literature, there have been ways to tackle precision explosion due to multiplication for multiply-accumulate operations like quantization [10].

Traditionally, IoT-based machine learning and neural networks train offline with full precision and deploy the inference at lower precision fixed point [11]. Even with quantization-aware training, the backpropagation in training is done in full precision, and only the forward pass is quantized [12]. 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 [13]. However, these systems fail to achieve convergence

979-8-3503-8440-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VLSID60093.2024.00038

during backpropagation [10], 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, are implemented at higher precision with only the classifier quantized.

This paper leverages the energy-efficient bird density detection tinyML system [14] which uses the in-filter computing [5] with template-based SVM architecture [15]. Here we apply the Margin Propagation (MP) principle [16] to this architecture to develop a multiplierless in-filter computing framework, which exploits the computing and nonlinear primitives in the feature extraction process. The multiplierless MP-based kernel machine has been proven to provide energy-efficient classification [17]. In our design, we implement an FIR filter bank, used as a feature extractor and kernel function, arranged in a multi-rate frequency model [18] using a multiplierless approach based on MP. 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 [19], which showcases the capabilities of potential deployment to identify wildlife sounds or even sounds that may indicate possible poaching or timber smuggling.

II. In-Filter Computation using Margin Propagation Kernel Machine

In-filter computation described in [5] and [14], combines the feature extraction and non-linear SVM kernel into a single function [15] as opposed to a traditional SVM. 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 [17].

A. Multiplierless Kernel Machine using MP

We develop a classification framework based on multiplierless kernel machine using the MP approximation [17]. Consider a vector $\mathbf{x} \in \mathbb{R}^d$, the decision function for kernel machines [20] is given as,

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

Here, $\mathbf{K}$ is a function of $\mathbf{x}$. Following the derivations in [17], we can rewrite eq.(1) in MP domain as,

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

where,

$$z^+ = MP([\mathbf{w}^+ + \mathbf{K}^+, \mathbf{w}^- + \mathbf{K}^-, \mathbf{b}^+], \gamma_1). \tag{3}$$
$$z^- = MP([\mathbf{w}^+ + \mathbf{K}^-, \mathbf{w}^- + \mathbf{K}^+, \mathbf{b}^-], \gamma_1). \tag{4}$$

γ_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 II-B. We normalize the values for z^+ and z^- for better stability of the system using MP,

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

Here, γ_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,

$$p = p^+ - p^-. \tag{6}$$

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 [21], which is generated by the MP function for each class,

$$p^+ = [z^+ - z]_+.$$
$$p^- = [z^- - z]_+. \tag{7}$$

As shown in the Fig.2, the kernel function forms a vector (p $\times$ 1) defined as $\mathbf{K}$. Using the principle of template based classification described in [15] and [5], we use the parallel FIR filterbank as the kernel as well as the feature extractor.

B. FIR Filter Bank as Kernel

Filter banks are commonly used for feature extraction in acoustic classification [22]. We use FIR filter bank due to it's stability and ease of implementation especially in approximate computing [23]. Each filter in the filter bank has resonators with center frequencies based on the Greenwood function [24]. Fig.2 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 the p^{th} bandpass FIR filter and $p \in P$, i.e., P is the total number of filters in the filter bank.

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

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 STD block is used for standardizing the values based on the inputs.

The filter bank has been divided into multiple octaves with a bank size of 5 filters per octave. Octaves are defined based

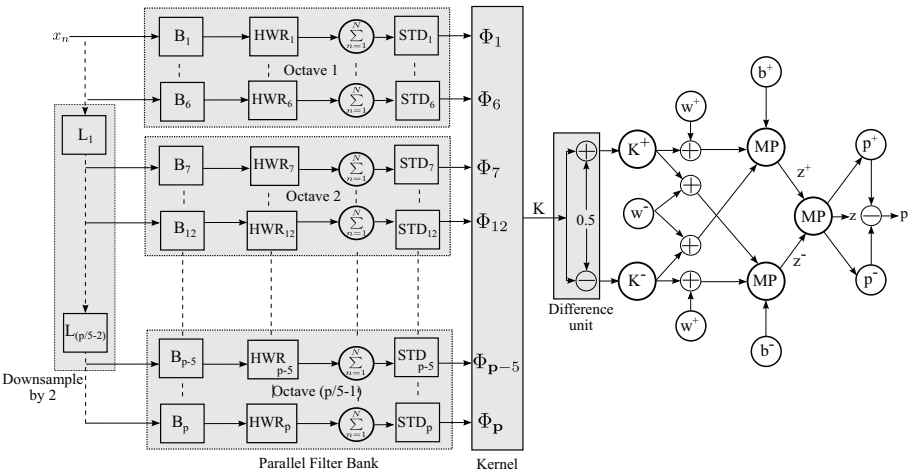

Fig. 2: FIR parallel filter bank framework for MP based classification for $p \times 1$ Kernel. Here, the input x_n is provided to the parallel FIR filter bank to generate a $p \times 1$ kernel. This kernel is used as input for a single layer classification network formed using MP modules. The parallel filter bank and the downsampling low pass filter blocks also use MP modules for computation.

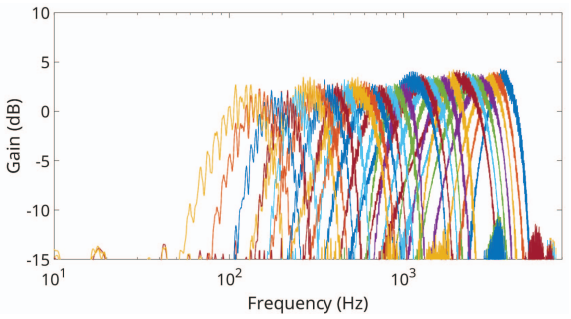

Fig. 3: MP Filter bank output (Gain Response) for chirp signal. This response shows distortion due to MP approximation errors.

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.2. 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 [18]. 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. The additional low pass filter required the same order as the bandpass filter of the filter bank. This down-sampling technique provides an efficient way of implementing an FIR filter bank for low-powered devices.

C. 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 [17], we can express this filtering operation in MP domain as below,

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

For this implementation, we have $\mathbf{h}^+ = \mathbf{h}$, $\mathbf{h}^- = -\mathbf{h}$, $\mathbf{x}^+ = \mathbf{x}$ and $\mathbf{x}^- = -\mathbf{x}$. γ_f is the MP parameter for the filtering operation. Since the property of MP inherently exhibits low pass filtering, based on the reverse water filling algorithm described in [21], 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.3.

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.

III. FPGA Implementation

The proposed design shown in Fig.2 is modeled by Verilog HDL and implemented in Spartan 7 series FPGA, as this

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

Related Work	Mahmoodi, et al. [25]	Cutajar, et al. [26]	Boujelben, et. al. [27]	Ramos-Lara et al. [28]	Nair et al. [5]	This work
Year	2011	2013	2018	2009	2021	2022
FPGA	Virtex4 xc4vsx35	Virtex-II xc2v3000	Artix-7 xc7a100T	Spartan 3 xcs2000	Spartan 7 xc7s6cpga196	Spartan 7 xc7s6cpga196
Operating Frequency	151.286 MHz	42.012 MHz	101.74 MHz	50 MHz	25 MHz	50 MHz[2]
Input Sampling Frequency	NA[1]	16 kHz	6 kHz	8 kHz	16 kHz	16 kHz
Flip Flop	11589	1576	17074	5351	2864	2376
LUTs	9141	11943	16563	6785	1517	1503
RAM (18 Kb)	99	NA[1]	4	NA[1]	0	0
DSP	81	64	87	21	4	0
Power (mW/MHz)	NA[1]	NA[1]	1.12	NA[1]	0.32	0.34
Techniques	SVM	DWT and SVM	MFCC and SVM	FFT and SVM	CAR-IHC IIR and SVM	FIR and Kernel Machine
Datasets	Persian Digits [29]	TIMIT Corpus [30]	Respiratory Sound [27]	Speaker Verification [28]	ESC-10 and FSDD [19]	ESC-10 and FSDD [19]
Average Accuracy (%)	98	61	94	95	88	88

[1] These works did not report this entity for their designs. [2] Maximum operating frequency of the proposed design is 166 MHz.

FPGA caters to edge applications. 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. There are two sections, kernel computation and inference, in the proposed design. Here, 3 MP modules (MP0-2) work simultaneously to compute kernel value and meet the time limit of 3125 clock cycles. The MP0 is used to implement 4 low pass (LP) filters and other two MP modules (MP1-2) are responsible for Band Pass (BP) filtering operation. The internal architecture and working principle of a MP module is described in [17]. 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 are used to store inputs for four LP filters. The multiplexers are used to select one of the registers. The coefficients for the LP filter are stored in ROM (ROM0). 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. II-B). 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 same register banks for the next iteration. The contents of the register banks are used for parallel BP filtering operation and generates kernel function Φ_5 to Φ_{30}.

One single MP module (MP1) is used repeatedly to generate outputs for octave 1. The window size of the BP filter is 16. The coefficients are stored in another ROM (ROM1). The MP2 is used for BP filter outputs of octaves 2,3,4 and 5. Here also, a single MP module is used repeatedly to generate desired outputs. The down sampling of the LP filter outputs provides more time span between two outputs which are the inputs to the BP filters generating octaves 2,3,4 and 5. Hence a single MP (MP2) is sufficient to produce Φ_5 to Φ_{30} in time multiplexed fashion. The output of the BP filters is stored in a Register bank which is the kernel function Φ_1 to Φ_{30} of the proposed design. The coefficients of BP filters for octaves 2, 3, 4, and 5 are stored in another ROM (ROM3).

The inference engine starts working after the completion of kernel computation. The three MP blocks MP3, MP4 and MP5 are used in the inference engine to generate the output p. The architecture and working principle of the inference engine are discussed in Section II-A. The w^+ and w^- are the weight matrix and are stored in a ROM (ROM4). The kernel function Φ and weight matrix w^+ and w^- are the inputs to the inference engine. The high-level block diagram of an MP module and the implementation details of the inference engine have been discussed in [17].

Table I shows the resource utilization and power consumption of our design compared with similar ML-based FPGA implementations. We clearly see advantages of our design in terms of resource and power over other designs. We were able to implement our design with 903 FPGA slices (less than 1K) and the dynamic power consumption is limited to 17 mW for a 50 MHz operating frequency. Table I 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 [5]. Due to multiplierless design, the proposed architecture does not consume any DSPs, whereas the design reported in [5] consumes 4 DSPs. We computed the number of LUTs required to replace 4 DSPs for fair comparisons. We have implemented 4×4, and 8×8 signed multipliers (Baugh Wooley) in FPGA and found that they have consumed 19 and 72 (4× more) LUTs, respectively. The design reported in [5] uses 4 signed multipliers and the dimensions are 20×12, 20×12, 12×12, 16×8 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 [5]. The power consumption (mW/MHz) is

TABLE II: 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
Dog (129/33)	42	90	94	89	90	91	94	91	94
Rain (119/40)	44	86	90	89	87	90	90	88	88
Sea_Waves (200/50)	80	87	90	84	78	89	88	88	88
Crying Baby (144/49)	37	93	84	91	87	92	87	89	88
Clock Tick (114/50)	54	81	76	92	85	85	86	85	84
Person Sneeze (101/44)	49	85	75	87	80	86	80	85	80
Helicopter (197/50)	45	92	88	95	90	88	85	85	86
Chainsaw (99/34)	41	90	85	93	82	92	81	92	80
Rooster (124/54)	40	93	93	93	96	90	94	91	94
Fire Crackling (152/66)	46	93	83	89	87	89	92	90	88

TABLE III: 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
Theo (761/254)	107	96	96	93	91	92	93	92	92
Nicolas(889/297)	15	100	100	98	97	99	99	98	98

almost same for the proposed design and the design presented in [5], 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.

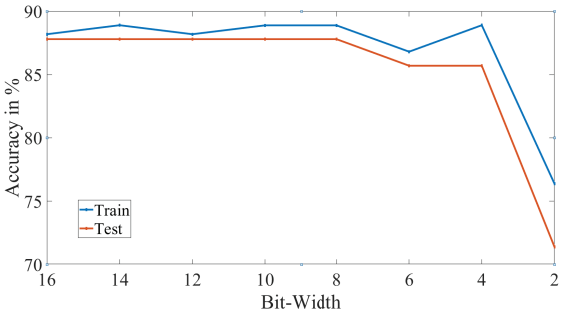

Fig. 4: Impact of bit-width on dataset accuracy for Crying Baby class from ESC-10.

IV. 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 wildlife conservation would result in rare data event detection, Environmental Sounds Classification (ESC-10) dataset [19] would be an ideal dataset to showcase the framework capabilities in case of ecological application. Also, we compared our system with [5] 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 II shows the results for this dataset, having classes like a dog bark, rain, sea waves, crying baby, clock ticking, person sneezing, helicopter, chainsaw, crawing 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 [5], [15]. 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 I.

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.4, the training and testing accuracy remains stable till 8-bit and decreases sharply for bit width lower than 8-bit. We use Keras 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 libraries for quantization.

V. 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. This makes the system highly efficient for deployment in battery-powered edge devices. Furthermore, the framework is tunable to any time series data by tuning the filter parameters in the FIR filter bank. 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.

Acknowledgment

The authors would like to acknowledge that this work was done under the joint IISc-WashU Memorandum of Understanding, to facilitate the collaboration between the two institutions. At IISc, this work was supported in part by the SPARC Funds (SP/MHRD-18- 0006), INAE (SP/INAE-22-2106), Pratiksha Trust and BCD Funds (FG/SMCH-22-2106).

References

[1] 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.

[2] G. W. Witmer, "Wildlife population monitoring: some practical considerations," *Wildlife Research*, vol. 32, no. 3, pp. 259–263, 2005.

[3] A. Zgank, "Iot-based bee swarm activity acoustic classification using deep neural networks," *Sensors*, vol. 21, no. 3, p. 676, 2021.

[4] 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.

[5] A. R. Nair, S. Chakrabartty, and C. S. Thakur, "In-filter computing for designing ultra-light acoustic pattern recognizers," *IEEE Internet of Things Journal*, 2021.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Y. Guo, "A survey on methods and theories of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.

[11] 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.

[12] 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.

[13] 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.

[14] H. Sabbella, A. R. Nair, V. Gumme, S. Yadav, S. Chakrabartty, and C. S. Thakur, "An always-on tinyml acoustic classifier for ecological applications," in *ISCAS*, 2022.

[15] 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.

[16] S. Chakrabartty and G. Cauwenberghs, "Margin propagation and forward decoding in analog vlsi," *A (A)*, vol. 100, p. 5, 2004.

[17] A. R. Nair, P. K. Nath, S. Chakrabartty, and C. S. Thakur, "Multiplierless mp-kernel machine for energy-efficient edge devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.

[18] 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.

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

[20] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[21] M. Gu, *Theory, Synthesis and Implementation of Current-mode CMOS Piecewise-linear Circuits Using Margin Propagation*. Michigan State University, Electrical Engineering, 2012.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] H. Khosravi and E. Kabir, "Introducing a very large dataset of handwritten farsi digits and a study on their varieties," *Pattern recognition letters*, vol. 28, no. 10, pp. 1133–1141, 2007.

[30] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.