applied sciences

MDPI

Article

MLoF: Machine Learning Accelerators for the Low-Cost FPGA Platforms

Ruiqi Chen [1], Tianyu Wu [1,2], Yuchen Zheng [3] and Ming Ling [4,*]

[1] VeriMake Innovation Lab, Nanjing Renmian Integrated Circuit Co., Ltd., Nanjing 210088, China; rickychen@verimake.com (R.C.); twu15@ece.ubc.ca (T.W.)
[2] Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada
[3] Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27514, USA; felix02@email.unc.edu
[4] National ASIC System Engineering Technology Research Center, Southeast University, Nanjing 210096, China
* Correspondence: trio@seu.edu.cn

Abstract: In Internet of Things (IoT) scenarios, it is challenging to deploy Machine Learning (ML) algorithms on low-cost Field Programmable Gate Arrays (FPGAs) in a real-time, cost-efficient, and high-performance way. This paper introduces Machine Learning on FPGA (MLoF), a series of ML IP cores implemented on the low-cost FPGA platforms, aiming at helping more IoT developers to achieve comprehensive performance in various tasks. With Verilog, we deploy and accelerate Artificial Neural Networks (ANNs), Decision Trees (DTs), K-Nearest Neighbors (k-NNs), and Support Vector Machines (SVMs) on 10 different FPGA development boards from seven producers. Additionally, we analyze and evaluate our design with six datasets, and compare the best-performing FPGAs with traditional SoC-based systems including NVIDIA Jetson Nano, Raspberry Pi 3B+, and STM32L476 Nucle. The results show that Lattice's ICE40UP5 achieves the best overall performance with low power consumption, on which MLoF averagely reduces power by 891% and increases performance by 9 times. Moreover, its cost, power, Latency Production (CPLP) outperforms SoC-based systems by 25 times, which demonstrates the significance of MLoF in endpoint deployment of ML algorithms. Furthermore, we make all of the code open-source in order to promote future research.

Keywords: Internet of Things; machine learning; embedded system; FPGA; hardware accelerator

Citation: Chen, R.; Wu, T.; Zheng, Y.; Ling, M. MLoF: Machine Learning Accelerators for the Low-Cost FPGA Platforms. *Appl. Sci.* **2022**, *12*, 89. https://doi.org/10.3390/app12010089

Academic Editors: Deliang Fan and Zhezhi He

Received: 2 November 2021
Accepted: 16 December 2021
Published: 22 December 2021

1. Introduction

Machine Learning (ML) algorithms are effective and efficient in processing Internet of Things (IoT) endpoint data with well robustness [1]. As data volumes grow, IoT endpoint ML implementations have become increasingly important. Compared to the traditional cloud-based approaches, they can compute in real-time and reduce the communication overhead [2]. There are some researches deploying ML algorithms on System on Chip (SoC) based endpoint devices in industry and academia. For example, the TensorFlow Lite [3], X-CUBE-AI [4], and the Cortex Microcontroller Software Standard Neural Network (CMSIS-NN) [5] are three frameworks proposed by Google, STM, and ARM for pre-trained models in embedded systems. However, these solutions cannot achieve a balance among power consumption, cost-efficiency, and high-performance simultaneously for IoT endpoint ML implementations.

Field Programmable Gate Array (FPGA) devices have an inherently parallel architecture that makes them suitable for ML applications [6]. Moreover, some FPGAs have substantially close costs and leakage power compared to those of Microcontroller Unit (MCU). Therefore, FPGA can be an ideal target platform for IoT endpoint ML implementations. Nowadays, some researches and supports have been done for the deployment of ML algorithms on FPGAs. For instance, the ongoing AITIA, led by the Technical University of Dresden, KUL, IMEC, VUB, etc. [7], is a preliminary project that investigated the

feasibility of ML implementations on FPGAs. On the other hand, in the industry, Xilinx and Intel have supported portions of the machine learning based Intellectual Property (IP) cores [8–11]. Unfortunately, these solutions are based on high-end FPGAs and require highly professional standards. It is difficult for many small and medium-sized corporations and individual developers to deploy their hardware platforms. Therefore, an ML library that can run on any FPGA platform is needed [12], especially those low-cost FPGAs. Low-cost FPGAs do not mean they have the lowest absolute value among all FPGAs, instead, they refer to the lowest-priced FPGA series in the most representative manufacturers. Meanwhile, the lack of comprehensive comparisons makes it difficult to demonstrate the benefits of FPGA ML implementations over conventional SoC-based solutions.

Therefore, we introduce Machine Learning on FPGA (MLoF) with a series of IP cores dedicated to low-cost FPGAs. The MLoF IP-cores are developed in Verilog Hardware Description Language (HDL) and can be used to implement popular machine learning algorithms on FPGAs, including Support Vector Machines (SVMs), K-Nearest Neighbors (k-NNs), Decision Trees (DTs), and Artificial Neural Networks (ANNs). The performance of seven FPGA producers (Anlogic *2, Gowin *1, Intel *2, Lattice *2, Microsemi *1, Pango *1, and Xilinx *1) is thoroughly evaluated using low-cost platforms. As far as we know, MLoF is the first case to implement machine learning algorithms on nearly every low-cost FPGA platform. Compared with the typical way of implementing machine learning algorithms on embedded systems, including NVIDIA Jetson Nano, Raspberry Pi 3B+, and STM32L476 Nucle, the advantage of MLoF is that it balances the cost, performance, and power consumption. Moreover, these IP cores are open-source, assisting developers and researchers in more efficient implementation of machine learning algorithms on their endpoint devices.

The contributions of this paper are as follows:

(1) To the best knowledge of the authors, this is the first time that four ML hardware accelerator IP-cores are generated using Verilog HDL, including SVM, k-NN, DTs and ANNs. The source code of all IP-cores is fully disclosed at github.com/verimake-team/MLonFPGA;
(2) The proposed IP-cores are deployed and validated on 10 mainstream low-cost and low-power FPGAs from seven producers to show its broad compatibility;
(3) Our designs are comprehensively evaluated on FPGA boards and embedded system platforms. The results prove that low-cost FPGAs are ideal platforms for IoT endpoint ML implementations;

The rest of this paper is organized as follows: Section 2 reviews prior work on the hardware implementations of machine learning algorithms. Section 3 introduces details of the proposed MLoF IP series. Section 4 provides a comparison of various FPGA development platforms. Section 5 contains experiments and analyses. Section 6 concludes the research results and future works.

2. Related Work

For decades, the implementations of machine learning algorithms on low-cost embedded systems have been vigorously investigated. Due to the limited computing resources on embedded systems, these approaches tend to lack outstanding performance. However, FPGA is an effective solution for machine learning algorithms [13]. Saqib et al. [14] proposed a decision tree hardware architecture based on FPGA. It improves data throughput and resource utilization efficiency by utilizing parallel binary decision trees and a pipeline. A 3.5× computing speed is achieved while only 2952 Look-up Tables (LUTs) of resources are consumed via an 8-parallelism 4-stage pipeline. Nasrin Attaran et al. [15] proposed a binary classification architecture based on SVM and k-NN. Over 10× computing speed and 200× power-delay are obtained as compared with ARM A53. Gracieth et al. [16] proposed a 4-stage, low-power SVM pipeline architecture capable of achieving 98% of accuracy on over 30 classification tasks. It consumes only 1315 LUTs of resources and operates at a system frequency of 50 MHz. The aforementioned works introduce the deployment of ML

algorithms on FPGA platforms, but there are still deficiencies: 1. None of the works integrate the mainstream ML algorithms for FPGA deployment; 2. None of the works compare the final results horizontally with other embedded platforms of various IoT terminals.

Due to the high inherent parallelism in FPGAs, more advanced machine learning algorithms, such as neural networks, are widely studied. Roukhami M et al. [17] proposed a Deep Neural Network (DNN) based architecture for classification tasks on low-power FPGA platforms. They thoroughly compared the performance with STM32 ARM MCU and designed a general communication interface for accelerators, such as SPI and UART. The entire acceleration process consumes only 25.459 mW of power with a latency of 1.99 s. Chao Wang et al. [18] proposed a Deep Learning Accelerating Unit (DLAU) that accelerates neural networks, e.g., DNN and CNN. Additionally, they developed an AXI-Lite interface for the acceleration unit to enhance its versatility. In general, the DLAU outperforms Cortex-A7 by 67%. Fen Ge et al. [19] developed a resource-constrained CNN accelerator for IoT endpoint SoCs that does not require any DSP resources. With a total resource overhead of 4901 LUTs, the data throughput reaches 6.54 GOP (Giga Operation per second). The above works present neural network-based FPGA deployments, but none of them is designed for low-cost FPGAs, which are oftentimes the most prevalent platforms used for IoT endpoints.

Although the previous works succeed in increasing the capability of endpoint computing by implementing only one or two machine learning algorithms on FPGAs, further analyses and comprehensive comparisons across low-cost FPGA platforms, as well as the integration of more commonly used machine learning algorithms are still required.

3. Machine Learning Algorithms Implementation on Low-Cost FPGAs

Normally, the post-process of IoT data mainly focuses on two tasks: Classifications and regressions [20]. By exploiting the parallelism and low power consumption of FPGAs, MLoF offers a superior solution for these workloads. Drawn from past designs, MLoF is designed with lower computation resources, and it includes a variety of common machine learning algorithms, namely ANN, DT, k-NN, and SVM. Details will be further presented in this section.

As shown in Figure 1, the system has consisted of the training and the MLoF modules as most machine learning models are used for inferencing and evaluating within IoT devices without requiring extensive training [21], all of the model training process is completed through PC. First, the IoT dataset is gained from the endpoints and will be sent to the training module. Then, an ML library (e.g., Scikit-Learn and TensorFlow Lite) [22] and an ML algorithm should be chosen as the first level of parameters. Thereafter, a set range of hyperparameters (the same set as in the MLoF module) are used to constrain the PC training process, as FPGA has limited local resources. Since hyperparameters are key features in ML algorithms [23–25], users could set them to different values to find the best sets for training according to Table 1. Next, with all the labeled data and hyperparameters, the best model and the best parameters (including the updated hyperparameters) are generated and sent to the MLoF module. After receiving and storing the parameters into ROM or external Flash, the algorithm is deployed on an FPGA. The language of FPGA is mainly Verilog HDL, thus the algorithms can literally be deployed on any known FPGA platform.

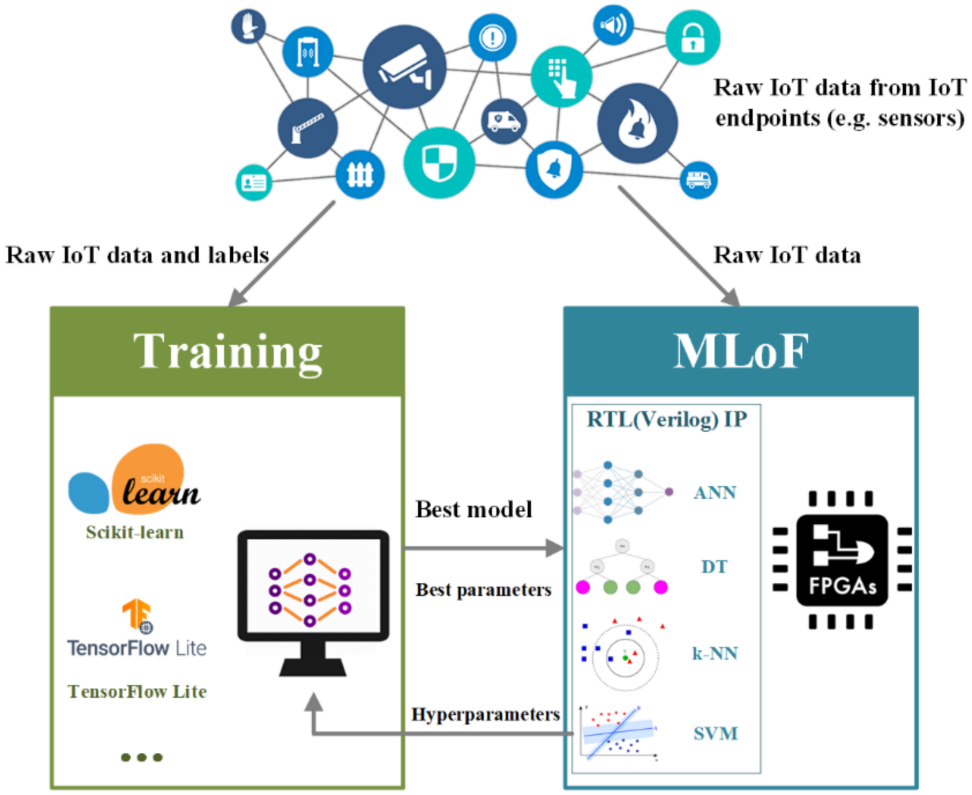

Figure 1. Block diagram of the MLoF system architecture.

Table 1. Configurable parameters.

ANN	DT	k-NN	SVM
Data width (max = 13); Number of inputs (max = 16); Number of hidden layers (max = 4); Number of neurons in each hidden layer (max = 8); Number of targets (max = 16); Activation functions	Data width (max = 13); Number of depth (max = 6); Number of leaf nodes (max = 64); Number of targets (max = 16)	Data width (max = 13); Number of inputs (max = 16); Number of neighbors (max = 16); Number of targets (max = 16)	Data width (max = 13); Number of inputs (max = 16); Number of targets (max = 16)

In this paper, we selected six datasets, the four most representative ML algorithms, and deployed them on 10 low-cost FPGAs, with a total of 240 combinations. The experiments and evaluations are described in detail in Section 5.

3.1. Artificial Neural Networks (ANN)

3.1.1. Overall Structure of ANN

The implementation of ANN, for example, with eight inputs and two hidden layers (eight neurons within each layer) is shown in Figure 2a. This ANN model includes a Memory Unit (Mem), a Finite State Machine (FSM), eight Multiplying Accumulator (MAC) computing units, Multiplexers (MUX), an Activation Function Unit (AF), and a Buffer Unit (BUFFER). The Memory Unit (Mem) stores the weights and biases after training. The FSM manages the computation order and the data stream. The MAC units are designed with multipliers, adders, and buffers for multiplying and adding operations within each neuron. Here, we implement eight MAC blocks to process in parallel the multiplication of the eight neurons. Initially, features are serially entered for registration. Then, the MAC is used to sequentially process the ANN from the first input feature to the eighth feature on the first

hidden layer (Figure 2b). Next, the second hidden layer is sequentially processed from the first hidden neuron to the eighth neuron. Finally, we process the output by the activation function. As demonstrated in Figure 3, the multiplexers (MUX) allocate the data stream. The AF computes the activation functions, which will be discussed in Section 3.1.2. The buffer stores data computed from each neuron.

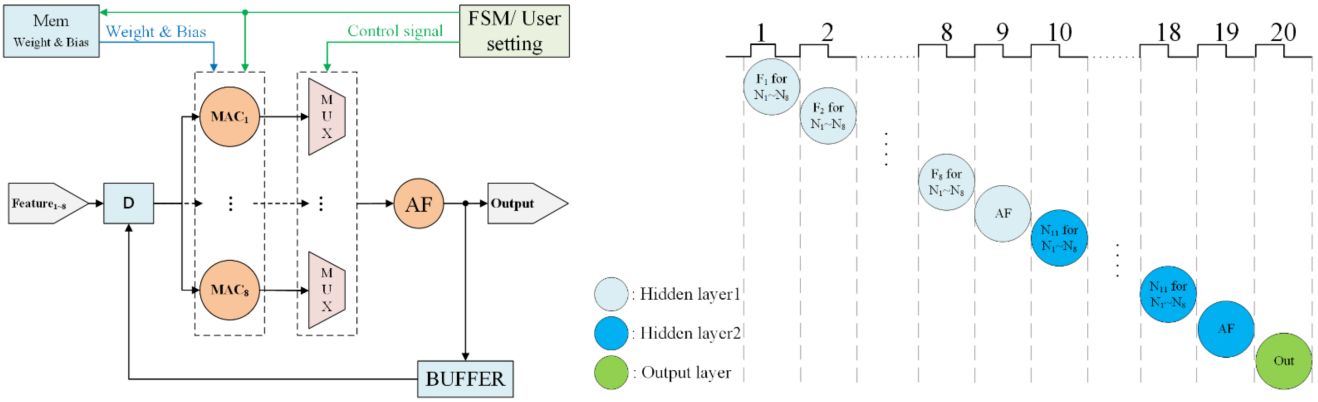

Figure 2. ANN algorithm implementation. (**a**) Block diagram of ANN. (**b**) Scheduling of the ANN processing.

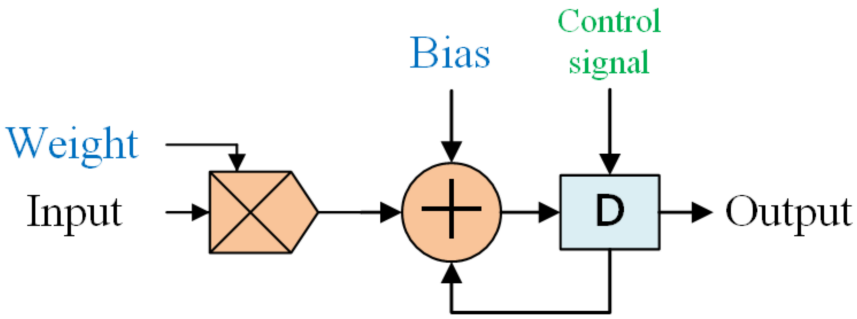

Figure 3. Block diagram of MAC.

The entire procedure for performing a hardware-based ANN architecture is described below. First, eight features are serially inputted to an ANN model. Each feature is entered into eight MAC units simultaneously and multiplications for eight neurons in the first layer are then completed. An inner buffer is used to store the multiplication results. Next, all eight results are added to a user-specified activation function. The output is further stored in the buffer unit as the input of the next hidden layer. Finally, the results are exported from the output layer following the second hidden layer.

3.1.2. Activation Function

The activation function is required within each neuron, which introduces non-linearity into the ANN, resulting in better abstract capability. Three typical activation functions include the Rectified Linear Unit (ReLU), the Hyperbolic Tangent Function (Tanh), and the Sigmoid Function [26]. All of three activation functions listed above are developed in hardware, and details are described as follows.

Rectified Linear Unit (ReLU)

The mathematical representation of the ReLU is described as Equation (1):

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (1)$$

The hardware implementation is shown in Figure 4 with a comparator and a multiplexer [27]:

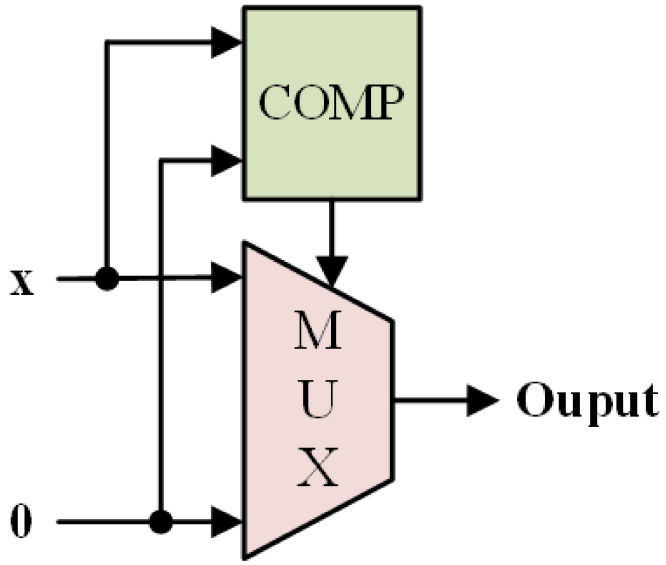

Figure 4. Block diagram of ReLU.

Hyperbolic Tangent Function (Tanh)

The mathematical representation of the Tanh function is described as Equation (2):

$$\mathrm{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

This cannot be achieved directly in the hardware using HDL. Therefore, we fit this functionality separately with five sub-intervals [28]. We divide the interval of [0, +∞] into five sub-intervals: [0, 1], (1, 2], (2, 3], (3, 4], (4, +∞). Table 2 contains the heuristic functions used to fit the Tanh function for each sub-interval. The performance of each sub-interval is shown in Figure 5 with an error kept within an acceptable range. The sub-intervals enable the Tanh function to be implemented using only adders and multipliers.

Table 2. Fitting function at different intervals.

Numerical Interval	Function	Absolute Error
[0, 1]	$y = -0.3275x^2 + 1.0977x - 0.0038$	0.0038
(1, 2]	$y = -0.1690x^2 + 0.7021x + 0.2324$	0.0039
(2, 3]	$y = -0.0282x^2 + 0.1703x + 0.7370$	0.0055
(3, 4]	$y = -0.0039x^2 + 0.0313x + 0.9363$	0.0101
(4, +∞)	1	/

Sigmoid Function

The mathematical representation of the Sigmoid function is described as Equation (3):

$$\mathrm{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Similar to the Tanh function, the Sigmoid function cannot be implemented directly in the hardware using HDL, as well. The Sigmoid function is equivalent to the tanh function [29] when the transformation in Equation (4) is applied:

$$\text{Sigmoid}(x) = \frac{\text{Tanh}(x/2) + 1}{2} \tag{4}$$

This transformation can be implemented easily on the hardware using a shift operation and an adder based on the tanh function.

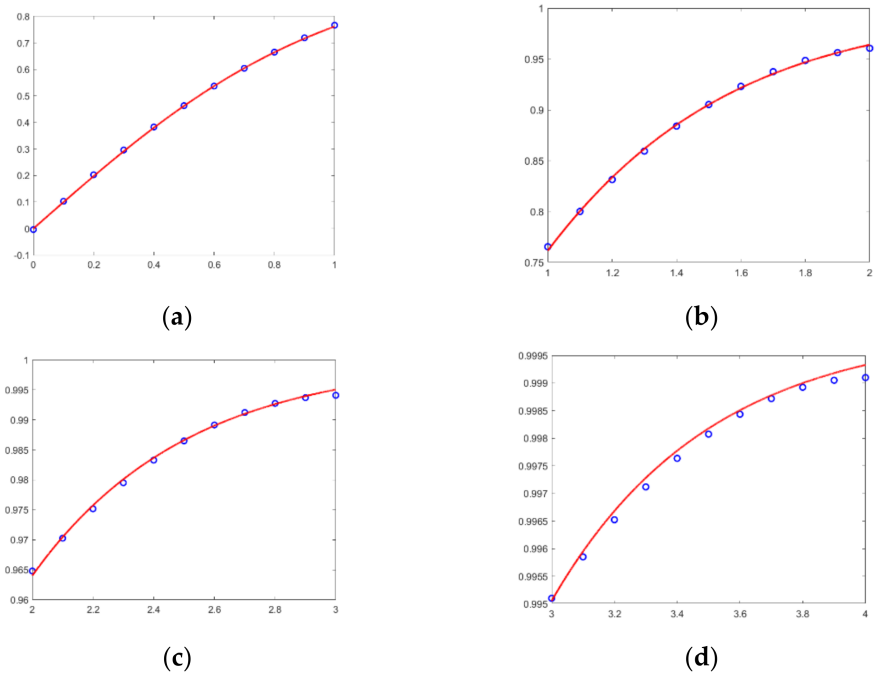

Figure 5. The images of the fitting effect of these functions. (**a**) Numerical interval [0, 1]. (**b**) Numerical interval (1, 2]. (**c**) Numerical interval (2, 3]. (**d**) Numerical interval (3, 4].

3.2. Decision Tree (DT)

Figure 6 illustrates the implementation of DT with multiple inputs, a depth of four, and a maximum of eight nodes on each layer. The DT consists of a Memory Unit (Rom), a Finite State Machine (FSM), eight compare units, and a dispatcher. The memory unit is used to store nodes' parameters from PC training. The FSM is used to determine which input node to use next based on the output. The compare unit serves as the selecting node. The distributor is used to distribute the input to each node.

3.3. The k-Nearest Neighbors (k-NN)

3.3.1. Overall Structure of k-NN

The k-NN method is used to classify the samples based on their distances. In this case, we use the squared Euclidean distance as defined in Equation (5). The structure of k-NN is demonstrated in Figure 7 with an example of eight inputs and a k-value of 6. It consists of a Memory Unit (Mem), a Finite State Machine (FSM), a subtractor, a multiplier, a buffer, an adder, and a Sorting Network and Label Finder (SNLF) module.

$$d = \sum_{i=1}^{8} (x_i - y_i)^2 \tag{5}$$

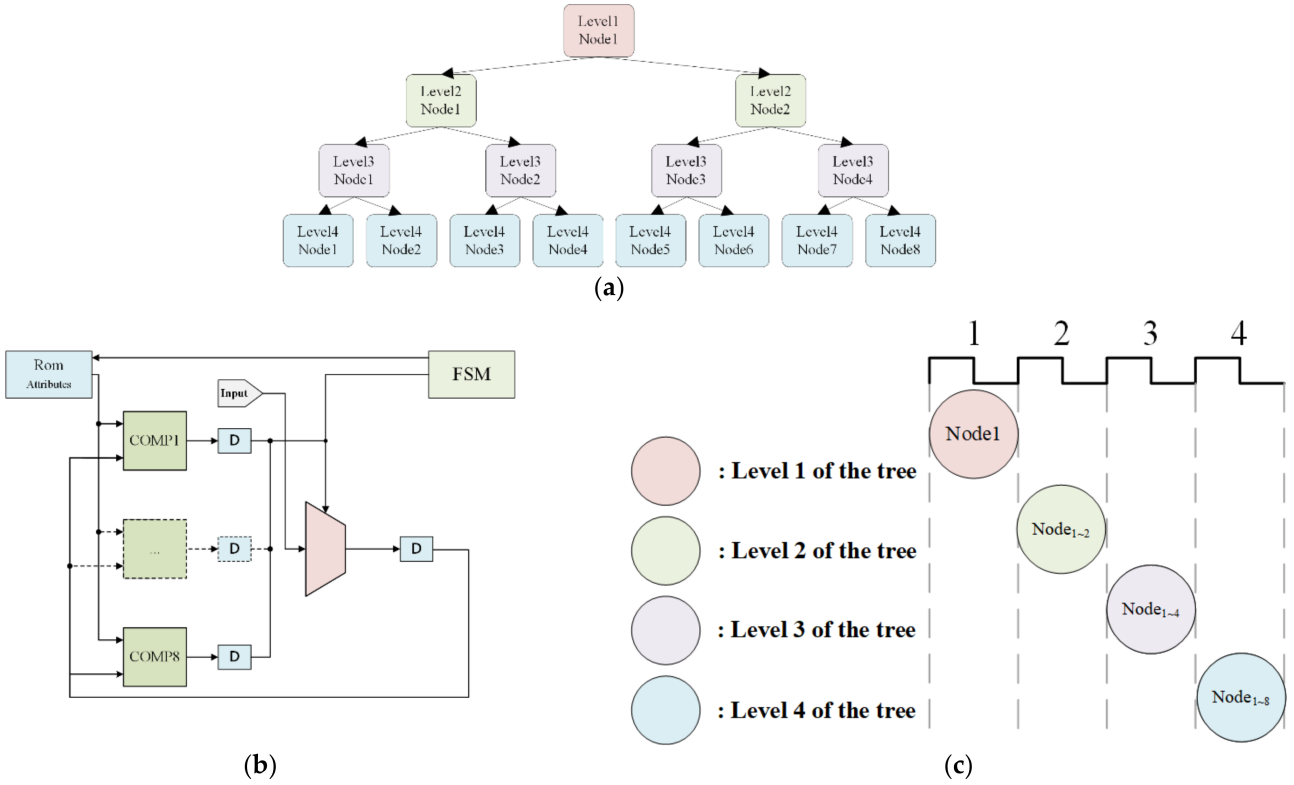

Figure 6. DT algorithm implementation. (**a**) The sample decision tree. (**b**) Block diagram of DT. (**c**) Scheduling of the DT processing.

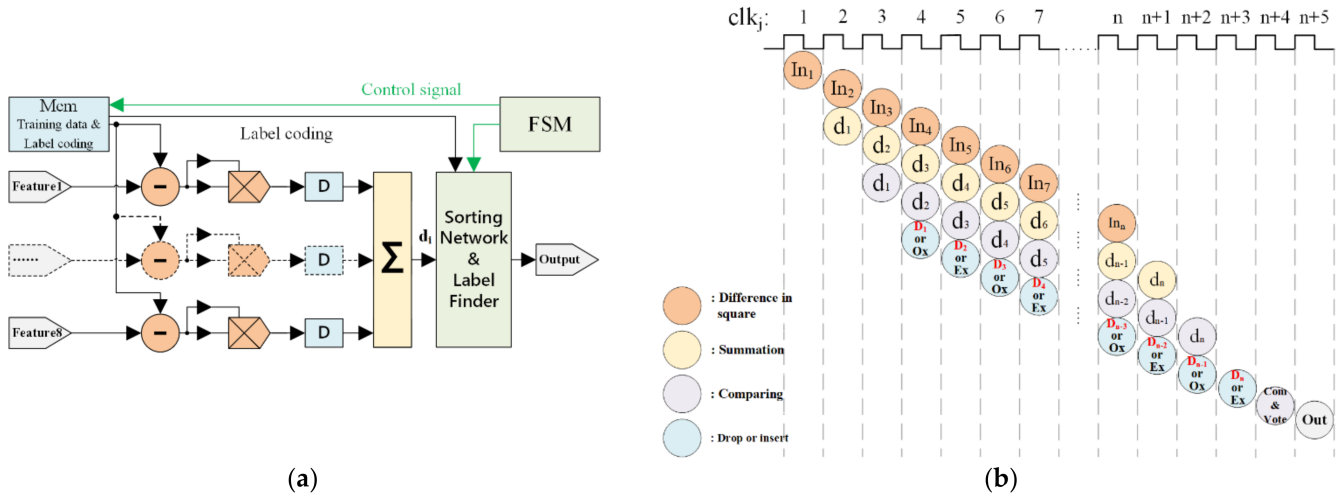

Figure 7. The k-NN algorithm implementation. (**a**) Block diagram of k-NN. (**b**) Scheduling of the k-NN processing.

3.3.2. Structure of Sort Network and Label Finder Module

The Sorting Network and Label Finder (SNLF) is a key module that completes the sorting operation of distance and then outputs the classification or prediction results. It balances the pipeline and parallel execution with a ping-pong operation. As shown in Figure 8, this module consists of three parts, MUX, comparators, and cache registers. The mux was used to control d_i transmit in different clock cycles. The comparator was used for comparison with the new d_i and for storing d_is in cache registers. There were 12 cache registers (Ox and Ex) used for storing d_i, as shown in Figure 7b. Specifically, Ox registers were used to store the six smallest d_is in 'odd' clock cycles. Ex registers were used to store

the six smallest d_is in 'even' clock cycles. The ping-pong cache was used to rank the d_i in different clock cycles for the SNLF. From those d_is, six could be identified to be the smallest. Initially, we set the register's value to the maximum. The d_i value was compared with each Ox's value when the clock cycle was jth (j = 3,5 . . .) period. If the d_i value was bigger than each Ox's values in the next period, it would be dropped. Otherwise, the d_i value was inserted into Ox and the biggest value in six Ox's would be dropped. Similarly, the Ex's values were updated in (j + 1)th (j + 1 = 4,6 . . .) period. The cycles were repeated until all of the 600 training samples had been calculated with input features. Finally, we compare all the 12 registers to get the smallest six values. The result was voted from the smallest six value of registers.

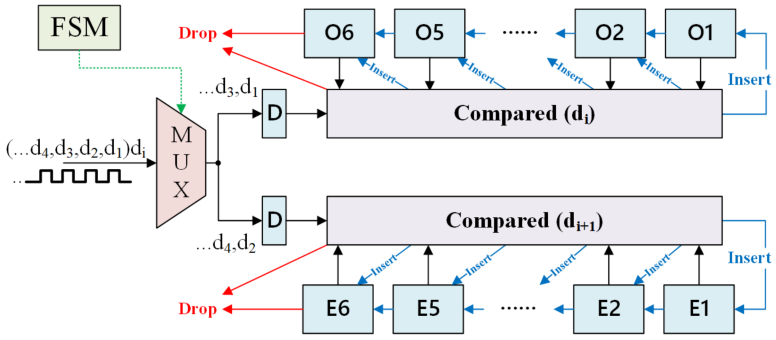

Figure 8. Block diagram of SNLF.

3.4. Support Vector Machine (SVM)

The SVM (with eight inputs) is composed of Memory Units (Mem), the Finite State Machine (FSM), multipliers, adders, and multiplexers. The pre-trained support vectors are stored in the memory unit. The FSM controls the order of output data and the running process. The FSM controlling process is mainly used in multi-class classification, where the support vector and bias are updated recursively within the structure shown in Figure 9. Multipliers and adders complete the support vector calculation in Equation (6), and the multiplexer is used for the sign function.

$$f(x) = sign\left(\sum_{i=1}^{8}(S_i \cdot Feature) + B\right) \tag{6}$$

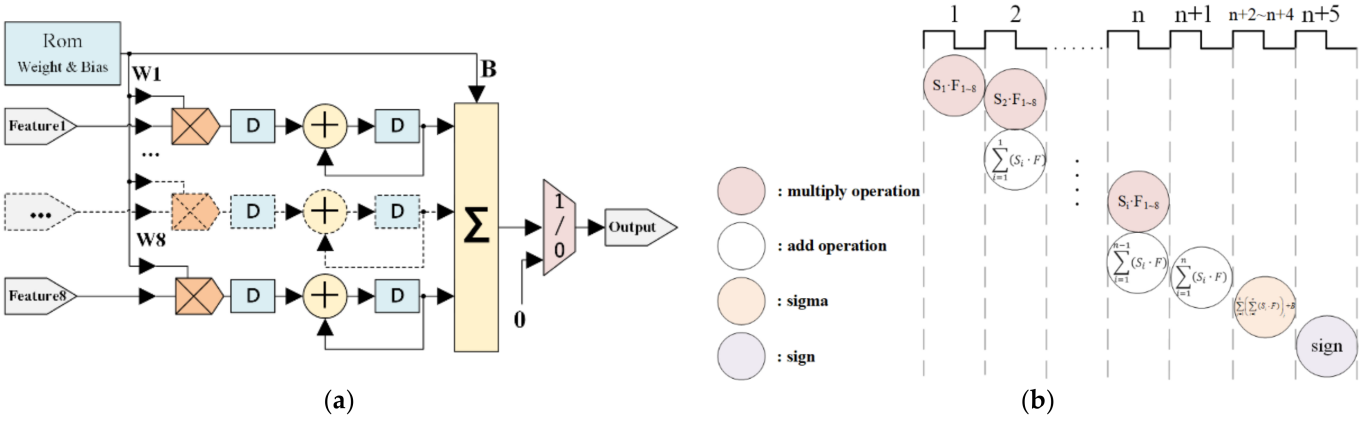

Figure 9. SVM algorithm implementation. (**a**) Block diagram of SVM. (**b**) Scheduling of the SVM processing.

4. Comparison of Development Platforms

The specifications of 10 different FPGA platforms from seven different producers are thoroughly analyzed. The key features of these FPGA cores are listed in Table 3. It is worth

noting that Intel MAX and Xilinx Artix-7 FPGA have the richest LUTs and DSPs, which are beneficial for the parallel implementation of multi-input machine learning algorithms. Additionally, on-chip Random Access Memory (RAM) resources are important. Otherwise, a large amount of pre-trained data must be pre-fetched from memory to the cache and limited LUTs resources cannot be used as buffers to cache data during the calculation. Pango PGL12G, Lattice MachXO2, and Anlogic EG4S20 all have limited RAM capacitances. Anlogic EG4S20 has an internal SDRAM module that satisfies the need for additional caching. In addition, static power consumption is a critical metric for endpoint platforms, and Anlogic FPGAs, Intel Cyclone 10LP, and Microchip M2S010 perform well in this regard. Moreover, two Lattice FPGA platforms consume the least static power, which is quite competitive for endpoint implementations. Finally, both Anlogic EF2M45 and Microchip MS010 are equipped with an internal Cortex-M3 core, which significantly improves their general performance in terms of driving external devices and communication.

Table 3. The resource of 10 different FPGAs.

Producer	Device	LUTs	DSPs	RAM	Static Current (mA at 12 V, 25 °C)	Distinction
Anlogic	EF2M45	4480	15	700 Kb	5	Inside Cortex-M3 unit
	EG4S20	19,600	29	156.8 Kb	5	Inside SDRAM unit
Gowin	GW2A	20,736	48	828 Kb	35	/
Intel	Cyclone10LP10CL	6272	30	270 Kb	5	Low-power design
	MAX10M50DAF	49,760	144	1638 Kb	35	/
Lattice	ICE40UP5	5280	8	128 Kb	0.075	Ultra low-power design
	MachXO2	6864	/	92 Kb	0.08	Low-power design
Microchip	M2S010	12,084	22	512 Kb	6.9	Inside Cortex-M3 unit
Pango	PGL12G	12,480	20	85 Kb	13	/
Xilinx	Artix-7	33,280	240	1800 Kb	14	/

On-board resources, external interfaces, and prices are the three main discriminative FPGA features that the developers pay most attention to. Therefore, in Table 4, we present the features of 10 FPGAs from seven producers.

Table 4. The specification of 10 different FPGA boards.

Producer	Device	Manufacturer	Board Name	Memory	Interface	Price
Anlogic	EF2M45	Nanjing Renmian Integrated Circuit	Sparkroad-M	/	/	$25.9
	EG4S20		Sparkroad	SPI Flash, microSD	ADC, VGA, DVP Arduino, Raspberry Pi	$49.9
Gowin	GW2A	MYMINIEYE	Combat	DDR3, SPI Flash, microSD	RJ45, HDMI, MIPI	$169
Intel	Cyclone10LP10CL	QMTECH	Starter Kit	SDRAM, SPI Flash	ADC, MIPI	$49.9
	MAX10M50DAF	Terasic	DE10-Lite	SDRAM,	VGA, Accelerometer	$85
Lattice	ICE40UP5	TinyFPGA	TinyFPGA BX	SPI Flash	/	$38
	MachXO2	STEPFPGA	STEP-MXO2	/	/	$29.9

Table 4. *Cont.*

Producer	Device	Manufacturer	Board Name	Memory	Interface	Price
Microchip	M2S010	Trenz electronic	SMF2000	SDRAM, SPI Flash	/	$59.9
Pango	PGL12G	ALINX	PGL12G Development Board	SDRAM, SPI Flash	ADC, HDMI, MIPI	$52
Xilinx	Artix-7	QMTECH	Artix-7 Development Board	SDRAM, SPI Flash	ADC, MIPI	$79.9

All seven producers develop their own Electronic Design Automation (EDA) software, among which Lattice develops a completely different EDA software for different devices. In addition, the final resource consumption is determined by synthesis tools. Most of the seven producers use either their own synthesis tools or Synplify [30], but the latter requires an individual supporting license. Table 5 summarizes the relative information of seven producers. It is worth noting that Lattice's iCEcube2 and Lattice Diamond are for ICE40UP5 and MachXO2 development respectively, and thus they cannot share the same EDA.

Table 5. The specification of EDA software.

Producer	EDA Software	Synthesis Tool	Availability
Anlogic	Tang Dynasty	TD Integrated Synthesis	Commercial
Gowin	GOWIN EDA	GowinSynthesis/Synplify	Commercial
Intel	Quartus Prime	Quartus Integrated Synthesis	Free License/Commercial
Lattice	iCEcube2	Synplify Pro	Free License/Commercial
	Lattice Diamond	Lattice Synthesis Engine	Free License/Commercial
Microchip	Libero SoC	Synplify Pro ME	Commercial
Pango	Pango Design Suite	ADS/Synplify	Commercial
Xilinx	Vivado	Xilinx Synthesis Technology	Free License/Commercial

5. Experimental Analysis and Result

To evaluate the performance of these IPs, we select six typical IoT endpoint datasets for different parameter combinations and tests. As shown in Table 6, the datasets include binary classifications, multi-classifications, and regressions. The Gutter Oil dataset proposed by VeriMake Innovation Lab aims to detect gutter oils [31], and contains six input oil features, including the pH value, refractive index, peroxide value, conductivity, pH value differences under different temperatures, and conductivity value difference under different temperatures. This dataset can serve both in a dichotomous and a polytomous way. The Smart Grid dataset for conducting research on electrical grid stability is from Karlsruher Institut für Technologie, Germany [32,33]. This is a dichotomous dataset with 13 input features used to determine whether the grid is stable under different loads. The Wine Quality dataset is proposed by the University of Minho [34] for classifying wine quality. This is a polytomous dataset, with 11 input dimensions (e.g., humidity, light, etc.), rating wines on a scale of 0 to 10. The rain dataset by the Bureau of Meteorology, Australia, is based on datasets of different weather stations for recording and forecasting the weather [35]. This is a dataset for regression prediction, using eight input parameters, such as wind, humidity, and light intensity to predict the probability of rain. Power Consumption is an open-source dataset created by the University of California, Irvine. It tracks the total energy consumption of various devices within families [36].

Table 6. Dataset specifications.

Dataset	Number of Input Features	Type
Gutter Oil	6	Binary classification
Smart Grid	13	Binary classification
Gutter Oil	6	Multiclass classification 6
Wine Quality	11	Multiclass classification 11
Rain	8	Regression
Power Consumption	7	Regression

We use a desktop PC with a 2.59 GHz Core i7 processor to train various models on the six datasets and export the best parameters with the best scores obtained during training. For binary classifications and multiclass classifications, the scores represent the classification accuracies. For linear regressions, the scores represent R2 [37]. Then, these trained parameters are fed to our machine learning IPs and implemented on 10 different FPGA boards using EDAs from seven different candidate producers. Each EDA is configured to operate in the balanced mode with identical constraints. As shown in Table 5, part of the EDAs is integrated with synthesis tools, such as Gowin and Pango. However, as Synplify requires an individual supporting license, in this paper, only their self-developed synthesis tools (GowinSynthesis, ADS) are used for analyzing FPGA implementations. The analysis of FPGA implementations is not limited to the computing performance, but encompasses all aspects of the hardware. While Power Latency Production (PLP) [38] is a common metric for evaluating the results of FPGA implementations, it does not consider the cost, which is a critical factor in IoT endpoint device development. As a result, we introduce the Cost Power Latency Production (CPLP) as an additional metric for evaluating the results.

In addition, we realize the same machine learning algorithms and parameters to the Nvidia Jetson Nano 2, the Raspberry Pi 3B+, and STM32L476 Nucleo, respectively [39], allowing for more comprehensive comparisons of the implementations within different FPGAs.

5.1. ANN

5.1.1. ANN Parameter Analysis

We intend to find the best user-defined ANN parameters in six datasets, including the number of hidden layers, neurons within each layer, and activation functions. Different combinations of these parameters are used to train our ANN model on the desktop PC and their corresponding results are shown in Appendix A, Table A1. There are only minor differences among all the combinations. We chose to apply parameters with the best score from the software to our hardware implementation. The hyperparameter values associated with the best scores for these datasets processed using the ANN algorithm are shown in Table 7.

Table 7. The highest-scoring parameters of ANN obtained in different datasets.

Dataset		Score	ANN Architectures	Activation Function
Binary classification	Gutter Oil	97.14%	[4,4,4]	Sigmoid
	Smart Grid	98.73%	[4,4,4]	Tanh
Multiclass classification	Gutter Oil	97.09%	[8,8,8]	Tanh
	Wine Quality	73.01%	[8,8,8,8]	Sigmoid
Regression	Rain	0.8583 (R2)	[8,8,8,8]	ReLU
	Power Consumption	0.9979 (R2)	[8,8]	Sigmoid

5.1.2. Implementation and Analysis of ANN Hardware

Based on the ANN architectures in Table 7, we use the corresponding EDA (with the Balanced Optimization Mode in Synthesis Settings) to implement ANN on 10 different FPGA boards. The results are summarized in Appendix A, Table A2. In terms of computing performance (latency), Intel MAX10M50DAF outperforms the others in five out of six datasets, while PGL12G outperforms the competition in the Rain task. The performance differences in time delay between 10 FPGAs are all at the millisecond level, which can almost be ignored. For the comprehensive comparisons, Lattice's ICE40UP5 has achieved first place in most application scenarios for its extremely low power consumption and cost-effectiveness among most of the datasets (five out of six). One exception is that in the Wine Quality task scenario, it was not implemented on ICE40UP5 due to the resource constraint. In addition, the device that performed the best on the Wine Quality task was Lattice MachXO2. The FPGA deployment results with the best comprehensive performance under each task are shown in Table 8.

Table 8. ANN deployment results on the best CPLP-performing FPGAs.

Dataset		Device	Score	LUTs	DSPs	Latency/us	Power/mW	PLP	CPLP
Binary classification	Gutter Oil	ICE40UP5	97.14%	1898	8	11.12	140	1556.10	59,131.80
	Smart Grid	ICE40UP5	98.73%	1855	8	11.12	140	1556.10	59,131.80
Multiclass classification	Gutter Oil	ICE40UP5	97.09%	3887	8	11	140	1540	58,520
	Wine Quality	MachXO2	73.01%	6114	/	12.73	192	2444.54	73,091.87
Regression	Rain	ICE40UP5	0.8583 (R2)	3553	8	8.73	135	1178.69	44,790.03
	Power Consumption	ICE40UP5	0.9979 (R2)	3653	8	11.98	140	1676.50	63,707

5.2. DT

5.2.1. Analysis of DT Parameters

For PC simulations, 12 different combinations of the maximum depth and the maximum number of leaf nodes are chosen. The results are shown in Appendix A, Table A3. Different DT structures produce nearly identical results. Adding the maximum depth and the maximum number of leaf nodes has no significant improvement on the score. Here, we chose the best results from various combinations for hardware deployment, and the results with the best scores for the six datasets are shown in Table 9.

Table 9. The highest-scoring parameters of DT obtained in different datasets.

Dataset		Score	Max Depth	Max Leaf Nodes
Binary classification	Gutter Oil	96.18%	5	16
	Smart Grid	97.98%	4	8
Multiclass classification	Gutter Oil	96.85%	6	32
	Wine Quality	83.69%	4	16
Regression	Rain	0.8481 (R2)	4	8
	Power Consumption	0.9976 (R2)	6	64

5.2.2. Implementation and Analysis of DT Hardware

Based on the DT architectures in Appendix A, Table A4, we use the appropriate EDA (with the Balanced Optimization Mode in Synthesis Settings) to implement DT on 10 different FPGA boards. In terms of computing performance, Intel MAX10M50DAF outperforms the competition in all six datasets. While in terms of comprehensiveness, Lattice's ICE40UP5 came to first place again in most application scenarios for its extremely low power consumption and cost-effectiveness. The FPGA DT deployment results with the best comprehensive performance under each task are shown in Table 10.

Table 10. DT deployment results on the best CPLP-performing FPGAs.

Dataset		Device	Score	LUTs	DSPs	Latency/us	Power/mW	PLP	CPLP
Binary classification	Gutter Oil	ICE40UP5	96.18%	451	0	7.92	133.90	1060.76	3882.38
	Smart Grid	ICE40UP5	97.98%	285	0	5.87	122	716.14	2873.01
Multiclass classification	Gutter Oil	ICE40UP5	96.85%	715	0	7.92	135	1069.47	3876.36
	Wine Quality	ICE40UP5	83.69%	305	0	5.60	121	677	2740.86
Regression	Rain	ICE40UP5	0.8481 (R2)	303	0	5.80	121	701.80	2838.75
	Power Consumption	ICE40UP5	0.9976 (R2)	765	0	9.65	128	1235.46	4723.10

5.3. K-NN

5.3.1. Analysis of k-NN Parameters

In our k-NN model, the parameter k is user-defined. We experiment with various k values when training our model on the PC, and the results are shown in Appendix A, Table A5. The increment of k has no significant effect on the score. In fact, on the contrary, it might decrease them. We deploy the architecture that is optimal in terms of k value for hardware deployment. The hyperparameter values associated with the best scores for these datasets processed using the k-NN algorithm are shown in Table 11.

Table 11. The highest-scoring parameters of k-NN obtained in different datasets.

Dataset		Score	k Values
Binary classification	Gutter Oil	99.36%	2
	Smart Grid	79.39%	16
Multiclass classification	Gutter Oil	99.36%	2
	Wine Quality	81.23%	8
Regression	Rain	0.8527 (R2)	16
	Power Consumption	0.9960 (R2)	2

5.3.2. Implementation and Analysis of k-NN

According to the k parameters analyzed in Section 5.3.1, we implement our model on 10 FPGA boards (with the Balanced Optimization Mode in Synthesis Settings). The corresponding results are shown in Appendix A, Table A6. Gowin's GW2A has the best computing performance in all of the task scenarios. By relying on extremely low power consumption and cost-effectiveness, Lattice's ICE40UP5 achieves the best comprehensive performance across all datasets.

Additionally, two things are worth noting: Anlogic's EF2M45 and Lattice's MachXO2 are unable to deploy k-NN in multiple mission scenarios due to resource constraints. Pango's PGL12G is also incapable of deploying k-NN. In addition, the reason is that the synthesis tool is unable to correctly recognize the current k-NN design, and therefore ignores the key path. This does not occur when using alternative development tools. The FPGA k-NN deployment results with the best comprehensive performance under each task are shown in Table 12.

Table 12. The k-NN deployment results on the best CPLP-performing FPGAs.

Dataset		Device	Score	LUTs	DSPs	Latency/us	Power/mW	PLP	CPLP
Binary classification	Gutter Oil	ICE40UP5	99.36%	979	6	7.81	134	1046.54	39,768.52
	Smart Grid	ICE40UP5	79.39%	4949	8	9.32	135.79	1265.58	48,092.09
Multiclass classification	Gutter Oil	ICE40UP5	99.36%	1077	6	7.81	135	1054.35	40,065.30
	Wine Quality	MachXO2	81.23%	4348	/	7.18	186	1335.67	39,936.41
Regression	Rain	Cyclone10LP10CL	0.8527 (R2)	5397	8	11.89	305.54	3633.79	181,325.98
	Power Consumption	ICE40UP5	0.9960 (R2)	1159	7	9.36	12.71	118.97	4520.69

5.4. SVM

In the experiment, the linear SVM is chosen for training, and the results are shown in Table 13. Due to the function similarity between the linear SVM and ANN, their simulation scores are very similar. The SVM deployment results on 10 FPGA boards with the best comprehensive performance under each task are shown in Table 14. The remaining implementation results are provided in the Appendix A, Table A7.

Table 13. The highest-scoring of SVM obtained in different datasets.

Dataset		Score
Binary classification	Gutter Oil	96.02%
	Smart Grid	98.35%
Multiclass classification	Gutter Oil	96.33%
	Wine Quality	72.68%
Regression	Rain	0.7932 (R2)
	Power Consumption	0.9979 (R2)

Table 14. SVM deployment results on the best CPLP-performing FPGAs.

Dataset		Device	Score	LUTs	DSPs	Latency	Power	PLP	CPLP
Binary classification	Gutter Oil	ICE40UP5	96.02%	765	8	15.56	151	2349.41	89,277.54
	Smart Grid	ICE40UP5	98.35%	955	8	15.90	154.98	2464.17	93,638.31
Multiclass classification	Gutter Oil	ICE40UP5	96.33%	765	8	14.90	152.99	2279.20	86,609.61
	Wine Quality	ICE40UP5	72.68%	976	8	15.56	155.14	2413.89	91,727.65
Regression	Rain	ICE40UP5	0.7932 (R2)	527	8	13.51	147.76	1996.77	75,877.43
	Power Consumption	ICE40UP5	0.9979 (R2)	516	8	13.47	148	1994.00	75,772.15

5.5. Comparisons with Embedded Platforms

To provide a more accurate assessment of our implementation, we also compare MLoF with three representative embedded platforms, namely Nvidia Jetson Nano, Raspberry Pi3 B+, and STM32L476 Nucleo. The specification of each platform is listed in Table 15. Jetson Nano is powered by a Cortex-A57 core running at 1.43 GHz and a 128-core Nvidia Maxwell-based GPU [40], while Raspberry features a Cortex-A53 core running at 1.2 GHz. STM32L476 Nucleo is a typical IoT development platform with a Cortex-M4 core running at 80 MHz [41]. Compared with Table 4, the prices of these three representative embedded development platforms are similar to the FPGAs, which indicates that all of them are comparable in terms of other indexes. Given the proper cost of FPGAs, they can be considered as competitive substitutes for past typical platforms.

Table 15. The specification of three representative embedded development platforms.

Platform	Processor	Clock	Price
Nvidia Jetson Nano	Cortex-A57	1.43 GHz	$89.00
Raspberry Pi3 B+	Cortex-A53	1.2 GHz	$54.99
STM32L476 Nucleo	Cortex-M4	80 MHz	$31.99

Based on the simulation results of previous desktop PCs, the Receiver Operating Characteristic (ROC) curve and Precision-Recall (PR) curve shown in Figure 10, we select the models with the highest score in each of the six task scenarios for deployment of the embedded platform [42]. The corresponding deployment models and architectures for each task are listed in Table 16. PyCuda is used for GPU parallel acceleration with fixed weight parameters on Jetson Nano. Moreover, we use the same Python code to implement it on the Raspberry Pi, as well.

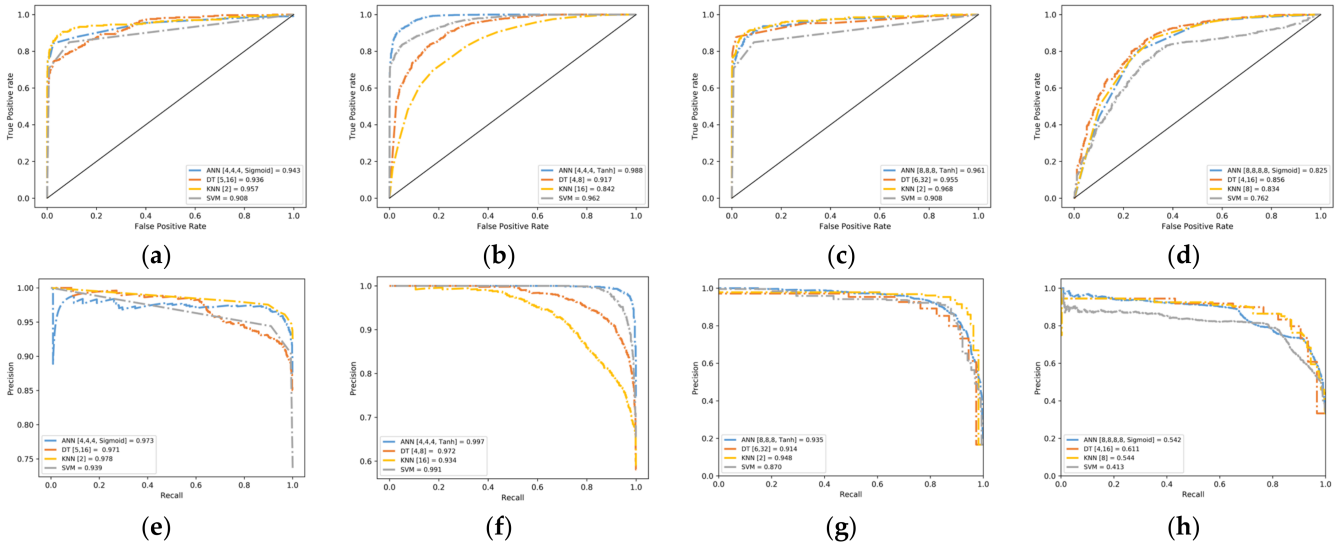

Figure 10. Comparison of the receiver operating characteristic curve and precision-recall curve for the six typical IoT endpoint datasets with different algorithms. (**a**) Receiver Operating Characteristic curve for Gutter Oil binary classification; (**b**) Receiver Operating Characteristic curve for Smart Grid binary classification; (**c**) Receiver Operating Characteristic curve for Gutter Oil multiclass classification; (**d**) Receiver Operating Characteristic curve for Smart Grid multiclass classification; (**e**) Precision-Recall curve for Gutter Oil binary classification; (**f**) Precision-Recall curve for Smart Grid binary classification; (**g**) Precision-Recall curve for Gutter Oil multiclass classification; (**h**) Precision-Recall curve for Smart Grid multiclass classification.

Table 16. The highest-scoring model obtained in different datasets.

Dataset		Model	Score	Architecture
Binary classification	Gutter Oil	k-NN	99.36%	K = 2
	Smart Grid	ANN	98.73%	[4,4,4], Tanh
Multiclass classification	Gutter Oil	k-NN	99.36%	K = 2
	Wine Quality	DT	83.69%	Max depth = 4, Max node = 16
Regression	Rain	ANN	0.8583 (R2)	[8,8,8], ReLU
	Power Consumption	SVM	0.9979 (R2)	N/A

Table 17 compares the performance of our FPGA and three embedded platform implementations. Jetson Nano takes the lead in terms of computing performance. On the other hand, Nucleo consumes the lowest power. While the power consumption of FPGA decreased by an average of 891%, and its performance improved by an average of 9 times compared to typical IoT endpoint platforms. Moreover, FPGAs outperform all other platforms in terms of Energy Efficiency (PLP) and Cost Efficiency (CPLP).

Table 17. Breakdown of platform implemented results.

Dataset and Type of Module			The Best of FPGA	Jetson Nano	Raspberry	Nucleo
Binary classification	Gutter Oil (k-NN)	Accuracy	99.36%	99.36%	99.36%	99.36%
		Latency	7.81 us	5.90 us	10.51 us	180.00 us
		Power	134 mW	2120 mW	1480 mW	102 mW
		PLP	1046.54	12,508	15,554.8	18,360
		CPLP	39,768.52	1,113,212	855,358.5	587,336.4
	Smart Grid (ANN)	Accuracy	98.73%	98.73%	98.73%	98.73%
		Latency	11.12 us	5.97 us	18.77 us	300.00 us
		Power	140 mW	2110 mW	1470 mW	101 mW
		PLP	1556.8	12,596.7	27,591.9	30,300
		CPLP	59,158.4	1,121,106	1,517,279	969,297
Multiclass classification	Gutter Oil (k-NN)	Accuracy	99.36%	99.36%	99.36%	99.36%
		Latency	7.81 us	5.44 us	10.25 us	180.00 us
		Power	135 mW	2120 mW	1470 mW	102 mW
		PLP	1054.35	11,532.8	15,067.5	18,360
		CPLP	40,065.3	1,026,419	828,561.8	587,336.4
	Wine Quality (DT)	Accuracy	83.69%	83.69%	83.69%	83.69%
		Latency	5.60 us	1.37 us	5.82 us	84.00 us
		Power	121 mW	2060 mW	1350 mW	101 mW
		PLP	677.6	2822.2	7857	8484
		CPLP	25,748.8	251,175.8	432,056.4	271,403.2
Regression	Rain (ANN)	R2	0.8583	0.8583	0.8583	0.8583
		Latency	8.73 us	7.81 us	35.77 us	261.00 us
		Power	135 mW	2140 mW	1480 mW	102 mW
		PLP	1178.55	16,713.4	52,939.6	26,622
		CPLP	44,784.9	1,487,493	2,911,149	851,637.8
	Power Consumption (SVM)	R2	0.9979	0.9979	0.9979	0.9979
		Latency	13.47 us	7.94 us	43.73 us	364.00 us
		Power	148 mW	2610 mW	1530 mW	103 mW
		PLP	1993.56	20,723.4	66,906.9	37,492
		CPLP	75,755.28	1,844,383	3,679,210	1,199,369

To demonstrate the benefits of FPGA implementation in IoT endpoint scenarios, we compare embedded and FPGA platforms using six datasets in terms of performance (latency), power consumption, PLP, and CPLP, as shown in Figure 11 with the ordinate-axis in logarithmic scale. Jetson Nano exceeds the others in performance, but the second-best FPGA is not far behind, only 38% lower in average, 100% ahead of Raspberry, and 2300% ahead of Nucleo. In terms of power consumption, Nucleo is quite competitive as a low-power MCU with 102 mW on average, 30 mW lower than FPGA. These two platforms advanced well beyond Jetson Nano and Raspberry. It can be clearly seen that in comparison to other platforms, FPGAs require significantly less PLP and CPLP on the ordinate-axis in logarithmic scale. The smallest PLP and CPLP are critical for IoT endpoint development and implementation, as response time, power consumption, and cost are all critical factors in IoT endpoint tasks. Furthermore, the FPGA PLP is 17× better than the average for embedded platforms and the FPGA CPLP is 25× better than the average for embedded platforms.

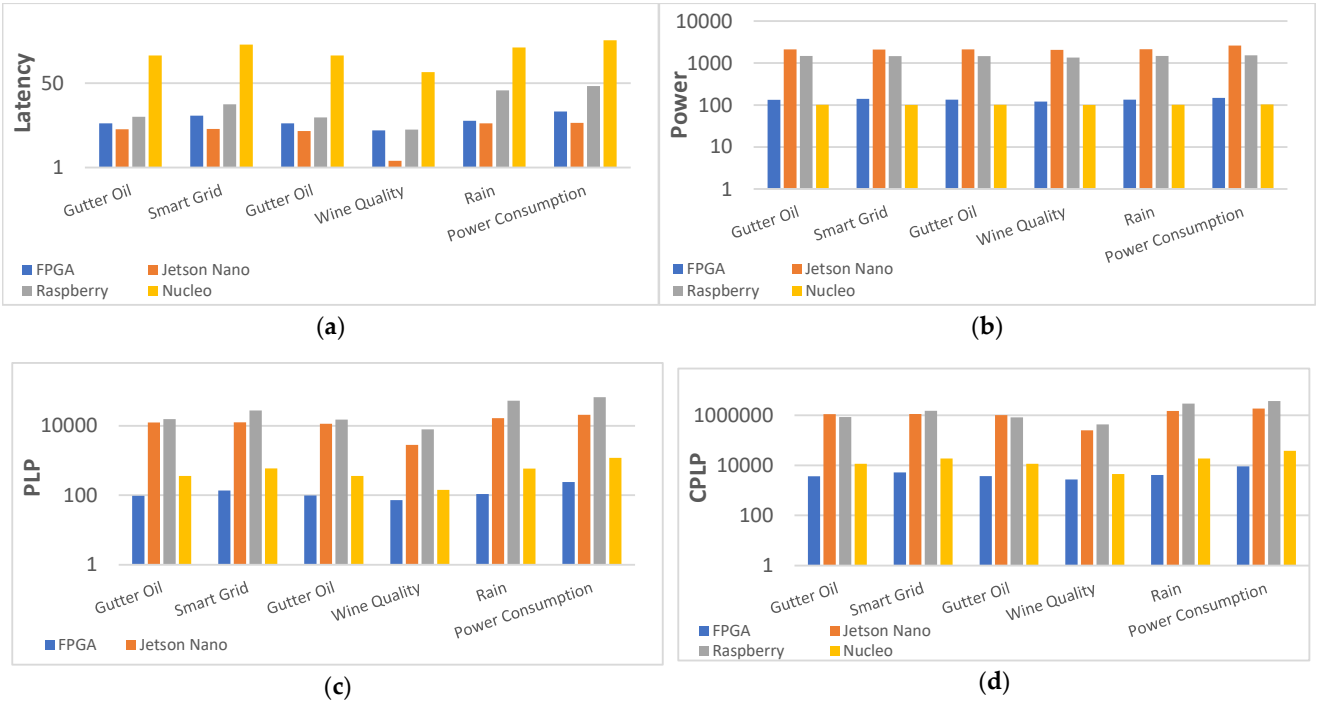

Figure 11. Comparison of performance (latency), power consumption, PLP, and CPLP for the six typical IoT endpoint datasets with different algorithms when implemented on FPGA and embedded platforms. (**a**) Comparison of performance (latency). (**b**) Comparison of power consumption. (**c**) Comparison of PLP. (**d**) Comparison of CPLP.

6. Conclusions

In this paper, the Machine Learning on FPGA (MLoF), a series of ML hardware accelerator IP cores for IoT endpoint devices was introduced to offer high-performance, low-cost, and low-power.

MLoF completes the process of making inferences on FPGAs based on the optimal parameter results from PC training. It implements four typical machine learning algorithms (ANN, DT, k-NN, and SVM) with Verilog HDL on 10 FPGA development boards from seven different manufacturers. The usage of LUTs, Power, Latency, Cost, PLP, as well as CPLP are used in comparisons and analyses of the MLoF deployment results with six typical IoT datasets. At the same time, we analyzed the synthesis results of different EDA tools under the same hardware design. Finally, we compared the best FPGA deployment results with typical IoT endpoint platforms (Jetson Nano, Raspberry, STM32L476). The results indicate that the FPGA PLP outperforms the IoT platforms by an average of 17× due to their superior parallelism capability. Meanwhile, FPGAs have 25× better CPLP compared to the IoT platforms. To our knowledge, this is the first paper that conducts hardware deployment, platform comparisons, and deployment result analysis. At the same time, it is also the first set of IP on open-source FPGA machine learning algorithms, and has been verified on low-cost FPGA platforms.

MLoF still has room for further improvements: 1. The adaptability of MLoF could be enhanced, thus more complex algorithms (kNN with k > 16) could also be deployed on low-cost FPGAs with few resources, such as MachXO2; 2. More options for user parameters configuration could be added, including more ML algorithms, larger data bit width, and more hyperparameters; 3. Usability could be improved by further providing a script file or a user interface, to help the users generate the desired ML algorithm IP core more easily. These existing shortcomings of MLoF point out the direction of our future work.

Author Contributions: Conceptualization, M.L. and R.C.; investigation, M.L. and R.C.; algorithm proposed, R.C.; hardware architecture, R.C.; data curation, R.C., T.W. and Y.Z.; validation, M.L. and R.C.; resources, M.L. and R.C.; supervision, M.L.; visualization, R.C., T.W. and Y.Z.; writing—original draft, M.L., R.C. and T.W.; writing—review and editing, M.L., R.C., T.W. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Innovation Funding of Agriculture Science and Technology in Jiangsu Province, under grant CX(21)3121.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The following are available online at https://github.com/verimake-team/MLonFPGA, the source code and data.

Acknowledgments: The authors would like to thank Jiangsu Academy of Agricultural Sciences for all the support, and would also like to acknowledge the technical support from VeriMake Innovation Lab.

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

Abbreviations

ANN	Artificial Neural Network
DT	Decision Tree
FSM	Finite State Machine
FPGA	Field Programmable Gate Array
GOP	Giga Operation per Second
HDL	Hardware Description Language
IoT	Internet of Thing
IP	Intellectual Property
k-NN	K-Nearest Neighbor
LUT	Look-up Table
MAC	Multiplying Accumulator
MLoF	Machine Learning on FPGA
MCU	Microcontroller Unit
ML	Machine Learning
RAM	Random Access Memory
ReLU	Rectified Linear Unit
SoC	System on Chip
SVM	Support Vector Machine

Appendix A

Table A1. The score for different ANN architectures and activation functions.

Activation Function	Dataset	The Score for ANN Architecture							
		1.4	**1.8**	**2.4**	**2.8**	**3.4**	**3.8**	**4.4**	**4.8**
ReLU	Gutter Oil	95.76	96.08	95.97	96.61	95.55	96.92	96.71	96.82
	Smart Grid	98.17	98.25	98.41	97.92	98.3	97.83	98.16	98.41
	Gutter Oil	95.28	96.55	96.46	96.28	96.28	96.97	93.1	96.19
	Wine Quality	70.84	71.58	63.01	71.45	70.22	71.36	70.7	72.29
	Rain	78.85	85.18	85.06	83.66	78.85	85.36	85.55	85.83
	Power Consumption	0.9976	0.9967	0.9894	0.997	0.9973	0.9964	0.9973	0.9973
Tanh	Gutter Oil	96.19	96.4	95.87	96.4	96.92	95.87	96.5	96.29
	Smart Grid	98.25	98.52	98.68	98.3	98.73	98.41	98.32	98.16
	Gutter Oil	95.73	96.46	96.37	96.82	96.37	96.91	96.53	96.55
	Wine Quality	71.85	70.44	71.76	72.95	72.02	72.29	72.46	72.94
	Rain	78.85	83.62	78.85	78.85	78.85	78.85	78.84	78.85
	Power Consumption	0.9948	0.995	0.9959	0.9965	0.9945	0.9956	0.9904	0.9958

Table A1. *Cont.*

Activation Function	Dataset	The Score for ANN Architecture							
		1.4	**1.8**	**2.4**	**2.8**	**3.4**	**3.8**	**4.4**	**4.8**
Sigmoid	Gutter Oil	96.29	96.4	96.19	96.4	97.14	97.14	97.03	96.92
	Smart Grid	98.07	98.07	98.52	98.03	98.64	98.26	97.87	97.56
	Gutter Oil	95.55	95.5	96.27	96.73	96.1	97.09	96.83	97
	Wine Quality	72.07	72.64	72.86	72.95	72.15	72.81	73	73.01
	Rain	78.85	85.14	85.12	85.17	78.85	85.46	78.85	84.37
	Power Consumption	0.9953	0.9966	0.9951	0.9979	0.9954	0.9956	0.9958	0.9964

Table A2. ANN deployment results on the low-cost FPGAs.

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Gutter Oil	EF2M45	1445	15	11.76	492.48	5789.10	149,937.75
	EG4S20	1165	25	11.40	371.82	4238.32	211,492.13
	GW2A	1833	12	11.45	451.47	5167.12	873,243.27
	Cyclone10LP10CL	1795	15	10.55	271.38	2863.87	142,907.27
	MAX10M50DAF	1400	25	10.17	312.53	3177.18	270,060.30
	ICE40UP5	1898	8	11.12	140	1556.10	59,131.80
	MachXO2	2717	/	11.48	190.98	2192.07	65,542.85
	M2S010	1398	22	11.63	292.47	3402.01	203,780.46
	PGL12G	1780	7	11.65	585.22	6820.15	354,648
	Artix-7	916	10	10.75	589	6331.75	505,906.83
Smart Grid	EF2M45	1371	15	15.59	494.18	7704.24	199,539.69
	EG4S20	1091	25	11.40	381.85	4353.48	217,238.81
	GW2A	1779	12	12.30	419.11	5155.09	871,210.19
	Cyclone10LP10CL	1530	15	10.62	267.11	2837.24	141,578.40
	MAX10M50DAF	1335	21	9.58	311.08	2978.90	253,206.68
	ICE40UP5	1855	8	11.12	140	1556.10	59,131.80
	MachXO2	2696	/	11.48	191	2192.30	65,549.71
	M2S010	1263	22	11.79	300.44	3542.49	212,195.03
	PGL12G	1116	10	11.61	586	6802.87	353,749.45
	Artix-7	919	10	10.90	588	6409.20	512,095.08
Gutter Oil	EF2M45	2651	15	12.44	542.82	6752.71	174,895.08
	EG4S20	1766	29	11.13	481.85	5360.60	267,494.11
	GW2A	3235	8	12.32	579.51	7136.70	1,206,102.74
	Cyclone10LP10CL	3069	15	10.36	272	2818.46	140,641.35
	MAX10M50DAF	2198	25	9.69	317	3072.05	261,124
	ICE40UP5	3887	8	11	140	1540	58,520
	MachXO2	5763	/	12.05	191	2301.17	68,804.92
	M2S010	2022	22	12	305	3658.78	219,160.92
	PGL12G	2546	7	11.97	585	7002.45	364,127.40
	Artix-7	2269	14	11	591	6501	519,429.90
Wine Quality	EF2M45	3139	15	16.37	622.83	10,195.78	264,070.60
	EG4S20	2231	29	12.31	491.84	6053.59	302,074.21
	GW2A	3696	16	14.27	682.89	9741.40	1,646,296.15
	Cyclone10LP10CL	3815	15	11.80	279	3290.81	164,211.17
	MAX10M50DAF	2895	33	9.66	322	3109.88	264,339.46
	ICE40UP5	/	/	/	/	/	/
	MachXO2	6114	/	12.73	192	2444.54	73,091.87
	M2S010	2560	22	13.99	302.21	4226.41	253,161.77
	PGL12G	2929	7	15.61	586	9146.87	475,637.45
	Artix-7	2320	14	12.47	588	7332.36	585,855.56

Table A2. *Cont.*

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Rain	EF2M45	1426	15	11.84	542.85	6428.45	166,496.94
	EG4S20	1354	16	10.77	431.85	4650.61	232,065.65
	GW2A	3245	8	11.45	579.73	6634.96	1,121,308.93
	Cyclone10LP10CL	2263	15	9.39	273	2562.38	127,862.66
	MAX10M50DAF	2292	16	8.12	313	2542.50	216,112.42
	ICE40UP5	3553	8	8.73	135	1178.69	44,790.03
	MachXO2	4392	/	9.14	189	1728.03	51,668.01
	M2S010	2196	16	9.12	311	2836.63	169,914.20
	PGL12G	2055	7	4.72	584	2754.73	143,245.86
	Artix-7	1794	8	9.25	595	5503.75	439,749.63
Power Consumption	EF2M45	2548	15	16.61	555.51	9227.56	238,993.80
	EG4S20	1614	29	14.61	454.12	6632.39	330,956.43
	GW2A	3245	8	14.45	561.69	8117	1,371,772.43
	Cyclone10LP10CL	2741	15	12.48	293	3657.23	182,495.58
	MAX10M50DAF	1832	33	11.67	344	4014.14	341,201.56
	ICE40UP5	3653	8	11.98	140	1676.50	63,707
	MachXO2	5621	/	12.80	209	2675.41	79,994.73
	M2S010	2323	22	12.21	322	3931.30	235,484.75
	PGL12G	2298	10	15.89	586	9309.78	484,108.66
	Artix-7	1110	14	12.12	588	7126.56	569,412.14

Table A3. The score for different DT architectures.

Number of Depths	Dataset	Number of Leaf Nodes			
		8	16	32	64
4	Gutter Oil	96.09	96.09	/	/
	Smart Grid	97.98	97.98	/	/
	Gutter Oil	96.46	96.46	/	/
	Wine Quality	81.98	83.69	/	/
	Rain	0.8481	0.848	/	/
	Power Consumption	0.98	0.992	/	/
5	Gutter Oil	96.09	96.18	96.18	/
	Smart Grid	97.98	97.98	97.98	/
	Gutter Oil	96.46	96.63	96.7	/
	Wine Quality	81.98	83.21	83.51	/
	Rain	0.8481	0.8451	0.8404	/
	Power Consumption	0.98	0.9928	0.9963	/
6	Gutter Oil	96.09	96.18	96.18	96.18
	Smart Grid	97.98	97.98	97.98	97.98
	Gutter Oil	96.46	96.63	96.85	96.85
	Wine Quality	81.98	82.5	83.51	83.25
	Rain	0.8481	0.8447	0.8366	0.839
	Power Consumption	0.98	0.9928	0.9966	0.9976

Table A4. DT deployment results on the low-cost FPGAs.

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Gutter Oil	EF2M45	681	/	12.28	416.85	5120.52	132,621.57
	EG4S20	661	/	11.52	325.63	3751.54	187,201.70
	GW2A	417	0	8.62	409.26	3528.20	596,265.12
	Cyclone10LP10CL	360	0	6.33	263	1664	83,033.65
	MAX10M50DAF	363	0	5.70	302	1722	146,370.34
	ICE40UP5	451	0	7.92	133.90	1060.76	40,309.02
	MachXO2	494	/	8.86	184.71	1635.83	48,911.23
	M2S010	396	0	9.87	279	2752.34	164,864.87
	PGL12G	271	0	9.77	548	5356.15	278,519.90
	Artix-7	264	0	9.40	587	5517.80	440,872.22
Smart Grid	EF2M45	272	0	7.95	387.38	3081.21	79,803.41
	EG4S20	272	0	7.62	281.08	2142.35	106,903.45
	GW2A	313	0	7.86	400.69	3150.21	532,385.33
	Cyclone10LP10CL	284	0	5.93	261	1546.95	77,192.66
	MAX10M50DAF	287	0	5.38	301	1620.28	137,724.06
	ICE40UP5	285	0	5.87	122	716.14	27,213.32
	MachXO2	285	/	5.95	184	1095.17	32,745.52
	M2S010	291	0	6.38	293	1868.75	111,938.36
	PGL12G	307	0	6.33	543	3436.65	178,705.64
	Artix-7	188	0	8.02	588	4715.76	376,789.22
Gutter Oil	EF2M45	1412	0	18.90	446.36	8436.73	218,511.20
	EG4S20	1463	0	17.63	374.87	6608.87	329,782.61
	GW2A	693	0	10.17	437.50	4450.69	752,166.19
	Cyclone10LP10CL	636	0	8.06	264	2128.10	106,192.39
	MAX10M50DAF	648	0	7.11	303	2153.72	183,066.54
	ICE40UP5	715	0	7.92	135	1069.47	40,639.86
	MachXO2	774	/	8.86	187	1656.07	49,516.55
	M2S010	685	0	9.35	316	2955.86	177,056.25
	PGL12G	679	0	12.82	553	7088.35	368,594.41
	Artix-7	655	0	10.20	590	6018	480,838.20
Wine Quality	EF2M45	266	0	8.26	387.38	3199.73	82,873.11
	EG4S20	266	0	8.39	286.44	2403.78	119,948.59
	GW2A	311	0	7.94	401.59	3187.39	538,668.59
	Cyclone10LP10CL	284	0	5.77	261.42	1507.09	75,203.61
	MAX10M50DAF	287	0	5.54	301.81	1670.82	142,019.71
	ICE40UP5	305	0	5.60	121	677.6	25,748.8
	MachXO2	304	/	5.54	179	991.66	29,650.63
	M2S010	301	0	6.72	291.47	1958.41	117,308.58
	PGL12G	316	0	7.22	443.83	3204.90	166,654.61
	Artix-7	177	0	8.20	488	4001.60	319,727.84
Rain	EF2M45	269	0	7.65	382.18	2925.24	75,763.62
	EG4S20	269	0	7.79	281.84	2196.36	109,598.15
	GW2A	315	0	7.94	400.68	3180.23	537,458.69
	Cyclone10LP10CL	280	0	5.77	261.29	1508.69	75,283.55
	MAX10M50DAF	283	0	5.09	301.38	1533.12	130,315.21
	ICE40UP5	303	0	5.80	121	701.80	26,668.40
	MachXO2	301	/	5.94	174	1033.21	30,893.04
	M2S010	306	0	6.57	289.28	1900.28	113,826.79
	PGL12G	312	0	7.23	443.86	3210	166,919.77
	Artix-7	178	0	7.42	487	3613.54	288,721.85

Table A4. *Cont.*

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Power Consumption	EF2M45	1437	0	19.76	436.51	8625.48	223,399.86
	EG4S20	1424	0	17.51	365.56	6399.56	319,338.21
	GW2A	679	0	10.24	439.19	4495.09	759,670.07
	Cyclone10LP10CL	655	0	8.52	264.80	2256.10	112,579.19
	MAX10M50DAF	665	0	6.87	303.44	2083.72	177,116.41
	ICE40UP5	765	0	9.65	128	1235.46	46,947.33
	MachXO2	783	/	9.96	194	1932.82	57,791.38
	M2S010	709	0	10.47	313.76	3286.36	196,853.21
	PGL12G	668	0	15.50	454.35	7044.24	366,300.60
	Artix-7	716	0	8.35	492	4108.20	328,245.18

Table A5. The score for different k-NN architectures.

Dataset	k Values (Number of Neighbors)			
	2	**4**	**8**	**16**
Gutter Oil	99.36	99.09	98.73	98.46
Smart Grid	75.78	77.3	78.38	79.39
Gutter Oil	99.36	99.07	98.73	98.46
Wine Quality	77.93	77.41	81.23	74.9
Rain	0.8366	0.8466	0.8494	0.8527
Power Consumption	0.996	0.9954	0.9944	0.9915

Table A6. The k-NN deployment results on the low-cost FPGAs.

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Gutter Oil	EF2M45	1251	15	17.26	446.36	7703.32	199,515.87
	EG4S20	942	18	16.08	315.81	5077.64	253,374.31
	GW2A	991	6	5.18	397.25	2059.34	348,029.14
	Cyclone10LP10CL	984	6	10.63	282.49	3001.74	149,786.76
	MAX10M50DAF	985	6	9.33	335.91	3135.05	266,479.08
	ICE40UP5	979	6	7.81	134	1046.54	39,768.52
	MachXO2	1062	/	8.84	185.47	1638.63	48,994.96
	M2S010	990	6	13.32	325.57	4337.50	259,816.40
	PGL12G	/	/	/	/	/	/
	Artix-7	592	18	11.92	597	7116.24	568,587.58
Smart Grid	EF2M45	/	/	/	/	/	/
	EG4S20	10,757	29	18.07	1371.32	24,783.83	1,236,713.13
	GW2A	5708	13	7.84	888.16	6960.47	1,176,319.55
	Cyclone10LP10CL	4810	13	11.89	304.67	3621.31	180,703.25
	MAX10M50DAF	4847	13	9.90	369.19	3653.14	310,516.48
	ICE40UP5	4949	8	9.32	135.79	1265.58	48,092.09
	MachXO2	/	/	/	/	/	/
	M2S010	5078	13	13.41	327.23	4386.45	262,748.42
	PGL12G	/	/	/	/	/	/
	Artix-7	3231	39	12.05	594	7157.70	571,900.23

Table A6. *Cont.*

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Gutter Oil	EF2M45	1361	15	16.92	452.82	7663.01	198,471.83
	EG4S20	1058	18	16.62	331.42	5507.27	274,812.90
	GW2A	1047	6	5.18	406.45	2107.04	356,089.22
	Cyclone10LP10CL	1076	6	10.63	282.79	3005.77	149,988.17
	MAX10M50DAF	1088	6	8.83	335.66	2964.55	251,986.68
	ICE40UP5	1077	6	7.81	135	1054.35	40,065.30
	MachXO2	1241	/	7.62	185.47	1414.02	42,279.30
	M2S010	1117	6	12.94	325.72	4213.81	252,407.44
	PGL12G	/	/	/	/	/	/
	Artix-7	673	18	11.57	597	6907.29	551,892.47
Wine Quality	EF2M45	/	/	/	/	/	/
	EG4S20	4435	29	18.96	672.69	12,752.20	636,334.94
	GW2A	3415	11	6.46	578.87	3739.51	631,977.72
	Cyclone10LP10CL	3073	11	11.10	194.79	2162.56	107,911.67
	MAX10M50DAF	3067	11	9.27	354	3280.87	278,874.12
	ICE40UP5	3848	8	9.13	135	1232.42	46,831.77
	MachXO2	4348	/	7.18	186	1335.67	39,936.41
	M2S010	3568	11	13.35	235.65	3145.95	188,442.66
	PGL12G	/	/	/	/	/	/
	Artix-7	2037	33	12.10	511	6183.10	494,029.69
Rain	EF2M45	/	/	/	/	/	/
	EG4S20	11,767	24	18	1491.21	26,843.27	1,339,479.23
	GW2A	6297	8	7.76	886.53	6882.98	1,163,223.64
	Cyclone10LP10CL	5397	8	11.89	305.54	3633.79	181,325.98
	MAX10M50DAF	5412	8	10.09	362.04	3651.17	310,349.74
	ICE40UP5	/	/	/	/	/	/
	MachXO2	/	/	/	/	/	/
	M2S010	6218	8	12.94	471.37	6097.62	365,247.23
	PGL12G	/	/	/	/	/	/
	Artix-7	3754	24	11.50	509	5853.50	467,694.65
Power Consumption	EF2M45	1735	15	17.75	485.83	8625.01	223,387.78
	EG4S20	1071	21	15.30	365.83	5598.36	279,358.05
	GW2A	1095	7	5.18	321.68	1667.57	281,819.93
	Cyclone10LP10CL	1160	7	9.07	288.61	2617.40	130,608.46
	MAX10M50DAF	1173	7	10.54	339.08	3573.90	303,781.77
	ICE40UP5	1159	7	9.36	127	1188.21	45,152.06
	MachXO2	1864	/	8.81	195.09	1719.12	51,401.82
	M2S010	1232	7	11.86	212.88	2524.48	151,216.63
	PGL12G	/	/	/	/	/	/
	Artix-7	734	21	11.95	534	6381.30	509,865.87

Table A7. SVM deployment results on the low-cost FPGAs.

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Gutter Oil	EF2M45	829	8	13.81	426.67	5890.66	152,568.13
	EG4S20	853	8	13.45	365.11	4909.28	244,973.20
	GW2A	671	8	6.51	329.78	2145.52	362,592.23
	Cyclone10LP10CL	740	8	10.59	274.36	2906.30	145,024.14
	MAX10M50DAF	741	8	10.06	342.58	3447.04	292,998.40
	ICE40UP5	765	8	15.56	151	2349.41	89,277.54
	MachXO2	2663	/	15.57	217	3378.91	101,029.32
	M2S010	766	8	12.66	315.13	3988.89	238,934.52
	PGL12G	663	8	11.73	589.16	6910.26	359,333.40
	Artix-7	416	8	9.12	590	5380.80	429,925.92

Table A7. *Cont.*

Dataset	Device	LUTs	DSPs	Latency	Power	PLP	CPLP
Smart Grid	EF2M45	950	8	13.69	456.22	6245.71	161,763.80
	EG4S20	1000	8	14.01	355.15	4976.29	248,316.96
	GW2A	890	8	6.75	361.03	2437.66	411,964.72
	Cyclone10LP10CL	963	8	11.33	307.44	3484.22	173,862.45
	MAX10M50DAF	964	8	10.46	364.08	3809.00	323,765.42
	ICE40UP5	955	8	15.90	154.98	2464.17	93,638.31
	MachXO2	3752	/	15.21	216	3284.71	98,212.89
	M2S010	975	8	13.48	325.74	4390.70	263,003.13
	PGL12G	775	8	11.22	587.97	6596.44	343,014.64
	Artix-7	482	8	8.78	589	5171.42	413,196.46
Gutter Oil	EF2M45	829	8	13.81	426.67	5890.66	152,568.13
	EG4S20	853	8	13.45	365.11	4909.28	244,973.20
	GW2A	671	8	6.51	329.78	2145.52	362,592.23
	Cyclone10LP10CL	740	8	10.59	274.36	2906.30	145,024.14
	MAX10M50DAF	741	8	10.06	362.58	3648.28	310,103.80
	ICE40UP5	765	8	14.90	152.99	2279.20	86,609.61
	MachXO2	2667	/	15.19	217.11	3297.02	98,580.82
	M2S010	766	8	12.07	319.07	3851.48	230,703.77
	PGL12G	663	8	11.73	569.16	6675.68	347,135.24
	Artix-7	416	0	9.12	487	4441.44	354,871.06
Wine Quality	EF2M45	918	8	13.52	444.32	6005.85	155,551.42
	EG4S20	994	8	13.72	355.15	4873.65	243,195.38
	GW2A	858	8	6.68	354.76	2369.78	400,493.40
	Cyclone10LP10CL	932	8	10.96	276.47	3030.39	151,216.34
	MAX10M50DAF	933	8	10.52	362.87	3817.76	324,509.20
	ICE40UP5	976	8	15.56	155.14	2413.89	91,727.65
	MachXO2	3070	/	15.91	217.11	3454.24	103,281.66
	M2S010	966	8	12.44	320.80	3990.75	239,046.04
	PGL12G	743	8	12.10	569.60	6891.02	358,333.08
	Artix-7	465	8	9.20	490	4508.00	360,189.20
Rain	EF2M45	523	8	13.27	400.72	5317.57	137,725.00
	EG4S20	533	8	12.95	312.13	4041.45	201,668.17
	GW2A	541	8	5.91	316.57	1869.98	316,027.45
	Cyclone10LP10CL	502	8	9.83	275.69	2710.86	135,271.90
	MAX10M50DAF	503	8	9.95	320.90	3193.92	271,483.00
	ICE40UP5	527	8	13.51	147.76	1996.77	75,877.43
	MachXO2	864		15.21	217.43	3307.55	98,895.60
	M2S010	570	8	11.38	184.51	2100.12	125,797.09
	PGL12G	540	8	12.50	567.52	7095.14	368,947.02
	Artix-7	344	8	8.92	490	4370.80	349,226.92
Power Consumption	EF2M45	523	8	13.85	397.72	5509.19	142,688.01
	EG4S20	533	8	12.66	315.82	3997.32	199,466.32
	GW2A	534	8	5.91	314.48	1857.63	313,939.04
	Cyclone10LP10CL	504	8	10.02	275.91	2765.72	138,009.52
	MAX10M50DAF	507	8	9.59	321.22	3081.14	261,897.09
	ICE40UP5	516	8	13.47	148	1994.00	75,772.15
	MachXO2	851	/	15.21	216.43	3292.33	98,440.76
	M2S010	519	8	12.53	193.70	2427.30	145,395.56
	PGL12G	531	8	10.86	467.43	5078.16	264,064.30
	Artix-7	342	8	8.66	490	4243.40	339,047.66

References

1. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
2. Sakr, F.; Bellotti, F.; Berta, R.; De Gloria, A. Machine Learning on Mainstream Microcontrollers. *Sensors* **2020**, *20*, 2638. [CrossRef]

3. Deploy Machine Learning Models on Mobile and IoT Devices. Available online: https://www.tensorflow.org/lite (accessed on 1 April 2021).
4. STMicroelectronics X-CUBE-AI—AI Expansion Pack for STM32CubeMX. Available online: http://www.st.com/en/embedded-software/x-cube-ai.html (accessed on 1 April 2021).
5. Lai, L.; Suda, N.; Chandra, V. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. *arXiv* **2018**, arXiv:1801.06601.
6. DiCecco, R.; Lacey, G.; Vasiljevic, J.; Chow, P.; Taylor, G.; Areibi, S. Caffeinated FPGAs: FPGA Framework for Convolutional Neural Networks. In Proceedings of the IEEE 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016; pp. 265–268.
7. Brandalero, M.; Ali, M.; Le Jeune, L.; Hernandez, H.G.M.; Veleski, M.; da Silva, B.; Lemeire, J.; Van Beeck, K.; Touhafi, A.; Goedemé, T. AITIA: Embedded AI Techniques for Embedded Industrial Applications. In Proceedings of the IEEE 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain, 31 August–2 September 2020; pp. 1–7.
8. Kathail, V. Xilinx Vitis Unified Software Platform. In Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 23–25 February 2020; pp. 173–174.
9. Aydonat, U.; O'Connell, S.; Capalija, D.; Ling, A.C.; Chiu, G.R. An Opencl™ Deep Learning Accelerator on Arria 10. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–25 February 2017; pp. 55–64.
10. Intelligent Automation, Inc. DeepIP-FNN. Available online: https://www.xilinx.com/products/intellectual-property/1-15kaxa2.html (accessed on 2 May 2021).
11. Intel Intel® FPGA Technology Solutions for Artificial Intelligence (AI). Available online: https://www.intel.com/content/www/us/en/artificial-intelligence/programmable/solutions.html (accessed on 2 May 2021).
12. Shawahna, A.; Sait, S.M.; El-Maleh, A. FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review. *IEEE Access* **2019**, *7*, 7823–7859. [CrossRef]
13. Holanda Noronha, D.; Zhao, R.; Goeders, J.; Luk, W.; Wilton, S.J. On-Chip Fpga Debug Instrumentation for Machine Learning Applications. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 24–26 February 2019; pp. 110–115.
14. Saqib, F.; Dutta, A.; Plusquellic, J.; Ortiz, P.; Pattichis, M.S. Pipelined Decision Tree Classification Accelerator Implementation in FPGA (DT-CAIF). *IEEE Trans. Comput.* **2013**, *64*, 280–285. [CrossRef]
15. Attaran, N.; Puranik, A.; Brooks, J.; Mohsenin, T. Embedded Low-Power Processor for Personalized Stress Detection. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 2032–2036. [CrossRef]
16. Batista, G.C.; Oliveira, D.L.; Saotome, O.; Silva, W.L. A Low-Power Asynchronous Hardware Implementation of a Novel SVM Classifier, with an Application in a Speech Recognition System. *Microelectron. J.* **2020**, *105*, 104907. [CrossRef]
17. Roukhami, M.; Lazarescu, M.T.; Gregoretti, F.; Lahbib, Y.; Mami, A. Very Low Power Neural Network FPGA Accelerators for Tag-Less Remote Person Identification Using Capacitive Sensors. *IEEE Access* **2019**, *7*, 102217–102231. [CrossRef]
18. Wang, C.; Gong, L.; Yu, Q.; Li, X.; Xie, Y.; Zhou, X. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2016**, *36*, 513–517. [CrossRef]
19. Ge, F.; Wu, N.; Xiao, H.; Zhang, Y.; Zhou, F. Compact Convolutional Neural Network Accelerator for Iot Endpoint Soc. *Electronics* **2019**, *8*, 497. [CrossRef]
20. Jindal, M.; Gupta, J.; Bhushan, B. Machine Learning Methods for IoT and Their Future Applications. In Proceedings of the IEEE 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019; pp. 430–434.
21. Qian, B.; Su, J.; Wen, Z.; Jha, D.N.; Li, Y.; Guan, Y.; Puthal, D.; James, P.; Yang, R.; Zomaya, A.Y. Orchestrating the Development Lifecycle of Machine Learning-Based Iot Applications: A Taxonomy and Survey. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–47. [CrossRef]
22. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
23. Meshram, V.; Patil, K.; Hanchate, D. Applications of Machine Learning in Agriculture Domain: A State-of-Art Survey. *Int. J. Adv. Sci. Technol.* **2020**, *29*, 5319–5343.
24. Gong, Z.; Zhong, P.; Hu, W. Diversity in Machine Learning. *IEEE Access* **2019**, *7*, 64323–64350. [CrossRef]
25. Yang, L.; Shami, A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *Neurocomputing* **2020**, *415*, 295–316. [CrossRef]
26. Venieris, S.I.; Bouganis, C.-S. FpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs. In Proceedings of the 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Washington, DC, USA, 1–3 May 2016; pp. 40–47.
27. Faraji, S.R.; Abillama, P.; Singh, G.; Bazargan, K. Hbucnna: Hybrid Binary-Unary Convolutional Neural Network Accelerator. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
28. Akima, H. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *J. ACM (JACM)* **1970**, *17*, 589–602. [CrossRef]

29. Chen, H.; Jiang, L.; Yang, H.; Lu, Z.; Fu, Y.; Li, L.; Yu, Z. An Efficient Hardware Architecture with Adjustable Precision and Extensible Range to Implement Sigmoid and Tanh Functions. *Electronics* **2020**, *9*, 1739. [CrossRef]
30. Ramachandran, S. Synthesis of Designs–Synplify Tool. In *Digital VLSI Systems Design: A Design Manual for Implementation of Projects on FPGAs and ASICs Using Verilog*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 255–292.
31. Verimake Gutter Oil Dataset. Available online: https://github.com/verimake-team/Gutteroildetector/tree/master/data (accessed on 2 May 2021).
32. Schäfer, B.; Grabow, C.; Auer, S.; Kurths, J.; Witthaut, D.; Timme, M. Taming Instabilities in Power Grid Networks by Decentralized Control. *Eur. Phys. J. Spec. Top.* **2016**, *225*, 569–582. [CrossRef]
33. Arzamasov, V.; Böhm, K.; Jochem, P. Towards Concise Models of Grid Stability. In Proceedings of the 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, Denmark, 29–31 October 2018; pp. 1–6.
34. Cortez, P.; Cerdeira, A.; Almeida, F.; Matos, T.; Reis, J. Modeling Wine Preferences by Data Mining from Physicochemical Properties. *Decis. Support Syst.* **2009**, *47*, 547–553. [CrossRef]
35. Climate Data Online-Map Search-Bureau of Meteorology. Available online: http://www.bom.gov.au/climate/data/ (accessed on 2 May 2021).
36. Individual Household Electric Power Consumption Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption (accessed on 2 May 2021).
37. Dellamonica, J.; Lerolle, N.; Sargentini, C.; Beduneau, G.; Di Marco, F.; Mercat, A.; Richard, J.-C.M.; Diehl, J.-L.; Mancebo, J.; Rouby, J.-J. Accuracy and Precision of End-Expiratory Lung-Volume Measurements by Automated Nitrogen Washout/Washin Technique in Patients with Acute Respiratory Distress Syndrome. *Crit. Care* **2011**, *15*, 1–8. [CrossRef]
38. Hu, Y.; Zhu, Y.; Chen, H.; Graham, R.; Cheng, C.-K. Communication Latency Aware Low Power NoC Synthesis. In Proceedings of the IEEE 43rd annual Design Automation Conference, San Francisco, CA, USA, 24–28 July 2006; pp. 574–579.
39. Garofalo, A.; Rusci, M.; Conti, F.; Rossi, D.; Benini, L. PULP-NN: Accelerating Quantized Neural Networks on Parallel Ultra-Low-Power RISC-V Processors. *Philos. Trans. R. Soc. A* **2020**, *378*, 20190155. [CrossRef] [PubMed]
40. Slater, W.S.; Tiwari, N.P.; Lovelly, T.M.; Mee, J.K. Total Ionizing Dose Radiation Testing of NVIDIA Jetson Nano GPUs. In Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 22–24 September 2020; pp. 1–3.
41. Lang, R.; Lescisin, M.; Mahmoud, Q.H. Selecting a Development Board for Your Capstone or Course Project. *IEEE Potentials* **2018**, *37*, 6–14. [CrossRef]
42. Crocioni, G.; Pau, D.; Delorme, J.-M.; Gruosso, G. Li-Ion Batteries Parameter Estimation with Tiny Neural Networks Embedded on Intelligent IoT Microcontrollers. *IEEE Access* **2020**, *8*, 122135–122146. [CrossRef]