electronics

MDPI

Article

Approximate Computing Circuits for Embedded Tactile Data Processing

Mario Osta [1,*], Ali Ibrahim [1,2] and Maurizio Valle [1]

[1] Department of Electrical, Electronic and Telecommunications Engineering and Naval Architecture, University of Genova, 16132 Genoa, Italy; ali.ibrahim@edu.unige.it (A.I.); maurizio.valle@unige.it (M.V.)
[2] Department of Electrical and Electronics Engineering, Lebanese International University, Beirut 14404, Lebanon
* Correspondence: mario.osta@edu.unige.it

Abstract: In this paper, we demonstrate the feasibility and efficiency of approximate computing techniques (ACTs) in the embedded Support Vector Machine (SVM) tensorial kernel circuit implementation in tactile sensing systems. Improving the performance of the embedded SVM in terms of power, area, and delay can be achieved by implementing approximate multipliers in the SVD. Singular Value Decomposition (SVD) is the main computational bottleneck of the tensorial kernel approach; since digital multipliers are extensively used in SVD implementation, we aim to optimize the implementation of the multiplier circuit. We present the implementation of the approximate SVD circuit based on the Approximate Baugh-Wooley (Approx-BW) multiplier. The approximate SVD achieves an energy consumption reduction of up to 16% at the cost of a Mean Relative Error decrease (MRE) of less than 5%. We assess the impact of the approximate SVD on the accuracy of the classification; showing that approximate SVD increases the Error rate (Err) within a range of one to eight percent. Besides, we propose a hybrid evaluation test approach that consists of implementing three different approximate SVD circuits having different numbers of approximated Least Significant Bits (LSBs). The results show that energy consumption is reduced by more than five percent with the same accuracy loss.

Keywords: approximate computing; digital multiplier; Singular Value Decomposition; embedded machine learning; tensorial kernel; tactile data processing; FPGA

Citation: Osta, M.; Ibrahim, A.; Valle, M. Approximate Computing Circuits for Embedded Tactile Data Processing. *Electronics* **2022**, *11*, 190. https://doi.org/10.3390/electronics11020190

Academic Editor: Veljko Pejovic

Received: 27 September 2021
Accepted: 1 January 2022
Published: 8 January 2022

1. Introduction

Embedding Machine Learning (ML) near the sensor is increasingly required for many application domains such as wearables [1], health care devices [2], and tactile sensing systems [3]. A tactile sensing system is composed of three main blocks: (1) a tactile sensor array that senses the mechanical input stimuli; (2) an interface electronic system for signal conditioning and data acquisition; (3) an embedded electronic system (EES) for digital signal processing. In particular, the EES extracts meaningful information from raw data [4]. ML algorithms provide effective solutions for nonlinear and complex problems through a "learning by examples" approach. Such methods are employed to design predictive systems that can make decisions on unseen input samples [5,6]. ML methods have been investigated to extract structured information from raw data, e.g., texture/touch modality classification. A tensorial kernel ML approach has proven its effectiveness in processing tactile data [3]. Despite the accuracy of the tensorial kernel approach [4], its high computational load [7] imposes high energy consumption. The computational bottleneck of the tensorial kernel approach lies in the SVD implementation.

On the other hand, SVD plays a significant role in ML algorithms: dimensionality reduction methods are required for ML algorithms with large size datasets. State-of-the-art studies report the usage of SVD in ML algorithms to extract low-dimensional

features [8–10]. The focus of this paper remains on improving the energy consumption of the SVD implementation by exploiting energy-efficient circuit design techniques. Therefore, the solution presented in this paper could be exploited in various embedded ML algorithms [8–10]. Our main focus lies in implementing an energy-efficient SVD for tactile sensing systems. To this aim, the approximate computing paradigm is exploited in this paper.

In recent years, approximate computing techniques (ACTs) have attracted much research since they may enhance the energy efficiency of embedded digital systems. ACTs target error-resilient applications to trade accuracy for power consumption, time latency, and hardware size [11,12]. Approximate computations are effective in many applications, e.g., image processing [13], data mining [14], and tactile data processing [15], where there is a possibility to take advantage of energy reduction with minimal loss of accuracy [16]. ACTs could be considered as a potential solution to reduce the computational cost of our targeted SVD implementation in the tensorial kernel approach. In this perspective, this paper aims to answer the following questions:

(1) What could be the impact of ACTs on the performance of the SVD?
(2) What could be the impact of ACTs in tactile data processing tasks?

Since digital multipliers are extensively used in SVD circuit implementation, we aim to optimize the implementation of the multiplier circuit by adopting ACTs. Most of the research works have focused on applying ACTs to the SVD at the algorithmic level [17–19] but till now, no research works have investigated the usage of ACTs in the SVD circuit. The main contribution in this work lies in implementing an efficient approximate SVD based on an approximate multiplier. In this perspective, the research objective of this paper is to demonstrate that approximate multipliers could be effectively used to decrease the energy consumption of the SVD and reduce consecutively the energy consumption of the embedded SVM-based tensorial kernel circuit implementation [4]. We systematically analyze the energy–accuracy trade-offs offered by different approximations in the tensorial SVM circuit. In this work, we have demonstrated the applicability of ACTs for embedded ML algorithms in tactile sensing systems at the circuit level. The main contributions of this work are summarized as follow:

1. We assess the performance of the Approx-BW multiplier with respect to the approximate multipliers presented in the state-of-the-art. The work presented in this paper is an extension of the work proposed in [20] where an extensive comparison among the BW multiplier with different state-of-the-art multipliers has been addressed. The results in [20] have shown that the Approx-BW multiplier achieves power consumption reduction up to 60% with respect to a Rounding based approximate multiplier (ROBA) [21] and multiplier based on inexact ETA adder (META) [22] multipliers with degradation of MRE of less than 4%.
2. We propose the implementation of the approximate SVD circuit based on the Approx-BW multiplier [20]. The approximate SVD circuit shows a reduction of energy consumption by up to 16% at the cost of an MRE increase of less than 5%.
3. We analyze the impact of the approximate SVD on the accuracy of the classification in a case study, i.e., classification of two touch modalities (sliding a finger vs. rolling a washer). We show that the Error increases from 1% to less than 8% when using approximate SVD circuits. We show that energy consumption could be reduced by more than 5% at the same accuracy loss when applying a hybrid approach, which consists of implementing three different approximate SVD having different numbers of approximated Least Significant Bits (LSBs).

2. Related Works

State-of-the-art works have addressed the hardware implementation of ML using different methods and approaches. Most of the hardware implementations are based on software implementation [23] or custom digital circuits [24]. Different state of the art works [23,25–27] have presented the hardware implementation of various embedded ML

algorithms (Extreme learning machine (ELM), Neural networks, etc.) on FPGA devices for various applications (e.g., epileptic seizure detection).

Few works have focused on implementing embedded ML algorithms based on SVM at the hardware level. To give some examples, Ortega and Anguita [28] presented the implementation of an efficient support vector machine (SVM) on FPGA for online classification applications. Fafoutis et al. [29] investigated the benefits of embedded ML algorithms for extracting features for wearable sensor devices. An energy-efficient SVM classifier using approximate computing is presented in [30]. An approximate adder is proposed, then an approximate fixed-width multiplier is introduced. The proposed circuits have been evaluated on the architecture of the support vector machine (SVM) classifier. Results have shown that the use of approximate computing can reduce the area and critical path delay, and the power delay product (PDP), by 18.7, 16.0, and 32.4%, respectively, compared to the exact classifier. Another energy-efficient-based approximate computing SVM classifier is proposed in [31]. The authors have proposed a hardware architecture with reconfigurable kernels where the optimum energy is achieved by choosing and configuring different kernels. Results show up to 15% energy and 14% area savings when compared to exact SVM implementation. An optimized approximate SVM FPGA accelerator is presented in [32]. The authors have proposed an approach by applying two algorithmic approximation techniques: precision scaling and loop perforation. Results show that the approximate SVM classifier achieves a speedup of 15× while preserving accuracy of 96.7%.

Ibrahim et al. [7] implemented a real-time SVM classifier based on the tensorial kernel approach [4] on FPGA, to distinguish three touch modalities classification: sliding a finger, rolling a washer, and brushing a brusher, which is three binary classification problems. The tensorial kernel SVM consumes a significant amount of power consumption, i.e., 1.14 W. The importance of this approach is that it maps the variation of the stimuli generated from the two-dimensional sensor array in terms of time. Most of the ML approaches deal with input vectors, and they are not able to handle tensorial inputs. The approach of [7] preserves the inherent tensorial structure of the signals provided by the sensing device. However, the main drawback of [7] is the high energy consumed.

To improve the energy consumption of embedded ML algorithms, some works in the state-of-the-art demonstrated the effectiveness of some low power techniques and ACTs approaches in embedded ML algorithms implementations. In [33], The authors have used the All Spin Logic Device (ASLD), which is a spintronic device that provides the properties of small area, zero leakage, and low operating voltage. Presented results have shown that ASLD has identical power dissipation through the switching operations. The authors in [34] have used silicon nanowire field-effect transistors (SiNW FETs) to preserve lower power and area consumption when compared to conventional CMOS technology. Ibrahim et al. [28] presented a survey showing the main techniques adopted at the circuit level for embedded ML algorithms such as approximate arithmetic circuits, approximate memory, and quantization techniques. Particularly, researchers have focused on introducing approximate multipliers in the embedded ML implementation since the latter requires a high number of multiplications. Several approximate multipliers for ML algorithms have been proposed in the literature [14,35–38]. Reference [36] evaluated the use of an Alphabet Set Multiplier (ASM) in a deep neural network: the conventional multiplication is substituted by simplified shift and add operations [36]. The power consumption is reduced by 18% to 27% at the cost of an accuracy loss of less than 0.4%. In [37], the energy efficiency is improved by 43.9 to 62.5% after implementing the approximate multiplier using the inexact logic minimization approach in a neural network. Hammad et al. [39] demonstrated that approximate multipliers could improve the performance of the VGGNet deep learning network. They showed that the cost of the multiplications in VGGNet could be reduced by 50% in terms of energy consumption with a mean relative error (MRE) of 1.5%.

Inspired by the approach adopted in [39], we have applied ACTs in our previous work [7] by implementing approximate multipliers in the tensorial SVM. The main differences in our work compared with [39] are: (1) Ref. [39] employs approximate multipliers

from the state-of-the-art, while in this work we use the approximate multiplier proposed in our recent work [20]; (2) In [39], the MRE of the approximate multipliers is estimated, while in our work the approximate multiplier is implemented directly in the SVD, (3) the study in [39] is applied for image classification, while in our work the study is applied for touch modalities classification.

3. Machine Learning-Based Tensorial Kernel Approach

3.1. General Approach

The tensorial kernel approach proposed in [40] is implemented in two stages: offline training and online classification. During the offline training, data are used to build the classifier. In the online classification, the system classifies the touch modalities. Figure 1 shows a sketch of the tensorial kernel approach. The tactile data generated from the sensor array are arranged in a three-dimensional tensorial representation. The first two dimensions represent a 4×4 sensor array, while the third dimension represents the time. This work employs the Support Vector Machine for the classification [4]:

$$\hat{y} = f_{SVM}(i_n) = \sum_{m=1}^{N_p} \beta_m \, K(i_m, i_n) + b \tag{1}$$

where i_n, $\hat{y}$, β_m, and $K(i_m, i_n)$ are respectively the input, the predicted category, the weights, and the kernel function. The tensorial kernel extended from the Gaussian kernel is defined and computed as follow:

$$K(i,j) = \prod_{n=1}^{N} K^n(i,j) \tag{2}$$

where K^n is the kernel factor defined as:

$$K(i,j) = \exp\left(\frac{-1}{2\sigma^2} \| V_i V_i^T - V_j V_j^T \|_F^2\right) \tag{3}$$

where $\|.\|_F$, V_i,V_j, and σ are, respectively, the Frobenius norm, the singular vectors of the unfolded matrix in the online stage, and the singular vectors of the unfolded matrix in the training stage and the bandwidth of the kernel function. The singular values in Equation (3) are generated after computing the SVD.

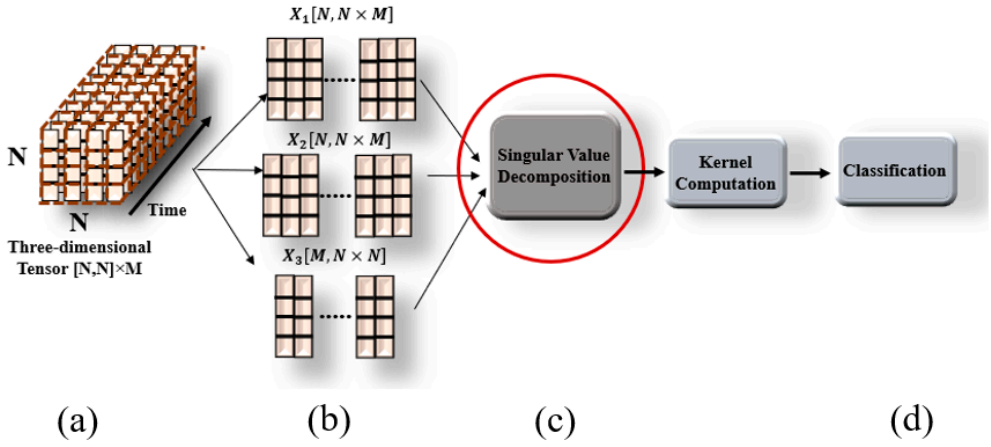

Figure 1. Sketch of the tensorial kernel approach, which is composed of three main phases: (**a**) a three-dimensional tensor (representing the tactile data generated from a sensor array), (**b**) X_1 X_2 X_3 are the unfolded matrices, (**c**) SVD to compute the eigenvalues of the unfolded matrices, (**d**) classifying the eigenvalues based on the SVM.

The SVD is the main block of the kernel computation, it factorizes each matrix M of size $(m \times n)$ into a product of matrices as follows:

$$M = USV^T \tag{4}$$

where M, U, S, and V are respectively the unfolded input matrix, an orthogonal matrix $(m \times m)$, a diagonal matrix $(m \times n)$, and a unitary matrix $(n \times n)$. U and V contain, respectively, the left and right singular vectors of M. The diagonal elements $(\sigma_0, \ldots, \sigma_{n-1})$ are the singular values of M. The SVD is implemented based on the one-sided Jacobi algorithm [41] where a sequence of rotations are applied to the matrix $U = M^T M$. Depending on the input data, the one-sided Jacobi algorithms needs between five to eight iterations to converge [7]. The sequence U_1, U_2, U_3, etc. Is generated as follow:

$$U_{a+1} = J(a, b, \theta)^T U_a \, J(a, b, \theta) \tag{5}$$

where the Jacobi rotation $J(a, b, \theta)$ consists of an identity square matrix with four elements on the intersection of rows a and columns b. A Jacobi rotation is generated for each sub-matrix to annihilate the off-diagonal elements (w) of the matrix U as follow:

$$\begin{bmatrix} \hat{x} & 0 \\ 0 & \hat{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T \begin{bmatrix} x & w \\ w & y \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{6}$$

3.2. Dataset Preparation

The dataset is adopted from [4]: seventy participants were asked to apply two touch modalities (sliding the finger vs. rolling a washer) on a 16 sensors array. Each touch modality was repeated twice for two directions (horizontally and vertically) as shown in Figure 2. The total number of collected touch modalities is 560 (70 participants, 2 modalities, 2 directions, 2 trials).

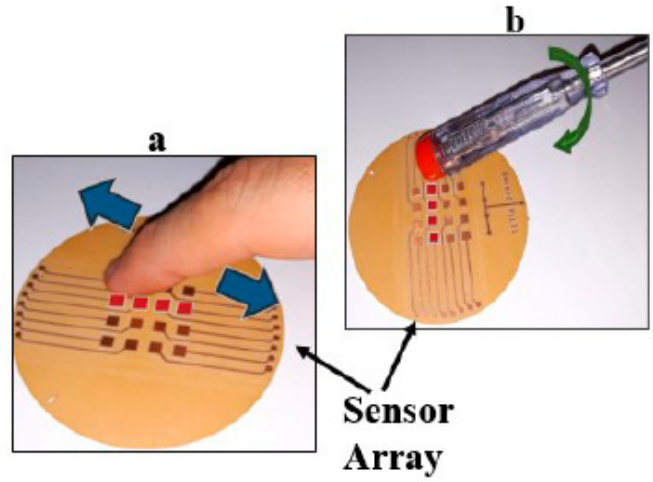

Figure 2. Touch modalities (**a**) finger sliding, (**b**) washer rolling.

3.3. Data Preprocessing

The collected touch modalities were represented by a tensor of three dimensions (4×4 sensor array and time) of size T ($4 \times 4 \times 30{,}000$). The third component of the tensor T was determined by a time interval of (10 s) in each experiment with an adopted sample rate of (3 k samples per second). Pre-processing is used to remap the original tensor mostly to reduce the dimensionality of the third component of T. Only a portion of the 30,000 samples in the third component of T carry information about the tactile stimulus. The relevant signal lies in a limited time window as shown in Figure 3. The localization of the relevant time window is defined after evaluating the amount of energy provided by the sensors as proposed in [4]. In the following, the tensor t obtained after extracting the relevant time window from T is as follows: t ($4 \times 4 \times 20$).

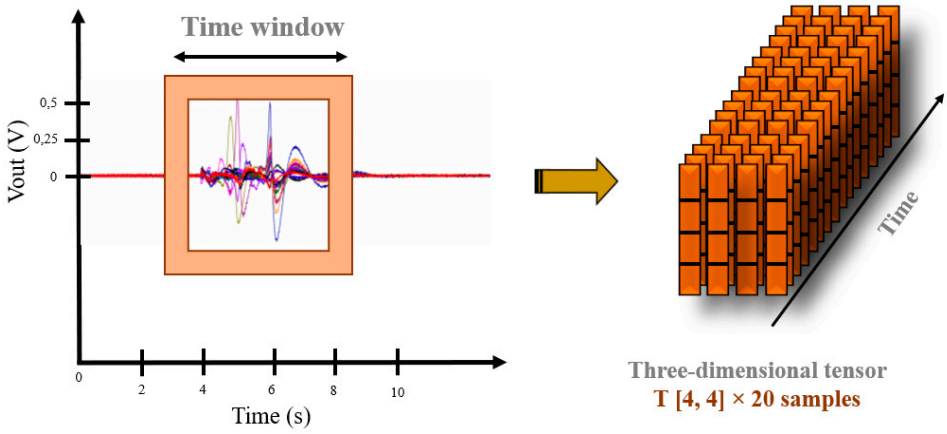

Figure 3. Schematization of the data preprocessing phase.

4. Proposed Methodology

This section will describe the methodology adopted to assess the impact of approximate SVD on online classification accuracy. Figure 4 presents the methodology, which is organized into three main steps:

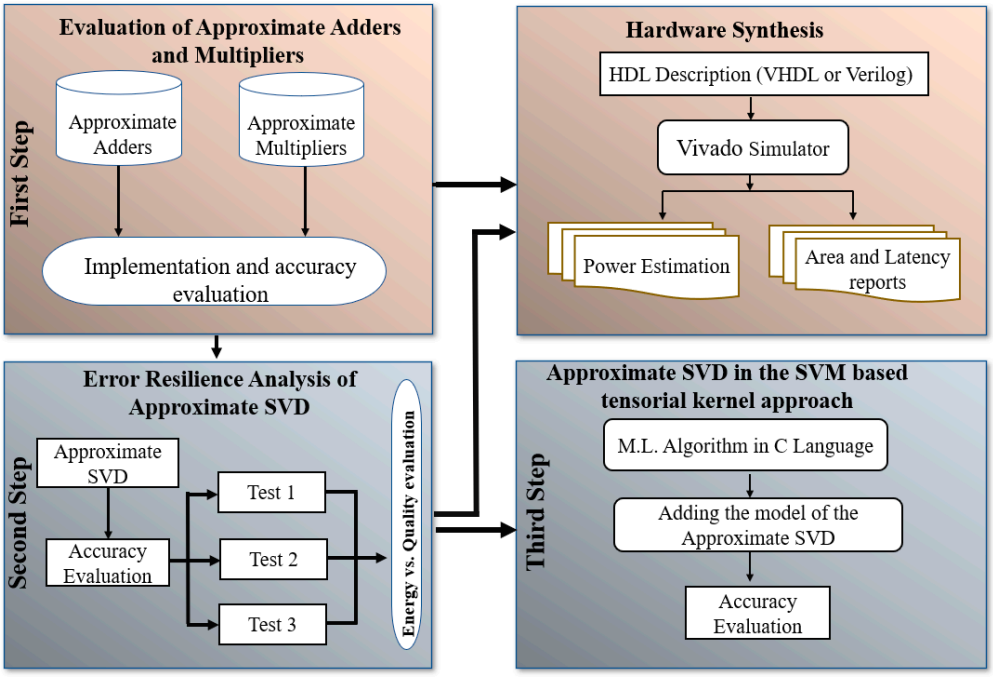

Figure 4. Overview of the proposed methodology.

In the first step, the approximate adder and multiplier circuits are implemented and simulated in Vivado Design Suite 2017.1 using VHDL Hardware Description Language. Six approximate adders are selected that belong to two classes: speculative adders and approximate full adders. Seven approximate multipliers are selected from the state-of-the-art, including two approximate multipliers proposed in our recent work. The source codes of the approximate multipliers and adders are found in [42]. The accuracy is evaluated after selecting 10^5 input patterns with uniform probability density distribution. The energy consumption and time delay are reported after synthesizing the designs by using the Xilinx Vivado synthesizer, with Virtex-7 xc7vx485tffg1157-1 device files. Then, we assess the performance of the approximate multipliers, to select the circuit to be used in the online classification.

In the second step, the selected circuit (i.e., Approx-BW) is implemented in the SVD since the SVD takes 95% of the operations of the tensorial SVM as demonstrated in [43]. The error-resilience [44] of the approximate SVD is analyzed after implementing Approx-BW compared with the exact multiplier in the SVD. The number of approximated LSBs is varied to extract the energy-quality trade-off of the approximate SVD. The error-resilience of approximate multipliers and approximate SVD is evaluated based on the mean relative error (MRE) [45] defined as follow:

$$Error = |Approximate\ result - Exact\ result| \tag{7}$$

$$RE\ (\%) = \frac{|\ Approximate\ result - Exact\ result|}{Exact\ result} \times 100 \tag{8}$$

$$MRE = \frac{\sum_{i=1}^{n} RE}{n} \tag{9}$$

where n represents the number of the inputs for the multiplier and the SVD.

The RE of the approximate SVD is analyzed based on a set of matrices selected based on uniform distribution ranging from 0 to 1, which represents the values generated from the sensor array in a real application. The probability density function for a uniform distribution [46] a and b represents the range of values.

$$f(x) = \begin{cases} \frac{1}{b-a}\ if\ a \le x \le b \\ 0 \quad otherwise \end{cases} \tag{10}$$

Three tests are implemented as follow:

Test 1: calculating the MRE of the eigenvalue S_i generated from each of the input matrices from the approximate SVD as follow:

$$RE\ (S_i) = \frac{|\ Approximate\ S_i - Exact\ S_i|}{Exact\ S_i} \times 100 \tag{11}$$

where S_i represents the eigenvalue such as S_1,

The MRE is computed as follows:

$$MRE\ (S_i) = \frac{\sum_{j=1}^{n} (RE\ (S_i))_j}{n} \tag{12}$$

where n represents the number of input matrices ($n = 20$). The goal is to find the MRE for each eigenvalue. Five eigenvalues are generated from the SVD; therefore, the same procedure is applied for each eigenvalue from S_1 till S_5.

Test 2: calculating the RE of the five eigenvalues generated from each of the input matrices from the approximate SVD as follow:

$$RE\ (M_{Eigen}) = \frac{\sum_{i=1}^{5} RE\ (S_i)}{5} \tag{13}$$

where M_{Eigen} represents the matrix of eigenvalues containing the five eigenvalues (S_1, S_2, S_3, S_4, S_5). The MRE of M_{Eigen} is computed as follow:

$$MRE\ (M_{Eigen}) = \frac{\sum_{j=1}^{n} RE\ (M_{Eigen})}{n} \tag{14}$$

Test 3: calculating the MRE by taking into consideration only the value of the eigenvalue having the highest RE from the five eigenvalues generated from each of the input

matrices from the approximate SVD, e.g., if S_5 has the highest RE among the remaining eigenvalues, then the RE of S_5 is selected as follow:

$$RE\left(M_{Eigen}\right) = RE\left(S_5\right) \tag{15}$$

$$MRE\left(M_{Eigen}\right) = \frac{\sum_{j=1}^{n} RE\left(M_{Eigen}\right)}{n} \tag{16}$$

The goal of the third test is to figure out the range of the maximum error that could be generated from the approximate SVD. The tensorial kernel algorithm is modeled in the C language in the third step. The approximate SVD model is introduced in the simulation program as shown in Figure 5. We test the impact of the approximate SVD on the accuracy of the classification. The MRE generated from the approximate SVD from each of the three tests done above (test 1, test 2, and test 3) is added to the simulation program. The MRE is applied through element-wise multiplication with the eigenvectors of the SVD, as follow:

$$\begin{aligned} error(1) &= V_{x1} \times MRE \\ error(2) &= V_{x2} \times MRE \\ error(3) &= V_{x3} \times MRE \end{aligned} \tag{17}$$

where V_{x1}, V_{x2}, *and* V_{x3} represent the right singular vectors. The approximate model generates the $error(1), error(2)$, *and* $error(3)$ as shown in Figure 5. Consecutively, the final eigenvalues are generated as follow:

$$\begin{aligned} X_1 &= U_{x1}\, S_{x1}\, V'_{x1} \\ X_2 &= U_{x2}\, S_{x2}\, V'_{x2} \\ X_3 &= U_{x3}\, S_{x3}\, V'_{x3} \end{aligned} \tag{18}$$

where

$$V'_{x3} = V_{x3} + error(3)$$

where V'_{x1}, V'_{x2}, *and* V'_{x3} are the modified right singular vectors after applying an MRE generated from the approximate SVD as shown in Figure 5. Finally, after classifying the touch modalities (sliding vs. rolling), the Error Rate (*Err*) of the SVM is computed as follows:

$$Err\ (\%) = \frac{number\ of\ incorrect\ touch\ modalities}{total\ number\ of\ touch\ modalities} \times 100 \tag{19}$$

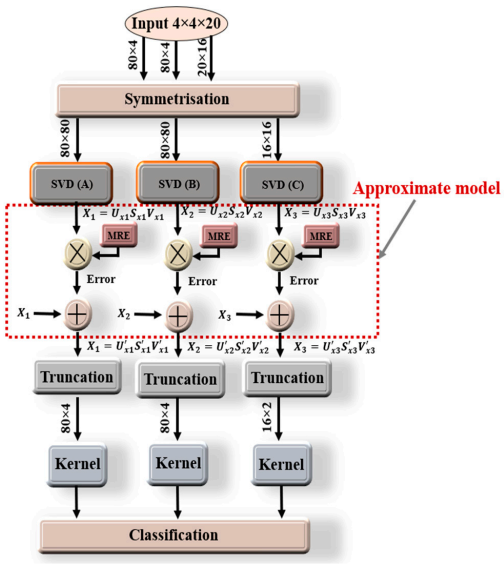

Figure 5. Tensorial SVM kernel approach based on approximate SVD.

5. Experimental Results

5.1. First Step Analysis

5.1.1. Approximate Adders

Efficient approximate adder circuits from the literature have been selected and implemented. The approximate adders belong to two classes: speculative adders and approximate full adders.

Speculative adders: (Approximate XNOR-based Adder (AXA) [47], Input Pre-Processing [48]).

Approximate full adders: (Approximate NAND-carry out bit [49], Approximate AND-carry out bit [49], (LOA) [38], and Error Tolerant Adder (ETA) [50]).

Energy consumption and delay have been evaluated for each adder implementation with a testbench of 3×10^{-6} s. Figure 6a illustrates the power consumption and time latency of the approximate adders concerning MRE. Table 1 shows the circuit synthesis results, i.e., Lookup Table (LUT) utilization, power-delay product (PDP), and PDP-MRE products: AFA and LOA adders (which have the lowest MRE of 1.52%) have also the lowest energy consumption (0.09 pJ, 0.11 pJ) since carry propagation has been omitted. While AXA has the worst MRE (4.95%) with higher energy consumption (0.17 pJ). AFA and LOA have been selected to be implemented in the approximate multiplier circuits.

Table 1. PDP and PDP-MRED of Adders Designs.

Approximate Adders	PDP (pJ)	PDP-MRED
AFA	0.09	0.13%
LOA	0.11	0.16%
AND-C	0.17	0.47%
NAND-C	0.16	0.71%
IPP	0.14	0.43%
AXA	0.17	0.83%

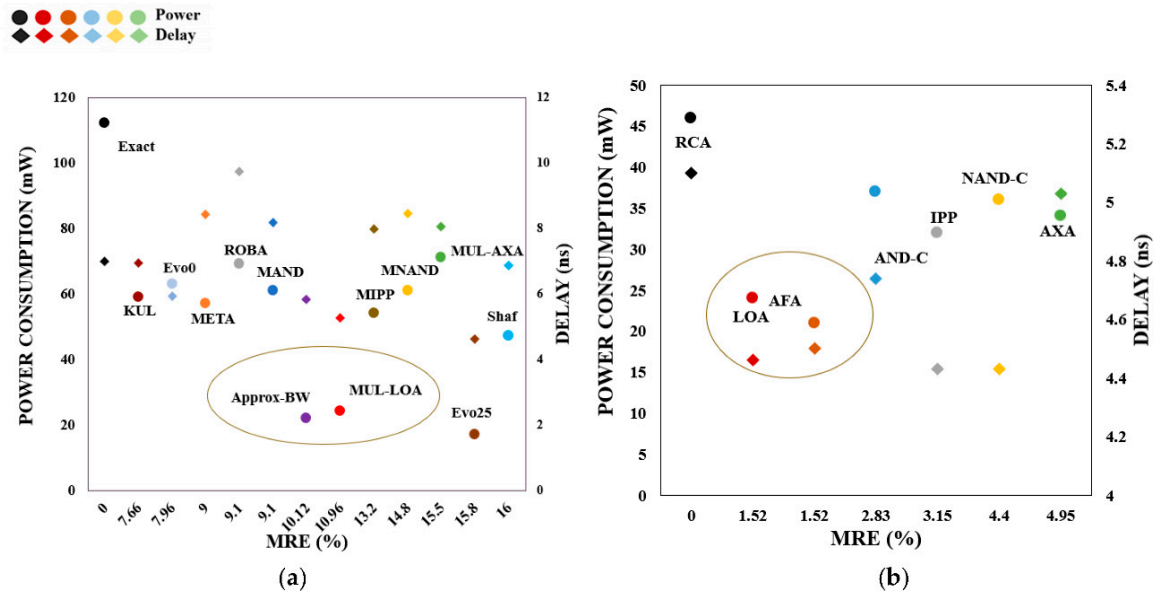

Figure 6. Power Consumption [mW], delay [10^{-9} s], and MRE of (**a**) approximate adder and (**b**) approximate multiplier circuits.

5.1.2. Approximate Multiplier Circuits

Seven relevant approximate multipliers from the state-of-the-art [20,21,51] including two approximate multipliers proposed in our recent work [20], Reference [22] have been selected for the analysis. META [22] multiplier is a rounding multiplier based on the

Error tolerant Adder (ETA). Approx-BW multiplier [20] is based on dividing the circuit architecture into two parts: accurate and approximate, as shown in Figure 7. From right to left, an exact addition is computed in the accurate part. Through a half adder, the partial products are added; the generated carry signal is propagated to the following partial column in the next column. The partial products in the second column are computed through the Ripple Carry Adder (RCA). While in the approximate part, the addition of the partial products is done through the AFA approximate adder [20]. Starting from the first column on the left, an input carry signal set to "0" is added to the first partial product. Then, in each column the values of the partial products are checked through the AFA [20] adder, which consists of two blocks: control block and carry free adder block as shown in Figure 8. The control block applies an AND logic operation for the two input bits A and B. Then, the result performs an OR operation with the control signal obtained from the previous control block. While the carry-free adder block is composed of an XOR and OR gates generating a sum bit based on a control signal obtained from the control block. As shown in Table 2, if all the values of the partial products are 0, or if one partial product is equal to 1, a normal addition is computed. Otherwise, if two partial products or more are equal to 1, then the sum of the partial products in the selected column and all the remaining right columns is set to 1. The approximation strategy adopted is inspired by the approach employed in [50]; where the control check approach has achieved good improvements in the Error Tolerant Adder (ETA). Therefore, we have applied this approach to the approximate multiplier.

To find the most optimal multiplier, we have implemented five other versions of META and Approx-BW based on approximate adders presented in Section 5.1.1. All the multipliers are implemented by respecting the same conditions and constraints. The configuration adopted for all the multipliers is as follows: half of the number of bits are approximated while the other half is accurate. The multipliers are the following:

- Mul-LOA and Mul-AXA, which are based on lower-part-OR and XNOR-based adders, respectively.
- MNAND, MAND, and MIPP multipliers are based, respectively, on NAND-carry out a bit, AND-carry out bit, and Input pre-processing approximate adders.

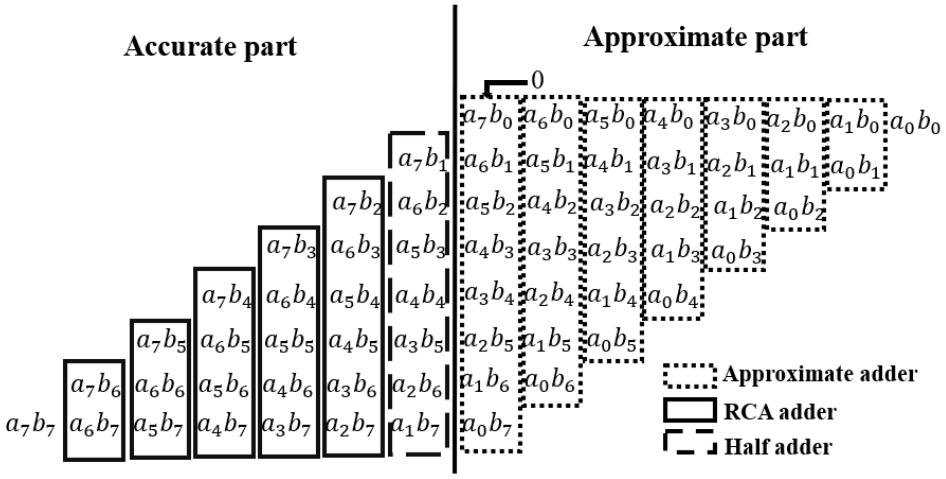

Figure 7. Architecture of Approximate Baugh-Wooley multiplier.

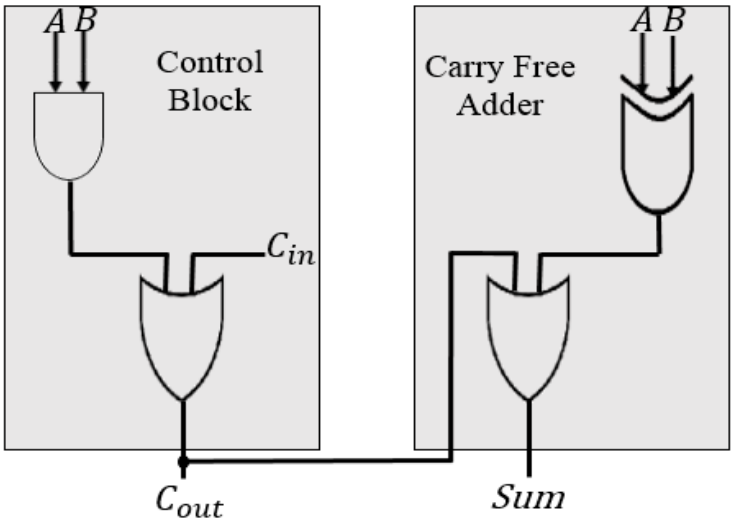

Figure 8. Approximate full adder block diagram.

Table 2. Output of Sum and Carry bits for Different Cases.

Cases	Partial Products	Sum Bit	Carry Bit
First	$a_{n-1}{\cdot}b_0 \mid a_{n-2}{\cdot}b_1 \mid \ldots \mid a_0{\cdot}b_{n-1} = 1$	$S_{n-1} = 1$	$C_{n-1} = 0$
Second	$a_{n-1}{\cdot}b_0 = a_{n-2}{\cdot}b_1 = \ldots = a_0{\cdot}b_{n-1} = 0$	$S_{n-1} = 0$	$C_{n-1} = 0$
Third	$a_{n-1}{\cdot}b_0 = a_{n-2}{\cdot}b_1 = 1$ Or $a_{n-1}{\cdot}b_0 = a_{n-2}{\cdot}b_1 = a_{n-3}{\cdot}b_2 = 1$ Or: $a_{n-1}{\cdot}b_0 = a_{n-2}{\cdot}b_1 = \ldots = a_0{\cdot}b_{n-1} = 1$	$S_{n-1} = S_{n-2} = \ldots = S_0 = 1$	$C_{n-1} = 1$

In total, 12 approximate multipliers have been implemented. The energy consumption and delay of the implemented multipliers have been determined after running a test for 3×10^{-6} s. Table 3 shows that the energy consumption of Approx-BW and Mul-LOA is reduced by more than 50% when compared to the exact BW multiplier. Approx-BW is 10% more accurate than Mul-LOA; thus outperforming Mul-LOA in terms of accuracy. Approx-BW and Mul-LOA are the most efficient with an MRE around 0.1. Table 4 shows the power consumption breakdown of Approx-BW. Among the twelve approximate multipliers, we conclude that Approx-BW is the most appropriate architecture to be employed for the target implementation.

Table 3. PDP and PDP-MRED of Multipliers Designs.

Approximate Multipliers	PDP (pJ)	PDP-MRED
Approx-BW	0.13	1.29%
Mul-LOA	0.13	1.38%
Mul-AXA	0.57	8.85%
MAND	0.5	4.53%
MNAND	0.51	7.61%
MIPP	0.43	5.67%
ROBA	0.67	6.09%
META	0.48	4.31%
Evo0	0.37	2.96%
Evo25	0.08	1.72%
Kulkarni	0.41	3.13%
Shafique	0.32	5.12%

Table 4. Power consumption breakdown of Approx-BW.

Characteristics	Power Consumption(mW)
Dynamic power	22
I/O	19
Signal	1
Logic	2

5.2. Second Step Analysis

5.2.1. SVD Hardware Implementation Details

Figure 9 illustrates a block diagram for the computational blocks of the SVD algorithm. As described in Section 3, the one-sided Jacobi algorithm expects a square and symmetric matrix at the input. For that, the Matrix Symmetrization block in the figure is in charge of multiplying the input matrix by its transpose, implying a square and symmetric matrix. In the next step, The Coordinate Rotational Digital Computer (CORDIC) Phase block computes the rotation angles to be then used by the one-sided Jacobi rotation block to rotate the row/columns of the symmetric matrix in each pre and post-rotation. After a series of rotations, the algorithm converges to provide the singular vectors at its output.

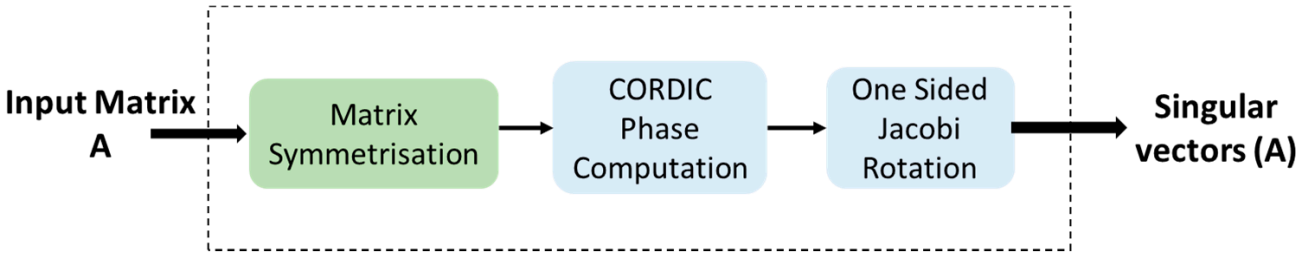

Figure 9. Block diagram for the SVD computational blocks.

Moreover, the hardware implementation of the SVD adopts a hybrid data representation. The architecture starts with 8 bits integer, then the data representation will be transformed to 23 bits integer after matrix symmetrization. CORDIC algorithm uses 32 bits fixed-point representation where 2 bits are used as integers, and 30 bits for the decimal part. Then, the output of the multiplier is 42 bits, which are to be truncated before starting the next iteration. Since SVD is an iterative algorithm, a truncation process is applied each time, the data needs to be reiterated. This approach has been adopted to not fix the data representation to the maximum number reducing the complexity as much as possible.

5.2.2. Error Resilience Analysis

The Approx BW has been selected as mentioned above. Four Approx BW multipliers have been implemented in place of the exact multipliers in the architecture of the SVD as shown in Figure 10. The number of bits for the exact and approximate multiplier is 42 bits in fixed-point representation. The scalability of the approximate multiplier has been assessed as follow:

- The approximation is enabled for the eight LSBs of the Approx-BW into the SVD (SVD-approx8), where 8 LSBs are approximated while the rest bits are exact.
- The number of the approximated LSBs is increased to 12 bits, where the rest MSBs are exact.
- The same procedure is applied until approximating 28 LSBs.

The error will increase when increasing the number of the approximated bits. Based on Test 1 presented in Section 4, simulations have shown that the MRE of the eigenvalues of the SVD (S1, S2, S3, and S4) remains lower than 1% after approximating 16 LSBs as is shown in Figure 11. While the MRE of the eigenvalues increases from 1% to 14.5% when

approximating 20 LSBs. When approximating 24 and 28 LSBs, the MRE of the eigenvalues (S2, S3, S4, S5) increases drastically to more than 50%. From this simulation, we assume that 20 LSBs can be approximated. Figure 12 shows the MRE of approximate SVD after simulating test 2 and test 3, presented in Section 4. The MRE (SVD-approx8, SVD-approx12, and SVD-approx16) remains less than 5% for both tests. The MRE of SVD-approx20 is 11%, while in the worst cases, the MRE increases by up to 30% based on test 3. The MRE of SVD-approx24 and SVD-approx28 reaches values far from being acceptable (more than 50%) for both tests. From these simulations, we may conclude that ACTs are applicable for the SVD showing an increment of the MRE by less than 11%. In the following, we evaluate the performance of the approximate SVD in terms of power/energy consumption, time latency, and LUT usage. Figure 13 indicates the performance analysis of the SVD in terms of LUT and power/energy.

In Figure 13a, the *x*-axis represents the numbers of output LSBs approximated in the SVD and the *y*-axis represents, respectively, the percentage of reduction of LUT and latency. The power/energy consumption concerning the exact SVD is shown in Figure 13b. *y*-axis denotes the MRE of the outputs generated from test 2, indicating the range of acceptable accuracy. Based on these experiments, we notice that the power/energy consumption, LUT utilization, latency have been reduced when increasing the number of approximated LSBs. For instance, when approximating 20 LSBs, the LUT utilization and latency have been improved by more than 4%. The power and the energy consumption have been reduced respectively by up to 14 and 16% after approximating 20 LSBs with an MRE of less than 11%.

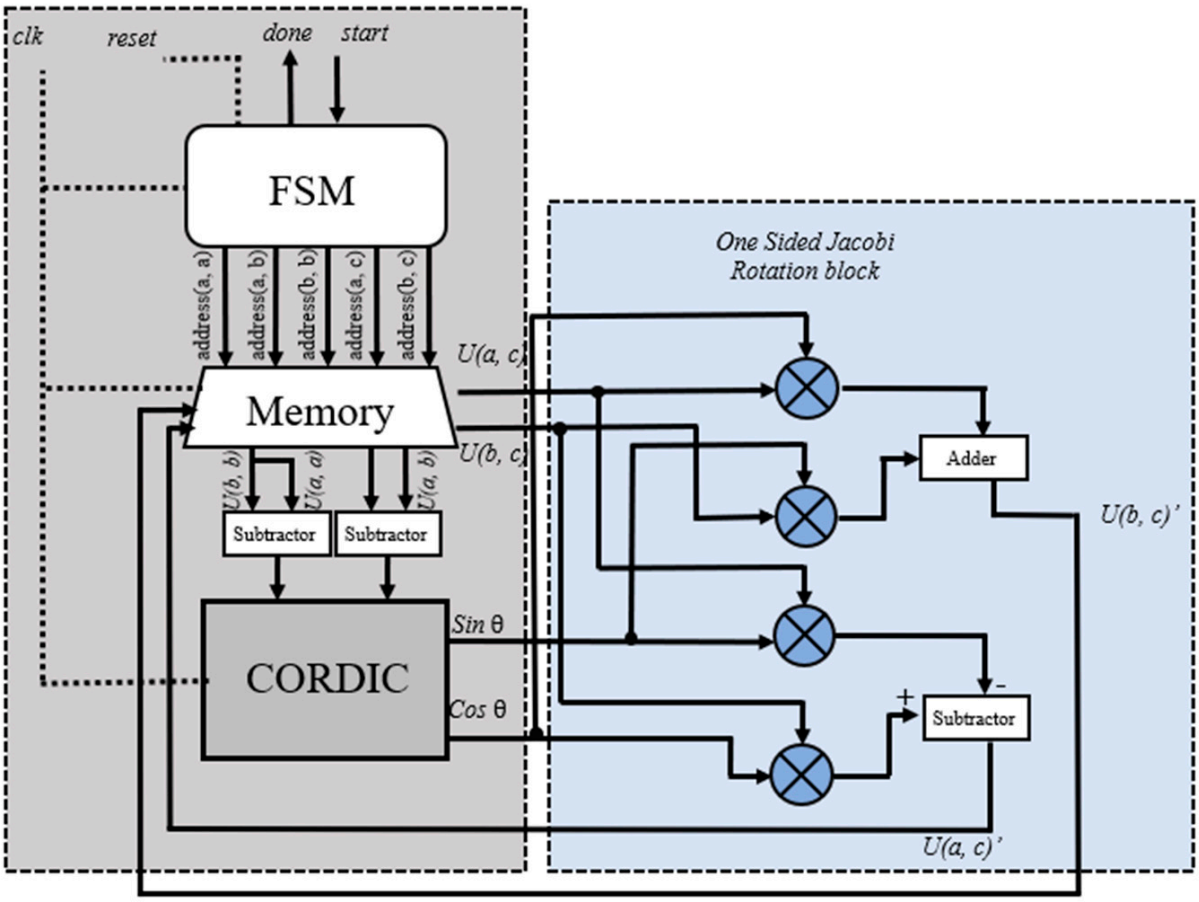

Figure 10. Block diagram of SVD hardware implementation.

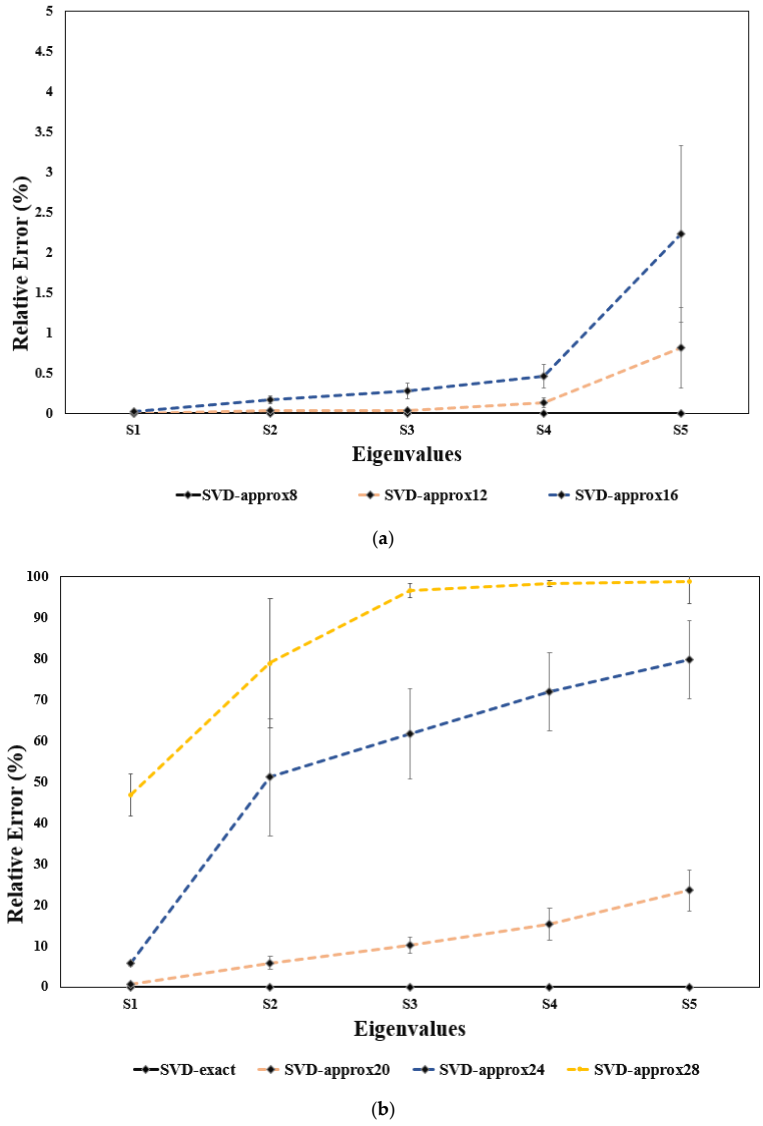

Figure 11. Mean Relative Error of the eigenvalues of the approximate SVD for number of approximated LSBs ranging from 8 till 20 approximated LSBs (**a**) and 24 till 28 approximated LSBs (**b**).

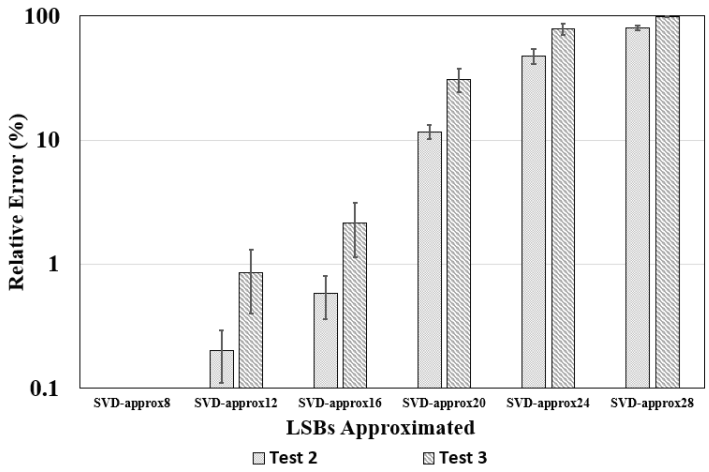

Figure 12. Mean Relative Error of the eigenvalues generated from the approximate SVD for different number of approximated LSBs ranging from 8 till 28 approximated LSBs for two different tests.

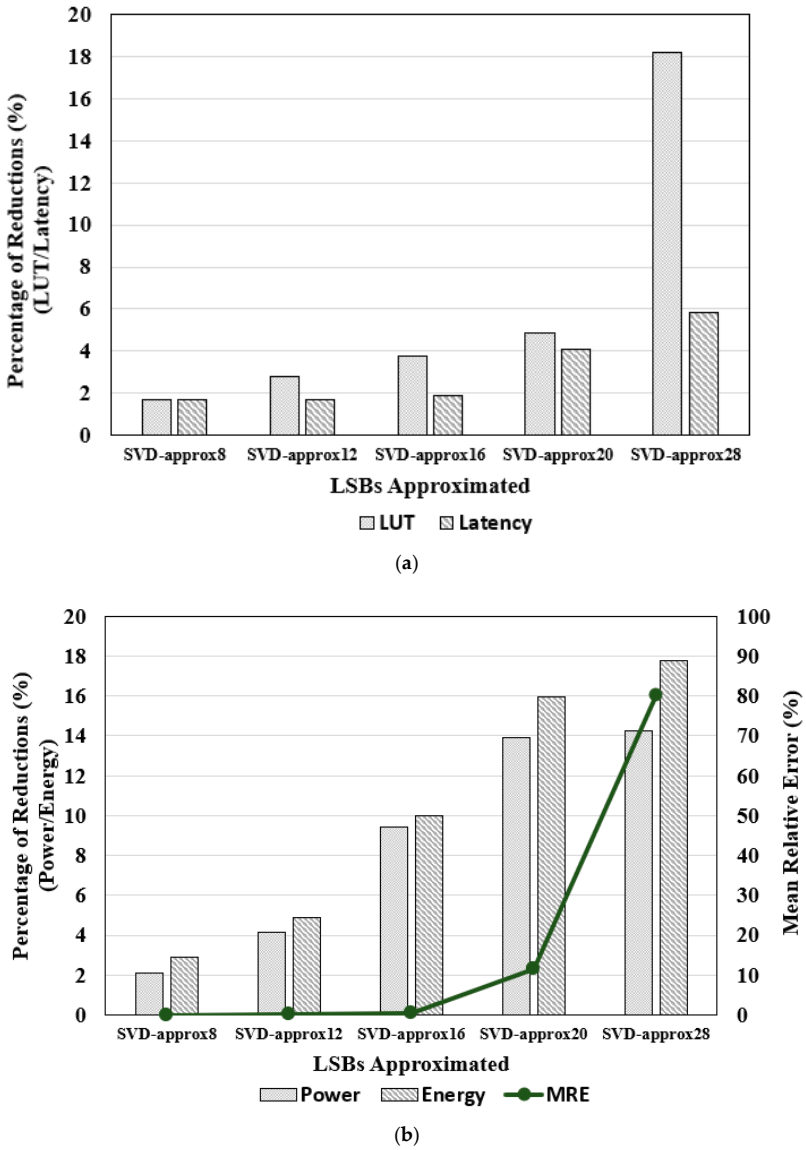

Figure 13. Performance analysis of the Singular Value Decomposition in terms of LUT/Latency (**a**) and Power/Energy (**b**).

5.3. Third Step Analysis

In this section, the values of the MRE generated from the second and the third tests presented in Section 5.2 are added to the simulation program of the tensorial SVM [4]. Figure 14 shows the results of the classification problem (sliding a finger vs. rolling a washer). For each test, the Err value of the SVM has been reported after increasing the number of approximate LSBs in the SVD from 8 to 28 bits. The simulation with the exact SVD shows an Err value of 12.5% for sliding and 20% for rolling. Such values have been assumed as a baseline to assess the impact of approximate SVD. Figure 14a shows that the Err value for sliding and rolling does not increase when employing SVD-approx8 and SVD-approx12. In the case of SVD-approx16, the Err value for sliding increases by 1.25%; while with SVD-approx20 the Err value reaches 28.75% for sliding and 21.25% for rolling. The Err reaches values far from being acceptable such as 46.25% when using SVD-appox24 and SVD-approx28 for sliding. Based on the third test, Figure 14b shows that the Err value for sliding increases by not more than 2% in the case of SVD-approx12 and SVD-approx20 concerning the Err value shown in Figure 14a. The Err value for sliding and rolling is fairly constant to the Err resulting in the second test in the case of SVD-approx8 and SVD-approx16.

The Err in Figure 14b is expected to be higher than the Err in Figure 14a; since in the third test, only the highest RE from the eigenvalues is taken into consideration.

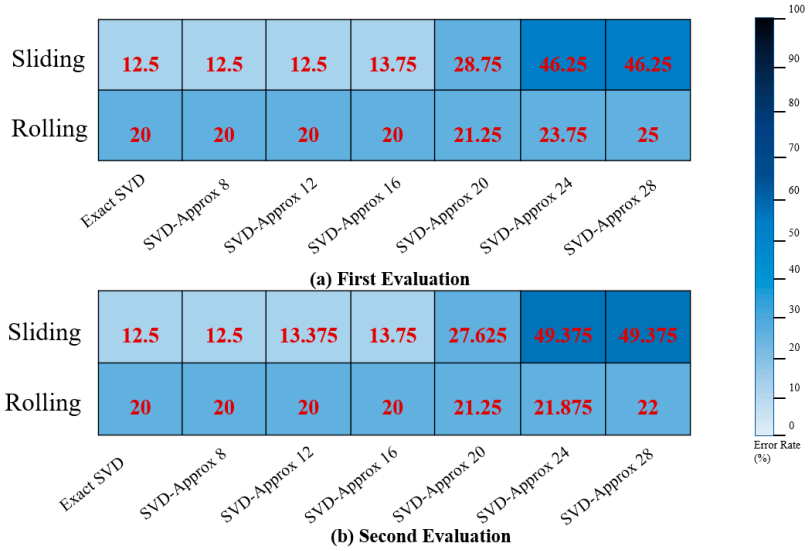

Figure 14. Error of the SVM based tensor kernel approach for different numbers of Approximate LSBs in the SVD for two cases.

Figure 15 illustrates the variation of the average additional Err of sliding and rolling when increasing the number of approximated LSBs for three different tests. Figure 15 shows that the Err is increased by less than 10% for test 2 when using Approx-SVD 20, while the Err is increased by around 15% when selecting eigenvalues having the highest RE (test 3). By mapping the simulation results from Figure 14 to the MRE results of the approximate SVD in Figure 11, the SVD could have savings, respectively, in energy of 16% at the cost of Err less than 10%, which is promising even if the SVD is considered as a sensitive algorithm and the errors in the iterative steps accumulate and may diverge from the unique singular values. This result motivates applying more ACT techniques at different levels of the SVD. A hybrid approach test is applied that consists of implementing three different approximate SVD having different numbers of approximated LSBs. The main idea in the hybrid approach is to implement Approximate SVDs with different configurations and check which configuration fits well in terms of performance and accuracy.

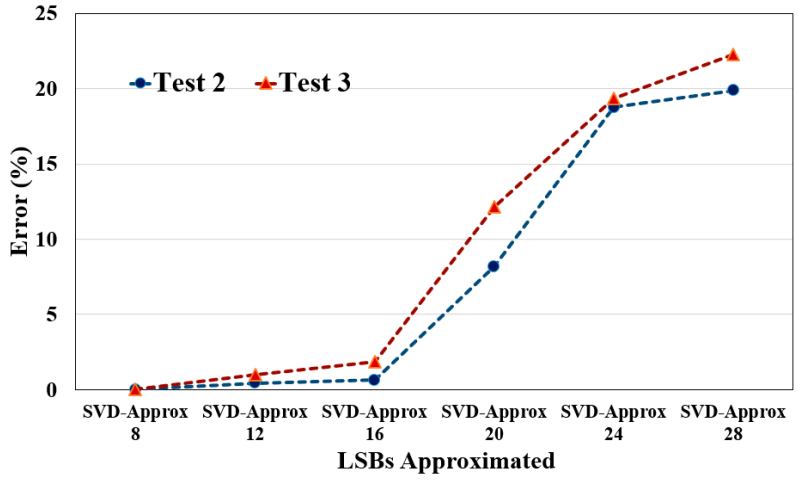

Figure 15. Additional error of the SVM-based tensor kernel approach when increasing the number of approximated LSBs in the SVD for two different tests.

Table 4 shows the hybrid approach simulation results. The results are extracted based on the MRE generated from the second test in Section 5.2. In this approach, SVD's of different values of approximated LSBs are applied to the SVM. The added Err is less than 10% when applying the hybrid approach in the first three cases, as shown in Table 5. The benefits of the hybrid approach lie in improving additionally the power consumption concerning the previous results. For example, the following combination (SVD-approx16, SVD-approx20, and SVD-approx12) shows an additional Err of 1.87%; the Err is approximately equal to the case of using only SVD-approx12 or SVD-approx16 in all SVD's blocks. By adopting the hybrid approach, energy consumption could be improved additionally by more than 5% with respect to previous results, i.e., SVD-approx12.

Table 5. Tests results for the hybrid approach (error comparison is the error difference between the error of the approximate SVD and the baseline error of the exact SVD).

SVD (A)	SVD (B)	SVD (C)	Error Rate (%)	Error Difference (%)
Exact	Exact	Exact	16.25	0
Approx12	Approx16	Approx20	18.12	1.88
Approx16	Approx20	Approx12	18.12	1.88
Approx20	Approx24	Approx16	25.62	9.38
Approx16	Approx20	Approx24	29.38	13.12
Approx20	Approx24	Approx28	35	18.75
Approx24	Approx28	Approx20	36.25	20

6. Discussion and Conclusions

An investigation of the applicability of ACTs in SVM-based tensorial kernel approach for tactile data processing was presented in this paper. The paper started by introducing the tensorial kernel approach adopted for tactile data processing, showing the main implementation challenges of the tensorial SVM. The main contribution in this paper focused on exploiting the applicability of ACTs to the SVD; since SVD is the main computational bottleneck in many ML algorithms [8–10]. To this aim, we demonstrated that the Approx-BW multiplier successfully improves the energy consumption of the SVD. Three different tests have been done to analyze the accuracy of the approximate SVD. We simulated the impact of the approximate SVD on the accuracy of the tensorial SVM when classifying different touch modalities. We proved that approximate SVD improves the energy consumption of the tensorial SVM by up to 16% within the range of an Err from 1% to 8%. Additionally, a hybrid approach has been proposed, which consists of three different approximate SVD of different numbers of approximated LSBs. The energy consumption has improved additionally by more than 5%, concerning the previous approach within the same range of an Err. It should be noted that the reduction of the energy consumption by 16% is considered acceptable for embedded ML algorithms when compared to other state of the art works, which have an energy consumption reduction within the range of 15–20%, such as [36,50,52].

In the context of the research area of approximate computing, it should be noted that the first exploration of applying ACTs to the SVD at the circuit level is presented in this work. The proposed approximate SVD could be adopted in different research works since SVD plays a significant role in various DSP and ML applications. Moreover, this research work demonstrated that any embedded ML algorithm that employs an SVD could be optimized using ACTs. However, one of the main challenges of the adopted approach is to study well the scalability of the approximate SVD before exploiting it in embedded ML algorithms.

To sum up, in this paper we evaluated the use of the approximate BW multiplier in the ML algorithm to validate its importance in such applications. However, the SVD uses only three multipliers which represents a low percentage compared to the whole circuit. This is the main reason behind the limited reduction in the overall application. Nevertheless, the results achieved in this work motivate us to investigate the implementation of different

ACTs to further reduce the energy consumption, such as approximate memory storage, voltage-scaled memory (circuit level), and quantization techniques (algorithmic level).

Author Contributions: Conceptualization, A.I. and M.O.; Methodology, M.O. and A.I.; Software, M.O.; Validation, M.V. and A.I.; Investigation, M.O. and A.I.; Data curation, M.O.; Writing—original draft preparation, M.O.; Writing—review and editing, A.I. and M.V.; Visualization, M.O.; Supervision, A.I. and M.V.; Funding acquisition, M.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: In The authors acknowledge financial support from Tactile feedback enriched virtual interaction through virtual reality and beyond (Tactility) project: EU H2020, Topic ICT-25-2018–2020, RIA, Proposal ID 856718.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, Z.; Yu, H.; Shi, H. Human Activity Recognition Based on Improved Bayesian Convolution Network to Analyze Health Care Data Using Wearable IoT Device. *IEEE Access* **2020**, *8*, 86411–86418. [CrossRef]
2. Benatti, S.; Montagna, F.; Kartsch, V.; Rahimi, A.; Rossi, D.; Benini, L. Online Learning and Classification of EMG-Based Gestures on a Parallel Ultra-Low Power Platform Using Hyperdimensional Computing. *IEEE Trans. Biomed. Circuits Syst.* **2019**, *13*, 516–528. [CrossRef] [PubMed]
3. Seminara, L.; Pinna, L.; Ibrahim, A.; Noli, L.; Caviglia, S.; Gastaldo, P.; Valle, M. Towards integrating intelligence in electronic skin. *Mechatronics* **2016**, *34*, 84–94. [CrossRef]
4. Gastaldo, P.; Pinna, L.; Seminara, L.; Valle, M.; Zunino, R. A tensor-based approach to touch modality classification by using machine learning. *Rob. Auton. Syst.* **2015**, *63*, 268–278. [CrossRef]
5. Vapnik, V. *Statistical Learning Theory*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1999.
6. Schölkopf, B. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
7. Ibrahim, A.; Valle, M. Real-Time embedded machine learning for tensorial tactile data processing. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *65*, 3897–3906. [CrossRef]
8. Fontenla-Romero, O.; Pérez-Sánchez, B.; Guijarro-Berdiñas, B. LANN-SVD: A Non-Iterative SVD-Based Learning Algorithm for One-Layer Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 3900–3905.
9. Wang, Y.; Zhu, L. Research and implementation of SVD in machine learning. In Proceedings of the 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 24–26 May 2017; pp. 471–475.
10. Narwaria, M.; Lin, W. SVD-based quality metric for image and video using machine learning. *IEEE Trans. Syst. Man. Cybern. Part B Cybern.* **2012**, *42*, 347–364. [CrossRef]
11. Mittal, S. A survey of techniques for approximate computing. *ACM Comput. Surveys.* **2016**, *48*, 1–33. [CrossRef]
12. Han, J.; Orshansky, M. Approximate computing: An emerging paradigm for energy-efficient design. In Proceedings of the 2013 18th IEEE European Test Symposium, ETS 2013, Avignon, France, 27–30 May 2013.
13. Seva, R.; Metku, P.; Kim, K.K.; Bin Kim, Y.; Choi, M. Approximate stochastic computing (ASC) for image processing applications. In Proceedings of the ISOCC 2016—International SoC Design Conference: Smart SoC for Intelligent Things, Jeju, Korea, 23–26 October 2016.
14. Zhang, Q.; Wang, T.; Tian, Y.; Yuan, F.; Xu, Q. ApproxANN: An approximate computing framework for artificial neural network. In Proceedings of the Design, Automation and Test in Europe, DATE, Grenoble, France, 9–13 March 2015.
15. Ibrahim, A.; Gastaldo, P.; Chible, H.; Valle, M. Real-time digital signal processing based on FPGAs for electronic skin implementation. *Sensors* **2017**, *17*, 558. [CrossRef]
16. Mohapatra, D.; Karakonstantis, G.; Roy, K. Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In Proceedings of the International Symposium on Low Power Electronics and Design, San Fancisco, CA, USA, 19–21 August 2009.
17. Irofti, P.; Dumitrescu, B. Pairwise Approximate K-SVD. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019.
18. Foster, B.; Mahadevan, S.; Wang, R. A GPU-based approximate SVD algorithm. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Parallel Processing and Applied Mathematics, Torun, Poland, 11–14 September 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 569–578.
19. Shim, K.; Lee, M.; Choi, I.; Boo, Y.; Sung, W. SVD-softmax: Fast softmax approximation on large vocabulary neural networks. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
20. Osta, M.; Ibrahim, A.; Chible, H.; Valle, M. Inexact Arithmetic Circuits for Energy-Efficient IoT Sensors Data Processing. In Proceedings of the IEEE International Symposium on Circuits and Systems, Florence, Italy, 27–30 May 2018.

21. Zendegani, R.; Kamal, M.; Bahadori, M.; Afzali-Kusha, A.; Pedram, M. RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 393–401. [CrossRef]
22. Osta, M.; Ibrahim, A.; Seminara, L.; Chible, H.; Valle, M. Low power approximate multipliers for energy-efficient data processing. *J. Low Power Electron.* **2018**, *14*, 110–117. [CrossRef]
23. Fukiage, T. Visibility-based blending for real-time applications. In Proceedings of the Visibility-Based Blending for Real-Time Applications, Munich, Germany, 10–12 September 2014; IEEE: Piscataway, NJ, USA; pp. 63–72.
24. Himavathi, S.; Anitha, D.; Muthuramalingam, A. Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. *IEEE Trans. Neural Netw.* **2007**, *18*, 880–888. [CrossRef] [PubMed]
25. Yeam, T.C.; Ismail, N.; Mashiko, K.; Matsuzaki, T. FPGA implementation of extreme learning machine system for classification. In Proceedings of the IEEE Region 10 Annual International Conference, TENCON, Penang, Malaysia, 5–8 November 2017.
26. Ortega-Zamorano, F.; Jerez, J.M.; Munoz, D.U.; Luque-Baena, R.M.; Franco, L. Efficient Implementation of the Backpropagation Algorithm in FPGAs and Microcontrollers. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *27*, 1840–1850. [CrossRef] [PubMed]
27. Wang, Z.; Schapire, R.E.; Verma, N. Error Adaptive Classifier Boosting (EACB): Leveraging Data-Driven Training towards Hardware Resilience for Signal Inference. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 1136–1145. [CrossRef]
28. Anguita, D.; Boni, A.; Ridella, S. A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation. *IEEE Trans. Neural Netw.* **2003**, *14*, 993–1009. [CrossRef]
29. Fafoutis, X.; Marchegiani, L.; Elsts, A.; Pope, J.; Piechocki, R.; Craddock, I. Extending the battery lifetime of wearable sensors with embedded machine learning. In Proceedings of the IEEE World Forum on Internet of Things, WF-IoT, Singapore, 5 February 2018; pp. 269–274.
30. Zhou, Y.; Lin, J.; Wang, Z. Energy-efficient SVM classifier using approximate computing. In Proceedings of the 2017 IEEE 12th International Conference on ASIC (ASICON), Guiyang, China, 25–28 October 2017; pp. 1045–1048. [CrossRef]
31. Van Leussen, M.; Huisken, J.; Wang, L.; Jiao, H.; de Gyvez, J.P. Reconfigurable Support Vector Machine Classifier with Approximate Computing. In Proceedings of the 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Bochum, Germany, 3–5 July 2017; pp. 13–18. [CrossRef]
32. Koliogeorgi, K.; Zervakis, G.; Anagnostos, D.; Zompakis, N.; Siozios, K. Optimizing SVM Classifier Through Approximate and High-Level Synthesis Techniques. In Proceedings of the 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Piscataway, NJ, USA, 13–15 May 2019; pp. 1–4. [CrossRef]
33. Alasad, Q.; Yuan, J.-S.; Bi, Y. Logic locking using hybrid CMOS and emerging SiNW FETs. *Electronics* **2017**, *6*, 69. [CrossRef]
34. Alasad, Q.; Lin, J.; Yuan, J.-S.; Fan, D.; Awad, A. Resilient and Secure Hardware Devices Using ASL. *ACM J. Emerg. Technol. Comput. Syst.* **2021**, *17*, 1–26. [CrossRef]
35. Ibrahim, A.; Osta, M.; Alameh, M.; Saleh, M.; Chible, H.; Valle, M. Approximate Computing Methods for Embedded Machine Learning. In Proceedings of the 2018 25th IEEE International Conference on Electronics Circuits and Systems, ICECS 2018, Bordeaux, France, 9–12 December 2018.
36. Sarwar, S.S.; Srinivasan, G.; Han, B.; Wijesinghe, P.; Jaiswal, A.; Panda, P.; Raghunathan, A.; Roy, K. Energy-efficient neural computing: A study of cross-layer approximations. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 796–809. [CrossRef]
37. Du, Z.; Lingamneni, A.; Chen, Y.; Palem, K.V.; Temam, O.; Wu, C. Leveraging the Error Resilience of Neural Networks for Designing Highly Energy Efficient Accelerators. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2015**, *34*, 1223–1235. [CrossRef]
38. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 850–862. [CrossRef]
39. Hammad, A.; El-Sankary, K. Impact of approximate multipliers on VGG deep learning network. *IEEE Access* **2018**, *6*, 60438–60444. [CrossRef]
40. Gastaldo, P.; Pinna, L.; Seminara, L.; Valle, M.; Zunino, R. Computational intelligence techniques for tactile sensing systems. *Sensors* **2014**, *14*, 10952–10976. [CrossRef] [PubMed]
41. Ibrahim, A.; Valle, M.; Noli, L.; Chible, H. FPGA implementation of fixed-point CORDIC-SVD for E-skin systems. In Proceedings of the 2015 11th Conference on Ph.D. Research in Microelectronics and Electronics, PRIME 2015, Glasgow, UK, 29 June–2 July 2015.
42. Available online: https://github.com/osta1/Multipliers-adders.git (accessed on 26 September 2021).
43. Osta, M.; Ibrahim, A.; Magno, M.; Eggimann, M.; Pullini, A.; Gastaldo, P.; Valle, M. An energy-efficient system for touch modality classification in electronic skin applications. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems, Hokkaido, Japan, 26–29 May 2019; Volume 2019, pp. 1–4.
44. Prabakaran, B.S.; Rehman, S.; Shafique, M. XBioSiP: A methodology for approximate bio-signal processing at the edge. In Proceedings of the 56th Annual Design Automation Conference, Las Vegas, NV, USA, 2–6 June 2019.
45. Liang, J.; Han, J.; Lombardi, F. New metrics for the reliability of approximate and probabilistic adders. *IEEE Trans. Comput.* **2013**, *62*, 1760–1771. [CrossRef]
46. Kozak, A.; Kozak, R.A.; Staudhammer, C.L.; Watts, S.B. Sampling distributions: The foundation of inference. In *Introductory Probability and Statistics: Applications for Forestry and Natural Sciences*; CABI: Wallingford, UK, 2019; pp. 111–145.
47. Yang, Z.; Jain, A.; Liang, J.; Han, J.; Lombardi, F. Approximate XOR/XNOR-based adders for inexact computing. In Proceedings of the 2013 13th IEEE International Conference on Nanotechnology, Beijing, China, 5–8 August 2013; pp. 690–693.
48. Liu, C.; Han, J.; Lombardi, F. A low-power, high-performance approximate multiplier with configurable partial error recovery. In Proceedings of the Design, Automation and Test in Europe, DATE, Dresden, Germany, 24–28 March 2014.

49. Allen, C.I.; Langley, D.; Lyke, J.C. Inexact computing with approximate adder application. In Proceedings of the NAECON 2014—IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 24–27 June 2014; Volume 2015, pp. 21–28.
50. Zhu, N.; Goh, W.L.; Zhang, W.; Yeo, K.S.; Kong, Z.H. Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2010**, *18*, 1225–1229.
51. Mrazek, V.; Hrbacek, R.; Vasicek, Z.; Sekanina, L. EvoApproxSb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017, Lausanne, Switzerland, 27–31 March 2017.
52. Kumar, M.; Zhang, X.; Liu, L.; Wang, Y.; Shi, W. Energy-efficient machine learning on the edges. In Proceedings of the 2020 IEEE 34th International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 18–22 May 2020; pp. 912–921.