MSDF-SVM: Advantage of Most Significant Digit First Arithmetic in FPGA Realization of SVM

***Abstract*—Support vector machine (SVM) is one of the most well-known machine learning algorithms, which is widely used in many areas, including computer vision, speech recognition, and data mining, for classification and regression. Processing high-dimensional datasets, achieving a reasonable accuracy, and preserving suitable response time are considered the most challenging issues with the implementation of SVM. These challenges are aggravated by the rapid growth in the size of today's large datasets, as well as the feature dimension of each data point. To tackle these challenges, this paper exploits the parallelism and digit level pipelining opportunities via FPGA devices and Online arithmetic. In our proposed approach, all the required dot-product operations for computing distances to support vectors and to detect their signs are applied on serially coming data samples. Using serial computation, the area required for each classifier instance and the memory footprint are minimized, which leads to utilizing the maximum parallel instances based on the target devices. Moreover, to achieve lower power consumption, we dynamically terminate the unnecessary computations once they are detected, which occurs in more than 65.6% of cases on average. To the best of our knowledge, MSDF-SVM is the first implementation that employs Online arithmetic effectively to design an accelerator for the SVM. Our evaluation results on two functional real-world applications, i.e., voice gender detection and emotion classification, demonstrate significant improvements in terms of performance and power consumption compared to the previous designs.**

Keywords— Machine Learning, Hardware Accelerator, Support Vector Machine, Computer Arithmetic.

I. INTRODUCTION

Nowadays, machine learning (ML) is playing a significant role in our today's life. ML techniques have been employed to address many problems in various domains ranging from computer science, to finance, agriculture, and medical sciences [1]. Support Vector Machine (SVM) is one of the supervised learning algorithms that is broadly used for classification and regression. SVM has indicated a better performance in terms of accuracy compared to well-known classification algorithms, such as Artificial Neural Networks [2]. The basis of the SVM is the linear classification of data, in which the task is to select the line that has the most reliable margin with data points in each of the classes. The optimal line equation for data is solved by Quadratic Programming methods, which are also known as methods for solving constrained problems [3].

One of the major challenges in this area is the volume of computations required facing today's large-size and high-dimension datasets, as well as the vast amount of data produced daily. To mitigate this severe obstacle, some hardware accelerators have been offered in recent years. Acceleration helps machine learning algorithms run novel applications, carry out time-consuming tasks faster, and improve resource efficiency [4]. In addition, domain-specific hardware acceleration can be beneficial for running machine learning algorithms on battery-powered devices, especially for the SVM, which is broadly employed in wearable healthcare gadgets [5], [6]. Hence, accelerated computing is always needed because of increasing demands for AI-powered applications in modern lifestyles, intelligent applications, and businesses [7].

In this paper, we focus on proposing a hardware accelerator for the SVM machine learning algorithm to accelerate the process of classification using the most digit first (MSDF) arithmetic on FPGA devices. Employing MSDF arithmetic, our proposed design can discard unnecessary computation and exploit the advantage of massive parallelism in FPGAs to speed up the time-consuming classification process in the SVM. Considering the large-size and high-dimensional today's dataset, the importance of our proposed method becomes more significant and evident. Our contribution and novelties can be summarized as (a) designing an FPGA-based hardware accelerator for SVM using MSDF arithmetic, (b) implementing an early termination mechanism to discard unnecessary computations once they are detected, (c) improving the performance of SVM classification, (d) lowering resources usage and energy consumption.

The rest of this paper is as follows. Section II describes the background and literature review of this field. Our proposed design and its details are elaborated in Section III, while Section IV provides two real-world applications as a case study. In Section V, we provide a deep evaluation of our proposed design and the best previous one. Section VI reviews the most related works, and the paper is concluded by Section VII.

II. BACKGROUND AND LITERATURE REVIEW

SVM is one of the well-established algorithms in machine learning in various domains, such as face detection [8], text and hypertext categorization to classify documents into different groups [9], and image classification. SVM provides better search accuracy for image classification compared to the traditional query-based searching techniques and even artificial neural networks in some cases [10]. SVM is also employed in bioinformatics for protein classification, cancer classification, as well as identifying the classes of genes [11]. Besides, SVM has widely been used for recognizing handwritten characters and vehicle classification [12]. SVM uses a technique called kernel trick to convert data; then, based on this conversion, SVM finds the optimal boundary between possible outputs. One of the main pros of the support vector machine method, which makes it very popular, is the ability to solve various problems without losing accuracy by enhancing the size and dimension of the problem.

Whereas the SVM has offered many advantages, including being effective in high dimensional spaces and cases where the number of dimensions is higher than samples, it severely suffers from a computationally intensive nature with a time-complexity of $O(n^3)$. In recent years, some custom kernel functions have been proposed to mitigate this challenge by leveraging the advantage of the versatile capability of SVM. This capability makes it possible to specify custom kernels, such as Sigmoid, Gaussian, Hyperbolic Tangent, and Radial Basis Function (RBF) [13]. The linear kernel is the most frequent type of SVM, which is typically used to deal with linearly separable problems, such as facial expression classification [14] and colorectal cancer detection [15]. The majority of customized kernels, such as sigmoid, RBF, and polynomial, can be implemented using linear equations. These types of the SVM have widely been employed in addressing real-world problems, e.g., linear Sigmoid for language recognition [16], linear RBF polynomial for pedestrian detection [17]. This is the underlying reason that we have focused on mitigating the challenge of the computationally-intensive nature of linear SVM

A. *Support Vector Machine Internals*

The SVM is a supervised classifier that finds hyperplanes in *n*-dimensions space to perform non-overlapping segmentation or regression utilizing all features. It allows a global classification model based on linear discrimination with maximum margin, not considering dependencies between attributes. Intuitively, the hyperplane that best separates data points is the case where the distance to the nearest training data is the largest, as indicated in Fig. 1.

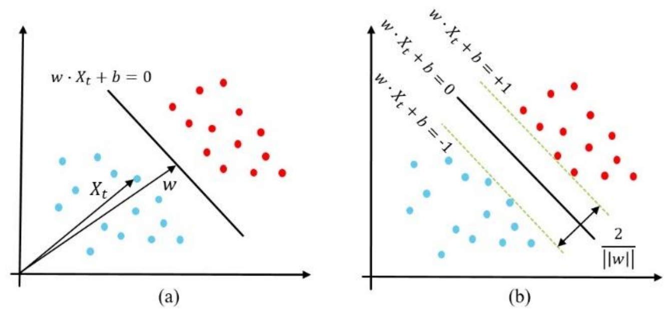


Fig. 1 The procedure of SVM (a) computing distances and (b) determining hyperplanes.

To determine the label (i.e., Y_{X_t}) of test data, Equation 1, or its expanded form in Equation 2, is used. In this equation, if the value of $f(X_t)$ is negative, the Sign function returns the value of -1 as the label; otherwise, if the value of $f(X_t)$ is positive, the label would be +1.

$$Y_{X_t} = \text{Sign}\big(f(X_t)\big) \quad (1)$$

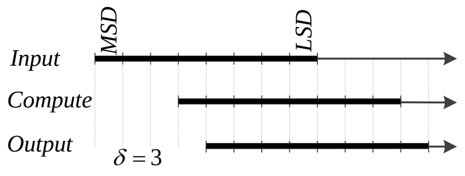
$$f(X_t) = \left(\sum_{i=1}^{n} \big(y_i \alpha_i K(X_t, x_i)\big) + b\right) \quad (2)$$

In Equation 2, x_i is the train data, X_t is the test data, and y_i indicates the label of each train data point. The values of α_i and b are calculated from the training phase and $K(X_t, x_i)$ is the kernel function, which is shown by Equation 3.

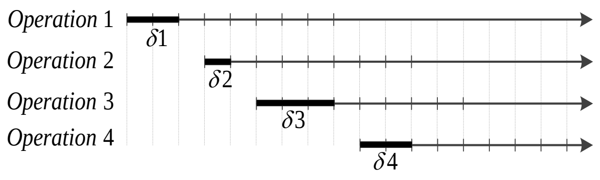
$$K(X_t, x_i) = X_t \cdot x_i \quad (3)$$

B. *Most Significant Digit First Arithmetic*

Online arithmetic is a type of serial computation where the most significant digits are processed first [18]. Indeed, contrary to conventional arithmetic, the result is generated from the right (the least significant positions) to the left (the most significant positions); hence, it is also called Most Significant Digit First (MSDF) arithmetic. In Online arithmetic, the result of the most significant position is made available first. In this fashion, the result digits are produced after receiving a limited number of digits (i.e., online delay δ), while the remaining parts of inputs are coming, as shown in Fig. 2 (a). Therefore, as illustrated in Fig. 2 (b), the execution of even dependent operations can be overlapped. It paves the way for digit-level massive parallelism.



(a) Serial timing with online delay δ



(b) Overlapping the execution of the dependent operations

Fig. 2 The timing diagram of online arithmetic [19].

This capability is due to redundant number representation, allowing several representations of a given value. Due to the serial manner of Online arithmetic, it minimizes the area of the arithmetic unit along with a low memory footprint and negligible interconnects. Additionally, Online arithmetic is well-suited for variable precision computations; once the desired precision is attained or a specific condition is satisfied, the current operation can be terminated, and the next operation starts. It has a significant impact on power consumption and performance [20].

III. PROPOSED DESIGN

This section describes our proposed approach and its implementation in detail. The main goal pursued in our design is to improve the performance, as well as reduce energy consumption and resource utilization compared to state-of-the-art methods.

In Equation 2, y_i and x_i are training samples, and the value of α_i and b are determined in the training phase. Considering Equation 3, it can be re-written in the form of Equation 4.

$$Y_{X_t} = \text{Sign}(wX_t + b) \quad (4)$$

where w indicates the weight and has a fixed value during the test phase for all data points so that it can be computed during the pre-processing phase. Algorithm 1 indicates how to calculate the value of w before running the inference phase and using it to determine the label of new data points. This precomputation paves the way for the real-time processing of linear SVM.

Algorithm 1

```
Input:
    $X[1\ to\ M][1\ to\ dim]$: $Test\ Samples$
    $y[1\ to\ N]$: $Labels\ of\ training\ data$
    $x[1\ to\ N][1\ to\ dim]$: $Training\ data$
    $\alpha[1\ to\ N]$: $Computed\ in\ the\ training\ phase$
    $b$: $Computed\ in\ the\ training\ phase$
Output:
    $Y_{X_t}[1\ to\ M]$
Pre-processing:
  $w[1\ to\ dim]$
  For $i = 1\ to\ dim$
      $Q = 0$
      For $j = 1\ to\ N$
          $Q = Q + y[j] \times x[j][i] \times \alpha[j]$
      End
      $w[i] = Q$
  End
Inference:
  For $i = 1\ to\ M$
      $temp = 0$
      For $j = 1\ to\ dim$
          $temp = temp + X[i][j] \times w[j]$
      End
      $Y_{X_t}[i] = sign(temp + b)$
  End
```

For hardware acceleration of the inference phase, we have used a hardware-software co-design approach that takes advantage of specific custom hardware based on MSDF arithmetic beside the embedded processor of the FPGA fabric. The abstract view of the proposed architecture is indicated in Fig. 3. All the interfacing of the components and the read/write operations from/to the memory are handled by the AXI interconnect. Due to independence between required operations for each sample data in the SVM algorithm to classify a large number of sample data, we can exploit massive parallelism over all the samples where an instance of the classifier is replicated several times. The amount of parallelism is defined by the available resources of the target FPGA and also application constraints.

In this architecture, the dot-product of precomputed value (w) and each sample are calculated, and the classification is done based on the result's sign. As shown in Fig. 3, the value of all dimensions of w are available in parallel, and the value of sample data comes serially from the most significant digit (MSDF). The results of the bit-serial multiplications are added via an adder tree where its input number is equal to the number of the features plus one (i.e., dim = the number of the features and the shifted sum). Due to fast carry chain hardware inside the FPGA fabric, in this design, we use carry-propagate adders at each level of the reduction tree. While in ASIC design, carry-save addition provides the best result.

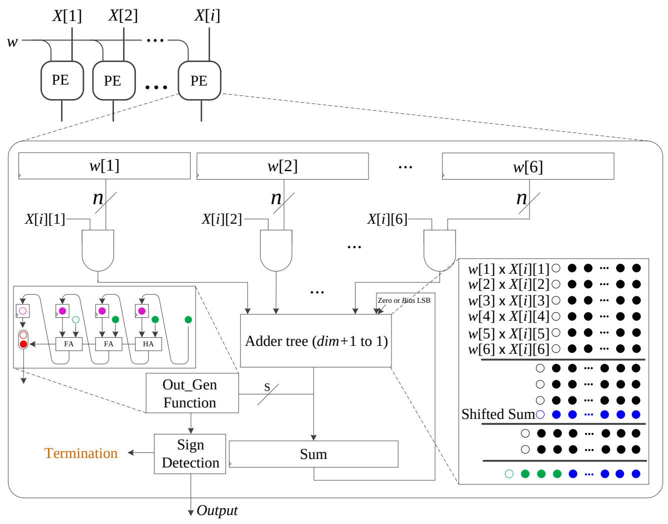


Fig. 3 The abstract view of the MSDF-SVM architecture beside the MSDF dot-product unit architecture with dot-notation clarification (dim = 6, n = precision, s = 4)

The shifted partial sum value, which contains the summation of previous cycles, is inserted in the second stage of the reduction tree. This value is equal to zero in the first cycle, so we use this empty room to add the bias value. This value needs a one-bit sign extension since in each level of reduction, the bit width of intermediate values is increased (i.e., n+1 instead of n in the second level of the reduction tree). The number of levels is dependent on the tree's depth, and finally, the value of the partial sum becomes available after $\lceil \log_2 dim + 1 \rceil$ level. The partial sum is divided into two parts, the most significant part (i.e., s in Fig. 3) goes to Out_Gen Function and participates in generating the dot-product result. Meanwhile, the least significant part is sent to the second level of the reduction tree for the next cycle's computation. As mentioned before, the operand's width in the second level is n+1 with 2's complement representation. On the other hand, the least significant part of the partial sum is a positive value, which is shifted one bit to the left. Therefore, $n - 1$ bits of the least significant part of the partial sum contribute to the result of the next cycle's computation, while it is padded with one "0" in the least significant position (i.e., left shift), and the sign bit is set to "1", due to inverted encoding of Negabits [21]. The bit width of the most significant part is $\lceil \log_2 dim + 1 \rceil + 1$, which feeds Out_Gen Function (i.e., green dots in Fig. 3).

In Out_Gen Function, we take advantage of the Binary Signed-Digit (BSD) redundant number representation to present the output of dot-product. In this representation, the value of each position is defined by two same weight bits, a Posibit (i.e., represented by ●, x, 0, 1 as a dot notation, a variable, a constant zero, and a constant one, respectively) and a Negabit (i.e., represented by ○, x^-, 0^-, 1^- as a dot notation, a variable, a constant zero, and a constant one with the arithmetic value of minus one, respectively) [21]. The digit set of BSD is $\{-1,0,1\}$ where -1, 0, and 1 are represented by 01^-, 00^- (or 11^-), and 10^-. It is worth mentioning that we use Inverted Encoding for Negabit (IEN), in which -1, 0, and 1 are represented by 00^-, 01^- (or 10^-), and 11^-, respectively. In each cycle, the Out_Gen Function's input is accumulated via a carry propagation adder, and the two most significant bits of the result (i.e., a BSD) are considered as the dot-product output. The other bits are formed as a partial remainder for the next cycle's computation, as shown in Fig. 3. The partial remainder is initialized with zero for the first cycle's computation (i.e., purple dots in Fig. 3).

The dot-product output is connected to the Sign Detection unit. In this unit, the leading zero values are ignored, and the classification result is determined by receiving the first non-zero value. Upon receiving the first non-zero value, the label of the classification result (i.e., $+$ or $-$) is sent as the output of the classifier instance since just the sign of dot-product output is enough for classification based on the SVM algorithm. Besides the output, a termination signal is also generated. After raising the termination signal, the classifier instance stops the current computation and starts processing a new data sample. This early termination provides significant performance improvement and meaningfully reduces power and energy consumption. These operations are applied in a pipelined digit serial manner and sweep all the test data points.

To convey a comprehensive understanding of the internal process of MSDF-SVM, we have provided two numerical examples, i.e. Example 1 and Example 2. The former is classified with negative labels and the latter is used to classify positive labels. The value of w, bias, and test data points for the above-mentioned examples are shown in Table I. The results of all intermediate signals during running the MSDF-SVM are shown in Table II. For ease of understanding, the same color as Fig. 3 was used for representing signals. Moreover, a few points should be considered: (1) all Negabits are represented in IEN, (2) All the computations are carried out based on MSDF arithmetic (e.g., in cycle #1, Example 1, the most significant position of test data points are "0", so the value of Serial Input is represented as "000000"), (3) The value of Shifted Sum in cycle #1 is zero, however, instead of zero we use the sign-extended value of bias, and finally (4) There is a possibility for early termination of the process, as shown by these two examples, after three cycles classification is terminated instead of processing all bits of test data points.

IV. MSDF-SVM IN REAL-WORLD APPLICATIONS

We have implemented our designs on two real-world applications of machine learning, i.e., gender detection and emotion classification, as case studies to indicate the effectiveness and performance of our proposed design.

A. *Voice Gender Detection*

Gender detection has widely been utilized in a wide range of real-time applications. Whereas much research has been conducted to improve the accuracy of gender detection, it is still a severe challenge to accurately distinguish genders in images [22] and voices [23].

The human voice is the most effective method for communication, which consists of unique features like gender, age, language, accent, and emotional state. The human ear has a natural analysis system that can automatically detect these features by volume and frequency, but it is a complicated task for computers.

Due to the difference in accents, gender recognition has become more complicated for machines. To address this challenge, sound waves must first be collected using devices like a microphone and converted to digital data. Then, these features can be extracted by signal processing algorithms or deep learning algorithms. Gender detection is usually run on mobile devices that have limited memory volume and power consumption. Besides, gender detection mainly needs real-time processing in industrial applications. MSDF-SVM, was employed to distinguish gender from audio data. Therefore, the speed of the program has been increased by early termination, while the power and memory consumption has also decreased due to the reduction of computation. In experiments, the audio data from [24] dataset consisting of 20 features such as length

TABLE I THE VALUE OF w, $bias$, AND TEST DATA POINTS.

	$w[1]$	$w[2]$	$w[3]$	$w[4]$	$w[5]$	$w[6]$	$\boldsymbol{bias}$	$X[i][1]$	$X[i][2]$	$X[i][3]$	$X[i][4]$	$X[i][5]$	$X[i][6]$
Example 1	$0^{-}0010$	$1^{-}0100$	$0^{-}0110$	$1^{-}0001$	$0^{-}1000$	$0^{-}0001$	$\mathbf{0^{-}0101}$	01011	01000	01100	00101	00111	01101
Example 2	$0^{-}0010$	$0^{-}0100$	$0^{-}0110$	$0^{-}1001$	$0^{-}1000$	$1^{-}1111$	$\mathbf{1^{-}1101}$	10000	11111	10110	11101	10000	00001

TABLE II THE RESULTS OF ALL INTERMEDIATE SIGNALS DURING RUNNING THE MSDF-SVM

# Cycle	Serial Input	$w[1] \times X[i][1]$	$w[2] \times X[i][2]$	$w[3] \times X[i][3]$	$w[4] \times X[i][4]$	$w[5] \times X[i][5]$	$w[6] \times X[i][6]$	Shifted Sum	Partial Sum	S	Out_Gen Function Reg.	CPA Output	z (Output Posibit)	z^{-} (Output Negabit)	Early Termination
								Example 1 (classified as negative label)							
1	000000	$1^{-}0000$	$1^{-}0000$	$1^{-}0000$	$1^{-}0000$	$1^{-}0000$	$1^{-}0000$	$\mathbf{0^{-}10101}$	$0^{-}1110101$	$0^{-}111$	$1^{-}000$	$00^{-}111$	0	1^{-}	0
2	111001	$0^{-}0010$	$1^{-}0100$	$0^{-}0110$	$1^{-}0000$	$1^{-}0000$	$0^{-}0001$	$1^{-}01010$	$0^{-}1100111$	$0^{-}110$	$0^{-}111$	$10^{-}100$	1	0^{-}	0
3	001111	$1^{-}0000$	$1^{-}0000$	$0^{-}0110$	$1^{-}0001$	$0^{-}1000$	$0^{-}0001$	$1^{-}01110$	$0^{-}1101110$	$0^{-}110$	$0^{-}100$	$01^{-}110$	0	0^{-}	1
								Example 2 (classified as positive label)							
1	111110	$1^{-}0010$	$1^{-}0100$	$1^{-}0110$	$1^{-}1001$	$1^{-}1000$	$1^{-}0000$	$\mathbf{1^{-}01101}$	$1^{-}0101010$	$1^{-}010$	$1^{-}000$	$01^{-}010$	0	1^{-}	0
2	010100	$1^{-}0000$	$1^{-}0100$	$1^{-}0000$	$1^{-}1001$	$1^{-}0000$	$1^{-}0000$	$1^{-}10100$	$1^{-}0100001$	$1^{-}010$	$1^{-}010$	$01^{-}110$	0	1^{-}	0
3	011100	$1^{-}0000$	$1^{-}0100$	$1^{-}0110$	$1^{-}1001$	$1^{-}0000$	$1^{-}0000$	$1^{-}00010$	$1^{-}0010101$	$1^{-}001$	$1^{-}110$	$10^{-}101$	1	1^{-}	1

of the signal, minimum and maximum dominant frequency measured across the acoustic signal, the standard deviation of frequency, and median frequency was used.

B. *Emotion Classification*

Another case study we have considered to implement the proposed design is emotion classification due to its high functions in various domains. Emotions have significant and undeniable effects on people's lives. Over the recent years, although plenty of research has been done to recognize visual semantics, there are still many open problems and challenges regarding recognizing visual emotion due to the higher levels of abstraction that exist in emotions compared to visual semantics [25]. Since emotion detection is a multi-emotion analysis model [26], it can be computationally intensive for running on embedded systems and edge devices, which have limited resources in terms of power supply and processing elements. We have employed our SVM accelerator to perform the task of emotion classification using [27] dataset of electroencephalography (EEG). The datasets include about two thousand features, and the main features for this classification are the Signal total energy, relative energy of Alpha, Beta, and Gamma band, EEG Channels (1-128), front, central, and back signal of the brain, as well as electrooculogram signals. The results of these experiments were used to conduct a real-world evaluation and fair comparison.

V. EVALUATION AND COMPARISON

This section evaluates our proposed design and compares it to the best state-of-the-art- methods. In order to indicate the effectiveness of our proposed design, we have performed several experiments and evaluations. Our design was implemented on Zynq UltraScale+ XCZU7EV MPSoC Xilinx device. Besides, we have included the best of previous designs in our experiments to make a valid and fair comparison under the same conditions and resources.

A. *Experimental Setup*

We select the best state-of-the-art implementation of linear SVM on FPGA [28] (i.e., SVM-Parallel). This implementation encounters a challenge in processing high-dimensional datasets, where the required hardware exceeds available FPGA resources. For instance, in the Emotions dataset [27], which has more than two thousand features, the required hardware resources for SVM-Parallel implementation pass the available budget. This is the main reason that the figures for *Speedup* and *Energy Consumption* for this design are not reported in some parts of the evaluation. To tackle this obstacle, we took advantage of time multiplexing and hardware re-using techniques in SVM-MAC, where the required hardware resources are sufficient. After implementing our proposed and SVM-Parallel using VHDL, we selected SVM-MAC as the baseline model and two different datasets, including Voices [24] and Emotions [27], for conducting a fair and valid comparison. The properties and characters of these datasets are reported in Table III.

TABLE III THE PROPERTIES OF DATASETS USED FOR THE EVALUATION

Dataset	# Features	#Test Samples	#Training
Voice	20	634	2536
Emotions	2509	284	1136

B. *Comparison of Dot-Product Module*

Dot-Product is one of the main modules in our architecture for the SVM. In order to indicate the efficiency of this module, we have compared our proposed architecture for computing the cost of dot-product (DP) operation (shown in Fig. 3) with the state-of-the-art serial DP circuit presented in [29]. This design consists of a barrel shifter for shifting partial products in a multiplication operation, so there is a constant cost for shifting even for datasets with a small number of features. Fig. 4 illustrates the number of LookUp Table (LUT) for each design.

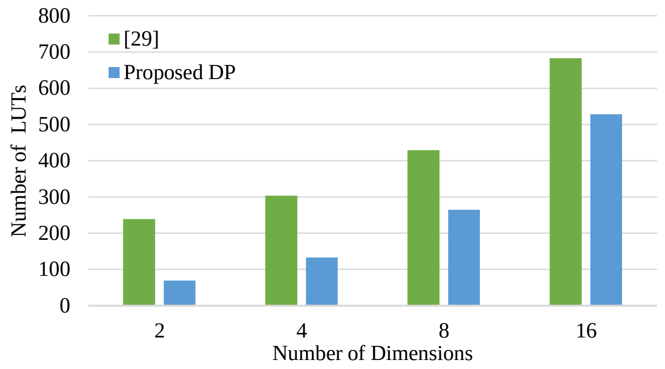


Fig. 4 The comparison of the number of LUTs of dot-product modules for various dimensions.

As shown in Fig. 4, our proposed DP has consumed a lower number of LUTs compared to the state-of-the-art DP design for various dimensions. Another important aspect in our comparison for evaluation is the number of FFs. Fig 5. depicts the number of FFs in our proposed architecture compared to the design presented in [29].

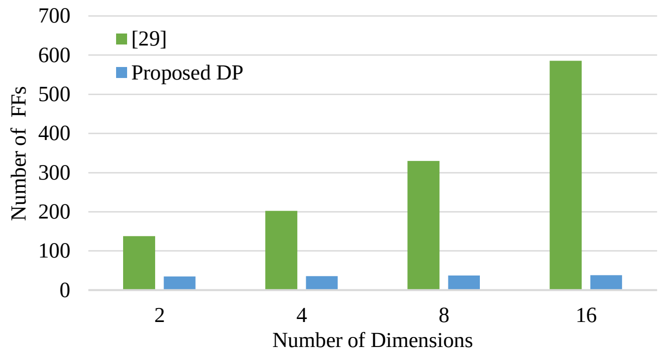


Fig. 5 The comparison of the number of FFs of dot-product modules for various dimensions.

As shown in Fig. 5, the number of FFs in [29] has grown drastically compared to our design; consequently, the number of FFs in our design is lower for different numbers of dimensions.

C. *Comparison of Resource Utilization*

Resource efficiency is essential aspect of our design. A comparison between our proposed design and other related works regarding resource usage has been reported in Table IV to indicate the improvement in our design.

TABLE IV COMPARISON BETWEEN OUR PROPOSED METHODS AND PREVIOUS RELATED WORKS

Design	Dataset	#LUT	#FF	#PE
SVM-MSDF	Voice	132172	7640	191
	Emotions	132520	47	1
SVM-MAC	Voice	132264	2816	88
	Emotions	132264	2816	88
SVM-Parallel Tree	Voice	150015	160	5
	Emotions	-	-	-

As indicated in Table IV, our proposed design consumed the lowest resources in terms of the number of LUT, and the number of Flip-Flops (FF) compared to the other designs running both datasets. This improvement would be more critical when the SVM classifier needs to be run on edge devices and portable devices with limited resources and areas.

D. *Comparison of Speedup*

Speedup is the most important factor in our evaluation since the main aim of our proposed design was to improve the speed of SVM classification, especially when it faces novel large-size datasets. We have measured the amount of *Speedup* for our design by running two Voice and Emotions datasets. The results of this comparison are reported in Fig. 6.

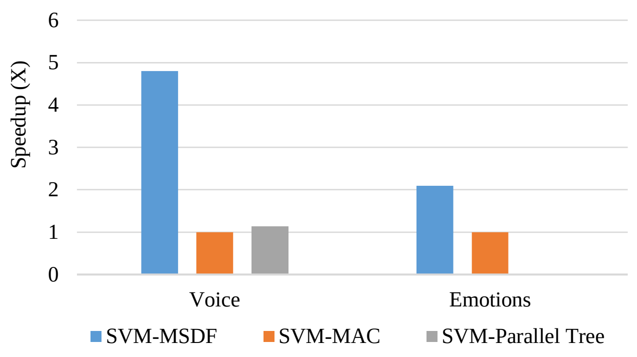


Fig. 6 Comparison of *Speedup* between the proposed design and previous related works.

As illustrated by Fig. 6, our proposed design was placed at the first rank compared to the state-of-the-art related design on both Voice and Emotions datasets. In the gender detection problem, our design indicated 5X improvements in *Speedup* against the SVM-MAC and SVM-Parallel Tree. This improvement was 2X in solving the problem of emotions classification. Also, as mentioned before, implementing SVM-Parallel was not possible for this experiment on the Emotions dataset.

E. *Comparison of Energy Consumption*

Another aspect of comparison in our paper is *Energy Consumption*, which is highly important for modern smart and portable devices. We have measured the improvements attained for each design against the base implementation. Fig. 7 illustrates the results of this evaluation.

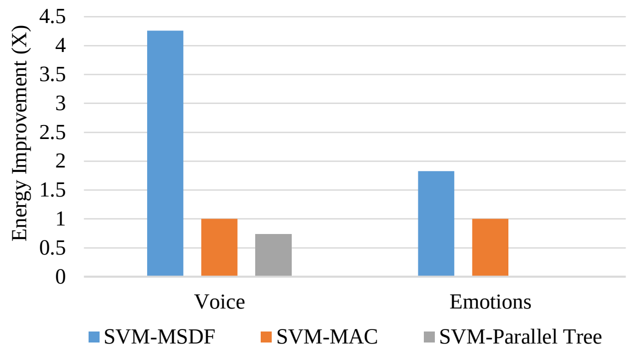


Fig. 7 Comparison of improvement in terms of *Energy Consumption* for each design.

As indicated in Fig. 7, our design has gained more improvements in terms of *Energy Consumption* compared to the state-of-the-art design. It is worth mentioning that the static power consumption had a constant value of 0.594 for all designs during the experiments. It can be inferred that our design had better performance in terms of *Speedup*, *Resource Utilization*, and *Energy Consumption* compared to other related designs in both experiments of voice gender detection and EEG emotion classification.

VI. RELATED WORKS

Machine learning algorithms have been widely employed to address and solve cumbersome challenges in various domains. Due to this high-level functionality, plenty of research has been done to improve the performance of ML algorithms, especially when a large number of computations is required facing large datasets. These works can be categorized from various perspectives. Here we have provided a categorization based on the type of architecture design.

A. *Available Architecture Types*

This subsection presents some of the most important and frequent architecture designs used for developing hardware accelerators.

(a) Parallel Pipelined Architectures: the main aim of the pipelining technique is to concurrent execution of different tasks. So, the main task needs to be split into multi subtasks. Whereas using pipeline techniques can remarkably hasten the performance, there is always a restriction fully parallelizing a task due to the existence of sequential parts and read-after-write (RAW) hazard. This technique has broadly been implemented on FPGAs to design new hardware accelerates for many applications in various domains. For instance, Qasaimeh *et al.* [30] have proposed a parallel pipelined design for real-time image classification using SVM. Their design was implemented on a Xilinx CORDIC IP and indicated an Accuracy up to 85%, while a *Speedup* of 5.7X was obtained against software implementation.

(b) Systolic Array Architectures: a systolic array technique takes advantage of both pipelining and parallelism approaches simultaneously, which can be used as a PE. This architecture can significantly reduce the complexity of hardware and enhance its scalability. The systolic array design has widely been used to accelerate the computation in many applications, especially where the matrix multiplication computations are required, such as fully connected layers in deep neural networks. Kyrkou and Theocharides [31] have suggested a systolic array able to perform vector and scalar processing for object detection. In three case study scenarios, their design obtained a performance of up to 122 fps and an Accuracy of up to 78% during the evaluation experiments.

(c) Multiplier-less Architectures: another architecture that was introduced to design accelerators for the SVM classifier is multiplier-less architecture. This architecture tries to mitigate computationally intensive multiplication operations in order to decrease the complexity of hardware and enhance the *Speedup* and *Resource Efficiency*. Anguita *et al.* [32] proposed a multiplier-free design for the SVM with a Gaussian kernel function. This design provided a desirable classification performance while less complex hardware was utilized.

(d) Cascaded Classification-Based Architectures: this type of hardware accelerator architecture allows multiple simple classifiers to be run on a cascading design in order to increase the speed of classification. This design suites FPGA-based implementation with a low hardware cost for the SVM classifier. This architecture provides higher classification accuracy and *Speedup* compared to a standalone SVM classifier. Also, this architecture can be deployed for real-time function on embedded systems. In this category, Kyrkou *et al.* [33] have presented cascaded architecture hardware, in which a reduction method was employed to decrease power consumption and area. Their implemented design for SVM demonstrated performance of about 70 fps, on average, as well as a *Speedup* of 5X against a single parallel SVM classifier.

B. *Previous Real-world Applications*

This subsection provides an in-detailed description and discussion of real-world applications that have employed one of the previously mentioned designs to develop hardware accelerators for the SVM classifier.

Song *et al.* [28] proposed an FPGA-based parallel implementation for SVM. Their proposed implementation accelerates the classification process of SVM to be used in Smart Grid systems to establish a secure two-way data transmission channel. The authors' experiments indicated an improvement in performance while a high level of accuracy was preserved.

Martins *et al.* [34] proposed a hardware accelerator to hasten classifying hyperspectral Images. These images are highly used in urban development management and ground recognition. This approach leverage the Entropy Multiple Correlation Ratio process in the training phase for feature selection. Then, in the classification step, which is the most computationally expensive part, they have suggested a custom processor based on FPGA to run an SVM classifier based on Hamming Distance function. The authors' evaluation indicates that their proposed accelerator, with an accuracy of 99.7% for correct classification, can surpass the state-of-the-art related works in terms of hardware cost and performance of real-time classification.

Ranawaka *et al.* [35] suggested an FPGA-based hardware accelerator for enhancing the performance of running SVM for object detection using the Histogram of Oriented Gradients. The authors' proposed method mitigates the computation load of this algorithm and leverages a based around Block RAM (BRAM) and deep pipelining technique. The authors have implemented their proposed design on Zynq 7000 FPGA with the ARM CPU using AXI memory interfaces. In this design, bare-metal device drivers encapsulate the behavior of the accelerator as a hardware thread to the applications running on the CPU. The paper has claimed that the customized processor has obtained better performance and power consumption compared to previous related designs with a throughput of 240fps on HD images.

Whereas many related works, including [28] and [30-35], have tried to deal with the challenge of computation load of SVM, specifically in real-time applications, their proposed design is limited to application-specific architectures for hardware acceleration. In contrast, our proposed design can accelerate the SVM classifier using the same type of hardware for generic domains. It means our proposed design can be employed generically in a wide range of applications. To indicate this important advantage, we used our system to classify data from two different domains, i.e., emotion classification using EEG data and gender detection based on human voices.

VII. CONCLUSION

Support vector machine is one of the most well-established machine learning algorithms, which has widely been employed to address classification and regression problems in a wide range of applications. In this paper, we proposed a novel hardware accelerator for the SVM to mitigate its computationally-intensive nature and improve its performance and efficiency in facing huge datasets. Our proposed approach leveraged Online arithmetic and massive parallelism via digit-level pipelining to allow an early termination mechanism, which can early detect and neglect unnecessary computations. Our design was implemented on an FPGA using VHDL, where two real-world applications, i.e., voice gender detection and emotion classification, were employed for evaluating the proposed design. Our proposed design significantly improved the performance of the SVM in terms of *Speedup*, *Resource Utilization*, and *Energy Consumption*. The results of the comparison also proved that our proposed approach could surpass the related works and the state-of-the-art designs. Considering the sheer amount of data produced daily, the importance of our proposed design becomes more significant. Also, our proposed accelerator can be effectively deployed in various scenarios where the SVM faces large-size datasets or has to be run on resource-limited devices, like wearable equipment and edge devices.

REFERENCES

[1] M. Ahmadi, A. Taghavirashidizadeh, D. Javaheri, A. Masoumian, S. Jafarzadeh Ghoushchi, and Y. Pourasad, "DQRE-SCnet: A novel hybrid approach for selecting users in Federated Learning with Deep-Q-Reinforcement Learning based on Spectral Clustering," *J. King Saud Univ. - Comput. Inf. Sci.*, 2021, doi: https://doi.org/10.1016/j.jksuci.2021.08.019.

[2] S. Chidambaram and K. G. Srinivasagan, “Performance evaluation of support vector machine classification approaches in data mining,” *Cluster Comput.*, vol. 22, no. 1, pp. 189–196, 2019, doi: 10.1007/s10586-018-2036-z.

[3] C. A. Floudas and V. Visweswaran, “Quadratic Optimization BT - Handbook of Global Optimization,” R. Horst and P. M. Pardalos, Eds. Boston, MA: Springer US, 1995, pp. 217–269.

[4] K. Givaki *et al.*, “Using Residue Number Systems to Accelerate Deterministic Bit-stream Multiplication,” in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2019, vol. 2160–052X, p. 40, doi: 10.1109/ASAP.2019.00-33.

[5] M. Kee and G.-H. Park, “A Low-Power Programmable Machine Learning Hardware Accelerator Design for Intelligent Edge Devices,” *ACM Trans. Des. Autom. Electron. Syst.*, Apr. 2022, doi: 10.1145/3531479.

[6] G. Bhat, Y. Tuncel, S. An, H. G. Lee, and U. Y. Ogras, “An Ultra-Low Energy Human Activity Recognition Accelerator for Wearable Health Applications,” *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, Oct. 2019, doi: 10.1145/3358175.

[7] H. Park and S. Kim, “Chapter Three - Hardware accelerator systems for artificial intelligence and machine learning,” in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, vol. 122, S. Kim and G. C. B. T.-A. in C. Deka, Eds. Elsevier, 2021, pp. 51–95.

[8] S. Kumar, S. Singh, and J. Kumar, “Multiple Face Detection Using Hybrid Features with SVM Classifier BT - Data and Communication Networks,” 2019, pp. 253–265.

[9] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, vol. 408, pp. 189–215, 2020, doi: https://doi.org/10.1016/j.neucom.2019.10.118.

[10] A. A. A. Ali and S. Mallaiah, “Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout,” *J. King Saud Univ. - Comput. Inf. Sci.*, 2021, doi: https://doi.org/10.1016/j.jksuci.2021.01.012.

[11] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, and W. Xu, “Applications of Support Vector Machine (SVM) Learning in Cancer Genomics,” *Cancer Genomics Proteomics*, vol. 15, no. 1, pp. 41–51, 2018, doi: 10.21873/cgp.20063.

[12] H. Fu, H. Ma, Y. Liu, and D. Lu, “A vehicle classification system based on hierarchical multi-SVMs in crowded traffic scenes,” *Neurocomputing*, vol. 211, pp. 182–190, 2016, doi: https://doi.org/10.1016/j.neucom.2015.12.134.

[13] S. Afifi, H. GholamHosseini, and R. Sinha, “FPGA Implementations of SVM Classifiers: A Review,” *SN Comput. Sci.*, vol. 1, no. 3, p. 133, 2020, doi: 10.1007/s42979-020-00128-9.

[14] S. Saurav, S. Singh, R. Saini, and A. K. Saini, “Hardware Accelerator for Facial Expression Classification Using Linear SVM BT - Advances in Signal Processing and Intelligent Recognition Systems,” 2016, pp. 39–50.

[15] T. Koide *et al.*, “FPGA implementation of type identifier for colorectal endoscopie images with NBI magnification,” in *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2014, pp. 651–654, doi: 10.1109/APCCAS.2014.7032865.

[16] Z. Nie, X. Zhang, and Z. Yang, “An FPGA Implementation of Multi-Class Support Vector Machine Classifier Based on Posterior Probability,” 2012.

[17] M. Berberich and K. Doll, “Highly Flexible FPGA-Architecture of a Support Vector Machine,” in *45. MPC-Workshop, Albstadt-Sigmaringen*, 2014, no. 45, pp. 25–32, [Online]. Available: https://www.mpc-gruppe.de/workshopbaende.

[18] M. D. Ercegovac, “On left-to-right arithmetic,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 750–754, doi: 10.1109/ACSSC.2017.8335445.

[19] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. USA: Elsevier Science Inc., 2004.

[20] M. D. Ercegovac, “On Reducing Module Activities in Online Arithmetic Operations,” in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 524–528, doi: 10.1109/IEEECONF51394.2020.9443576.

[21] G. Jaberipur, “Redundant number system-based arithmetic circuits,” in *Arithmetic Circuits for DSP Applications*, John Wiley & Sons, Ltd, 2017, pp. 273–312.

[22] R. Ranjan, V. M. Patel, and R. Chellappa, “HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, 2019, doi: 10.1109/TPAMI.2017.2781233.

[23] A. Lausen and A. Schacht, “Gender Differences in the Recognition of Vocal Emotions,” *Front. Psychol.*, vol. 9, 2018, doi: 10.3389/fpsyg.2018.00882.

[24] I. E. Livieris, E. Pintelas, and P. Pintelas, “Gender Recognition by Voice Using an Improved Self-Labeled Algorithm,” *Mach. Learn. Knowl. Extr.*, vol. 1, no. 1, pp. 492–503, 2019, doi: 10.3390/make1010030.

[25] H. Zhang and M. Xu, “Weakly Supervised Emotion Intensity Prediction for Recognition of Emotions in Images,” *IEEE Trans. Multimed.*, vol. 23, pp. 2033–2044, 2021, doi: 10.1109/TMM.2020.3007352.

[26] X. Zeng, Q. Chen, X. Fu, and J. Zuo, “Emotion Wheel Attention-Based Emotion Distribution Learning,” *IEEE Access*, vol. 9, pp. 153360–153370, 2021, doi: 10.1109/ACCESS.2021.3119464.

[27] J. J. Bird, D. R. Faria, L. J. Manso, A. Ekárt, and C. D. Buckingham, “A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction,” *Complexity*, vol. 2019, p. 4316548, 2019, doi: 10.1155/2019/4316548.

[28] X. Song, H. Wang, and L. Wang, “FPGA Implementation of a Support Vector Machine Based Classification System and Its Potential Application in Smart Grid,” in *2014 11th International Conference on Information Technology: New Generations*, 2014, pp. 397–402, doi: 10.1109/ITNG.2014.45.

[29] J. Albericio *et al.*, “Bit-Pragmatic Deep Neural Network Computing,” in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 382–394.

[30] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, “FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification,” *IEEE Trans. Comput. Imaging*, vol. 1, no. 1, pp. 56–70, 2015, doi: 10.1109/TCI.2015.2424077.

[31] C. Kyrkou and T. Theocharides, “A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines,” *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 831–842, 2012, doi: 10.1109/TC.2011.113.

[32] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, “Feed-Forward Support Vector Machine Without Multipliers,” *IEEE Trans. Neural Networks*, vol. 17, no. 5, pp. 1328–1331, 2006, doi: 10.1109/TNN.2006.877537.

[33] C. Kyrkou, T. Theocharides, and C.-S. Bouganis, “An embedded hardware-efficient architecture for real-time cascade Support Vector Machine classification,” in *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2013, pp. 129–136, doi: 10.1109/SAMOS.2013.6621115.

[34] L. A. Martins, G. A. M. Sborz, F. Viel, and C. A. Zeferino, “An SVM-Based Hardware Accelerator for Onboard Classification of Hyperspectral Images,” 2019, doi: 10.1145/3338852.3339869.

[35] P. Ranawaka, M. Ekpanyapong, A. Tavares, J. Cabral, K. Athikulwongse, and V. Silva, “Application Specific Architecture for Hardware Accelerating HOG-SVM to Achieve High Throughput on HD Frames,” in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2019, vol. 2160–052X, pp. 131–134, doi: 10.1109/ASAP.2019.00-18.