IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 11, November 2015.
www.ijiset.com
ISSN 2348 – 7968

Hardware Implementations of SVM on FPGA: A State-of-the-Art Review of Current Practice

Shereen Moataz Afifi, Hamid GholamHosseini and Roopak Sinha

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand

Abstract

The Support Vector Machine (SVM) is a common machine learning tool that is widely used because of its high classification accuracy. Implementing SVM for embedded real-time applications is very challenging because of the intensive computations required. This increases the attractiveness of implementing SVM on hardware platforms for reaching high performance computing with low cost and power consumption. This paper provides the first comprehensive survey of current literature (2010-2015) of different hardware implementations of SVM classifier on Field-Programmable Gate Array (FPGA). A classification of existing techniques is presented, along with a critical analysis and discussion. A challenging trade-off between meeting embedded real-time systems constraints and high classification accuracy has been observed. Finally, some key future research directions are suggested.

Keywords: *SVM, FPGA, Hardware Implementation, Embedded Systems, Image Processing.*

1. Introduction

Support Vector Machine (SVM) model is a powerful supervised machine learning method that is used for efficient classification with high accuracy in various applications like object detection, speech recognition, bioinformatics, image classification, medical diagnosis and others [1]. Supervised learning methods are normally composed of two main phases: training/learning, and classification. The SVM training phase builds a model for classifying any future data based on the Support Vectors (SVs) identified from a training dataset. The SVs are then used in the classification phase to predict the appropriate class of an input test data. Interestingly, SVMs have shown high classification accuracy rates outperforming other popular classification algorithms in numerous cases and applications [2-5].

A growing interest exists for exploiting SVMs in many embedded detection systems and various image processing applications. SVM model is computationally expensive and time-consuming especially for large-scale problems, which raises a vital need for acceleration. While software implementations of SVM produce high accuracy rates, they cannot efficiently meet real-time embedded systems constraints. In such embedded real-time applications, special dedicated hardware architectures are required to meet constraints like limited resources utilization and low power consumption. This has motivated plethora of research towards implementing and accelerating SVM in hardware such as using parallel computing platforms.

Special purpose hardware such as reconfigurable hardware is promising for speeding up computations, and provides High Performance Computing (HPC) at low cost and low power consumption [6]. Field-Programmable Gate Arrays (FPGAs) are powerful and highly parallel processing reconfigurable devices which are used for achieving HPC in embedded systems with efficient utilization of hardware resources. Interestingly, FPGAs have recently shown significant performance gains outperforming General-Purpose-Processors (GPPs) for many applications in a growing range of areas such as image processing, digital signal processing, pattern recognition, computer vision, bioinformatics, machine learning algorithms, etc. [7-11].

Graphics Processing Unit (GPU) also offers an alternative platform for high performance computing [12]. Many performance comparisons of FPGA and GPU implementations of different algorithms and applications have been studied in literature [13-19]. FPGAs demonstrated superior performance in most cases, however in some applications, GPUs slightly outperformed FPGAs [19]. The availability of open source libraries such as OpenCV helps achieve much faster development time for GPUs than for FPGAs. However, for more complicated algorithms that use shared arrays and high memory accesses, GPUs cannot provide good performance due to memory access limitations caused by their memory architecture [13]. Although GPUs benefit from lower cost and shorter development time compared to FPGAs, they are inferior to FPGAs in terms of power consumptions (FPGAs consume approximately an order of magnitude less power [14]). Moreover, existing GPU implementations are challenging and very hard to be mapped efficiently to the energy-efficient embedded GPUs because of the fixed hardware and limited available resources (less memory, registers, cache and cores) [20].

ISSN 2348 – 7968

Accordingly, implementations on power hungry GPUs are difficult to be deployed in embedded environments, and this has motivated a move towards FPGA implementations.

Recent FPGA technology makes it possible to include processor cores and various useful Intellectual Property (IP) blocks onto a single chip, which ensures more flexibility for designing high performance embedded systems and Multi- Processor Systems on Chip (SoC) [21]. Also, modern development tools that have been recently released allow simplified embedded systems design by using high-level languages that decrease hardware development effort and shorten time-to-market with no need for expert hardware designers (e.g. Xilinx Vivado HLS tool [22]). Accordingly, FPGA has become a good choice for substantially accelerating intensive computations like SVM to achieve HPC with more flexibility at low cost, while meeting hard constraints of embedded real-time systems.

Some existing research work aims to implement and speed up the SVM model (training or classification phase) using FPGAs by exploiting the inherent parallelism of the SVM algorithm. Various hardware architectures and designs have been introduced achieving high level of parallelization as well as high performance. This paper reviews existing literature for hardware implementations of the SVM classifiers on FPGAs, as to the best of our knowledge no survey or review paper on this topic currently exists in literature. This paper reviews a total of 53 current papers published within the period 2010-2015. These papers are grouped into two main groups. The first group considers FPGA-based implementations of the SVM training phase, while implementations regarding the classification phase are presented in group two. An additional third group is presented that contains application-specific implementations that use hardware implementation of SVM as part of a larger algorithm. Group three is narrowed to concern SVM-based applications that target classification or analysis of images only, due to the common use of SVM in image processing applications.

This survey paper aids hardware designers to understand this area and possibly choose the best-fit solution for their context. In addition to simply presenting and collecting various techniques, we also provide a critical analysis and discussion on the strengths and limitations of existing works as well as future research directions in the area.

This paper is organized in five sections. Section 2 provides a brief background of SVM. The research methodology used is presented in section 3. Different hardware techniques assembled into the three groups mentioned above are presented in section 4 followed by critical analysis for each group. Finally, this article ends with a discussion and concluding remarks in section 5.

2. SVM Background

Support Vector Machine was first introduced by Cortes and Vapnik in 1995, and is based on the concept of a decision boundary that separates two different classes of data in order to discriminate classes with high accuracy [23]. A separating hyperplane is constructed in the training phase by using an input training data set containing data samples. The hyperplane that best separates the samples belonging to the two classes is called a maximum-margin hyperplane that forms the decision boundary. The class samples that are on the boundary are called Support Vectors (SVs) as depicted in Fig.1, where SVs are encircled. These SVs obtained from training phase are then used in the classification phase to classify new data [24].

Consider training data labelled as (x_i , y_i), $i = 1, 2,....., N$, $y_i \epsilon \{-1, +1\}$, and $x_i \epsilon R^d$. The hyperplane is the plane that separates the two classes of positive and negative samples as shown in Fig.1. The pattern x lies on the hyperplane in the feature space can be described by Eq. (1), where w is a normal vector to the hyperplane and b is a constant:

$$w \cdot x + b = 0 \quad (1)$$

By selecting the two hyperplanes described in Eq. (2) and (3), the data points are separated in the margin region, and the aim is to maximize the distance between them.

$$w \cdot x + b = +1 \quad (2)$$
$$w \cdot x + b = -1 \quad (3)$$

The Euclidian distance between these two hyperplanes is given as $2/||w||$ (see Fig.1), and so the distance $||w||$ needs to be minimized. The optimum separation hyperplane conditions can be formulated into the following expression that represents a linear SVM, minimize $||w||^2/2$ under the following constraints, which are added for all the training samples to prevent from falling into the margin:

$$w \cdot x_i + b \geq +1 \text{ , for } y_i = +1 \quad (4)$$
$$w \cdot x_i + b \leq -1 \text{ , for } y_i = -1 \quad (5)$$

This can be rewritten in the following equivalent form:

$$y_i (w \cdot x_i + b) \geq 1 \text{ , } i=1,\ldots N \quad (6)$$

The optimization problem represents the minimization of a quadratic function under linear constraints (quadratic programming (QP)). A convenient way to solve constrained minimization problems is by using a Lagrange

ISSN 2348 – 7968

function, where a Lagrange multiplier α_i is assigned to each training pattern via the constraints represented in Eq. (6). Solving the SVM training problem by using the QP techniques is computationally expensive, especially for large high-dimensional datasets. Hence, numerous algorithms have been proposed in literature for solving the QP problem like the Sequential Minimal Optimization (SMO) decomposition method [25]. The most important aspect of these solutions is that the complicated computation of the dot-products calculation is required for each iteration.

In many real-world classification problems, it is not possible to linearly separate the training data in the original space. So, the input space is mapped to a higher-dimensional one where a linear separation is feasible, which is a computationally expensive task, especially in large-scale problems. Therefore, SVMs go towards utilizing kernel tricks/functions $K(x_i, x_j)$ replacing the inner products in the optimization problem in Eq. (6) as in Eq. (7).

$$y_i\,(K(x_i\,,x_j\,)+b)\geq 1\,,\ i=1,\ldots N \qquad (7)$$

The SVM computational requirements depend on the used Kernel function. The most common kernel designs which are widely used because of their efficiency in mapping data to higher dimensional space are illustrated as follows (X and Z are vectors) [26]:

1. Linear: $K(X,Z)= (X\cdot Z)$
2. Polynomial: $K(X,Z)= (1+(X\cdot Z))^d$
3. Sigmoid: $K(X,Z)= \tan((X\cdot Z)+\theta)$
4. Gaussian Radial Basis Function (RBF): $K(X,Z)= \exp(\ ||(X-Z)||^2 / (2\sigma^2))$

On the SVM classification phase, the classification function is universal and straightforward. A new data sample x is classified according to the output (sign) of the main decision function in Eq. (8). For large datasets, a massive number of complicated dot-product calculations is needed, which offers significant parallelization potential that could be exploited by parallel hardware resources as FPGAs.

$$F(x) = sign\,(\sum \alpha_i\, y_i\, K(x_i\,, x)\ + b) \qquad (8)$$

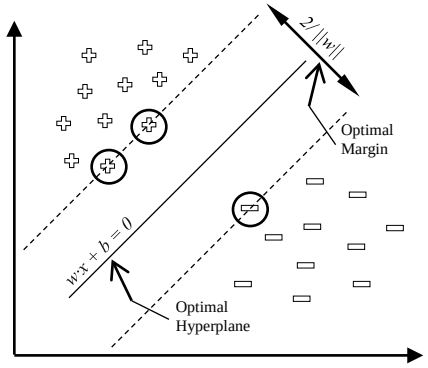

Fig. 1 SVM separating hyperplane.

SVM is originally designed for binary classification and the use for multiclass classification is more problematic, either several binary classifiers have to be built or a larger optimization problem is required. As multiclass problems are commonly encountered, many multiclass SVM classification strategies have been proposed in literature like “one-against-all”, “one-against-one” and other methods [27].

3. Research Methodology

In order to review literature that deals with “Hardware Implementations of SVM Classifiers on FPGAs”, some keywords were used to cross-search for thousands related papers in five databases of scientific publications: Google Scholar, IEEE Xplore, ScienceDirect, Scopus and ACM Digital Library. The search keywords include FPGA, hardware implementation, embedded systems, Support Vector Machine, SVM classification/classifier, and SVM training/learning.

This review is confined to conference papers and journal articles published within the period 2010-2015 in order to explore and analyze the current state-of-the-art of hardware implementations of SVM classifiers on FPGAs. Many recent works highlight the importance of real-time embedded applications in different areas like bioengineering, healthcare, digital signal processing, wireless sensor networks, multimedia, and others. Therefore, demonstrating the importance of boosting SVM classifier to meet embedded systems constraints is required to be considered and added to the current literature. There already exist a recent survey shows SVM in data mining tasks [1], regarding applications only and some of the challenges. Some of the considerable limitations of implementing the SVM model that are stated in [1] are the processing speed for the training and testing phase as well as large memory space requirements, which prove the necessity of an efficient implementation of SVM. To the

ISSN 2348 – 7968

best of our knowledge, there is currently no review or survey paper on FPGA-based hardware implementations of SVM targeted towards various embedded applications. Consequently, this review examines current and recent techniques in hardware implementations targeting efficient implementations of the SVM model on FPGA.

The methodological framework of this research is defined by analyzing a variety of techniques in hardware implementations on FPGA for implementing SVM classifiers. Thus, a total of 53 articles is classified into two main groups regarding hardware implementations on FPGA for the training/learning phase of the SVM as the first group (12 papers) and for the classification phase as the second group (28 papers), where training is performed offline in the software. In addition, a third group (13 papers) is also included to gather a miscellany of hardware implementations for applications that barely use the SVM classification as a part of a larger implemented algorithms covering different domains and areas. We restrict group 3 to image analysis algorithms only, as it results in a compact but robust set of techniques that can be qualitatively compared.

4. FPGA Implementations of SVM

Existing literature for the hardware implementations of SVM model on FPGA was reviewed and roughly subdivided into two main groups according to the implemented phase of the SVM; training phase and classification phase. In this section, different techniques for FPGA-based hardware implementations were presented for the two groups and followed by critical analysis for each group. In addition, a third group is added regarding SVM classification-based application-specific implementations, which is considered part of the second group but from an application point of view and so analyzed with group two at the end of the section.

4.1 Group 1: SVM Training Phase Implementations

The training/learning phase of the SVM model has motivated many researchers to use hardware accelerators targeting a reduction in total training time. In this section, a variety of different FPGA-based architectures is presented in a systematic manner for the first group with a total number of 12 papers.

Many SVM implementations were presented in literature that were based on the common SMO decomposition method [25]. T. Kuan et al. [28] proposed a fully functional circuit design for accelerating the SVM learning phase based on SMO algorithm. The proposed architecture consists of three main circuit modules functioning for the SMO process with a memory block and a cache block that were controlled by a designed Finite-State Machine (FSM) based controller. Experimental results showed that lesser processing time was recorded from using the cache, and the recognition performance of the proposed fixed-point design (FPGA) was similar to that of the floating-point SMO running on MATLAB.

K. Cao et al. [29] proposed a parallel scalable digital architecture for training SVM based on SMO algorithm, aiming to overcome the lack of necessary flexibility in previous implemented embedded applications in literature. A modified version of the traditional SMO algorithm [30] (to improve the efficiency of the working set selection method) was adopted in the proposed digital system, where a multiple processing units working in parallel was mapped to the error cache updating task in the algorithm. The memory size of the hardware and the number of processing units are adjustable, aiming to achieve scalable architecture for handling different sizes of the training problems. The synthesizable Verilog code adopted for FPGA synthesis was generated automatically from the Simulink Stateflow using Simulink HDL Coder. Experiment results based on two different datasets demonstrated that SVM training problems could be solved effectively with inexpensive fixed-point arithmetic offering better flexibility and scalability results.

J. Filho et al. [31] presented a dynamically reconfigurable SVM architecture for general purpose training that supports different sizes of training sets. Based on the SMO algorithm, a modular architecture was designed reaching interchanging modules by dynamic reconfiguration. The hardware-friendly kernel function proposed in [32] was adopted for the system and so the Coordinate Rotation Digital Computer (CORDIC) algorithm [33] based on shift and add operations was employed for kernel implementation. The proposed reconfigurable architecture achieved 22.38% area saving with acceptable reconfiguration time penalty. The effect of fixed-point data representation on precision and classification error was studied on three different learning benchmarks. Based on simulation results, the hardware implementations of the three benchmarks achieved acceleration factors of more than 12.53 times faster than the software implementation for the total training time.

C. Peng et al. [34] proposed a novel reconfigurable and efficient chip design for accelerating SMO-based SVM learning. Two novel methods were used in the proposed design; trimode coarse-grained reconfigurable architecture and triple finite-state-machine with dynamic scheduling, targeting improvement of the baseline design proposed in [28]. The first method amended the baseline design by proposing trimode reconfigurable architectures with

ISSN 2348 – 7968

parallel and pipeline computing capabilities, while the second method offered a schedule for efficient reconfiguration. Compared to the baseline design, simulation results demonstrated that the proposed design achieved improvements in area size (50% less memory usage from removing the kernel cache and 31% fewer gate counts), power consumption (17-fold improvement), and training speed (16-fold improvement) with satisfactory recognition accuracy (85%).

L. Martinez et al. [35] proposed a hardware/software architecture to accelerate SVM training phase based on SMO algorithm. The high time-consuming dot-product computation in the SMO algorithm was executed on the hardware coprocessor in parallel, while the heuristic hierarchy of SMO was implemented in software on the GPP. The bottleneck dot-product calculation was mapped to a logical AND operation and a counter to be executed in three clock cycles. The proposed coprocessor architecture achieved a speedup of 178.7x compared to a software-only implementation on a GPP.

Another hardware-software co-design system for fast SVM training based on SMO algorithm was proposed for embedded speaker identification system [36]. The modular design proposed in [28] was exploited and improved for realizing the computational bottleneck SMO training on hardware, whilst other processes including pre-processing, feature extraction and SVM-based voting analysis were implemented in software on an ARM processor (embedded C code). Also, a data-packed/unpacked mechanism was proposed to improve the efficiency of communication and data transmission between software and hardware (around 5% reduction in delivery time). Compared to an ARM embedded C code, the proposed system achieved 90% decline in training time with a slight decrease in identification rate, where about 89.9% identification rate was achieved.

A. Patel et al. [37] proposed also a hardware/software co-design system to speedup SVM training based on SMO algorithm. A fully scalable co-processor architecture was implemented on Xilinx Virtex-7 FPGA for handling kernel computations based on a proposed SMO algorithm which effectively designed for exploiting the parallel computing power of the proposed co-processor and caching kernel columns through a grid of processing units. A speedup of 20-25x was achieved for the kernel computations on the designed co-processor, while an application speed-up of up to 15x was achieved by comparing results of experiments of the proposed system on the ADULT dataset with LIBSVM.

Other researchers implemented SVM based on different decomposition algorithms replacing the traditional SMO. A hardware-software co-design system for accelerating the SVM learning phase was presented based on another decomposition algorithm instead of the common SMO algorithm [38]. A Hybrid Working Set (HWS) algorithm was proposed based on the extended working set decomposition algorithm [39], taking advantage of cached kernel values and the fast convergence nature in order to decrease the number of iterations. A fully scalable coprocessor architecture consisting of a grid of cores was proposed to achieve parallelism for kernel computations similar to their previous proposed architecture [37]. A speed-up for kernel computations of up to 25x was achieved from the implemented coprocessor (32 cores) over a single threaded core i5 CPU. A reduction in iterations of 50% and 60% was achieved compared to LIBSVM and SVMLight software programs due to the implemented HWS algorithm. Finally, the proposed coprocessor with the HWS algorithm achieved an application speedup of up to 15x and 23x compared to software implementations of LIBSVM and SVMLight respectively.

M. Papadonikolakis et al. [40] proposed a fully scalable heterogeneous FPGA architecture for boosting the SVM learning, which fully exploits the device parallel processing power and the dynamic range diversities of the precision requirements among training problem features. The proposed design fully utilized the custom-precision arithmetic and heterogeneous components supported by the FPGA device for handling kernel computations. The proposed architecture used parallel custom precision multipliers feeding a pipelined adder tree for the fixed-point inner products which are then interpreted into floating point format for further calculations. Experimental results demonstrated the efficiency of the proposed heterogeneous architecture which increased with the precision diversities of the homogenous/heterogeneous datasets attributes. Also, the proposed design showed a speedup of more than 6x compared to other proposed designs.

M. Rabieah et al. [41] presented a complete FPGA-based system for boosting nonlinear SVM training by utilizing ensemble learning, in addition to a proposed cascaded multi-precision training flow that exploits FPGA reconfigurability and the training problem heterogeneity for handling large datasets. The proposed hardware module was designed for implementing Gilbert's training algorithm (simpler than SMO [42]), where numerous processing elements are used for kernel dot product operations, each with its own on-chip memory blocks. The architecture of the processing element followed the previously proposed architecture in [40], where the kernel computation was divided between fixed point and floating point domains, and was improved by using caching and

ISSN 2348 – 7968

running the solution update in parallel. The proposed FPGA system achieved significant speedups compared to other CPU and GPU-based implementations across three different datasets with acceptable accuracy and lower power consumption.

M. Ning et al. [43] used a different design methodology from the previous works presented; High-Level Synthesis (HLS) for developing the hardware accelerator. A demonstration was presented to show that the hardware development time for an embedded system could be reduced by applying the HLS method using C language instead of traditional Hardware Description Language (HDL). A SOPC (System on Programmable Chip) system was designed to implement the Least Square SVM (LS-SVM) on the recent Xilinx Zynq platform using Vivado HLS tool with C language. The LS-SVM has lower computational complexity as it solves a set of linear equations instead of a quadratic programming for standard SVM [44]. The proposed algorithm was divided into three parts; a generating Kernel Matrix module, a solving linear equations module (SOLE), and a forecasting module. The first and third modules were realized in ARM processor which also controls computing modules and data path, whilst the compute-intensive second module was implemented in programmable logic utilizing HLS method. A speed up factor of 2-8x was achieved with 0.06% maximum relative error for trained data, compared to a Matlab based CPU implementation for experiments with a real dataset.

S. Wang et al. [45] exploited the Run Time Reconfiguration (RTR) technology of the FPGA for boosting the online training of the LS-SVM. The proposed design was divided into two parts to be swapped using RTR and applying pipeline in the modules design, reaching high parallelization. The first part is the kernel matrix formulation where a piecewise linear interpolation method was applied. The second part is the least-square problem solving, where a modified Cholesky Decomposition was proposed improving large memory requirements and the long latency caused by square roots operations. Experimental results illustrated speed up from 6 to 218x compared to a Xeon CPU implementation. Regarding time cost percentage analysis, the proposed architecture was shown to be suitable for large-scale problems with more than 1000 samples, due to the large reconfiguration time.

4.1.1 Critical Analysis of group 1

The total training time is considered the main bottleneck of the overall SVM performance in real-time applications, which motivates many researchers for speeding-up this consuming task through implementation on parallel FPGA devices. This section presents a critical analysis for the previous FPGA implementations of the SVM classifier that were introduced in the previous section and summarized in Table 1.

Most of the proposed hardware architectures for implementing the SVM learning were based on the common SMO decomposition algorithm [28, 29, 31, 34-37], whilst an alternative algorithm was employed by one work in [38] and another simpler algorithm was implemented in [41]. Two research papers implemented the improved SVM; LS-SVM instead of the standard SVM due to its low computational complexity [43, 45], but generally most previous work focused on boosting the whole training process including complicated kernel computations.

Various parallel digital designs were presented where the common pipelining approach was mostly applied, reaching a high level of parallelization. Additionally, some studies employed the dynamic reconfiguration technique for the designs aiming speed improvement with more flexibility [31, 45], and dynamic scheduling was exploited as well for efficient reconfiguration in [34]. Moreover, the hardware-friendly kernel was used in [31] and implemented by common CORDIC algorithm based on shifters and adders replacing expensive multipliers, which led to a remarkable reduction in resources utilization.

Many implemented systems employed the hardware/software co-design method for running the compute intensive task of the algorithm on the FPGA as a hardware accelerator (co-processor) aiming to reach real-time SVM training [35-38, 43]. In addition, researchers in [43] used different design methodology; HLS for developing the hardware co-processor instead of using the traditional HDL, where an outstanding reduction in hardware development time and effort for realizing an embedded system was emphasized.

Concerning scalability, some proposed architectures were designed to meet scalability issue, offering more flexibility for efficient usage in embedded environment [29, 37, 38, 40]. Interestingly, a unique heterogeneous FPGA architecture for fully exploiting custom-precision arithmetic and heterogeneous components of the device was proposed [40, 41], offering scalability and adaptability to the classification problem nature.

Finally regarding the achieved results, many proposed implementations for SVM training phase achieved significant speedup results which outperformed similar software implementations [31, 34-38, 41, 43, 45], where some reached acceptable accuracy with slightly rate loss [34, 36]. On the other hand, some hardware

ISSN 2348 – 7968

implementations gained remarkable speedup improvement compared to previous hardware designs [34, 40]. Furthermore, great area saving results were realized in [31, 34] and an improvement in power consumption was shown by [34, 41], which could meet some important embedded systems constraints.

4.2 Groups 2 and 3

4.2.1 Group 2: SVM Classification Phase Implementations

Regarding the second group, various techniques for hardware implementations have been developed for boosting the online classification process on different FPGA boards. The SVM model was trained first offline on software (mostly using Matlab), then the trained data were extracted to be used for online classification on hardware. In this section, different hardware techniques used in implementations are presented in an organized manner for 28 papers in group two.

Systolic array architecture was widely implemented on FPGA for achieving high levels of parallelism, which was exploited by numerous SVM implementations. R. Patil et al. [46] used the systolic array architecture to implement a multiclass SVM classifier on Xilinx Virtex-6 FPGA for facial expression recognition system after performing training phase in Matlab. In addition, difference-based partial reconfiguration technique was utilized in order to achieve power optimization of the FPGA design. Power reduction up to 3 to 5 percent was achieved after applying reconfiguration by using Xilinx EDA tool.

Table 1. Group 1: SVM Training Phase Implementations

Ref.	***Training Implementation Method***	***Platform/FPGA board and tools***	***SVM Type***	***Kernel Function Type***	***Application Domain / dataset***	***Important Results***
[28]	Functional circuit design	Altera Cyclone II DE2-70 Quartus II 7.2 sp3	Multiclass	Linear	Speaker recognition, SMD and FMMD speech datasets	Fully functional prototype
[29]	Parallel scalable digital architecture	Xilinx Virtex-4 (XC4VLX100) Simulink Stateflow +HDL Coder	Binary	Gaussian	Sonar dataset, Telecommunicatio n problem dataset	-
[31]	General-purpose DR architecture, CORDIC	Xilinx Virtex-IV (XC4VLX25)	Binary	Hardware friendly kernel	Breast Cancer, Dermatology, Tic Tac Toe benchmarks	22.38% area saving Speedup > 12.53x
[34]	Reconfigurable architecture with dynamic scheduling	Spartan c3s4000	Multiclass	Linear	Speaker recognition, SMD and FMMD speech datasets	Improvements in power, area, memory efficiency
[35]	Co-processor	XtremeDSP Virtex- IV Development Kit Xilinx ISE 9.2 ModelSIM SE 6.5	Binary	Linear	ADULT dataset	Speedup 178.7x GPP
[36]	Co-processor	Xilinx Spartan-c3s4000 Xilinx ISE 10.1	Multiclass	Linear	NIST 2010 speaker recognition database	90% less training time
[37]	Co-processor	Xilinx Virtex-7 (VC707)	Binary	Gaussian	ADULT dataset	Speedup 20-25x App speedup 15x
[38]	Co-processor	Xilinx Virtex-7 (XC7VX485T) Xilinx PlanAhead	Binary	Gaussian	Different datasets	Speedup 25x CPU App speedup 15x LIBSVM, 23x SVMLight
[40]	Heterogeneous architecture(custom-arithmetic)	Altera's Stratix III (EP3SE260)	Binary	Linear Gaussian polynomial	Various datasets	Speedup 6x

ISSN 2348 – 7968

				sigmoid		
[41]	Heterogeneous architecture(custom-arithmetic)	Xilinx board ML605 (Virtex-6 XC6VLX240T	Binary	RBF	ADULT, Forest covertype, MNIST datasets	Speedup >1 order
[43]	SOPC (HLS)	Xilinx XC7Z020 Vivado HLS	LS-SVM	RBF	Electricity Demand dataset	Speedup 2-8x CPU
[45]	RTR	Xilinx ML510 (XC5VFX130T) Xilinx PlanAhead	LS-SVM	RBF	mobile communication traffic dataset	Speedup 6-218x CPU

Dynamic Partially Reconfigurable (DPR) technique was exploited by H. Hussain et al. [47] to implement an SVM classifier for classifying microarray data for bioinformatics applications. That SVM classifier was also implemented using the systolic array architecture (4 main blocks), but on an old FPGA board; Xilinx ML403, where the kernel computation was implemented using 2-pipelined stages. A speed-up of up to 85x was achieved over an equivalent GPP software implementation based on Matlab bioinformatics toolbox. For changing the SVM core with different parameters, DPR was applied which was 8x faster than reconfiguring the whole FPGA device.

H. Hussain et al. [48] extended their work lately based on their previous work [47] by proposing another systolic array architecture for different dataset sizes (number of SVs is greater than the dimension), where more computation was required. The two architectures were compared with an equivalent Matlab based implementations on GPP and a speedup of ~61x and ~49x were achieved for both respectively. Moreover, DPR was applied for a multi-core SVM architecture based on their previous architecture [47], providing flexibility for microarray based applications.

C. Kyrkou et al. proposed an optimized parallel array architecture [49] targeting real-time SVM-based object detection, by finalizing their initial work of implementing a systolic chain of processing elements [50]. They also showed that the proposed array processing engine is scalable and flexible that could be extended and adapted to meet multiclass classification and different applications demands. The implemented array architecture was evaluated using three different object detection applications; face, pedestrian, and car side view detection. The results demonstrated a high performance of 40, 46, 122 fps for the three applications, with no accuracy loss regarding the software detection accuracy of the SVM model implemented in Matlab (77, 76, 78%).

Many implementations adopted the ***multiplier-less*** approach for optimizing hardware complexity. A hardware implementation for boosting SVM classification was developed on the modern Xilinx Virtex-7 FPGA, that was also based on parallel pipelined systolic array architecture [51]. The researchers were targeting a reduction in hardware complexity and power consumption by implementing simplified multiplier-less kernel using shifts and adds operations instead of traditional vector product kernel for classification. They presented a different approach of applying the CSD (Canonic Signed Digit) and CSE (Common Subexpression Elimination) representation methods for vectors data in order to reduce the number of required adders leading to hardware complexity reduction [52]. A comparative analysis was performed regarding resources utilization for three implemented classifiers; binary linear, binary non-linear and multiclass classifiers using the proposed CSD-based multiplier-less kernel versus the vector product kernel [51]. Also, power reductions of 1, 2.7, and 3.5% were achieved from the three implemented classifiers than that using the conventional vector product kernel.

M. Ruiz-Llata et al. [53] presented an FPGA-based hardware design of SVM for classification as well as regression (regression out of our scope). The proposed system adopted the ***hardware friendly kernel*** function presented in [32], which significantly simplifies the hardware design of the feed-forward SVM classification phase by avoiding the use of computationally intensive multiplications, providing good classification performance compared to the traditional Gaussian kernel. The proposed architecture employed the ***CORDIC*** iterative algorithm [33], based on using only shift and add operations instead of multiplications required by the kernel computation. The implemented SVM classification system utilized 75% of the FPGA logic (Cyclone II) and an external memory was used for storing support vectors leading to 2ms limitation in the classification speed, with an error rate of 4%.

An embedded hardware SVM implementation on FPGA was proposed [54], which was based also on exploiting the hardware friendly Kernel [32] to simplify the hardware design targeting satellite onboard applications. Similarly, the resource consuming multiplications were replaced by simple shift operations, which demonstrated lower resources utilization of 167 slices.

J. Sarciada et al. [55] introduced an FPGA implementation of fast SVM that was based on CORDIC algorithm,

ISSN 2348 – 7968

following the design of the proposed hardware friendly Kernel. The proposed system consists of three sub-circuits (data preparation, CORDIC, and classification circuits) to implement the kernel calculations that was based on a proposed iterative algorithm which is a simplification of the CORDIC method through adders and shifters (without multiplications and exponentials). The implemented system achieved speed improvement over their previous CORDIC circuit implemented in [56] with a factor of 6, with limited hardware resources utilization.
Another hardware design for the SVM decision function was presented for classification and regression problems [57], that was based also on adopting the hardware friendly Kernel [32]. A tree structure based on common Sum of Absolute Differences (SAD) module was exploited for decreasing clock cycles of the 1-norm computation between vectors [58]. Preliminary simulation study was performed on the precision of input parameters by choosing fixed-point arithmetic, keeping the same classification accuracy level with no loss.

D. Anguita et al. [59] proposed an FPGA core generator tool for automatically generating an optimized hardware description of a digital architecture for implementing SVM, according to the user requirements and the constraints of the target FPGA device. The proposed architecture for the feedforward phase of the SVM was based on the hardware-friendly kernel [32] [60], where the bottleneck 1-norm (Manhattan norm) computation was implemented using the parallel tree structure of the common SAD modules [58]. Three architectures was presented for implementing the kernel computation. The first kernel architecture exploited the polynomial approximation method (parabolic and linear piecewise approximations) that was computed by multiple cascade pipelined stages of Multiply-and-Accumulate (MAC) blocks. The second architecture was based on CORDIC-like iterative algorithm with no multipliers implementation [32]. The third architecture was based on the Look-Up Tables (LUTs (memory blocks)) based approach, where kernel values were stored. The tool was tested for an automotive application, showing results with trade-off among latency, hardware resources and maximum clock frequency for the three architectures on low-cost, intermediate-class and high-end FPGAs. The LUT-based and CORDIC-like approaches achieved higher classification rates.

V. Vranjkovic et al. [61] presented a digital architecture for SVM classification with a kernel that avoids multiplication, which is similar to the previous hardware-friendly kernel [32] and looks like the common radial kernel. In the proposed architecture, the expensive exponential function was realized by CORDIC algorithm and the multiplications were implemented using shifters. The results showed that the proposed kernel had comparable classification performance with the similar original radial kernel (No hardware implementation of the proposed digital architecture on FPGA was presented).

Other implementations aimed to exploit the common ***pipelined*** fashion for reaching efficient designs. V. Vranjkovic et al. [62] proposed a universal coarse-grained reconfigurable architecture for implementing various types of machine learning including SVM, decision trees and artificial neural networks. The proposed architecture was organized as 1D or 2D array of simple reconfigurable blocks in a pipelined structure for implementing one of the three classifiers. Concerning the SVM implementation, the classification function was divided into partial sums to be implemented by the reconfigurable blocks using multipliers and adders (and a subtractor in the case of a radial kernel), which was controlled by FSM model. Experimental results showed that the implemented architecture achieved significant classification speedup of 1-2 orders of magnitude, compared with R project-based software implementation, with modest hardware resources utilization.

A high performance unified circuit supporting both linear and non-linear SVM classification was designed, based on sharing most of the resources required for both types, leading to a reduction in circuit size [63]. The proposed unified circuit was designed using a parallel architecture with two-stage pipeline providing shared multipliers and adders required for inner product calculation to support both linear and non-linear SVM classification. For the nonlinear case, the table-driven algorithm proposed in [64] was adopted for the RBF kernel computation, which accelerates the operating speed for an exponential function (fixed-point arithmetic operation) as well as increasing the accuracy. The proposed circuit was synthesized using 65nm standard cell library, demonstrating 661,261 gates with 152 MHz maximum operating frequency. Also, high performance was achieved from processing up to 33.8 640x480 image frames per second.

A highly flexible architecture for a complete SVM classifier regarding the input data and kernel function selection was proposed for multi-purpose classification that could be integrated into different projects [65]. The SVM core was designed as a pipelined architecture, where selection of linearly, polynomial and RBF kernel was performed dynamically at run-time. The dot-product calculation was performed with parallel embedded DSP48E-based MAC units with a LUT-based adder tree, and the exponential function was calculated by using the Xilinx CORDIC IP core. The proposed flexible SVM core was simulated and verified in hardware, achieving accuracy for the RBF kernel of a bound of 10ppm with maximum frequency 92 MHz.

ISSN 2348 – 7968

A hardware design of SVM-based type identification module in a CAD system was presented for colorectal endoscopic images with narrow band imaging (NBI) magnification [66, 67]. A 2-step identifier based on binary SVM was proposed to develop a 3-class identifier using one-versus-one technique targeting real-time processing of full HD images. The SVM identification module was implemented by a fixed-point number and evaluated with different bit length, showing the tradeoff between hardware size and accuracy. A customized architecture was first introduced based on combining 2-class classifiers, which was implemented in their previous work [68], resulted in increasing the inner-product process by 18% due to the overlap of input data. Then, an improved architecture was presented preventing the duplication in the inner-product computation that was structured in two pipeline stages, where common inner-products were computed at the first stage (similar to the approach in above paper described [63]). Accordingly, the hardware size was decreased and the system throughput was increased (>21.2 fps at 100 MHz [66]), with sufficient identification accuracy for CAD system.

An FPGA implementation of multi-class SVM classifier based on posterior probability was introduced that implemented Lin's second improved classification algorithm [69]. The proposed simplified algorithm was implemented in a pipelined design that used a LUT-based method to produce the sigmoid function output, in addition to adders, multipliers and dividers used for the other computations. Experimental results showed that the mean absolute error is 10-4 order of magnitude between the implemented simplified algorithm on FPGA and the C-based implementation, with a slight loss in recognition rate. Also, a decrease in computation complexity was achieved from implementing the simplified algorithm with 0.7 ms time delay, which meet the real-time requirements.

Y. Ago et al. [70] presented a new approach of effectively using cascaded DSP slices and block RAMs embedded in FPGA to design a fully pipelined DSP architecture for accelerating SVM classification. The proposed processor core utilizing 768 DSPs and 800 block RAMs was implemented in Xilinx Virtex-6 FPGA, for the SVM classifier with 760 support vectors that supported three types of kernel functions; sigmoid, polynomial, and RBF kernels. The experimental results showed high throughput of 2.89x106 times per second for classifying 128-dimension feature space running at 370.096 MHZ.

Beside using FPGA, a hardware architecture for accelerating SVM classification phase was realized in ASIC exploiting pipelined adder for the accumulation process in the main processing unit instead of common adders [71]. As a result, speedups of 1.44x and 1.21x were achieved using pipelined adder in compare to using RCA and KS adders respectively, with slightly area overhead. Additionally, the implemented architecture offered 3.5x GMACs compared to other FPGA-based architectures in literature.

Some work in the surveyed literature targeted comparisons between FPGA and GPU implementations. Hardware architectures for FPGA and GPU implementing a human skin SVM classifier and comparing their performance with the software was presented [7, 17]. For FPGA implementation, the critical hardware components were designed using HDL in a fully pipelined structure, whilst other standard components (interfaces, FIFO, FSMs) were implemented in HLL (Impulse C). The preliminary implementation results illustrated that the implemented fully pipelined FPGA architecture outperformed GPU and CPU for a small number of image pixels, while the GPU implementation was the fastest for a large number of pixels. But, GPU implementation consumed significantly higher power than the FPGA implementation, which reduces the use of GPUs in power-constrained embedded systems [17].

Apart from using the traditional HDL approach in many research work for designing hardware implementations, some other work used available powerful tools on the market which simplified the process of hardware designs. One of the common tools is the ***Xilinx System Generator*** that offers high-performance system modelling and automatic code generation from Matlab/Simulink. D. Mahmoodi et al. [72] used the System Generator to design and implement a simple hardware architecture of a 3-class pairwise SVM classifier for Persian handwritten digits dataset. The training phase was performed using LIBSVM model in Matlab, then testing phase was implemented using a combination of serial and parallel designs of simple blocks and functions of System Generator reaching a parallel simultaneously hardware architecture of the classifier. Also, the CORDIC block in System Generator was exploited for implementing the exponential function. The FPGA simulation results demonstrated 202.840 MHz maximum frequency for linear classification and 98.67% classification accuracy for nonlinear classification, with considerable computation time compared to software implementation in Matlab.

A digital hardware implementation for multi-speaker phoneme recognition system based on SVM was designed using building blocks of the Synopsys Signal Processing Workbench (***SPW***) software, which generated the corresponding VHDL code to be analysed with the Xilinx Simulator [73]. The priority scheme was included in the design with the multiclass one-against-one SVM method

ISSN 2348 – 7968

to produce the three most likely phonemes at the output (first, second and third possible phoneme representations), targeting an increase in the accuracy. The implemented hardware system was faster than the software implementation by about 2.53 times, with 16.2% reduction in accuracy.

Some researchers considered the ***cascaded classification*** scheme as an alternative approach for accelerating the SVM classification process. In this scheme, the majority of data are rejected in the early classification stages with less computationally demanding, leaving small amounts of data to be classified at the latter stages with higher accuracy and computational complexity. Accordingly, the usage of the cascaded classification structure could achieve great speedups over a single SVM classifier. Therefore, many researchers were motivated towards hardware implementations for the cascaded structure, in order to reach online real-time classification meeting embedded systems constraints of high performance and low cost.

M. Papadonikolakis et al. [74] presented the first FPGA-based cascaded SVM classifier that exploited custom-arithmetic characteristic (bit-precision) of the device heterogeneous nature, targeting SVM classification acceleration (following their previous proposed architecture in [40] for training acceleration). A fully scalable heterogeneous architecture was proposed that effectively utilized the parallel processing power of the device heterogeneous resources and dynamic range diversities among the classification problem's features. The proposed heterogeneous architecture achieved a speed-up of 2-3 orders of magnitude compared to the CPU classification execution time, and a speed-up over 7x other previous implementations on FPGAs and GPUs.

Later, M. Papadonikolakis et al. [75] applied the FPGA reconfigurability approach to their previous proposed architecture [74] for the cascaded classifier in order to switch from low- to high-precision classification. As a result, an increase in the performance was achieved, as well as gaining an expansion of the potential design space for implementing large-scale cascaded classifiers.

C. Kyrkou et al. [76, 77] presented an optimized architecture for the cascaded SVM classifier based on a proposed hardware reduction method to implement additional stages in the cascade, resulting in reducing hardware resources utilization and power consumption for embedded applications. The proposed method replaced all multiplication operations of the early cascade stages with shift operations by rounding off the training data to the nearest power of two values, with classification accuracy savings. A hybrid architecture was implemented (Virtex 5 FPGA) using the cascaded structure for sequential and parallel processing of input data, after training in Matlab targeting face detection on 640x480 images. As a result, an average performance of 70 fps was achieved, reaching a speed-up of 5x over a single parallel SVM classifier implementation. The employment of the proposed hardware reduction method achieved 43% fewer hardware resources utilization and 20% saving in power, with only 0.7% reduction in classification accuracy.

Next, C. Kyrkou et al. [78] extended their previous work for accelerating the cascaded SVM classifier [76] by proposing an optimized hybrid architecture with the proposed hardware reduction method and an additional novel response evaluation method. The proposed response evaluation process was developed by using the Neural Network (NN) model to classify the responses of the preceding simple stages in the cascade in order to remove samples before the final complicated classification stage, leading to classification speed improvement. In addition, the architecture employed local binary pattern (LBP) descriptors for applying feature extraction prior to the final stage in order to improve detection accuracy. The presented architecture was implemented on Spartan-6 FPGA (replacing the old one used before), targeting embedded face detection using higher resolution of 800x600 images than that in their previous work and other hardware implementations. The implemented hybrid architecture achieved real-time processing of 40 fps with 80% detection accuracy, as well as 25% and 20% reduction in area and peak power respectively, with only 1% reduction in classification accuracy. But compared to their previous work, it seems that lower figures were achieved for both area and accuracy. This is because they are evaluating a big test set of higher resolution images, targeting real-time processing of online video classification as an embedded benchmark application.

4.2.2 Group 3: SVM-based Applications Implementations

This group was constructed to demonstrate the usage of the SVM classification in a wide range of applications, in which the research papers are focused more on the main application's implementation, rather than the classification purpose implementation as in group two. Different hardware architectures have been introduced in literature for implementing algorithms including classification task targeting particular applications. This group consists of 13 papers to introduce some research work of different applications that deal with images.

Many works for object detection have been developed on FPGAs, utilizing the SVM-based Histograms of Oriented Gradients (HOG) algorithm. A real time pedestrian detection was implemented on FPGA targeting high resolution images of 1920 x 1080 pixels to be processed at

ISSN 2348 – 7968

twice the pixel frequency [79]. The proposed design of the detection system was based on a time-multiplex method to construct multiscale detection, with a simply designed SVM that consists of a module for the dot product calculation, an address decoder controlling the memory accesses, an adder for the intermediary summation, and an adder for the bias. Experimental results showed high performance of processing HD resolution images at 64 fps, which outperformed existing FPGA implementations by a factor of 4 with good resources utilization.

A real-time human detection using HOG algorithm with linear SVM classifier was implemented that based on adopting a binarization process [80, 81]. As a result of the binarization process, all multiplication operations in the classifier were replaced with addition operations, which decreased hardware complexity. The proposed pipelined architecture achieved a processing rate of 293 fps with a detection accuracy of 1.97% miss rate and 1% false positive rate, in addition to a low power consumption rate of 353mW.

A processor based approach was employed for implementing SVM-based HOG algorithm [82], which was based on a developed IPPro (Image Processing Processor) from Rathlin research project [83]. The designed multi-core IPPro system (coded by the help of Matlab Simulink model) achieved throughput of 2.6x compared to a handed coded design targeting Xilinx Zed board, and 3.2x compared to relevant recent work in literature.

An object detection processor was presented that based on a proposed simplified HOG algorithm which employed a simultaneous SVM calculation, targeting a reduction in amount of required computations [84, 85]. The proposed simultaneous SVM calculation architecture was designed as 15 parallel classification cores, where each managed 7 blocks of MAC operations and allowed the reuse of intermediate results. Accordingly, experimental results showed 99% reduction of cycle counts for the proposed SVM module compared to architecture without parallelization and pipeline. The proposed system achieved objects detection for SVGA resolution video at 72 fps with 40MHz. Also, the proposed simplified algorithm reduced required computation (from 89.2 to 2.25 GOPS) with minimal memory usage and 3% decline in accuracy.

An embedded design using the HOG-SVM combination was presented for multiple object detection [86]. Single precision 32-bit floating point representation was used in the proposed implementation without making any simplification of the algorithm design in order not to reduce the detection accuracy as occurred in some previous work. Xilinx single precision IP cores (adders and multipliers) were exploited to be designed with the longest latency in a fully pipelined structure with no need of external memory storage. Multiple binary SVM classifiers were instantiated in the proposed system to be capable of detecting multiple objects. The implemented system was capable of detecting three different objects in 640x480 images at 60 fps (outperformed the speed of the software implementation), with a computational performance of 9.47 Giga Floating Point Operations per Second (GFLOPS).

An FPGA-based hardware design for head-shoulder detection was presented [87], which was based on Local Binary Patterns (LBPs) for feature extraction and SVM for classification. In addition, foreground object detection was exploited for improving detection accuracy. A different scheme of unrolling loops required by SVM computations was used for the implementation to meet the sequential flow of data in the FPGA. And so, the SVM computation module was designed based on a FIFO, multiple multipliers, and a pipelined adder tree. The integrated system was implemented on Xilinx Virtex 6 FPGA, demonstrating real-time video stream processing of 640x480 images at 60 fps, with a 15% drop of detection accuracy.

A digital hardware architecture was presented for face detection in IR images based on LBP, SVM algorithms, and generating bounding boxes of detected faces [88]. Concerning SVM module implementation, the dot product calculation was implemented as a pipelined sum using integer adders replacing multipliers (no DSP slices) and buffers were exploited for decreasing memory requirements. The implemented system outperformed software implementation with processing 640x480-pixels video at 313 fps with a false positive rate between 4.5 and 7.2%. Also, low resources utilization were achieved of less than 25% with 266 mW power consumed.

A hardware implementation for pedestrian detection algorithm was presented that was based on a sliding window- based SVM classifier using feature covariance matrices as descriptors [89]. The SVM classifier was designed as 15 units processing in parallel, where 147 scalar products and accumulations were computed that are proportional to the number of the descriptor features. The implemented detection system showed encouraging results for hardware implementation with a maximum frequency of 213 MHz and 9% loss in accuracy.

A complete parallel hardware architecture targeting real-time image classification (object detection) using SVM model was implemented on FPGA, which was based on Scale-Invariant Feature Transform (SIFT) and Bag of Features (BoFs) algorithms for feature extraction [90].

ISSN 2348 – 7968

Regarding SVM implementation, the proposed architecture of the SVM module was designed by using multipliers and accumulators, exploiting parallelism in computing the RBF kernel to accelerate the processing time. Also, the Xilinx IP core CORDIC block was used for implementing the exponential function (sum of sin and cos). Two different challenging datasets with high image variation were used to evaluate the proposed hardware system in which 85% and 78% classification accuracy were achieved for Caltech-256 and KUL Belgium traffic sign datasets respectively, with less than 3% loss in accuracy from the software implementation. In addition, a speedup of 5.7x was achieved for classifying 640x480 images compared to software implementation, with reasonable hardware resources utilization compared to related implementations.

C. Kyrkou et al. [91] presented a hardware architecture for real-time SVM-based object detection using a proposed depth and edge accelerated search method targeting improvement in speedup and detection accuracy. The proposed architecture exploited the proposed SVM processing architecture in their previous work that was based on an array of processing elements [49], and the reduced set method was employed to decrease the number of support vectors used for reducing memory utilization and increasing classification accuracy. The implemented system achieved real-time frame rates of 271, 42, and 23 fps for 320x240, 640x480 and 800x600 image sizes respectively, with a 52% decline reached for the false-positive rate.

An FPGA implementation of a CAD system for skin cancer was introduced that was based on feature extraction and SVM classification for histo-pathological images [92]. A block diagram of hardware architecture of the CAD system flow was presented, which used the RAM memory for storing input image (no hardware design details). Simulation results of the proposed system implemented on Xilinx Virtex-7 FPGA was presented for applying CAD system phases on an input test image.

4.2.3 Critical Analysis of Group 2 and 3

Regarding parallel architectures targeting acceleration of the SVM classification phase, both group two and three are analyzed in this section and summarized in Table 2 and Table 3 respectively.

Many works exploited the FPGA-based parallel systolic array architecture in their implementations [46-49, 51], resulting in good results of classification speedups that mostly outperformed software implementations on GPPs/CPUs. In addition, other works employed the DPR technique [46-48, 75], achieving more flexibility and design space expansion besides gaining speedups. Also, the hardware implementation in [46] achieved a significant reduction in power consumption as a result of applying the difference-based partial reconfiguration technique.

Interestingly, many studies adopted the multiplier-less approach [51, 76-78, 80, 81, 88], where computational intensive multipliers required for computations are replaced with conventional adders and/or shifters in order to decrease the hardware complexity. Similarly, others [53-55, 57, 59, 61] utilized the hardware friendly kernel function for simplifying the hardware design by using the simple shift and add operations instead of resource consuming multiplications. Some of them employed the common CORDIC iterative algorithm for implementing the hardware friendly kernel computations, which is also based on using only shifters and adders [53, 55, 59], while other implementations used the CORDIC algorithm for solving the exponential function of the kernel [61, 65, 72, 90]. As a result of multiplier-less implementation, significant reduction in hardware resources utilization was achieved by [51, 54, 55, 76-78], in addition to remarkable power consumption decrease was demonstrated in [31, 51, 76-78, 80, 81, 88].

Moreover, the common pipelining technique was exploited by most previous hardware designs, taking advantage of the parallel processing capabilities of the FPGA that led to throughput increase of the implemented classification process. Some researchers designed a pipeline stage for common and shared multipliers required for computations to decrease usage of duplicate multiplications [63, 66, 67], achieving lower hardware resources utilization. Additionally, some pipelined designs like in [65, 70, 86] were based on exploiting the embedded IP cores in the FPGA device for efficient resources utilization. Furthermore, the parallel pipelined (adder) tree structure was used by various designs aiming to reach processing speed improvement [57, 59, 65, 74-78].

Table 2. Group 2: SVM Classification Phase Implementations

Ref.	***Classification Implementation Method***	***Platform/FPGA board and tools***	***SVM Type***	***Kernel Function Type***	***Application Domain / dataset***	***Important Results***
[46]	Systolic array architecture, difference-based PR	Xilinx Virtex-6 (6vlx240tff1156-2) Xilinx ISE +	Multiclass	Polynomial	Facial expression recognition system	Power reduction 3-5%

		EDA				
[47]	Systolic array architecture, DPR	Xilinx ML 403(XC4VSX35) Xilinx ISE	Binary	Linear	Classifying microarray (biomedical)data	Speedup 85x GPP DPR 8x faster
[48]	Systolic array architecture, DPR	Xilinx ML 403(XC4VSX35) Xilinx ISE + PlanAhead + ChipScope 12.2	Binary	Linear	Classifying microarray (biomedical)data	Speedup ~61x, ~49x GPP DPR 8x faster
[49]	Parallel systolic array architecture	Xilinx ML505 (Virtex 5-LX110T)	Binary	Polynomial RBF	Object detection	40-122 fps 76-78% accuracy
[51]	Systolic array architecture, multiplierless kernel	Xilinx Virtex-7 Xilinx XPE 14.1	Binary and multiclass	Linear Polynomial	Fisheriris dataset	Power reduction 1, 2.7, 3.5%
[53]	Multiplierless kernel, CORDIC, external memory	Cyclone II (EP2C20)	Multiclass	Hardware friendly kernel	Image recognition COIL dataset	75% logic
[54]	Multiplierless kernel	Xilinx Virtex-5, Spartan-3E Modelsim	Binary	Hardware friendly kernel	Satellite onboard application/ NASA database	167 slices < others
[55]	Multiplierless architecture, CORDIC	-	Binary	Hardware friendly kernel	-	Speedup 6x previous circuit
[57]	SAD-based tree	Altera Cyclone III ModelSim	Binary	Hardware friendly kernel	UCI standard data set Breast Cancer	-
[59]	Pipelined MAC/CORDIC/LUT-based	Xilinx Spartan-IIE, Spartan-3, Virtex-II Pro, Virtex-4	Binary	Hardware friendly kernel	Automotive application/pedestrian detection Daimler-Chrysler dataset	-
[61]	Multiplierless architecture, CORDIC	-	Binary	Digital kernel	4 UCI datasets	-
[62]	1D, 2D pipelined streams	Xilinx Virtex-7 Vivado 2014.2	Binary	Polynomial RBF	18 UCI datasets	Speedup 1-2 orders
[63]	2-stage pipelined parallel architecture	-	Binary	RBF	-	33.8 fps
[65]	Pipelined Structure (CORDIC and MACs)	Xilinx ML505 (XC5VLX110T)	Binary	Linear Polynomial RBF	Pedestrian detection	-
[66, 67]	2-stage pipelined	Altera Stratix-IV (EP4SE360F35C2)	Multiclass	Linear	Colorectal cancer detection	> 21.2 fps
[69]	Pipelined	Xilinx Virtex-5 ISE 10.1	Multiclass	Linear Sigmoid	Language Recognition	-
[70]	Fully pipelined DSP-based architecture	Xilinx Virtex-6 (6VLX240T-FF1156) Xilinx ISE 14.1	Binary	Sigmoid Polynomial RBF	-	768 DSPs $2.89x10^6$ throughput
[71]	Pipelined adder-based architecture	ASIC Synopsys tools	Binary	Linear	MNIST dataset	Speedup 1.44x,1.21x other adders usage
[7, 17]	Fully Pipelined architecture	Xilinx Virtex-5 (LX220), Xilinx Spartan 6 (XC6SLX75)	Binary	Gaussian	Human skin classification	Performance >> CPU
[72]	System Generator blocks-based design	Xilinx Virtex4 (xc4vsx35) System Generator	Multiclass	Linear Gaussian	Persian handwritten digits dataset	202.840 MHz Freq. 98.67% accuracy
[73]	Synopsys SPW	Xilinx Virtex-II	Multiclass	RBF	Multi-speaker	Speedup 2.53x

ISSN 2348 – 7968

	blocks-based	(XC2V3000) Synopsys SPW, Xilinx Simulator			phoneme recognition, TIMIT corpus	
[74, 75]	Heterogeneous architecture(custom-arithmetic), DR	Altera's Stratix III (EP3SE260) Altera tools	Cascaded (Binary)	Gaussian polynomial sigmoid	MNIST dataset	Speedup 2-3 CPU Speed-up >7x others
[76, 77]	Hybrid processing architecture, hardware reduction method	Xilinx ML505 (Virtex 5-LX110T)	Cascaded (Binary)	Linear polynomial	Face detection	70 fps performance Speedup 5× 43% less resources 20% less power
[78]	Hybrid processing architecture, hardware reduction method, novel response evaluation method	Xilinx Spartan-6 XC6SLX150T	Cascaded (Binary)	Linear polynomial	Face detection	40 fps performance 80% accuracy 25% less resources 20% less power

One research group is discriminated with implementing the novel cascaded SVM classifier on FPGA [74-78]. The researches started with using the custom-arithmetic potential of the heterogeneous architecture to design low- and high-precision classification modules. Then, they proposed a hardware reduction method to be used in an optimized hybrid architecture. Finally, they achieved remarkable results of high performance, low resources utilization and less power consumption despite a slight loss in classification accuracy rate.

Some research work studied the quantization effect of using the fixed-point number representation in the hardware implementation on the classification accuracy, aiming to reach the optimal number representation (minimum number of bits) that reduces hardware resources and increases speed with minimal reduction in overall classification accuracy [57, 62, 66, 67]. Accordingly, some hardware implementations recorded relatively loss in classification accuracy as in [69, 73, 76-78, 80, 81, 84, 85, 87, 89-91].

Many of the previous various hardware implementations achieved relatively significant results of accelerating SVM classification process that outperformed similar software implementations [47, 62, 72-75, 79, 84, 86, 88, 90]. However, some researchers were interested to present fair comparison with selected hardware implementations in literature like in [74, 75, 78, 79, 82], demonstrating better performance achievement.

Table 3. Group 3: SVM-based Applications Implementations

Ref.	***Classification Implementation Method***	***Platform/FPGA board and tools***	***SVM Type***	***Kernel Function Type***	***Application Domain / dataset***	***Important Results***
[79]	Multiscale and time-multiplex parallel architecture	Xilinx Virtex-5 (XC5VFX200T)	Binary	Linear	Pedestrian detection INRIA dataset	64 fps
[80, 81]	Pipelined binarization-based adder architecture	Xilinx Spartan-3e (XC3S500E)	Binary	Linear	Human detection INRIA dataset	293 fps
[82]	(IPPro) Processor-based	Xilinx Zed board (Zynq 7020) Xilinx ISE 14.6	Binary	Linear	Object detection	Throughput 3.2x
[84, 85]	Pipelined parallel architecture	Altera Cyclone IV EP4CE115	Binary	Linear	Object detection	72 fps
[86]	Fully pipeline of Xilinx single precision IP cores	Xilinx ML605 (Virtex 6 XC6VLX240T)	Multiclass	Linear	Multiple object detection	60 fps
[87]	Pipelined architecture	Xilinx Virtex 6 (XC6VLX 240T-1FF1156) Xilinx ISE 13.4	Binary	Linear	Head-shoulder detection	60 fps
[88]	Pipelined multiplierless	Xilinx Spartan-6 (XC6SLX45) Synopsys Synplify, Xilinx ISE	Binary	Linear	Face detection	313 fps

[89]	Parallel processing units	Xilinx Virtex-4 LX80	Binary	Linear	Pedestrian detection, INRIA dataset	9% accuracy loss
[90]	Parallel architecture	Xilinx ML509 (Virtex 5 LX110T) Xilinx ISE	Multiclass	RBF	Caltech-256 dataset, Belgium Traffic Sign dataset	Speedup 5.7x 3% accuracy loss
[91]	Parallel systolic array architecture	Xilinx ML505 (Virtex 5-LX110T)	Binary	Polynomial RBF	Object detection	271, 42, 23 fps
[92]	Block diagram	Xilinx Virtex-7 (XC7VX980) Xilinx ISE 14.2	Binary	Linear	Skin cancer detection	-

5. Discussion and Concluding Remarks

Various FPGA-based hardware techniques have been used for implementing the SVM classifier. These techniques exploit the inherent parallel capability of the device for reaching efficient parallel pipelined designs. It is clear that most of existing research works are focused on implementing the SVM classification phase online on FPGA after executing the training phase in software (mostly using Matlab), and limited research aims at implementing the training phase online.

The main trend used for implementing the SVM training phase is the hardware/software co-design approach, where FPGA acts as a hardware coprocessor for accelerating the most computationally intensive parts in the training process. Numerous designs for implementing and boosting SVM classification phase use the parallel pipelined systolic array architecture of processing elements. This provides a configurable modular platform for parallel processing with efficient memory management and data flow, allowing for scalable design and reduced complexity [49].

Most previous research works agree that the main bottleneck of implementing the SVM classifier (for both training and classification phases) is the complicated kernel computation. So, numerous designs have been proposed for mainly implementing these most time and resource consuming computations on FPGA aiming to achieve high performance. The kernel functions basically require a dot-product calculation to process large vectors using expensive multipliers. Accordingly, a lot of research has been focused on replacing complex multipliers by simple shift and add operations, introducing the multiplier-less approach that was widely implemented. In addition, the proposed hardware-friendly kernel was commonly adopted by many implementations targeting simple hardware designs without multiplications. Furthermore, the CORDIC iterative algorithm was employed in many designs for implementing the hardware-friendly kernel as well as the complicated exponential function of the kernel. Therefore, significant area and power savings are achieved by using these simplified methods for reducing hardware complexity.

Furthermore, some researchers have studied the bit-width precision and quantization impacts on the classification accuracy, where some loss in accuracy rating occurs. This emphasizes a trade-off between hardware resources and classification rate. Additionally, meeting embedded systems constraints is considered the main challenging goal of the majority of previous implementations. Consequently, the main trade-off that exists is between classification accuracy and meeting real-time embedded systems constraints of high performance, flexibility, scalability and low levels of area, cost and power consumption.

Regarding limitations of previous implementations, most of the previously presented SVM classification systems are application/problem specific and lack simplicity, flexibility and scalability. These cannot be easily extended and adapted for other applications. In addition, the problem of large memory storage requirements for implementing large-scale applications with big numbers of support vectors is not effectively addressed. Also, fully parallel processing of high vector dimensionality has not been effectively realized, where parallelism depends on the vector dimensionality of a given problem in terms of computational resources that increases processing cycles in the case of high dimensionality and limited hardware resources. Different simplification methods utilizing the multiplier-less approach targeting reduction in hardware complexity result in some loss of classification accuracy, which needs to be improved for reaching acceptable classification rates. Furthermore, many architectures are developed without taking into consideration important embedded systems constraints like the low power consumption constraint that was measured for few number of previous implementations. Moreover, most designs in existing literature were implemented on old versions of

ISSN 2348 – 7968

FPGAs and only a very limited number used recent ones that reflect the current modern technologies for better achievements [37, 38, 43, 51, 82, 92].

From the above, we can conclude that the main challenges are the difficulty of meeting important embedded systems constraints such as real-time, high performance and low cost, and reaching reliable effective classification system with a high classification accuracy.

For future research suggestions, reconfigurable hardware architectures of SVM classifiers for embedded real-time applications that require both high performance and low cost are required. New effective approaches are needed for reducing memory demands and resources utilization especially for problems with high dimensionality. Also, an optimum hardware-friendly kernel needs to be defined for better SVM acceleration. Also, using the DPR technique needs to be effectively employed for reaching optimized flexible and scalable systems, which has so far been poorly applied in a very few implementations. A combination of different techniques presented in literature could be explored for developing an optimized design. Efficient methods are required for improving and increasing the hardware-based classification system accuracy with at least keeping the same accuracy of software SVM implementations. In addition, more research work is required for addressing multi-class classification problems, as most of the existing targeted binary classification. More attention could be paid to the evolvable hardware approach for realizing effective online real-time SVM training. Moreover, an optimization method should be investigated for efficient data transfer and flow control in hardware implementations. Recent FPGA devices and SoCs could be exploited for realizing optimized real-time embedded system, complying with new technologies for achieving better efficiency and results. Furthermore, modern tools and IDEs for designing and developing hardware systems are recommended to be used replacing traditional HDLs, which simplify embedded systems design and reduce hardware development effort and time-to-market.

In conclusion, further future research is required for accelerating SVM on FPGA, aiming to reach an efficient real-time embedded system taking into account the challenging trade-off between high classification accuracy and meeting significant embedded systems constraints.

References

[1] J. Nayak, B. Naik, and H. Behera, "A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges," International Journal of Database Theory and Application, vol. 8, pp. 169-186, 2015.
[2] P. Sabouri, H. GholamHosseini, T. Larsson, and J. Collins, "A Cascade Classifier for Diagnosis of Melanoma in Clinical Images," in 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2014, pp. 6748-6751.
[3] G. M. Foody and A. Mathur, "A Relative Evaluation of Multiclass Image Classification by Support Vector Machines," IEEE Transactions on Geoscience and Remote Sensing, vol. 42, pp. 1335-1343, 2004.
[4] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of Classification Methods Based on the Type of Attributes and Sample Size," Journal of Convergence Information Technology, vol. 4, pp. 94-102, 2009.
[5] J. KIM, B.-S. Kim, and S. Savarese, "Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines," Ann Arbor, vol. 1001, pp. 48109-2122, 2012.
[6] M. P. Véstias, "High-Performance Reconfigurable Computing Granularity," Encyclopedia of Information Science and Technology, pp. 3558-3567, 2015.
[7] M. Wielgosz, E. Jamro, D. Zurek, and K. Wiatr, "FPGA Implementation of The Selected Parts of The Fast Image Segmentation," in Studies in Computational Intelligence vol. 390, ed, 2012, pp. 203-216.
[8] T. Saegusa, T. Maruyama, and Y. Yamaguchi, "How Fast is an FPGA in Image Processing?," in International Conference on Field Programmable Logic and Applications, FPL 2008 2008, pp. 77-82.
[9] H. M. Hussain, K. Benkrid, and H. Seker, "The Role of FPGAs as High Performance Computing Solution to Bioinformatics and Computational Biology Data," AIHLS2013, p. 102, 2013.
[10] K. Nagarajan, B. Holland, A. D. George, K. C. Slatton, and H. Lam, "Accelerating Machine-Learning Algorithms on FPGAs using Pattern-Based Decomposition," Journal of Signal Processing Systems, vol. 62, pp. 43-63, 2011.
[11] L. Woods, J. Teubner, and G. Alonso, "Real-Time Pattern Matching with FPGAs," in 2011 IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 1292-1295.
[12] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, "Medical Image Processing on The GPU–Past, Present and Future," Medical Image Analysis, vol. 17, pp. 1073-1094, 2013.
[13] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance Comparison of FPGA, GPU and CPU in Image Processing," in International Conference on Field Programmable Logic and Applications, 2009. FPL 2009, 2009, pp. 126-131.
[14] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-Window Applications," in Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, 2012, pp. 47-56.
[15] M. Papadonikolakis, C.-S. Bouganis, and G. Constantinides, "Performance Comparison of GPU and FPGA Architectures for the SVM Training Problem," in International Conference on Field-Programmable Technology, 2009 - FPT 2009, 2009, pp. 388-391.
[16] B. Cope, P. Y. Cheung, W. Luk, and L. Howes, "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study," IEEE Transactions on Computers, vol. 59, pp. 433-448, 2010.

ISSN 2348 – 7968

[17] M. Pietron, M. Wielgosz, D. Zurek, E. Jamro, and K. Wiatr, "Comparison of GPU And FPGA Implementation of SVM Algorithm for Fast Image Segmentation," in Architecture of Computing Systems–ARCS 2013, ed: Springer, 2013, pp. 292-302.
[18] S. Che, J. Li, J. W. Sheaffer, K. Skadron, and J. Lach, "Accelerating Compute-Intensive Applications with GPUs and FPGAs," in Symposium on Application Specific Processors-SASP 2008, 2008, pp. 101-107.
[19] E. Fykse, "Performance Comparison of GPU, DSP and FPGA Implementations of Image Processing and Computer Vision Algorithms in Embedded Systems," M.Sc. thesis, Department of Electronics and Telecommunications, Norwegian University of Science and Technology, 2013.
[20] A. Maghazeh, U. D. Bordoloi, P. Eles, and Z. Peng, "General Purpose Computing on Low-Power Embedded GPUs: Has It Come of Age?," in 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013, pp. 1-10.
[21] Zynq-7000 All Programmable SoC. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
[22] Vivado High-Level Synthesis. Available: http://www.xilinx.com/products/design-tools/vivado.html
[23] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, pp. 273-297, 1995.
[24] V. N. Vapnik, "An Overview of Statistical Learning Theory," IEEE Transactions on Neural Networks, vol. 10, pp. 988-999, 1999.
[25] J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," Advances in Kernel Methods—Support Vector Learning, vol. 3, 1999.
[26] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, vol. 2, pp. 121-167, 1998.
[27] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," IEEE Transactions on Neural Networks, vol. 13, pp. 415-425, 2002.
[28] K. Ta-Wen, W. Jhing-Fa, W. Jia-Ching, L. Po-Chuan, and G. Gaung-Hui, "VLSI Design of an SVM Learning Core on Sequential Minimal Optimization Algorithm," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, pp. 673-683, 2012.
[29] K.-k. Cao, H.-b. Shen, and H.-f. Chen, "A Parallel and Scalable Digital Architecture for Training Support Vector Machines," Journal of Zhejiang University SCIENCE C, vol. 11, pp. 620-628, 2010.
[30] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," Neural Computation, vol. 13, pp. 637-649, 2001.
[31] J. G. Filho, M. Raffo, M. Strum, and W. J. Chau, "A General-Purpose Dynamically Reconfigurable SVM," in 2010 VI Southern Programmable Logic Conference (SPL), 2010, pp. 107-112.
[32] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-Forward Support Vector Machine without Multipliers," IEEE Transactions on Neural Networks, vol. 17, pp. 1328-1331, 2006.
[33] R. Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," in Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, 1998, pp. 191-200.
[34] C. H. Peng, B. W. Chen, T. W. Kuan, P. C. Lin, J. F. Wang, and N. S. Shih, "REC-STA: Reconfigurable and Efficient Chip Design With SMO-based Training Accelerator," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, pp. 1791-1802, 2014.
[35] L. Bustio-Martínez, R. Cumplido, J. Hernández-Palancar, and C. Feregrino-Uribe, "On the Design of a Hardware-Software Architecture for Acceleration of SVM's Training Phase," in Advances in Pattern Recognition, ed: Springer, 2010, pp. 281-290.
[36] W. Jhing-Fa, P. Jr-Shiang, W. Jia-Ching, L. Po-Chuan, and K. Ta-Wen, "Hardware/Software Co-design for Fast-trainable Speaker Identification System Based on SMO," in 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2011, pp. 1621-1625.
[37] A. Patel, V. Sriram, and K. Varghese, "Hardware Accelerator for Support Vector Machine Training," Training, vol. 1, p. 2.
[38] S. Venkateshan, A. Patel, and K. Varghese, "Hybrid Working Set Algorithm for SVM Learning With a Kernel Coprocessor on FPGA," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014.
[39] T. Joachims, "Advances in kernel methods," chapter Making large-scale support vector machine learning practical, pp. 169-184, 1999.
[40] M. Papadonikolakis and C. S. Bouganis, "A Heterogeneous FPGA Architecture for Support Vector Machine Training," in Proceedings - IEEE Symposium on Field-Programmable Custom Computing Machines-FCCM 2010, 2010, pp. 211-214.
[41] M. B. Rabieah and C.-S. Bouganis, "FPGA Based Nonlinear Support Vector Machine Training Using an Ensemble Learning," in 2015 25th International Conference on Field Programmable Logic and Applications (FPL), 2015, pp. 1-4.
[42] S. Martin, "Training Support Vector Machines Using Gilbert's Algorithm," in Fifth IEEE International Conference on Data Mining, 2005, p. 8 pp.
[43] M. Ning, W. Shaojun, P. Yeyong, and P. Yu, "Implementation of LS-SVM with HLS on Zynq," in 2014 International Conference on Field-Programmable Technology (FPT), 2014, pp. 346-349.
[44] L. Wei, Z. Chen, J. Li, and W. Xu, "Sparse and robust least squares support vector machine: a linear programming formulation," in IEEE International Conference on Grey Systems and Intelligent Services-GSIS 2007, 2007, pp. 1134-1138.
[45] W. Shaojun, P. Yu, Z. Guangquan, and P. Xiyuan, "Accelerating On-Line Training of LS-SVM With Run-Time Reconfiguration," in the International Conference on Field-Programmable Technology (FPT), 2011, pp. 1-6.
[46] R. Patil, G. Gupta, V. Sahula, and A. Mandal, "Power Aware Hardware Prototyping of Multiclass SVM Classifier Through Reconfiguration," in 2012 25th International Conference on VLSI Design (VLSID), 2012, pp. 62-67.
[47] H. M. Hussain, K. Benkrid, and H. Seker, "Reconfiguration-Based Implementation of SVM Classifier on FPGA for Classifying Microarray Data," in 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2013, pp. 3058-3061.

ISSN 2348 – 7968

[48] H. Hussain, K. Benkrid, and H. Seker, "Novel Dynamic Partial Reconfiguration Implementations of the Support Vector Machine Classifier on FPGA," Turkish Journal of Electrical Engineering and Computer Sciences, 2014.
[49] C. Kyrkou and T. Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines," IEEE Transactions on Computers, vol. 61, pp. 831-842, 2012.
[50] C. Kyrkou and T. Theocharides, "SCoPE: towards a systolic array for SVM object detection," IEEE Embedded Systems Letters, vol. 1, pp. 46-49, 2009.
[51] B. Mandal, M. P. Sarma, and K. K. Sarma, "Implementation of Systolic Array Based SVM Classifier Using Multiplierless Kernel," in International Conference on Signal Processing and Integrated Networks (SPIN), 2014, pp. 35-39.
[52] B. Mandal, M. P. Sarma, and K. K. Sarma, "Design Of A Systolic Array Based Multiplierless Support Vector Machine Classifier," in 2014 International Conference on Signal Processing and Integrated Networks (SPIN), 2014, pp. 35-39.
[53] M. Ruiz-Llata, G. Guarnizo, and M. Yébenes-Calvino, "FPGA Implementation of a Support Vector Machine for Classification and Regression," in The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1-5.
[54] A. H. M. Jallad and L. B. Mohammed, "Hardware Support Vector Machine (SVM) for Satellite on-Board Applications," in 2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2014, pp. 256-261.
[55] J. Gimeno Sarciada, H. Lamel Rivera, and M. Jiménez, "CORDIC Algorithms for SVM FPGA Implementation," in Proceedings of SPIE - The International Society for Optical Engineering, 2010.
[56] H. Lamela, J. Gimeno, M. Jiménez, and M. Ruiz, "Performance Evaluation of a FPGA Implementation of a Digital Rotation Support Vector Machine," in SPIE Defense and Security Symposium, 2008, pp. 697908-697908-8.
[57] X. Pan, H. Yang, L. Li, Z. Liu, and L. Hou, "FPGA Implementation of SVM Decision Function Based on Hardware-friendly Kernel," in International Conference on Computational and Information Sciences , ICCIS 2013 Proceedings, 2013, pp. 133-136.
[58] S. Wong, S. Vassiliadis, and S. Cotofana, "A sum of absolute differences implementation in FPGA hardware," in Euromicro Conference, 2002. Proceedings. 28th, 2002, pp. 183-188.
[59] D. Anguita, L. Carlino, A. Ghio, and S. Ridella, "A FPGA Core Generator for Embedded Classification Systems," Journal of Circuits, Systems and Computers, vol. 20, pp. 263-282, 2011.
[60] D. Anguita, A. Ghio, S. Pischiutta, and S. Ridella, "A support vector machine with integer parameters," Neurocomputing, vol. 72, pp. 480-489, 2008.
[61] V. Vranjkovic and R. Struharik, "New Architecture for SVM Classifier and Its Application to Telecommunication Problems," in 19th Telecommunications Forum (TELFOR), 2011, pp. 1543-1545.
[62] V. S. Vranjković, R. J. R. Struharik, and L. A. Novak, "Reconfigurable Hardware for Machine Learning Applications," Journal of Circuits, Systems and Computers, vol. 24, 2015.
[63] S. Kim, S. Lee, and K. Cho, "Design of High-Performance Unified Circuit for Linear and Non-Linear SVM Classifications," Journal of Semiconductor Technology and Science, vol. 12, pp. 162-167, 2012.
[64] P.-T. P. Tang, "Table-driven implementation of the exponential function in IEEE floating-point arithmetic," ACM Transactions on Mathematical Software (TOMS), vol. 15, pp. 144-157, 1989.
[65] M. Berberich and K. Doll, "Highly Flexible FPGA-Architecture of a Support Vector Machine," in 45. MPC-Workshop, 2014, pp. 25-32.
[66] T. Koide, H. Anh-Tuan, T. Okamoto, S. Shigemi, T. Mishima, T. Tamaki, 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.
[67] S. Shigemi, T. Mishima, A.-T. Hoang, T. Koide, T. Tamaki, B. Raytchev, et al., "Customizable Hardware Architecture of Support Vector Machine in CAD System for Colorectal Endoscopic Images with NBI Magnification," SASIMI 2013 Proceedings, the 18th Workshop on Synthesis and System Integration of Mixed Information Technologies, pp. 298-203, 2013.
[68] S. Shigemi, "An FPGA Implementation of Support Vector Machine Identifier for Colorectal Endoscopic Images with NBI Magnification," in Proc. of the 28th International Conference on Circuits/Systems, Computers and Communications (ITC-CSCC2013), 2013, pp. 571-572.
[69] Z. Nie, X. Zhang, and Z. Yang, "An FPGA Implementation of Multi-Class Support Vector Machine Classifier Based on Posterior Probability," in Proceedings of 2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010 no. 2), 2010.
[70] Y. Ago, K. Nakano, and Y. Ito, "A Classification Processor for a Support Vector Machine with Embedded DSP Slices and Block RAMs in the FPGA," in IEEE 7th International Symposium on Embedded Multicore Socs (MCSoC), 2013, pp. 91-96.
[71] C. Liu, F. Qiao, X. Yang, and H. Yang, "Hardware Acceleration With Pipelined Adder For Support Vector Machine Classifier," in 2014 Fourth International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), 2014, pp. 13-16.
[72] D. Mahmoodi, A. Soleimani, H. Khosravi, and M. Taghizadeh, "FPGA Simulation of Linear and Nonlinear Support Vector Machine," Journal of Software Engineering and Applications, vol. 4, pp. 320-328, 2011.
[73] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Hardware-based Support Vector Machine for Phoneme Classification," in IEEE EuroCon 2013, 2013, pp. 1701-1708.
[74] M. Papadonikolakis and C.-S. Bouganis, "A Novel FPGA-based SVM Classifier," in International Conference on Field-Programmable Technology (FPT) 2010, pp. 283-286.
[75] M. Papadonikolakis and C. Bouganis, "Novel Cascade FPGA Accelerator for Support Vector Machines Classification," IEEE Transactions on Neural Networks and Learning Systems, vol. 23, pp. 1040-1052, 2012.
[76] C. Kyrkou, T. Theocharides, and C.-S. Bouganis, "An Embedded Hardware-Efficient Architecture for Real-Time Cascade Support Vector Machine Classification," in 2013

ISSN 2348 – 7968

International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013, pp. 129-136.
[77] C. Kyrkou, T. Theocharides, and C. S. Bouganis, "A Hardware-Efficient Architecture for Embedded Real-Time Cascaded Support Vector Machines Classification," in Proceedings of the 23rd ACM international conference on Great lakes symposium on VLSI, 2013, pp. 341-342.
[78] C. Kyrkou, C.-S. Bouganis, T. Theocharides, and M. M. Polycarpou, "Embedded Hardware-Efficient Real-Time Classification with Cascade Support Vector Machines," IEEE Transactions on Neural Networks and Learning Systems, 2015.
[79] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-Based Real-Time Pedestrian Detection on High-Resolution Images," in 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2013, pp. 629-635.
[80] X. Shuai, L. Yibin, J. Zhiping, and J. Lei, "Binarization Based Implementation for Real-Time Human Detection," in 2013 23rd International Conference on Field Programmable Logic and Applications (FPL), 2013, pp. 1-4.
[81] S. Xie, Y. Li, Z. Jia, and L. Ju, "Binarization-based Human Detection for Compact FPGA Implementation," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 8299 LNCS, ed, 2013, pp. 119-131.
[82] C. Kelly, F. M. Siddiqui, B. Bardak, and R. Woods, "Histogram Of Oriented Gradients Front End Processing: An FPGA Based Processor Approach," in 2014 IEEE Workshop on Signal Processing Systems (SiPS), 2014, pp. 1-6.
[83] F. M. Siddiqui, M. Russell, B. Bardak, R. Woods, and K. Rafferty, "IPPro: FPGA based image processing processor," in Signal Processing Systems (SiPS), 2014 IEEE Workshop on, 2014, pp. 1-6.
[84] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection," in 2012 IEEE Workshop on Signal Processing Systems (SiPS), 2012, pp. 197-202.
[85] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "An FPGA Implementation of a HOG-based Object Detection Processor," IPSJ Transactions on System LSI Design Methodology, vol. 6, pp. 42-51, 2013.
[86] M. Komorkiewicz, M. Kluczewski, and M. Gorgon, "Floating Point HOG Implementation for Real-Time Multiple Object Detection," in 2012 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, pp. 711-714.
[87] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "FPGA Implementation of Real-Time Head-Shoulder Detection Using Local Binary Patterns, SVM and Foreground Object Detection," in 2012 Conference on Design and Architectures for Signal and Image Processing (DASIP), 2012, pp. 1-8.
[88] M. Vergara, A. Wolf, and M. Figueroa, "A Texture-based Architecture for Face Detection in IR Images on an FPGA," in Proceedings of SPIE - The International Society for Optical Engineering, 2014.
[89] S. Martelli, D. Tosato, M. Cristani, and V. Murino, "FPGA-based Pedestrian Detection Using Array of Covariance Features," in 2011 5th ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2011, 2011.
[90] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," IEEE Transactions on Computational Imaging, vol. 1, pp. 56-70, 2015.
[91] C. Kyrkou, C. Ttofis, and T. Theocharides, "A Hardware Architecture for Real-Time Object Detection Using Depth and Edge Information," Transactions on Embedded Computing Systems, vol. 13, 2013.
[92] S. S Dhar and K. Sreeraj, "FPGA Implementation of Feature Extraction Based on Histopathalogical Image and Subsequent Classification by Support Vector Machine," IJISET-International Journal of Innovative Science, Engineering & Technology, vol. 2, pp. 744-749, 2015.