SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I10P116

Volume 12 Issue 10, 196-211, October 2025

Review Article

FPGA-Driven Machine Learning Models: Trends, Challenges, and Implementations

Sattenapalli Kalyani[1], Vydeki D[2]

[1,2]*School of Electronics Engineering, Vellore Institute of Technology (VIT), Chennai, Tamil Nadu, India.*

[2]*Corresponding Author : vydeki.d@vit.ac.in*

Received: 16 August 2025 Revised: 18 September 2025 Accepted: 17 October 2025 Published: 31 October 2025

Abstract *- FPGAs represent a robust platform for accelerating ML algorithms because they enable parallel computation and short latency while minimizing power usage. Every aspect follows computerization, and most items achieve smart functionality at present. The IoT technology of the present allows network connection through the use of IoT platforms for objects. IoT defines an innovative information system of linked devices that perform automated exchanges between equipment independent of human input. IoT systems require flexible platforms. The connection capability of IoT devices to external environments depends on Field Programmable Gate Array (FPGA) technology, which provides easy user access using low-power systems with minimal delays and exceptional precision. The scalability feature of FPGAs allows SoC implementation since designers can place various hardware clocks onto one single chip. The FPGA functions as a particular type of programmable mainframe since it receives indicators through its input pins before transforming them into outputs at its output pins. This evaluation explores recent FPGA implementation methods of ML algorithms with a specific focus on Support Vector Machines (SVMs) and their classification precision. The research evaluates various hardware system designs while evaluating their performance tradeoffs and identifies noteworthy research areas for improvement. The final part addresses directions for enhancing FPGA-based ML implementations.*

Keywords *- SVM, Embedded Systems, FPGA, Hardware Architecture, System on Chip.*

1. Introduction

The Support Vector Machine (SVM) has emerged as one of the most powerful supervised learning algorithms for classification problems across diverse domains. Its capability to create optimal separating hyperplanes and maximize classification margins makes it highly effective for applications such as image and face recognition, object detection, bioinformatics, and medical diagnostics, including cancer classification [1]. The success of SVM models largely depends on the quality and quantity of the training data, as the model relies on support vectors derived from this dataset to predict unseen samples with high precision. Compared to other machine learning algorithms, SVM consistently demonstrates superior accuracy and robustness in handling high-dimensional and complex datasets [2–5].

Despite its strong theoretical foundation and proven accuracy, deploying SVMs in embedded or real-time systems remains a significant challenge. Conventional software-based implementations, though flexible, demand high computational resources and memory capacity, making them unsuitable for low-power, resource-constrained platforms such as IoT devices and edge computing systems. Embedded environments must balance stringent performance requirements, including low latency, energy efficiency, and compact design, while maintaining acceptable classification accuracy. Achieving this balance using general-purpose processors or GPUs often leads to high power consumption and cost, thereby restricting practical deployment [6]. To overcome these challenges, Field Programmable Gate Arrays (FPGAs) have gained prominence as reconfigurable computing platforms capable of accelerating computationally intensive tasks. Their parallel processing nature, energy efficiency, and design flexibility make FPGAs ideal for implementing machine learning algorithms like SVMs in embedded contexts. Empirical studies confirm that FPGA-based implementations can outperform GPU and CPU solutions in terms of throughput, latency, and power efficiency, particularly in edge or IoT-based applications [7–12].

However, despite the increasing number of FPGA-based SVM studies, a comprehensive synthesis of existing architectures and their comparative analysis remains limited. Prior reviews, including our earlier work [13], primarily focused on the training and classification phases up to 2015. Yet, the rapid evolution of reconfigurable architectures,

hardware design tools, and real-time data processing demands calls for a broader and more updated assessment. There remains a clear research gap in understanding.

- Which FPGA-based architectural frameworks best meet embedded system constraints.
- How do these implementations trade off between accuracy, speed, and hardware cost?
- Can current FPGA technologies ensure the scalability required for modern data-driven applications?

This study addresses these gaps by systematically reviewing over forty FPGA-based SVM classification architectures proposed between 2010 and 2019. The implementations are categorized into six architectural types—parallel pipeline, systolic array, dynamic partial reconfiguration, multiplier-less, tool-assisted, and cascaded classification frameworks. The review not only evaluates their respective advantages and drawbacks but also highlights unresolved challenges, key research directions, and design considerations for future embedded SVM systems. In particular, this work seeks to answer five core questions related to architectural suitability, performance efficiency, design tradeoffs, scalability, and real-world integration of FPGA-based SVM classifiers.

The remainder of this paper is structured as follows: Section 2 introduces the fundamental concepts of SVM. Section 3 details FPGA architectures relevant to machine learning implementation. Section 4 discusses the deployment of multiple ML algorithms across various domains. Section 5 explores FPGA applications for IoT environments, and Section 8 concludes with insights and recommendations for future research directions.

2. Related Work

Machine learning has gained significant attention in various applications, including image processing, biomedical engineering, and cybersecurity. However, conventional computing platforms face challenges in meeting the real-time constraints of these applications. The computational requirements of ML models continue to grow as datasets become larger and models become more complex. GPUs and TPUs have traditionally been used to accelerate ML computations, but they often suffer from high power consumption and limited adaptability in embedded environments. FPGAs offer a promising alternative due to their ability to execute computations in parallel, high-energy efficiency, and reconfigurability.

Unlike GPUs, which are optimized for general parallel processing, FPGAs can be customized at the hardware level to optimize performance for specific ML tasks. This customization enables efficient hardware utilization, reducing latency and improving throughput. Moreover, FPGAs provide flexibility in algorithm implementation, allowing real-time adaptations without requiring extensive reprogramming. Recent advancements in FPGA-based ML acceleration have demonstrated significant performance improvements in various domains. For example, FPGA implementations of deep learning models have enabled real-time inference in autonomous vehicles, industrial automation, and edge computing applications.

Furthermore, FPGA-based Support Vector Machines (SVMs) have been successfully employed in hyperspectral image classification, medical diagnostics, and financial fraud detection. Despite these advantages, FPGA implementations of ML models come with their own set of challenges. The design complexity, need for specialized hardware expertise, and resource constraints of FPGAs require careful optimization to achieve maximum efficiency. High-Level Synthesis (HLS) tools have helped bridge the gap between software and hardware design, making FPGA development more accessible to ML researchers. However, there remains a tradeoff between accuracy, computational speed, and power consumption that must be addressed when implementing ML models on FPGAs.

This paper provides an extensive review of FPGA implementations of ML algorithms, analyzing their efficiency, accuracy, and resource utilization. Systematically explore the architectural approaches, recent innovations, and future research directions that can enhance FPGA-based ML acceleration. By examining the strengths and limitations of various FPGA-based techniques, we aim to offer insights that can guide future advancements in this field. Rupani Ajay and Pandey Diskshant published their research about IoT FPGA implementation in the International Journal of Science Technology & Engineering (IJSTE) in [14]. Their article presents a future perspective of IoT. The authors explain IoT functions as a utility that requires advanced sensing elements, together with improved actuation capabilities and enhanced communication methods, and enhanced data-driven knowledge creation from vast amounts of data.

The paper explains the capability of FPGAs to expand Internet of Things functionality across changed requests. Sang Don Kim 2015, together with Seung Eun Lee, designed an IoT platform with an Altera FPGA. A temperature and humidity sensing system can be found in the paper because the authors use an external sensor with an FPGA connecting through USART2 to display outcomes through a VGA monitor. [15]. This reference explores FPGA-based web service concepts that use a network reconfigurable FPGA for the design and architecture of web servers. A. Ruta et.al [16] explore the Fast development of FPGA-based Service-Oriented Architecture (SOA) services, enabling rapid deployment of machine vision applications. By leveraging reconfigurable hardware, modular design, and parallel processing, FPGA-based SOA enhances performance, scalability, and flexibility. This approach accelerates image processing, improves real-time analysis,

and supports adaptive machine vision solutions across various industrial applications. Multiple proposed frameworks of IoT founded on FPGA are reviewed in [17], where authors present a cost-efficient approach to implement IoT components, including TCP/IP protocols and switch systems, and data-gaining features. The authors proposed developing a multi-sensor management system that incorporates data logging and control, as well as an Internet server and server application for dedicated IP facilities for validation purposes. Gasim Alandjani, together with other authors, presents design recommendations for an ECG machine implemented on an FPGA by combining capacitance scaling technology when operating at different WLAN-specific frequencies. The purpose is to build an energy-efficient FPGA device used for ECG measurement that provides fundamental heart function evaluations.

This research uses an Xilinx Kintex-7 FPGA device for designing the ECG system. [18]. Comparative studies indicate that FPGA-based ML models outpace software-based approaches in speed and energy efficiency, making them ideal for embedded and real-time requests. Recent advancements include CNN acceleration on FPGAs [4], low-power neural networks for edge computing [5], and hybrid CPU-FPGA co-processing for deep learning [6].

3. FPGA Architecture for ML Implementation

The article starts with a general presentation of FPGA-based SVM implementation methods targeting performance requirements in real-time biomedical applications. High-accuracy solutions through advanced computational SVM software implementations should be considered the best approach for operating embedded real-time systems. The flexible reconfigurable computing platform holds the answer to developing high. The combination offers inexpensive, power-efficient computing methods [19].

The hardware implementation of SVM classifiers contains six different structural designs, which serve as categories.

1. Parallel pipelined.
2. Systolic array.
3. Multiplier-less.
4. Dynamic Partial Reconfiguration (DPR).
5. Cascaded classification.
6. Development tool-based.

A number of papers have investigated different architectures and therefore belong to a distinct class. Reconfigurable computing allows for the cost-effective and efficient implementation of SVM algorithms, making them applicable to embedded systems.

3.1. Parallel Pipelined Structure

Various researchers have developed pipelined architectures that speed up SVM classification through FPGA implementations for parallel processing. A research team applied a Virtex-6 FPGA from Xilinx to create their complete pipelined system [6]. The configuration of 760 support vectors needed 768 DSP48E1 slices and 800 Block RAMs (BRAMs) to achieve implementation of an SVM classifier. The system running at 370.096 MHz frequency produced 2.89106 classifications per second. Special design structures within a pipelined architecture aim to create a flexible SVM that effectively combines data input modifications and dimensional adjustments and support vector adaptability with kernel choice adaptability [20].

During runtime, this design enabled the dynamic selection among linear polynomial or Radial Basis Function (RBF) kernel execution. The system executed exponential computations through a combination of embedded Multiplication-Accumulation MAC units, combined with adder trees made from Look-Up Tables and Xilinx CORDIC IP cores [21]. The RBF kernel system reached an operation frequency of 50 MHz through the implementation of systems that mix single-precision floating-point number calculations everywhere with fixed-point mathematics for dot-products. A pipeline design for a universal coarse-grained reconfigurable architecture was proposed in the study [22] in order to run various machine learning tools, including SVM.

SVM: Depending on an FSM model, SVM acceleration is achieved by the use of reconfigurable blocks, partial sum multipliers, partial sum adders, and partial sum subtractors. This one demonstrated good acceleration and a respectable use of hardware resources in comparison to the software counterparts. The pipelining and parallelism capabilities of FPGAs have been used in another hardware design that was put out in [23]. In order to perform desired computations, it reads from BRAMs using generation counters and a typical single-precision floating-point representation. With a maximum clock rate of 200 MHz, the architecture achieved 97.87% simulation performance.

A two-stage pipeline-based parallel architecture with the ability to support resource sharing for executing linear and nonlinear SVM groups has been proposed in the article [24]. Typical adder and multiplier-based inner product computation circuit used a 33.8 fps frame rate and 152 MHz frequency in scanning image sizes of 640 * 480. Three-class SVM-based identification systems were proposed with various structures [25, 26]. Fixed-point operation-based variable bit-width architectures were used for achieving maximum identification accuracy and hardware area provision. Also, two-class classifiers were incorporated to achieve an 18% improvement in the processing efficiency of inner-product operations. [27] employed a more effective two-pipelined-stage architecture that achieved a system quantity of more than 100 MHz, more than 21.2 edges per additional, with more than 90% accuracy. Reference [28] used a pipelined FPGA architecture to implement a low-

complexity SVM procedure on the basis of posterior probability. Sigmoid function and addition, multiplication, and division were managed by LUTs in the architecture. Experimental evidence utilized was that there was no loss of recognition rate, and computational complexity was reduced while computing compared to the native shape's floating-point algorithm, and hence, such an algorithm can be reorganized for real-time. Implementations of a human epidermal classifier using FPGA and GPU were compared [11, 29].

Pipelined architectures were completely implemented in the FPGA implementation, and the execution on the GPU consumed more power. The implication was that the FPGA implementation was faster compared to the GPU implementation when the number of pixels in the images processed was small. But the GPU handled large pixel totals better at the expense of power consumption, thus not appropriate to implement in embedded systems.

3.2. Systolic Array Styles in SVMs

Systolic array architecture uses parallel operating methods combined with pipelining protocols to achieve faster computation times. A system that consists of a distributed array of processor elements creates efficient applications to handle data communication and memory operations [13].

The application of systolic array architecture for matrix multiplication applications uses FPGAs in parallel. The Systolic Chain of Processing Elements (SCOPE) technology was introduced by Kyrkou and Theoharides [30] in order to provide a generic systolic array for SVM object classification in integrated image and video systems. A scalable, flexible, and adaptive parallel architecture was developed following the initial concept according to [31].

The hardware components used in the tested applications managed to operate at respective frame rates of 40, 46, and 122 fps without compromising detection accuracy levels. Figure 1 shows that a Systolic Array is a parallel computing architecture primarily used for efficient implementation of matrix operations, especially in deep learning, signal processing, and scientific computing. It consists of a grid of Processing Elements (PEs) that compute in a synchronized, rhythmic fashion – much like the beating of a heart, hence the term Systolic. The diagram shows a 3×3 Systolic Array, where:

- Each block labelled PE is a Processing Element.
- Inputs A0, A1, A2... (typically matrix A columns) are fed vertically.
- Inputs B0, B1, B2... (typically matrix B rows) are fed horizontally.
- Data flows rightward and downward, performing partial computations at each PE.

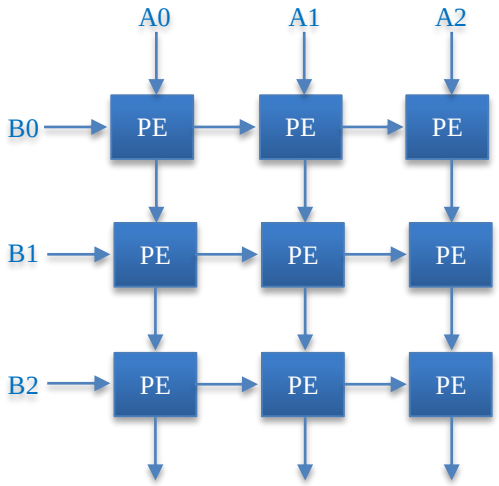

Fig. 1 Systolic array architectures

3.3. The Multiplier-Less Approach

A basic design of hardware implements computationally expensive multipliers to achieve greater simplicity. The authors in [32] developed a hardware-friendly kernel with the purpose of delivering outstanding classification results without Gaussian kernels. The research described in [33] presents a low hardware complexity design through its hardware-friendly kernel for tasks involving regression and categorization. The system increases processing speed by applying parallel shifting and addition instead of multipliers through the CORDIC algorithm, along with processing in parallel. Simulation tests on the 30 MHz 4-class classifier with a 4 % mistake rate used 75 % of the device logic for execution. Each binary classification used approximately thirty to forty support vectors while achieving zero percent floating-point error.

The multiplier-less kernel presents an alternate hardware implementation as described in [34], which operates comparably to the earlier hardware-friendly kernel described in [32]. It applies direct shifters to perform the multiplications and employs the CORDIC algorithm in the exponentiation case. Although it is not based on an FPGA-based implementation discussed within channel equalization tests, its new kernel achieves the same classification performance as the base radial kernel. Accelerated SVM has been discussed based on hardware implementation in [35].

The architecture is constructed from three kernel values computation subcircuits and employs an iterative algorithm, shifters, and adders. Six times lower computation time than other existing CORDIC circuit implementations [36] was needed for the developed scheme, and it used very few hardware resources. [37] discusses hardware SVM implementation on board with the hardware-friendly kernel. The Sum of Absolute Differences (SAD) Calculator is one of the six modules that make up the system. With just about 167 slices in the target FPGA chips, the system uses very

little power and is hence appropriate for onboard picture reduction and analysis. Accordingly, to save processing time, 1-norm vector calculations are performed using SAD-based tree building in [38]. The study investigates SVC input strictures by using fixed-point mathematics and achieves high regression accuracy with mean square error values under 0.004.

The paper proposes a parallel pipelined systolic array design with no multiplier for implementing the kernel [39]. Hardware complexity and energy consumption decrease with the shift and add operations implemented. Three different classifiers in the article reduce their electric consumption below the vector product kernel level. Research establishes hardware-friendly seeds and multiplier-less approaches as effective methods to develop SVM algorithms, which result in better FPGA-based implementation performance and reduced resource usage and system complexity.

3.4. SVM Buildings Based on DPR Technology

The modern FPGA technology incorporates DPR as a feature to enhance its hardware resource efficiency [40]. DPR enables the creation of computing systems with randomly interchangeable FPGA components that continue to operate other system components without disruption. Through the mechanism enabled by DPR, the run-time reconfiguration feature lets users share physical resources for executing different design modules at runtime [41].

The paper [42] demonstrates an SVM architecture built from four blocks through DPR and systolic array design implementation that includes Kernel Computation, Memory, Decision Making blocks, and Accumulation. A maximum of 85 times faster processing emerges from the same GPP operated on Xilinx ML 403 FPGA boards.

The application of FPGAs proves they work effectively with SVM-based bioinformatics analytical systems. The research paper [43] demonstrates methods for Xilinx partial reconfiguration, which decreases power consumption. The systolic array implementation technique simplifies design complexity, enhances memory retrieval efficiency, and enables better data infrastructure. Package reconfiguration leads to a 3–5% power consumption decrease that lowers system usage from 2.042 W to 2.021 W.

Hussain et al. [44] constructed two systolic array-based models that utilize two different dataset sizes according to their prior research [42]. The authors develop two DPR designs of SVM classifiers that operate with approximately 49x and 61x faster speeds than GPP implementations. The implementation of DPR allows execution systems to achieve substantial reductions in reconfiguration duration as opposed to traditional non-DPR deployments.

Hussain et al. introduce in their extended research an FPGA implementation of a metaclassifier architecture with SVM/KNN, which features adaptability. The implementation design allows switching between different parameters of SVM and KNN classifiers dynamically. The DPR metaclassifier employs fewer hardware resources to deliver reconfiguration times eight times faster than standard DPR use. Studies show FPGA implementations benefit significantly from DPR and systolic array approaches to implement SVM because they achieve order-of-magnitude speedups, together with reduced power consumption, optimized memory usage, and dynamic classifier switching capabilities. Figure 2 shows that Dynamic Partial Reconfiguration (DPR) architectures are a feature of modern FPGAs that allow the modification of a portion of the FPGA logic (called a Reconfigurable Region) while the rest of the FPGA (Static Region) continues to operate without interruption.

- Static FPGA Region: The part of the FPGA that runs continuously and is not affected by reconfiguration.
- PR (Partial Reconfiguration) Region: The reconfigurable part that can be dynamically modified by loading different Reconfigurable Modules (RMs).
- RM1, RM2, RM3: Different Reconfigurable Modules, each implementing a different functionality.
- Arrow (↔) between RMs and PR: Indicates that these modules can be loaded dynamically into the PR region one at a time during operation.

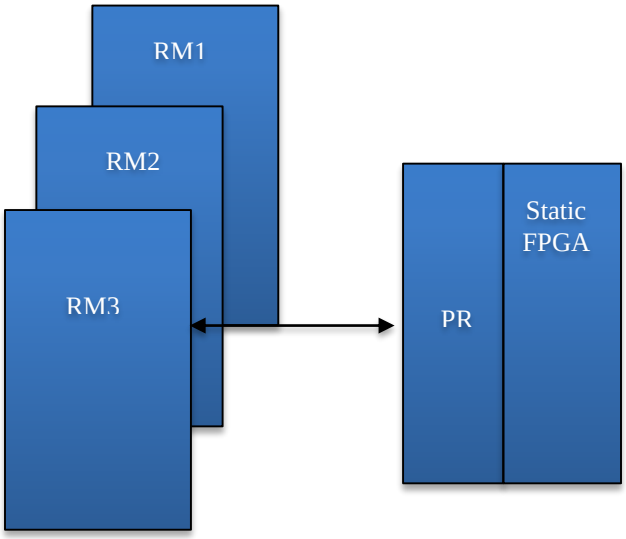

Fig.2 DPR architectures

3.5. Structures based on Cascade Classification System

To accomplish quicker organization, a cascaded classification architecture consists of many classification layers that operate sequentially Figure 3. A Cascaded Classifier is a sequential classification model where multiple classifiers are arranged in a series (cascade). Each classifier in the sequence is responsible for filtering out negative (non-

relevant) samples early, allowing only potential positive (relevant) samples to pass through to the next stage. This design is especially efficient and fast, reducing computational overhead by stopping the classification process as soon as a sample fails any stage. The diagram illustrates a 3-stage cascaded classifier system.

Stage 1: Classifier 1

- The input sample is passed to Classifier 1.
- If the sample fails, it is immediately rejected (no further processing).
- If it passes, it moves to the next stage.

Stage 2: Classifier 2

- Similar logic: a failed sample is rejected early, saving computation.
- Only those who pass go further.

Stage 3: Classifier 3

- The most refined classifier evaluates the final few candidates.
- Only if the sample passes all classifiers is it declared as positive.

Figure 3 reference in order to operate on a Virtex 5 FPGA, [45] built a hardware model for SVM cascade dispensation and reduction hardware. The multiplier-less methodology, which transforms shift operations into multiplication operations, is used in this method. The cascaded hybrid architecture combines sequential execution with parallelism through its usage of simple pipelined PEs that perform basic operations and follow up with an SVM classifier of advanced order. Parallel SVM performs recognition of 640 × 480 images at a rate of about 70 fps on average and shows five times higher speed than the single version parallel SVM. The implemented hardware reduction technique cuts down traditional logic capitals by 43% while reducing power by 20% while causing only a 0.7% accuracy decrease in an 84% classification success rate.

A parallel FPGA-based SVM classifier using cascaded design and pipelined multipliers and calculator tree for kernel calculation was reported in [46]. The information path contained precision sections for fixed-point along with single-floating-point values. The heterogeneous architecture features became an opportunity to develop a new cascade classifier system that followed design principles suitable for hardware applications.

The proposed system used two interconnected classifiers, where a basic low-precision unit was combined with an area-expensive higher-precision component. Continuous data processing on heterogeneous deployment yielded seven times better performance than CPU-based systems while surpassing the performance of FPGA and GPU-based systems.

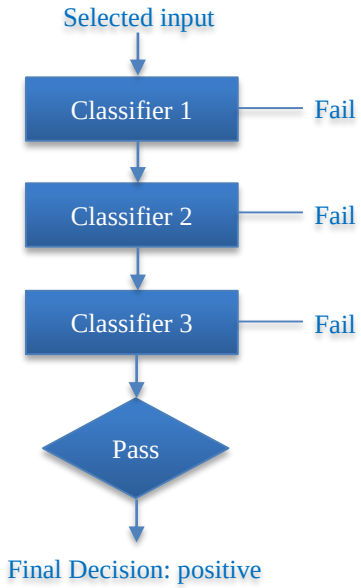

Fig. 3 Cascaded classifier architectures

The researchers in [47] expanded the problem by adding FPGA reconfigurability features that allowed. Higher performance was enabled through the implemented improvement. The heterogeneous FPGA classifier used customized word sizes for the dataset dynamic ranges, which enabled both parallelization optimization and circuitry customization. The present work [45] benefits from feature extraction capabilities. According to [48, 49] to advance detection accuracy in its proposed architectural design. The design emphasizes cascaded SVMs' effective hardware implementation, which drives clever embedded devices that perform real-time task classification.

The response evaluation process utilizes Neural Networks-based Classifiers, which decreases the number of examples used ahead of the final complex stage, thus improving classification speed. High-resolution image classification running at real-time reaches an 80% detection rate through a Spartan 6 FPGA platform while operating at 5 times the speed of a parallel SVM classifier operating individually. The approach using FPGA hardware reduction results in lower custom-logic resource consumption and lower maximum power levels that reach 25% and 20% respectively, as compared to traditional hardware design.

The classification accuracy exhibits minimal deterioration while the reduction reaches 0.7% points. This paper details the design of a Zynq SoC hybrid platform for melanoma clinical image classification through a cascade SVM classifier obtained from an extensible multicore architecture [50]. The digital system employs 34% of FPGA resources and operates at 2 watts of power since it demands minimal energy and system resources. The classifier functions

at high efficiency (1.8 μ s) with 97.9% classification accuracy, which makes it suitable as an essential part of a cost-effective portable medical scanner for diagnosing skin cancer.

3.6. Software Evolving SVMs Utilizing Design Tools

Most authors have not used the innovative software project tools that reduce complicated hardware design workloads and reduce development time while departing from standard HDL programming. The reviewed papers utilize the Generator Xilinx System alongside High-level Synthesis (HLS) and Synopsys Signal Processing Workbench (SPW) for new software design and development purposes. According to [51], Xilinx System Generator by Xilinx enables researchers to design similar hardware configurations for Xilinx FPGAs.

The platform enables users to create system models using MATLAB/Simulink for high-performance automatic design generation. The process of quantization utilizes serial and parallel methods for implementing fixed-point numbers. The paper describes how Vivado HLS integrates an enhanced approximations SVM accelerator that detects arrhythmia from ECG signals. The HLS-SVM classifier achieved 96.7% accuracy in identifying arrhythmia in ECG signals using 15 x acceleration speed.

The designers used HLS [52] to build an Energy-Efficient Embedded Binarized SVM (eBSVM) system, which operates on binarized weight and input data [53]. The solution used 3.6x more efficient energy than CPU and GPU executions and finished 45x faster with a minor 1.6x drop in accuracy. All the SVM configurations investigated for analysis appear in Table 1 for complete reference.

4. FPGA Implementation of ML Algorithms for Various Applications

Machine learning is a rapidly increasing technique, which not only creates machines that are intelligent but also machines that can work independently of any command. The machine learning techniques are employed in various applications such as health care, smart cities, and automobiles.

4.1. Health Care

In medical environments, FPGAs may be used to provide real-time processing to clinical decision support systems, telemedicine equipment, predictive health monitoring, and wearable medical technology. High-speed and accurate processing of gigabytes of information is required in these applications, and FPGAs provide the brawn and low latency required for their accomplishment. Real-time Electrocardiogram (ECG) classification is probably the most significant use of machine learning-based FPGA technology in medical environments. Analysis of ECG signals through ECG classification determines the identification of various heart disorders, ranging from arrhythmias to heart diseases.

Table 1. A Summary of different architectures for SVM

Reference	Hardware implementation	SVM Type	Kernel FPGA	Platform	Application
[6] [20] [22] [23] [24] [25,26] [28] [11, 29]	Parallel pipelined Parallel pipelined Parallel pipelined Parallel pipelined Parallel pipelined Parallel pipelined Parallel pipelined Parallel pipelined	Binary Binary Binary Multiclass Binary Multiclass Multiclass Binary	RBF Polynomial Sigmoid Linear RBF Polynomial RBF Polynomial Linear RBF Linear Linear Sigmoid Gaussian	Virtex-6 Xilinx ML505 Xilinx Virtex-7 Xilinx ML510 (Virtex-5 FXT) ------- Altera Stratix-IV Xilinx Virtex-5 Xilinx Virtex-5/Xilinx	– Pedestrian detection UCI datasets Facial expression classification ------ Colorectal cancer detection Language Recognition Skin classification
[31] [33] [34] [35] [38] [39] [42, 44, 53]	Systolic array Multiplier-less Multiplier-less Multiplier-less Multiplier-less/Systolic array DPR	Binary Multiclass Multiclass Binary Binary Binary/multi Binary	RBF Polynomial Hardware-friendly Digital kernel Hardware-friendly Hardware-friendly Linear polynomial Linear	Spartan 6 Xilinx ML505 Cyclone II (EP2C20) – – – Xilinx Virtex-5/Spartan-3E Xilinx Virtex-7 Xilinx ML403	Object detection Image recognition UCI datasets – Satellite onboard Fisher's iris dataset biomedical data classification
[45] [46,47] [48] [49] [50]	Cascaded classification Cascaded classification Cascaded classification Cascaded classification Cascaded	Binary Binary Binary Binary Binary	Linear polynomial Gaussian polynomial sigmoid Linear polynomial	Xilinx ML505 Altera's Stratix III Xilinx Spartan-6 Xilinx Spartan-6 Xilinx Zynq-7	Face detection MNIST dataset Face detection Face detection/pedestrian detection

	classification/Development tool		Linear polynomial Linear		Melanoma detection
[51] [54] [52] [53] [54,57] [56] [54] [55]	based on development tools based on development tools Development tool-based Development tool-based Development tool-based based on development tools based on development tools	Multiclass Binary Multiclass Multiclass Binary Binary Binary Multiclass	Linear Gaussian RBF Linear RBF Linear RBF RBF Linear	Xilinx Virtex-4 Xilinx Zynq-7000 Xilinx Virtex-5 Xilinx Virtex-II Xilinx Zynq-7ZC702 Zed board Zynq Zynq-7 ZC706 Xilinx ML605	Persian handwritten digits dataset Ultrasonic flaw detection Female facial expression classification Multi-speaker phoneme recognition Melanoma detection ECG-based arrhythmia detection ECG-based arrhythmia detection MNIST and CIFAR-10 datasets

Real-time ECG classification works through FPGAs and Deep Neural Networks (DNNs), which shows accuracy in real-time ECG signal processing [54]. Real-time ECG analysis represents one of the medical applications that utilizes FPGA-based machine learning technologies.

ECG analysis includes signal processing to discover whether epilepsy, along with brain tumors and other neurological conditions, exists in patients. FPGAs assist in processing real-time ECG through DNNs that function effectively in analyzing EEG signals in real-time.

4.2. Agricultural

Water scarcity, excessive rainfall, plastic items, and artificial fertilizer substances have polluted the agricultural lands across different regions. Farmers receive support from researchers who apply machine learning methods to various agricultural aspects. The primary applications of machine learning algorithms focus on four main aspects of agriculture, starting from plant surveillance through soil evaluation to detecting or predicting in agricultural operations and monitoring livestock.

4.3. Smart Cities

Real-time decisions in Autonomous driving solutions make use of DNN implementations on FPGAs to analyze sensor data in automotive applications. The combination of object detection, tracking, and lane detection defines FPGAs as the dependable choice for autonomous driving applications since they execute complex computations in real time with quick response times. The ability for FPGAs to implement custom functionality makes them perfect choices to run autonomous driving systems at their required performance levels.

The power usage of FPGAs stands out against conventional computing systems because of their superior efficiency rate. Autonomous vehicles need powerful systems, yet their limited power consumption must enable long battery life, making this an important issue in autonomous driving technology [56, 57]. The integration of FPGAs to conduct ML functions in autonomous vehicles produces several drawbacks despite their positive aspects. Prototyping requires developers to optimize machine learning algorithms for FPGAs, guarantee system reliability, and test methods to secure the system [58].

5. FPGA For IoT

FPGA stands as a semiconductor device made up of permanent transistors that amount to thousands or millions to execute logical computing commands. Several FPGA-based systems include computers, automobiles, radars, missiles, and aircraft. Xilinx, together with Altera and Quick Logic, represents some of the companies that manufacture FPGA kits [59].

The application of FPGAs in Internet of Things systems brings several advantages, which combine hardware and software capabilities through flexibility and reliability, while providing cost reduction, quick market entry, and stable maintenance benefits. Microsemi Power Meters presents most of the FPGA-based IOT system designs in various types of devices [60].

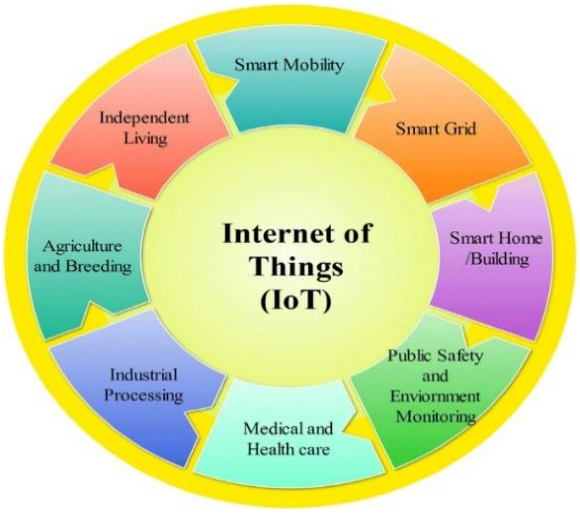

Fig. 4 IoT Applications

FPGAs have both harmonized hardware components with software elements and deliver three significant advantages and five supplementary benefits, which are:

1. Flexibility,
2. Reliability,
3. Low cost,
4. Rapid time-to-market and
5. Long-term maintenance.

The observation demonstrates that an FPGA serves as the optimal hardware choice for implementing IoT applications. The company, previously known as Altera but now called Intel FPGA, provides various FPGA-based IoT systems. Altera website [61] delivers machine vision systems together with car automotive systems, healthcare system infrastructure, smart grid control, and smart city infrastructure and building automation. The manufacturers gained multiple advantages from Altera, which include:

1. Cost reduction by avoiding ASICs' lengthy.
2. A quicker time to market by bypassing the drawn-out and dangerous development procedure.
3. Combining many ASSP functionalities within FPGAs reduces costs and adds uniqueness.
4. Design time and field programmability.
5. Reusing a single hardware platform for several systems with a single original design.
6. Making use of a number of industry protocols and standards.

An FPGA for IoT is also available from Xilinx. All of the test designs are available on the Xilinx website.

1. The Xilinx Spartan-7 FPGA kit is used to provide single-axis motor control.
2. Zynq-7000s are used in the design of multi-axis motor control. Every single one of them is adjustable.
3. Zynq-7000s All Accessible Soc or Xilinx Kinetex-7 FPGA may be used to control motors.

5.1. Advantages of FPGA for IoT

An FPGA has a grid of reconfigurable gate arrays that offer a logic motherboard. Once set up in the logic arrays, the gates are interconnected so they form a hardware application of software code. Increasingly more facilities are allowing entrenched control system developers to design and alter FPGA-based systems faster and easier.

FPGAs are not computers; FPGAs utilize dedicated hardware for processing logic and don't need an operating system. Processing pathways are parallel, which eliminates the need to employ the same resources for different processes. FPGAs' reconfigurability gives designers unlimited freedom. Christian Fritz, product manager for National Instruments motion control and mechatronics, said: “While in the past there were special Printed Circuit Board (PCB) designs that were taken from special hardware resources, FPGA-based systems can essentially rewire the internal circuits in attempting to offer configuration after control system installation to the field”.

FPGA-based platforms may be used for IoT prototypes for two purposes: Depending on the intended usage, a designer may first choose to employ an FPGA-based stage or target an FPGA to create a fully customized SoC request for which the FPGA serves as a stand-in. This allows for real-time functional and timing verification in SoC prototyping. The second issue is that early low-cost platform design development cannot be safely done with an FPGA-based prototype; instead, design space exploration, component selection, modelling, design entry, and verification must all be automated. Furthermore, because it is an end-of-life SoC prototype, the performance, power, and size that are attained may not be ideal. There should be a guarantee that all design stages are made automated in such a way that a prototype is created quickly and effectively, and it is easy to project the prototype to a finished SoC design [14].

5.2. Ransomware Detection

Ransomware is typically injected into a victim's system when they open a malicious file or click on a compromised link, often delivered through phishing emails or infected web pages. Once executed, the ransomware begins encrypting accessible files on the system, rendering them unusable and initiating the incident.

When it operates, the ransomware initiates an encryption process for system data. The ransomware intrusion method determines whether it has encryption keys and host information when connecting with the Control and Command (C & C) server. The system screen displays a ransom note which shows both security restrictions about the system locks and details about how to make payment. Data loss becomes a risk if the victim ignores the payment deadline set by attackers. After payment of ransom, the victim receives a decryption key to restore data, but the attacker might not deliver the data as promised [60].

5.3. Malware Detection

With the aid of malware software, attackers execute multiple unlawful operations, including data theft, denial-of-service assaults, and root access procurement. The number of dangerous malware threats grows exponentially today. Reported by McAfee threats report [61] from March 2016 shows that the previous quarter of 2015 saw 42 million malware samples total, with new threats occurring at a pace of 316 per minute. Modern computational devices within mobile technology and IoT distributions create an immediate need to enhance malware detection methods. The removal of harmful programs through Anti-virus (AV) software faces multiple substantial disadvantages.

The main limitation of traditional AV software emerges from its static signature-based detection method, which

identifies malware. Such a detection mechanism searches for suspicious byte patterns in the program. Hackers program malware to fake the signature of innocent software, thus tricking AV software into providing protection. AV software remains vulnerable to identical security exploits like other programs, thus putting the protection at potential risk. Security researchers have directed their focus toward hardware-based security solutions in recent times. Demme et al. [62] demonstrated how supervised ML classification of Hardware Performance Counters (HPC) traces from both malware and benign programs achieved highly accurate running application identification.

Hardware detectors enable fast online identification and efficient resource utilization, as well as remaining inaccessible to malicious infections, which qualifies them as threat mitigation tools against recent cyber risks. A set of design issues prevents the use of hardware-based detectors for HPC monitoring because they require online HPC observation capability with low misdiagnosis rates and minimal power usage during processor implementation, along with prompt HPC reading and classifier processing.

6. Computing Process

The computing process serves as our starting point since it directs both the usage of optimization methods, evaluation methods, and the selection of target devices.

6.1. Reasons for Dominance of Inference

6.1.1. Low Latency Requirements

The response time and speed of inference constitute a critical requirement in specified application domains, including real-time image recognition. The ability of FPGAs to be customized and process multiple operations simultaneously helps satisfy low-latency application needs. The implementation of ML in real-time systems requires the resolution of major latency and performance impediments. Safety in autonomous driving and video surveillance directly depends on latency, which should be below one millisecond according to [63-68]. The growth of the Internet of Things (IoT) right alongside edge computing [69-74] imposes the need to analyze massive sensor data through resource-limited devices effectively. The real-time performance needs of applications must increase because they must analyze enormous data streams [68, 75, 76]. Academic fields have proposed different hardware acceleration models because FPGAs have distinct customizability and parallel computing abilities.

6.1.2. Efficiency Considerations

The design of systems with low power consumption stands as a paramount consideration for edge computing and other applications that especially need it in IoT and mobile devices. Battery-operated IoT devices need to achieve maximum energy efficiency [68, 71-76]. FPGAs deliver energy-efficient solutions that benefit mobile AI applications [77, 78] as well as those mentioned in [79-82]. The fixed inference tasks performed by FPGAs achieve superior energy efficiency through multiple operational mechanisms. A custom-made hardware system reduces power waste through design optimizations that target specific computational operations [79-82]. The inference operates with deterministic data flow, which enables efficient data transmission procedures [83-86] as a complement to this stability.

6.2. Challenges and Potential of Training Acceleration Research

6.2.1. Data Processing Complexity

The data processing needs encounter multiple obstacles when deploying FPGA accelerators intended for training purposes in AI applications. The three main obstacles that face FPGA accelerators in AI training include the computational intensity, data handling complexities, and training across multiple machines. Large-scale dataset processing puts leading-edge stress on FPGAs to maintain their computational applications and memory storage capacity. The major limitation of FPGAs' memory capabilities ensues from the restricted on-board memory capacity that reduces data processing throughput [87-90]. The off-chip memory communication bottlenecks intensify data transmission's performance difficulty, leading to processing inefficiency during AI training operations.

6.2.2. The Hardware Systems

Hardware systems' design experiences problems when executing complex algorithms, including backpropagation. The backpropagation algorithm displays complexity during the operating process because it contains intricate steps. The gradient calculation process requires attached complexity because it includes mathematical procedures with complex steps at various propagation levels [91]. Complexity increases as a result of multi-level computational structures, with a special emphasis on executing them for large models. Stochastic Gradient Descent (SGD), together with its variants, introduces operational difficulties when dealing with random elements and automatic learning rate adjustments [91].

6.2.3. The Fundamental Role of Matrix Operations

Research papers use matrix multiplication in small numbers, yet this essential ML operation substantially affects performance despite its minimal distribution (6%). Matrix operations demonstrate automatic performance improvement potential for the majority of models because they undergo optimization. Deep learning, along with neural network operations, runs primarily on matrix multiplication as its fundamental computational base. The performance speed of sophisticated neural networks faces a key limitation from matrix multiplication because these models require complex computational operations. The FPGA platform reaches optimized matrix multiplication performance gains at 32-bit floating point precision through its parallel processing ability.

The current architectures address various classification and detection tasks, with the exception of detecting objects other than the ones specified. The implementation of SVM operates on distinct data sets through specified approaches or functions within the system without explicit application-based attempts. The fundamental objective of implementing SVM on an FPGA revolves around executing complex mathematical operations of kernel-based computations.

Among the research works, researchers have applied multiple kernel types, leading to a total of 23 linear kernel-based studies and 12 multinomial kernel implementations, alongside 4 sigmoid kernel implementations. Out of the literature that is currently accessible, 15 studies use the Gaussian RBF kernel that is derived from the complex exponential function. Five published papers use the suggested hardware-friendly kernel as an RBF substitute for the rudimentary FPGA/hardware implementation. The authors proposed an alternative kernel to operate as an opposing hardware-friendly kernel, but developers have not applied it to FPGA hardware or implemented it using this kernel design [20].

The implementations of hardware occur on previous FPGA versions with conventional design approaches, even though using contemporary growth tools. Each researcher adopts Xilinx series-7 technology for their work [51-53]. One of the studies is notable in that it uses the mixture character of the newest Zynq-7 SoC platform and exploits the newest Ultra-Fast HLS design practice ology [51, 52] with a follow-up new study on design space exploration using SVM implementation in [42].

In addition to the typical FPGA parallel architecture, the authors also employed several hardware methods to speed up their SVM applications, which are categorized into six categories in this paper. 30 papers employ parallel pipelined standard architecture (category A) through general-purpose. 11 designs employ multiplier-less structure (category D), highly interesting towards a multiplier-free design with decreased hardware complexity. 6 designs make use of parallel pipelined systolic array architecture (category B) with high interest for minimizing the complexity of multiplication. There are 6 implementations based on the common FPGA-based DPR feature (category C) to enable improved design and hardware outcomes [42, 44, 53] (3 papers) who visualize heterogeneous architecture and hybrid architecture realized by various approaches of categories A, C, and D.

Scientists from a different research group [45, 46] contributed a cascade SVM design to literature which executes integrated hardware strategies from categories A, C and E. A number of studies examined how bit-width precision with fixed-point number formatting affects classification accuracy rates during hardware implementation [29, 31, 39, 45, 51, 53].

The goal of these implementations is to minimize hardware space requirements without reducing accuracy performance. Afifi et al. [51, 52] examined the relationship between implementation speed and hardware resources consumption and area for several hardware architectures (HLS optimization directives). They later conducted research to explore design alternatives through similar analysis in [56, 57]. The implementation of multiple acceleration techniques together with hardware simplification strategies led to reduced accuracy levels in specific classification evaluations.

7. Conclusion and Future Directions

The paper provides a distinct survey investigation that demonstrates different hardware structures employed for implementing SVM classifiers on FPGAs. The survey study includes detailed analytical comparisons as well as identification of leading research groups, and it observes research limitations and challenges alongside gaps. This research establishes its primary goal as developing a superior solution to handle the tradeoff between efficient classification results and fulfilling embedded system demands for superior performance with reduced area usage, along with minimum power consumption and cost. The identified research directions provide guidelines to hardware designers who need to solve the issues highlighted throughout this review study:

- A hardware structure needs optimization to break the tradeoff between cost-efficient, low-cost, and high-performance real-time embedded classification.
- Several existing methods can function together efficiently in order to achieve optimized results. The online classification accuracy rate of hardware systems requires enhancement through new hardware-based solutions. The hardware-suitable kernel requires improvements to maintain its classification accuracy level.
- Experts should discover practical approaches that decrease the need for memory and hardware resources during the processing of large applications.
- Very few implementations report power consumption results, making it challenging to achieve a low-power embedded system.
- The DPR feature encompasses extensive potential to deliver better speed alongside power efficiency, area efficiency, and adaptability when it is tapped effectively.
- The scientific community requires additional research effort to implement multiclass classifiers and nonlinear kernel-based classifiers.
- Engineers need to conduct research into evolvable hardware techniques to develop adaptive classification systems.
- A thorough investigation of memory management and flow control systems will lead to optimal data transmission capabilities within the classification system.

- Researchers need to investigate multicore systems to function either as an ensemble of multiclass classifiers or cascaded classification classifiers by implementing voting/controller mechanisms.

New generation FPGA devices/SoCs and contemporary development tools, including HLS, should be leveraged in hardware implementations to achieve highly efficient designs with optimized results.

References

[1] Janmenjoy Nayak, Bighnaraj Naik, and H.S. Behera, "A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges," *International Journal of Database Theory and Application*, vol. 8, no. 1, pp. 169-186, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[2] P. Sabouri et al., "A Cascade Classifier for Diagnosis of Melanoma in Clinical Images," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Chicago, IL, USA, pp. 6748-6751, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[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, no. 6, pp. 1335-1343, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[4] Reza Entezari-Maleki, Arash Rezaei, and Behrouz Minaei-Bidgoli, "Comparison of Classification Methods Based on the Type of Attributes and Sample Size," *Journal of Convergence Information Technology*, vol. 4, no. 3, pp. 94-102, 2009. [Google Scholar]

[5] Jinho Kim, Byungsoo Kim, and Silvio Savarese, "Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines," *Proceedings of the 6th WSEAS International Conference on Computer Engineering and Applications, and Proceedings of the 2012 American Conference on Applied Mathematics*, pp. 133-138, 2012. [Google Scholar] [Publisher Link]

[6] Mário P. Véstias, *High-Performance Reconfigurable Computing Granularity*, 3rd ed., Encyclopedia of Information Science and Technology, IGI Global Scientific Publishing, pp. 3558-3567, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hanaa M. Hussain, Khaled Benkrid, and Huseyin Seker, "The Role of FPGAs as High Performance Computing Solution to Bioinformatics and Computational Biology Data," *AIHLS2013*, pp. 1-4, 2013. [Google Scholar]

[8] Shuichi Asano, Tsutomu Maruyama, and Yoshiki Yamaguchi, "Performance Comparison of FPGA, GPU and CPU in Image Processing," *2009 International Conference on Field Programmable Logic and Applications*, Prague, Czech Republic, pp. 126-131, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[9] Jeremy Fowers et al., "A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-Window Applications," *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey California USA, pp. 47-56, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ben Cope et al., "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study," *IEEE Transactions on Computers*, vol. 59, no. 4, pp. 433-448, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[11] Marcin Pietron et al., "Comparison of GPU and FPGA Implementation of SVM Algorithm for Fast Image Segmentation," *Architecture of Computing Systems – ARCS 2013*, pp. 292-302, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[12] Egil Fykse, "*Performance Comparison of GPU, DSP and FPGA Implementations of Image Processing and Computer Vision Algorithms in Embedded Systems*," Master Thesis, Norwegian University of Science and Technology, pp. 1-76, 2013. [Google Scholar] [Publisher Link]

[13] Shereen Moataz Afifi, Hamid GholamHosseini, and Roopak Sinha, "Hardware Implementations of SVM on FPGA: AState-of-the-Art Review of Current Practice," *IJISET - International Journal of Innovative Science, Engineering & Technology*, vol. 2, no. 11, pp. 733-752, 2015. [Google Scholar] [Publisher Link]

[14] Ajay Rupani, Dikshant Pandey, and Gajendra Sujediya, "Review and Study of FPGA Implementation of Internet of Things," *IJSTE - International Journal of Science Technology & Engineering*, vol. 3, no. 2, pp. 104-107, 2016. [Google Scholar] [Publisher Link]

[15] Sang Don Kim, and Seung Eun Lee, "Little Core Based System on Chip Platform for Internet of Thing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 4, pp. 695-700, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[16] A. Ruta, R. Brzoza-Woch, and K. Zielinski, "On Fast Development of FPGA-Based SOA Services—Machine Vision Case Study," *Design Automation for Embedded Systems*, vol. 16, pp. 45-69, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[17] Ajay Rupani, and Gajendra Sujediya, "A Review of FPGA Implementation of Internet of Things," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 9, pp. 16203-16207, 2016. [Google Scholar] [Publisher Link]

[18] Gasim Alandjani et al., "Energy Efficient VLSI Design on FPGA Using Capacitance Scaling Technique," *Indian Journal of Science and Technology*, vol. 9, no. 36, pp. 1-5, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[19] B. Schoelkopf, K. Tsuda, and J.P. Vert, *Support Vector Machine Applications in Computational Biology*, MIT Press, pp. 71-92, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yuki Ago, Koji Nakano, and Yasuaki Ito, "A Classification Processor for a Support Vector Machine with Embedded DSP Slices and Block RAMs in the FPGA," *2013 IEEE 7th International Symposium on Embedded Multicore Socs*, Tokyo, Japan, pp. 91-96, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[21] Markus Berberich, and Konrad Doll, "Highly Flexible FPGA-Architecture of a Support Vector Machine," *MPC Workshop*, no. 45, pp. 25-32, 2014. [Google Scholar] [Publisher Link]

[22] Ray Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, Monterey California USA, pp. 191-200, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[23] Vuk S. Vranjković, Rastislav J.R. Struharik, and Ladislav A. Novak, "Reconfigurable Hardware for Machine Learning Applications," *Journal of Circuits, Systems and Computers*, vol. 24, no. 5, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[24] Sumeet Saurav et al., "Hardware Accelerator for Facial Expression Classification Using Linear SVM," *Advances in Signal Processing and Intelligent Recognition Systems*, pp. 39-50, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[25] Sujin Kim, Seonyoung Lee, and Kyeongsoon Cho, "Design of High-Performance Unified Circuit for Linear and Non-Linear SVM Classifications," *The Institute of Electronics and Information Engineers*, vol. 12, no. 2, pp. 162-167, 2012. [Google Scholar] [Publisher Link]

[26] Tetsushi Koide et al., "FPGA Implementation of Type Identifier for Colorectal Endoscopie Images with NBI Magnification," *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Ishigaki, Japan, pp. 651-654, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[27] Satoshi Shigemi et al., "Customizable Hardware Architecture of Support Vector Machine in CAD System for Colorectal Endoscopic Images with NBI Magnification," *Proceedings of the 18th Workshop on Systhesis and System Integration of Mixed Information Technologies (SASIMI2013)*, pp. 298-303, 2013. [Google Scholar] [Publisher Link]

[28] S. Shigemi, "An FPGA Implementation of Support Vector Machine Identifier for Colorectal Endoscopic Images with NBI Magnification," *Proceedings of the 28th International Conference on Circuits/Systems, Computers and Communications*, pp. 571-572, 2013. [Google Scholar]

[29] Nie Zhiliang, Zhang Xingming, and Yang Zhenxi, "An FPGA Implementation of Multi-class Support Vector Machine Classifier Based on Posterior Probability," *Proceedings of International Conference on Civil and Environmental Engineering*, Chengdu, China, 2012. [Google Scholar]

[30] Maciej Wielgosz et al., *FPGA Implementation of the Selected Parts of the Fast Image Segmentation*, Intelligent Tools for Building a Scientific Information Platform, Springer, Berlin, Heidelberg, pp. 203-216, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[31] Christos Kyrkou, and Theocharis Theocharides, "SCoPE: Towards a Systolic Array for SVM Object Detection," *IEEE Embedded Systems Letters*, vol. 1, no. 2, pp. 46-49, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[32] Christos Kyrkou, and Theocharis Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 831-842, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[33] D. Anguita et al., "Feed-Forward Support Vector Machine without Multipliers," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1328-1331, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[34] Marta Ruiz-Llata, Guillermo Guarnizo, and Mar Yébenes-Calvino, "FPGA Implementation of a Support Vector Machine for Classification and Regression," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Barcelona, Spain, pp. 1-5, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[35] Vuk Vranjković, and Rastislav Struharik, "New Architecture for SVM Classifier and its Application to Telecommunication Problems," *2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers*, Belgrade, Serbia, pp. 1543-1545, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[36] Jesús Gimeno Sarciada, Horacio Lamel Rivera, and Matías Jiménez, "CORDIC Algorithms for SVM FPGA Implementation," *Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering VIII*, Orlando, Florida, United States, vol. 7703, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[37] Horacio Lamela et al., "Performance Evaluation of a FPGA Implementation of a Digital Rotation Support Vector Machine," *Independent Component Analyses, Wavelets, Unsupervised Nano-Biomimetic Sensors, and Neural Networks VI*, Orlando, Florida, United States, vol. 6979, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[38] Abdul-Halim M. Jallad, and Lubna B. Mohammed, "Hardware Support Vector Machine (SVM) for Satellite on-Board Applications," *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Leicester, UK, pp. 256-261, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[39] Xipeng Pan et al., "FPGA Implementation of SVM Decision Function Based on Hardware-Friendly Kernel," *2013 International Conference on Computational and Information Sciences*, Shiyang, China, pp. 133-136, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[40] B.H.A.S.W.A.T.I. Mandal et al., "Implementation of Systolic Array Based SVM Classifier Using Multiplierless Kernel," *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 288-294, 2014. [Google Scholar]

[41] Luca Pezzarossa et al., "Using Dynamic Partial Reconfiguration of FPGAs in Real-Time Systems," *Microprocessors and Microsystems*, vol. 61, pp. 198-206, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[42] Trailokya Nath Sasamal, and Rajendra Prasad, "Module Based and Difference Based Implementation of Partial Reconfiguration on FPGA: A Review," *International Journal of Engineering Research and Applications (IJERA)*, vol. 1, no. 4, pp. 1898-1903, 2011. [Google Scholar] [Publisher Link]

[43] Hanaa M. Hussain, Khaled Benkrid, and Huseyin Seker, "Reconfiguration-Based Implementation of SVM Classifier on FPGA for Classifying Microarray Data," *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, pp. 3058-3061, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[44] Rajesh A. Patil et al., "Power Aware Hardware Prototyping of Multiclass SVM Classifier through Reconfiguration," *2012 25th International Conference on VLSI Design*, Hyderabad, India, pp. 62-67, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[45] Hanaa Hussain, Khaled Benkrid, and Hüseyin Şeker, "Novel Dynamic Partial Reconfiguration Implementations of the Support Vector Machine Classifier on FPGA," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 5, pp. 3371-3387, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[46] Christos Kyrkou, Theocharis Theocharides, and Christos-Savvas Bouganis, "An Embedded Hardware-Efficient Architecture for Real-Time Cascade Support Vector Machine Classification," *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, Agios Konstantinos, Greece, pp. 129-136, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[47] Markos Papadonikolakis, and Christos-Savvas Bouganis, "A Novel FPGA-Based SVM Classifier," *2010 International Conference on Field-Programmable Technology*, Beijing, China, pp. 283-286, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[48] Markos Papadonikolakis, and Christos-Savvas Bouganis, "Novel Cascade FPGA Accelerator for Support Vector Machines Classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1040-1052, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[49] Christos Kyrkou et al., "Embedded Hardware-Efficient Real-Time Classification with Cascade Support Vector Machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 99-112, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[50] Christos Kyrkou et al., "Boosting the Hardware-Efficiency of Cascade Support Vector Machines for Embedded Classification Applications," *International Journal of Parallel Programming*, vol. 46, pp. 1220-1246, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[51] Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "SVM Classifier on Chip for Melanoma Detection," *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jeju, Korea (South), pp. 270-274, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[52] Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "FPGA Implementations of SVM Classifiers: A Review," *SN Computer Science*, vol. 1, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[53] Konstantina Koliogeorgi, "Optimizing SVM Classifier Through Approximate and High Level Synthesis Techniques," *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, Greece, pp. 1-4, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[54] Mostafa Rahimi Azghadi et al., "Hardware Implementation of Deep Network Accelerators towards Healthcare and Biomedical Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 6, pp. 1138-1159, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[55] Ahmed K. Jameil, and Hamed Al-Raweshidy, "Efficient CNN Architecture on FPGA Using High Level Module for Healthcare Devices," *IEEE Access*, vol. 10, pp. 60486-60495, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[56] Yi Sun et al., "Adaptive Multi-Lane Detection Based on Robust Instance Segmentation for Intelligent Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 888-899, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[57] Kenichi Harada, Kenji Kanazawa, and Moritoshi Yasunaga, "FPGA-Based Object Detection for Autonomous Driving System," *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, pp. 465-468, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[58] Gerlando Sciangula et al., "Hardware Acceleration of Deep Neural Networks for Autonomous Driving on FPGA-Based SoC," *2022 25th Euromicro Conference on Digital System Design (DSD)*, Maspalomas, Spain, pp. 406-414, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[59] Gina Smith, *FPGAs 101: Everything You Need to Know to Get Started*, Newnes, pp. 1-245, 2010. [Google Scholar] [Publisher Link]

[60] Archit Gajjar et al., "RD-FAXID: Ransomware Detection with FPGA-Accelerated XGBoost," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 17, no. 4, pp. 1-33, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[61] "*McAfee Mobile Threat Report 2021*," pp. 1-12, 2021. [Publisher Link]

[62] John Demme et al., "On the Feasibility of Online Malware Detection with Performance Counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559-570, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[63] Hongxiang Fan et al., "A Real-Time Object Detection Accelerator with Compressed SSDLite on FPGA," *2018 International Conference on Field-Programmable Technology (FPT)*, Naha, Japan, pp. 14-21, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[64] Frank Ridder, Kuan-Hsun Chen, and Nikolaos Alachiotis, "Accelerated Real-Time Classification of Evolving Data Streams using Adaptive Random Forests," *2023 International Conference on Field Programmable Technology (ICFPT)*, Yokohama, Japan, pp. 232-237, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[65] Shuanglong Liu, and Wayne Luk, "Towards an Efficient Accelerator for DNN-Based Remote Sensing Image Segmentation on FPGAs," *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, Barcelona, Spain, pp. 187-193, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[66] Caiwen Ding et al., "REQ-YOLO: A Resource-Aware, Efficient Quantization Framework for Object Detection on FPGAs," *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Seaside CA USA, pp. 33-42, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[67] Hiroki Nakahara et al., "A Lightweight YOLOv2: A Binarized CNN with A Parallel Support Vector Regression for an FPGA," *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey California USA, pp. 31-40, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[68] Kizheppatt Vipin, "ZyNet: Automating Deep Neural Network Implementation on Low-Cost Reconfigurable Edge Computing Platforms," *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, pp. 323-326, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[69] Panagiotis Mousouliotis, Ioannis Papaefstathiou, and Loukas Petrou, "SqueezeJet-3: An Accelerator Utilizing FPGA MPSoCs for Edge CNN Applications," *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Fayetteville, AR, USA, pp. 236-236, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[70] Yuhao Liu et al., "NetPU: Prototyping a Generic Reconfigurable Neural Network Accelerator Architecture," *2022 International Conference on Field-Programmable Technology (ICFPT)*, Hong Kong, pp. 1-1, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[71] Yu Gong et al., "N3H-Core: Neuron-designed Neural Network Accelerator via FPGA-based Heterogeneous Computing Cores," *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Virtual Event USA, pp. 112-122, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[72] Nikhil P Ghanathe et al., "MAFIA: Machine Learning Acceleration on FPGAs for IoT Applications," *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, Dresden, Germany, pp. 347-354, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[73] Cecilia Latotzke, Tim Ciesielski, and Tobias Gemmeke, "Design of High-Throughput Mixed-Precision CNN Accelerators on FPGA," *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, Belfast, United Kingdom, pp. 358-365, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[74] Philip Colangelo et al., "Exploration of Low Numeric Precision Deep Learning Inference Using Intel® FPGAs," *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Boulder, CO, USA, pp. 73-80, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[75] Di Wu et al., "A High-Performance CNN Processor Based on FPGA for MobileNets," *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, Barcelona, Spain, pp. 136-143, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[76] Panagiotis Mousouliotis, Ioannis Papaefstathiou, and Loukas Petrou, "SqueezeJet-3: An Accelerator Utilizing FPGA MPSoCs for Edge CNN Applications," *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines*, Fayetteville, AR, USA, pp. 236-236, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[77] Yang Yang et al., "FPNet: Customized Convolutional Neural Network for FPGA Platforms," *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, pp. 399-402, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[78] Glenn G. Ko et al., "Accelerating Bayesian Inference on Structured Graphs Using Parallel Gibbs Sampling," *2019 29th International Conference on Field Programmable Logic and Applications*, Barcelona, Spain, pp. 159-165, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[79] Mengshu Sun et al., "Hardware-Friendly Acceleration for Deep Neural Networks with Micro-Structured Compression," *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines*, New York City, NY, USA, pp. 1-1, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[80] Mathew Hall, and Vaughn Betz, "From TensorFlow Graphs to LUTs and Wires: Automated Sparse and Physically Aware CNN Hardware Generation," *2020 International Conference on Field-Programmable Technology (ICFPT)*, Maui, HI, USA, pp. 56-65, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[81] Dionysios Diamantopoulos, and Christoph Hagleitner, "A System-Level Transprecision FPGA Accelerator for BLSTM Using On-Chip Memory Reshaping," *2018 International Conference on Field-Programmable Technology (FPT)*, Naha, Japan, pp. 338-341, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[82] Yingxue Gao et al., "SDMA: An Efficient and Flexible Sparse-Dense Matrix-Multiplication Architecture for GNNs," *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, Belfast, United Kingdom, pp. 307-312, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[83] Lucian Petrica et al., "Memory-Efficient Dataflow Inference for Deep CNNs on FPGA," *2020 International Conference on Field-Programmable Technology (ICFPT)*, Maui, HI, USA, pp. 48-55, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[84] Mohamed Ibrahim et al., "Extending Data Flow Architectures for Convolutional Neural Networks to Multiple FPGAs," *2023 International Conference on Field Programmable Technology (ICFPT)*, Yokohama, Japan, pp. 132-141, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[85] Shuo Wang et al., "C-LSTM: Enabling Efficient LSTM using Structured Compression Techniques on FPGAs," *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey California USA, pp. 11-20, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[86] Yi-Chien Lin, and Viktor Prasanna, "A Framework for Graph Machine Learning on Heterogeneous Architecture," *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines*, Marina Del Rey, CA, USA, pp. 245-246, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[87] Shreyas Kolala Venkataramanaiah et al., "Automatic Compiler Based FPGA Accelerator for CNN Training," *2019 29th International Conference on Field Programmable Logic and Applications*, Barcelona, Spain, pp. 166-172, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[88] Tamon Sadasue, and Tsuyoshi Isshiki, "Scalable Full Hardware Logic Architecture for Gradient Boosted Tree Training," *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines*, Fayetteville, AR, USA, pp. 234-234, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[89] Frank Ridder, Kuan-Hsun Chen, and Nikolaos Alachiotis, "Accelerated Real-Time Classification of Evolving Data Streams using Adaptive Random Forests," *2023 International Conference on Field Programmable Technology (ICFPT)*, Yokohama, Japan, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[90] Changjun Song et al., "MSDF-SGD: Most-Significant Digit-First Stochastic Gradient Descent for Arbitrary-Precision Training," *2023 33rd International Conference on Field-Programmable Logic and Applications*, Gothenburg, Sweden, pp. 159-165, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[91] Duncan J.M Moss et al., "A Customizable Matrix Multiplication Framework for the Intel HARPv2 Xeon+FPGA Platform: A Deep Learning Case Study," *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey California, USA, pp. 107-116, 2018. [CrossRef] [Google Scholar] [Publisher Link]