ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326005988

Deep Learning and Reconfigurable Platforms in the Internet of Things: Challenges and Opportunities in Algorithms and Hardware

Article *in* IEEE Industrial Electronics Magazine · June 2018
DOI: 10.1109/MIE.2018.2824843

CITATIONS
0

READS
92

4 authors:

Roberto Fernandez Molanes
University of Vigo
8 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Kasun Amarasinghe
Virginia Commonwealth University
20 PUBLICATIONS **121** CITATIONS
SEE PROFILE

J.J. Rodriguez-Andina
University of Vigo
115 PUBLICATIONS **871** CITATIONS
SEE PROFILE

Milos Manic
Virginia Commonwealth University
189 PUBLICATIONS **1,706** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

resilient control and instrumentation systems View project

SUBMIT - conference submission content management system View project

All content following this page was uploaded by J.J. Rodriguez-Andina on 03 July 2018.

The user has requested enhancement of the downloaded file.

BELOW YOU MAY FIND THE ACCEPTED VERSION OF THE ARTICLE:

Roberto Fernández Molanes, Kasun Amarasinghe, Juan J. Rodríguez-Andina, and Milos Manic, “Deep Learning and reconfigurable platforms in the Internet of Things: Challenges and opportunities in algorithms and hardware”, *IEEE Industrial Electronics Magazine*, vol. 12, no. 2, pp. 36-49, June 2018.

DOI: 10.1109/MIE.2018.2824843

THE PUBLISHED VERSION IS AVAILABLE IN IEEE Xplore, AT:
http://ieeexplore.ieee.org/document/8396317/

Deep Learning and Reconfigurable Platforms in IoT

Challenges and Opportunities in Algorithms and Hardware

As the Internet of Things (IoT) continues its run as one of the most popular technology buzzwords of today, the discussion really turns from how the massive data sets are collected to how value can be derived from them, i.e., how to extract knowledge out of such (big) data. IoT devices are used in an ever-growing number of application domains (see Figure 1), ranging from sport gadgets (Fitbit, Apple Watches) or more serious medical devices (such are pacemakers or biochips) to smart homes, cities, and self-driving cars, to predictive maintenance in mission-critical systems (e.g., in nuclear power plants or airplanes). Such applications introduce endless possibilities for better understanding, learning from, and informedly acting upon (aka situational awareness and actionable information in government lingo). While rapid expansion of devices and sensors brings terrific opportunities for taking advantage from terabytes of machine data, the mind-boggling tasks of understanding growth of data remain, and heavily rely on, artificial intelligence and machine learning [1], [2].

Where traditional approaches do not scale well, artificial intelligence techniques have evidenced great success in applications of machine and cognitive intelligence (such as image classification, face recognition, or language translation). While we recognize the widespread usage of various well-known machine learning algorithms in IoT (such as fuzzy systems, support vector machines, Bayesian networks, reinforcement learning, and others), we focus here on the most recent and highly advantageous type of machine learning in IoT: deep learning.

The success of deep learning and, in particular, deep neural networks, greatly coincides with the advent of highly specialized, powerful parallel computing devices, namely Graphics Processing Units (GPUs) [4]. Although the overwhelming processing and memory requirements can be met

with high-performance computing hardware, the resulting sheer size, cost, and power consumption would make the goal of "deep neural network-enabled" IoT and embedded devices unattainable.

In this scenario, Field Programmable System-on-Chip (FPSoC) platforms, which combine in a single chip one or more powerful processors and reconfigurable logic (in the form of Field Programmable Gate Array –FPGA– fabric), are emerging as a very suitable implementation alternative for the next generation of IoT devices. The fine-grained structure of FPGAs has proven to provide powerful implementations of machine learning algorithms with less power consumption than comparable platforms (in terms of cost or size) [5], making them ideal for machine and cognitive intelligence in strict resource-limited applications, like many IoT ones (while GPUs remain as the dominant platforms for other IoT scenarios). Moreover, FPSoCs allow processing load to be balanced between processors and reconfigurable logic, the most suitable implementation (hardware or software) being used for each specific functional building block to be optimized, and functionality to be easily reconfigured on-site. In addition, reconfigurable platforms dramatically ease system scalability and upgrading. Hence they provide high levels of flexibility, as demanded by the IoT market.

In this regard, the paper identifies hardware implementation challenges and thoroughly analyzes the aforementioned suitability of FPSoCs for a broad range of IoT applications involving machine learning and artificial intelligence algorithms, which is demonstrated in two case studies, one related to deep learning and another related to the more "classical" Evolutionary Computing techniques.

Deep Learning for IoT

In the era of IoT, the amount of sensing devices that are deployed in every facet of our day-to-day life is enormous. In recent years, many IoT applications have risen in various different domains such as health, transportation, smart homes, and smart cities [6]. It is predicted by the US National

Intelligence Council that, by 2025, internet nodes will reside in everyday things such as food packages, furniture, and documents [7]. This expansion of IoT devices, together with cloud computing has led to a creation of an unprecedented amount of data [8], [9]. With this rapid development of IoT, cloud computing, and the explosion of big data, the most fundamental challenge is to store and explore these volumes of data and extract useful information for future actions [9].

The main element of most IoT applications is an intelligent learning methodology that senses and understands its environment [6]. Traditionally, many machine learning algorithms were proposed to provide intelligence to IoT devices [10]. However, in the recent years, with the popularity of deep neural networks / deep learning, using deep neural networks in the domain of IoT has received increased attention [6], [11]. Deep learning and IoT were among the top three technology trends for 2017 announced at Gartner Symposium/ITxpo [12]. This increased interest of deep learning in the IoT domain is due to the fact that traditional machine learning algorithms have failed to address the analytic needs of IoT systems [6], which produce data at such a rapid rate and volume that they demand artificial intelligence algorithms with modern data analysis approaches. Depending on the predominant factor, rate or volume, data analytics for IoT applications can be viewed in two main categories: 1) Big data analysis and 2) Data stream analysis, as discussed below.

When focusing on data volume, IoT is one of the major sources of big data. Analytics of the generated massive datasets directly benefit the performance and enhance capabilities of IoT systems. Extracting knowledge from such big data is not a straightforward task. It requires capabilities that go beyond the traditional inference and learning techniques [13], generally expressed with the "6V's" [14], [15]:

- Volume, which refers to the ability to ingest, process, and store large data sets (petabytes or even exabytes).

- Velocity, which refers to the speed of data generation and frequency of delivery (sampling).
- Variety, which refers to the data from different sources and types (structured or unstructured). Even the types of data have been growing fast.
- Variability, which refers to the need of getting meaningful data considering scenarios of extreme unpredictability.
- Veracity, which refers to bias, noise, and abnormality in data (only the relevant, usable data within analytic model is to be stored).
- Value, which refers to the purpose the solution has to address.

Figure 2 shows the 6V's of big data and how the advantages of deep learning techniques can be used to meet these challenges in big data. More specific applications of deep learning techniques in big data in IoT are presented in next section.

The latest considerations add 3 additional Vs to the mix: Vulnerability (of data), Volatility (relevance of data before becoming obsolete), and Visualization (ways of meaningful visualization).

As mentioned, in addition to performing data mining on massive collections of data produced by IoT systems, another important aspect is dealing with real-time data streams that require fast learning algorithms. IoT applications, such as traffic management systems and supply chain logistics of super markets, involve large datasets that have to be analyzed in near real time [16]. Mining fast generated data streams require the algorithms to be adaptable to the change of data distributions as the environment changes around the devices [17]. This context/concept drift occurs due to the changes in factors such as location, time, and activity. In addition to the requirement of speed adaptability, the lack of labeled data in IoT data streams adds to the difficulty because it makes supervised learning methods inadequate for analysis [17], [18]. Therefore, highly adaptable

unsupervised and semi supervised deep learning techniques are required for mining the fast-changing data streams in IoT devices.

Applications of Deep Learning in IoT

Deep neural networks have revolutionized a multitude of fields because of its ability for learning through multiple layers of abstraction [19], [20]. This enables learning of complex patterns that are hidden in complex datasets, a capability ideal for mining massive heterogeneous datasets. Different deep neural network algorithms have been used to good effect in a range of areas, very difficult to tackle in the past. For instance, long short terms memory algorithms have been shown to be extremely useful in speech recognition and natural language processing [21], [22], [23] and convolutional neural networks have been used to produce state-of-the-art performance in many vision applications such as image classification [24], [25]. Therefore, deep learning is applied extensively in a range IoT devices for human interaction.

One of the most important derivatives of the IoT is the concept of smart cities. Improving cities is becoming a global need with the rising and urbanization of the population [26]. The concept of "smart cities" has been in the features since the early 2000s. Smart cities claim to contain thousands of sensing devices, which generate massive amounts of data that can be harnessed to optimize and improve the operations of these cities [27]. Smart cities try to accomplish goals such as reducing pollution and energy consumption or optimizing transportation [28]. IoT devices can help collect data about how people use cities and machine learning algorithms can be used to understand that [26]. Adding further intelligence to the embedded sensing nodes allows local storage needs and network congestion to be reduced. One of the most important aspects of smart cities powered by IoT is smarter energy management. With the advent of smart meters, there are massive amounts of data being collected on energy consumption. Therefore, it enables research on energy consumption prediction, which can lead to optimizing energy usage and the way energy is

generated in smart cities and smart grids. Machine learning algorithms are indispensable in this area and deep learning algorithms such as long short terms memory algorithms, Restricted Boltzmann machines, and convolutional neural networks have been proposed to perform data driven predictions of energy usage at both individual consumer and aggregate level [32]-[34].

Another important aspect of smart cities is using machine learning and IoT for traffic management. Optimized traffic management targets reducing congestion, long queues, delays, and even the carbon footprint of cities [33]. To that end, driverless or self-driving cars have become a much-discussed topic in the recent past with major car companies like Tesla, BMW, or Ford and tech giants such as Google and Apple stepping up to the plate of developing truly intelligent autonomous cars. Self-driving cars have a plethora of devices continuously sensing its environment and a suit of machine learning algorithms for understanding and fusing the various data sources such as LIDAR depth maps and images. Deep neural networks have been extensively explored in this domain as they have the capability of automatically learning features to pick out obvious ones such as lane marking and road edges to other subtle ones that exist on the roads [34].

Computer vision is a highly sought-after application in many use cases in the IoT domain. Smart cameras, especially in smart security systems, play an important role in smart homes [35] and vision applications such as face recognition are very crucial [36]. Machine learning algorithms have been used extensively in image processing applications and, in that, convolutional neural networks have been deemed the gold standard since the advent of LeNet [37]. Ko et al. presented a framework for energy efficient neural networks to be used in IoT edge devices [38]. The authors claim that in deploying deep neural networks-based image processing, energy efficiency can be the performance bottleneck, and hence present the recent technological advantages for making deep neural networks such as convolutional and recurrent neural networks more energy-efficient.

Another area in which machine learning-driven vision applications is sued in IoT is human activity

recognition in smart homes. Fang et al. proposed a deep learning-based framework for human activity recognition in smart homes especially helping people with diseases [39]. Context awareness is another important aspect of IoT, closely tied with mining data streams. Machine learning has a very crucial role to play in understanding the environment and the context of the device from the data. In recent years, we have seen commercial IoT devices or edge devices emerging in the market such as Nest Thermostat [40] and Amazon devices powered by Alexa [41], which have the ability of sensing its environments and using machine learning to understand data. Context-aware devices or "things" have the ability of understanding the environment and adapting their reasoning capabilities [10]. Further, machine learning algorithms are extremely crucial for areas as intelligent health trackers for medicine. Examples are intelligent pacemakers or ppg systems [42], [43] that can monitor the heartbeat of the patient. Adding intelligence to this devices is very important as it permits improved and faster preventive detection of pathologies. Compared with the option to send data via internet to remote sensors for analysis or saving data for post processing, this option enables a dramatic reduction of data transmission and storage (with the respective reduction of energy consumption) and the possibility to work offline (very useful for remote or rural areas).

Safety and Security in IoT

In addition to enabling and facilitating IoT applications, deep learning plays a crucial role in keeping the highly-connected devices safe. Due to their ubiquity in the modern technological ecosystem, the IoT becomes a very attractive target for cyber attackers. Therefore, cyber security is one of the most important research areas in the field of IoT [44], [45]. It is known that a large amount of zero-day attacks are emerging continuously due to the various protocols added to the IoT [46]. The multiple level feature learning capabilities of deep learning has been exploited in this domain to good effect. Diro et al. presented a deep neural networks-based distributed

methodology for cyber-attack detection in IoT [46]. They compared their distributed deep model with a shallow neural network and a centralized deep model and they concluded that the distributed deep model outperforms the others significantly. Another area of cyber security is malware detection. Pajouh et al. presented a deep recurrent neural network-based malware detection methodology for IoT [47]. The authors implemented three different long short terms memory configurations and showed that their algorithm can achieve 98.18% accuracy in malware detection for the tested dataset. In all aspects of cyber security, when taking a data-driven approach, anomaly detection algorithms are very useful tools. Canedo et al. presented an artificial neural network-based anomaly detection methodology tailored for IoT cyber security [48]. They recognized that the main challenges for anomaly detection in IoT data are quantity and heterogeneity. They showed that the artificial neural network-based methodology was able to overcome those challenges in detecting anomalies in the data sent from edge devices.

Hardware Implementation Challenges

The implementation of machine learning algorithms has been a hot topic in research for several years but has recently boomed, mainly thanks to the opportunities created by the advancements in chip fabrication technologies, which enabled solving design problems at a cost and with a time-to-market that were unthinkable just a few years ago. The resolution of Google Challenge by AlexNet using 8-layer deep neural network [24] is usually cited as an inflexion point that boosted the research on new chips and applications of machine learning algorithms, especially in the field of neural networks. This explosion coincides with the deceleration of Moore´s Law[1], which now makes it economically reasonable to work on optimized software and hardware structures, as opposed to the trend of the last 30 years, where waiting for the next generation of devices was

[1] Even Gordon Moore himself predicted the end of his Moore's Law [92]

more profitable than investing in optimization. All these facts combined make more difficult than ever for designers to decide the best possible architecture for their applications.

The digital processing platforms currently available in the market are summarized in Figure 3, where they can be compared in terms of performance and flexibility. Flexibility refers here to ease of development, portability, and possibility for adapting to changes in specifications. For high-end deep neural network applications, where performance is the most important parameter, General Purpose Graphics Processing Units (GPGPUs) are the dominant solution. Their parallel structure, the latest efforts by manufacturers to compete for machine learning applications (e.g., adding specific instructions for fast neuron inference), and its reduced cost due to the mass production for personal computers made them ideal for training and inference of deep neural networks. The latest NVIDIA Volta™ GV100 GPU platform, including 21.1 billion transistors within a die size of 815 mm^2, is capable of doing inference 100 times faster than the fastest current CPU in the market [49]. This unparalleled brute power force comes at a price: high power consumption, the need for custom data types (not necessarily float), irregular parallelism (alternating sequential and parallel processing), and divergence (not all cores executing the same code simultaneously). That is why some companies are investing on neural network application-specific integrated circuits (ASICs) for improved performance at the expense of losing flexibility. Examples are 1st and 2nd generation (optimized for inference and both inference and training, respectively) of Google Tensor Processing Unit (TPU), slowly "stealing" high-performance computing applications from GPUs.

While this is the pace for high-performance computing, the lack of flexibility in ASICs and the high power consumed by GPUs do not fit in wide areas of the IoT world, which demand power-efficient, flexible embedded systems. This explains why many IoT devices are currently based on microcontrollers (μCs), Digital Signal Processors (DSPs), and multicore CPUs. However, as the IoT market grows, both manufacturers and designers face a problem due to the diversification of

applications and increasing demand for computing power (particularly for machine learning algorithms) leading a transformation from “sense making” to “decision making” [50]. Offering a wider portfolio of devices to cover the different applications means less market share per device, increasing manufacturing costs. On the other hand, offering complex heterogeneous devices that can be used in several applications implies higher integration of functionality and a waste of silicon, also increasing the overall cost [51]. In this scenario, FPGAs, located in the middle of Figure 3, appear as a balanced solution to add flexibility and efficient computing power for machine learning algorithms to the next generation of IoT devices. Combining processors and FPGAs in a single package results in the FPSoC concept. In the following sections, FPSoC architecture is presented along with an analysis of the usefulness of its hardware resources for implementing machine learning algorithms in IoT devices.

FPSoC Architecture

FPSoCs feature a Hard Processing System (HPS) and FPGA fabric on the same chip. Both parts are connected by means of high-throughput bridges, which provide faster communications and power savings compared to multichip solutions [53]. The HPS in first-generation FPSoCs featured single- or dual-core ARM application processors and some widely used peripherals, such as timers and controllers for different types of communication protocols, namely Ethernet, USB, I2C, UART, and CAN. Pushed by increasing application requirements, some devices in the newest FPSoC families include quad-core ARM processors, GPUs, and real-time processor in the HPS, FPSoCs becoming complex heterogeneous computing platforms. Resources in the FPGA fabric also evolved from the “basic” structure consisting of standard logic resources and relatively simple specialized hardware blocks (fixed point DSP multipliers, memory blocks, and transceivers, to name just a few). Current devices include much more complex blocks, e.g., DSP blocks with floating point capabilities, video codecs for video compression, Soft-Decision Forward Error

Recovery (SD-FEC) units to speed-up encoding/decoding in wireless applications, or Analog-to-Digital Converters (ADCs). Figure 4 shows the generic block diagram of a modern FPSoC device, where the location and connection of the aforementioned elements is depicted. All computing elements (processors and GPU) have their own cache memory and share common SDRAM external memory, usually controlled by a single multiport controller. A main switch interconnects masters and slaves in the HPS. The FPGA fabric can be accessed as any other memory-mapped peripheral from the HPS through the HPS-to-FPGA bridges. On the other hand, there are several options to access the HPS from the FPGA fabric: FPGA-to-HPS bridges to access HPS peripherals, the Accelerator Coherency Port (ACP) to coherently access processor cache, and FPGA-to-SDRAM bridges to access main memory in a non-coherent way.

Not all FPSoCs include all blocks in Figure 4. Table I shows a summary of characteristics of the most relevant currently available FPSoC families. Intel FPGA and Xilinx offer powerful devices with application processors and large FPGA fabrics, focused on higher end applications, such as 5G communications, artificial intelligence, data centers, or video processing. Microsemi and Quicklogic offer simpler devices with real-time processors, focusing on data acquisition, wearables and smartphones.

Despite the additional components that manufacturers provide in some devices targeting specific applications, the most important in an FPSoC are still the HPS processors and the FPGA fabric. To successfully deploy an application taking the most possible advantage of these devices, processors and FPGA should smoothly cooperate with each other executing the parts of the functionality that best fit their respective architectures, sharing data between them when needed. A designer typically starts with a software implementation in HPS and moves to the FPGA those parts of the code that need acceleration. Communication between HPS and FPGA is not a trivial task, and depends on several factors, such as data size, operating system (OS), or FPGA operating

frequency, among others. It is very important to choose the best possible mechanism for HPS-FPGA data exchange, otherwise it can impair the acceleration achieved by moving portions of the algorithms to hardware. In [57]-[59], different analyses of the influence of these factors in the transfer rate are carried out. In [56], the results of the analysis are elaborated into design guidelines to maximize the performance of FPSoC implementations.

Table I. Characteristics of modern FPSoC families.

Company	Family	Transistor size	Application Processor		Real Time Processor		FPGA		Other
			Type	Max f (GHz)	Type	Max f (MHz)	Max Size	Max f (MHz)	
Intel FPGA	Cyclone V SoC	28 nm	Single/Dual 32-bit ARM Cortex-A9	0.925	-	-	301 K LEs	200	
	Arria V SoC	28 nm	Single/Dual 32-bit ARM Cortex-A9	1.05	-	-	462 K LEs	300	
	Arria 10 SoC	20 nm	Dual 32-bit ARM Cortex-A9	1.5	-	-	1.15 M Les	500	Floating point DSP blocks in FPGA
	Stratix 10 SoC	14nm tri-gate	Quad 64-bit ARM Cortex-A53	1.5	-	-	5.5 M LEs	1000	Floating point DSP blocks in FPGA
Xilinx	Zynq-7000 Artix	28 nm	Single/Dual 32-bit ARM Cortex-A9	0.866	-	-	85 K LCs	-	ADC
	Zynq-7000 Kintex	28 nm	Dual 32-bit ARM Cortex-A9	1	-	-	444 K LCs	-	ADC
	Ultrascale+ Kintex	20 nm	Dual/Quad 64-bit ARM Cortex-A53	1.5	Dual Cortex-R5	600	1143 K LCs	-	Option to GPU, Video codec, ADC, DAC, SD-FEC
Microsemi	SmartFusion	130nm	-	-	Single Cortex-M3	100	6 K LEs	350	ADC, non-volatile FPGA
	SmartFusion 2	130nm	-	-	Single Cortex-M3	166	150 K LEs	350	ADC, non-volatile FPGA
QuickLogic	S3	-	-	-	Single Cortex M4-F	80	-	-	DSP-processor, Power management unit

FPGA design is typically based on Hardware Description Languages (HDLs), which require from designers good knowledge of digital hardware. Fortunately, nowadays it is also possible to

automatically compile code for both the FPGA and the HPS from high-level languages, namely C/C++ (using high level synthesis tools, either commercial or open-source like Legup [57]), OpenCL, Matlab, and Labview. This gives designers with limited or no experience in digital design access to the excellent characteristics of FPSoCs. Code generated by these tools is not as optimized as that resulting from HDL workflows, but they allow design time to be dramatically reduced [58].

FPSoCs and IoT

FPSoC characteristics make them very suitable for many IoT applications. The availability of HPS peripherals for the most popular communication protocols enables interoperability among a broad range of devices [59]. For example, the HPS can simultaneously connect with sensors using I2C and with other devices via Ethernet or Wi-Fi. On the other hand, the FPGA fabric adds great flexibility, enabling the implementation of communication protocols not included in HPS, as well as specific functionalities that achieve higher performance in hardware than in software, such as PWM, capture and compare, or frequency measurement units.

Connectivity of IoT devices raises serious security and privacy concerns. At hardware level, one possible way to address them is with ARM's TrustZone Technology [60], which defines some peripheral slaves as secure, so only trusted masters can access them. For instance, a secure interrupt controller may be used to create a non-interruptible task that monitors the system and a secure keyboard may ensure secure password entries. This concept has also been extended to software, as shown in Figure 5. A trusted firmware layer controls context switching of the processor from trusted OS and apps to regular OS and apps, which may run malicious software completely isolated from trusted software and secure hardware.

To protect intellectual property, current FPSoCs also allow the FPGA configuration bitstream as well as the boot image for the HPS to be encrypted [61]. In addition to the solutions provided by manufactures, extra functionalities can be implemented to prevent hacker attacks. These include

physically unclonable functions, useful for unique network identification, traceability and access control [62].

FPSoCs enable the design of embedded systems with very small size, low power consumption, and performance sometimes even equal or higher than desktop platforms [64]. Regarding energy, FPSoCs largely outperform computer systems in terms of operations per second and watt [65]. FPSoCs are also more power-efficient than GPU-based SoC designs [66], particularly for neural network implementations [67], [68]. However, poor usage of the available FPGA resources may result in some cases in CPUs and GPUs outperforming them [69]. With this concern in mind, FPSoCs are the best option for implementing machine learning in battery-powered systems with strict size limitations, like drones [70] or wireless sensor networks [71].

Regarding economic and marketing issues, FPSoCs are inexpensive since they are mass-produced components. Time to market is short and, thanks to the new high level synthesis tools (like OpenCL and C/C++ compilers), similar to that of pure software solutions. Because of its reconfigurable nature, functionality can be upgraded without the need for changing the hardware platform, improving post-sale support compared to non-configurable devices like ASICs.

FPSoCs and Machine Learning

FPGAs exhibit some unique features for efficiently implementing portions of machine learning algorithms in hardware:

- Parallelism. Most machine learning algorithms include parallelizable portions of the code that can take advantage of this property of the hardware. For example, each neuron in a neural network layer can be computed in parallel. In evolutionary computing, fit functions can also be concurrently executed for the whole population of genes/particles.
- Pipelining. Although this technique is also used in processors and GPUs to fetch and execute instructions, much more advantage of it can be taken in FPGAs, where the output of an

operation can feed directly the input of the next one, avoiding the extra clock cycles required to compute the same operations in the Arithmetic/Floating Point Units of processors and GPUs.

- Scalability and upgrading. It is usual for machine learning algorithms to change structure or size (e.g., adding layers or inputs to a neural network) to improve performance from knowledge gained during test or normal operation. In a hardware/software coprocessing implementation, this may mean to port more (or new) parts of the algorithm to hardware. The same may happen in the context of IoT when new functionality, whether related to the target machine learning algorithm or not (such as a web server or an encryption algorithm), needs to be added to the system. FPGAs abundance of standard logic resources and specialized hardware blocks, together with their reconfiguration capabilities, facilitates system scalability and upgrading.

Current FPGAs include tens to hundreds of DSP blocks usually equipped with fixed-point multipliers and adders. Other operations, e.g., floating point, are implemented by a combination of these blocks and standard FPGA logic elements (LEs). FPGAs are very powerful for fixed point operations [72], but achieve less performance in number of floating point operations per second than GPUs for most machine learning implementations [73]. However, in some cases the configurable FPGA architecture compensates this drawback and achieves faster execution times [74]. In an effort to make FPSoCs more competitive, newer devices from Intel FPGA (Arria 10 and Stratix 10 families) include DSP blocks with single floating point capabilities in the FPGA fabric. Table II summarizes the size (LE and DSP block usage) and performance (latency and maximum operating frequency, f_{MAX}) of floating point operators in Arria V and Arria 10 FPGAs for some usual floating point operations in machine learning algorithms. It can be seen that double precision operations require more than twice the resources and have almost twice the latency of single precision ones. It can also be noticed that addition, subtraction, and multiplication make low usage of resources, whereas other operators are less efficiently implemented. Using floating point

DSP blocks results in improvements in terms of either significant reduction of logic resource usage or increase of maximum operating frequency. The exception is the exponential operation, because it does not suit well the fixed structure of floating point DSP blocks.

In low-level design with HDLs it is easy to estimate the performance of a given algorithm implementation in a given device from the information regarding available hardware resources and latency of the different operations. This is not the case when using high level synthesis tools, where the compiler can make an inefficient use of hardware resources. To achieve acceptable performance when using these tools it is a must to consider all the available options to help the tool efficiently fit the design in the FPGA fabric [76].

Table II. Resource usage and latency for usual floating point operations in Arria FPSoCs [75].

Operation	Floating Point Precision	Arria V (fixed point DSP blocks)				Arria 10 (floating point DSP blocks)			
		Latency (clock cycles)	LEs	DSP blocks	f_{MAX} (MHz)	Latency (clock cycles)	Les	DSP blocks	f_{MAX} (MHz)
Addition/ Subtraction	Single	9	1193	0	250	5	1208	0	319
	Double	12	2903	0	252	7	2765	0	290
Multiplication	Single	5	390	1	281	3	123	1	289
	Double	7	848	4	186	5	780	4	289
Division	Single	18	1140	4	249	16	985	4	347
	Double	35	3523	15	185	30	3020	15	258
Exponential Base e	Single	14	1795	2	217	26	745	6	365
	Double	28	5335	10	185	28	5390	10	260
Sine	Single	12	1463	3	240	11	1463	3	280
	Double	29	4370	14	185	29	4795	14	260

The aforementioned hardware features are complemented in FPSoCs with those provided by the application processors in HPS. Those range from the real-time processors with fixed-point arithmetic capabilities available in simpler devices, to DSP-like processors for speeding up signal processing tasks, or to dedicated floating point units or single instruction multiple data coprocessors for vector arithmetic in more advanced devices.

Case Study I: Implementation of Deep Neural Networks in FPSoC

Neural network algorithms and, in particular, deep neural networks are executed in two phases: training (where network weights are adapted to achieve the desired functionality) and inference (deployment operation of the network). Training is highly computationally-demanding, so it is typically implemented by processing batches of data (several patterns at the same time) offline, for which GPUs are very suitable. The inference phase is suitable for FPGA implementation, because it typically has to be implemented over single patterns in real time and, as it can be concluded from Figure 6, the neurons in one layer can be executed in parallel. Moreover, the operations to be performed by each neuron can be very efficiently implemented using DSP blocks. These operations are:

$$a_x(y) = \sigma\left(\sum_{i=0}^{n-1} a_i(y-1) * w_{ix}\right) \quad (1)$$

where $a_x(y)$ is the output of neuron x in layer y, w_{ix} is the weight between neuron i in layer y-1 and neuron x in layer y, and σ is the so-called activation function of the neuron.

The classical neuron activation functions are $Sigmoid(x) = \frac{1}{1+e^{-x}}$ and $Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. These operations involve divisions and exponentials so, according to Table II, its FPGA implementation is not particularly efficient. Because of that, some works addressed their efficient hardware implementation using linear approximations. The use of Taylor approximations and reuse of the multipliers and adders for the linear part of the neuron is proposed in [77], reducing the additional hardware needed for the activation function to almost none. The solution in [78] incurs just 0.03% error with regard to an implementation using true exponential and division cores. On the other hand, the activation function $ReLu(x) = \max(0, x)$ has recently been shown to provide better classification results and shorter training times than the former ones for deep neural networks [79], simplifying their implementation in all platforms.

Although most implementations use floating point operations, recent works have shown that fixed point approximations provide equal performance in some cases [80]. Moreover, for some applications it is possible to aggressively scale down (what is called quantization) the number of bits in fixed point representations. For instance, in [81] it is reported that with only 5-bit integer resolution for the weighting coefficients, performance degradation is negligible compared with the original 32-bit floating point resolution. Other operations that can be used to reduce FPGA logic resource usage are network pruning (removing non-important connections) [81], network clustering (fusing neurons) [82], and retraining (adding a penalty term in the training cost function to maximize not only the network fitting to inputs and outputs but also the bit depth needed for the network weights) [83]. These techniques, together with the use of simpler activation functions like ReLu, will surely boost the number of implementations in FPGA-based devices in the near future.

FPSoC platforms have already been used to improve pure FPGA implementation. In [84] a Zynq-7000 is used to implement an image classifier based on a deep convolutional neural network. The network layers (convolutional, pooling, and fully connected layers) are executed in the FPGA, whereas the HPS is responsible for synchronization (controlling DMA in the FPGA) and the final steps of the classification process. A set of configurable processing elements (PEs) performs all network operations (see Figure 7). This implementation is compared against others using an Intel Xeon CPU @2.9GHz, an Nvidia TK1 mobile GPU with 192 CUDA cores, and an Nvidia K40 GPU with 2880 Cuda cores, respectively. Results show that the FPSoC is 1.4 times faster than the CPU, with 14 times less power consumption; 2 times faster than the mobile GPU, with the same power consumption; and 13 times slower than the GPU, but consuming 26 times less power. This shows that FPSoCs achieve excellent performance-power consumption tradeoffs.

In [85], a Zynq-7000 is used to implement a Deep-Q network (Figure 8) that learns how to play a board game called Trax. Starting from a pure C/C++ software implementation and using high level

synthesis, the most time-consuming parts of the algorithm, in this case matrix multiplication of the convolutional layers, were moved to hardware. Each layer has its own matrix multiplication core that uses a double precision floating point multiply-accumulate (MAC) module to perform operations and two FPGA-SDRAM ports to share data with the processor in the HPS. One port is used to read operands from the processor and the other to write results back. The processor executes the rest of the algorithm. Results show a 26x acceleration with respect to the pure software implementation. Design time was very short, since hardware was directly compiled from C/C++ code using high level synthesis and only the most time-consuming parts of the algorithm were migrated to hardware. This example shows that high level synthesis tools may allow impressive performance improvements to be achieved by migrating software implementations to hardware ones with little programming effort.

Artificial neural network implementation in FPGA-based devices is becoming so popular that a neural network compiler, which generates HDL code from high level specifications, has been recently created [86]. Designers have only to select the structure, activation function, and other parameters of the artificial neural network and the compiler automatically generates the HDL code applying the most suitable optimization options in each case. This reduces the design time compared to using high level synthesis, where a deep analysis of the network and the FPGA is needed to optimize the implementation.

Case Study II: Implementation of Evolutionary Computing in FPSoC

FPSoCs are suitable implementation platforms not only for deep learning algorithms such as deep neural networks, but also for other machine learning algorithms (such as Evolutionary Computing ones) used in a wide range of IoT applications.

Evolutionary computing algorithms are used for complex optimization problems. In them, a population of individuals (e.g., “particles” or “genes”) is spread through the solution space and a

fit function is evaluated for them, the goal being minimize or maximize it. Depending on the values of the fit function for the different individuals in the current and past iterations, these move towards a possible solution. After some iterations the algorithm should converge to the global solution. Several families of such algorithms exist, which are characterized by the search policy of the individuals: ant colony optimization (which emulates ant colony food search), particle swarm optimization (which emulates the movement of a flock of birds where the distance between individuals is important), or genetic algorithms (where particles experience gene evolution through, e.g., mutation and crossover), to name just the most popular ones.

Although the fit function can be evaluated in parallel for each individual, evolutionary computing algorithms are not always as suitable for FPGA implementation as artificial neural networks, because their arithmetic operations are completely dependent on the application and the algorithm used. The application defines the fit function and, depending on the operations involved, it will be more or less appropriate for FPGA implementation. Generally speaking, the more pipelineable and parallelizable the fit function is the better. Also, according to Table II fit functions involving multiplications and additions are more suitable for FPGA implementation than those using exponentials and divisions. The operations involved in particle movement in the aforementioned evolutionary computing algorithms are:

- Ant colony: addition, multiplication, division, exponential, square root, and random number generation [87]. Hence, these algorithms are not particularly suitable for FPGA implementation.
- Particle swarm optimization: multiplication, addition, and random number generation [88], which can be efficiently implemented in FPGA.
- Genetic algorithms: random number generation and movement or modifications of chromosomes [89]. Processing of chromosomes perfectly fits in FPGA hardware, to the extent that it can be concurrently executed for all individuals in a single clock cycle.

Until recently, when considering the use of configurable platforms for implementing Evolutionary Computing algorithms, both the algorithm itself (particle movement) and the evaluation of the fit function were typically executed in hardware [88], [90]. In some cases where simple fit functions can be used, a soft processor (i.e., a processor implemented using standard FPGA logic resources) may be in charge of evaluating the fit function in software, as reported for instance in [91]. However, in real-life problems it is very usual that fit function evaluation takes most of execution time and soft processors are not fast enough to justify a software implementation, therefore most designers opted for pure hardware implementations.

The situation is different nowadays with the availability of powerful FPSoC devices, whose embedded hard processors work much faster than soft ones and have in many cases floating point capabilities. In this scenario, the most efficient solution is to implement the evaluation of the fit function in hardware and execute the algorithm in software.

In [64], a particle swarm optimization algorithm is proposed for evaluating the state of health of solar panels located in remote areas, where human intervention is difficult. In a pure software implementation, the evaluation of the fit function takes 83% of the execution time. Using a Cyclone V SoC device the evaluation of the fit function is moved to hardware. In a first approach, the processor waits in idle state for the FPGA to finish this evaluation. Even though in this particular case the fit function is neither internally parallelizable nor pipelineable, it can be concurrently computed for 12 particles, resulting in 3.4x acceleration with regard to the pure software implementation. An improved solution takes advantage of idle processor time for it to generate the random numbers to be used in subsequent iterations of the algorithm, resulting in 4.8x acceleration. The achieved performance is comparable to that obtained with a desktop computer, but with much lower size, cost, and power consumption, as shown in Figure 9(a). The whole monitoring system fits in a small electric box [Figure 9(b)] and can be located under each panel.

Closing discussion

The ubiquitous deployment of machine learning and artificial intelligence across IoT devices has introduced various intelligence and cognitive capabilities. One may conclude that these capabilities have led to the success of a wide and ever growing number of applications such as object/face/speech recognition, wearable devices and biochips, diagnosis software, or intelligent security and preventive maintenance.

Developments in other areas, such as humanoid robots, self-driving cars, or smart buildings and cities will likely revolutionize the way we live in the very near future. This new realities come with significant advantages, but also with many challenges related with the acquisition, processing, storage, exchange, sharing, and interpretation of the continuously-growing overwhelming amount of data generated by the IoT.

Up to now, complex applications involving deep neural networks have mainly used the brute force of GPUs for both training and inference. In the last 2 years, some companies have produced ASICs with better performance and lower power consumption than GPUs. These solutions are suitable for high-performance computing applications, but neither the low flexibility of ASICs nor the high-power consumption of GPUs are suitable for many IoT applications, which demand energy-efficient, flexible embedded systems capable of coping with the increasing diversification of IoT.

In contrast, FPSoC architectures, which include processors and FPGA fabric in the same chip, are a balanced solution to implement machine learning applications for IoT devices. The latest advancements in FPGA hardware allow a wide range of machine learning algorithms to be efficiently implemented. FPGAs are very suitable to perform deep neural network inference because of the parallel arrangement of neurons in layers and the type of mathematical functions they have to compute. This will be even more so in the future because of the trend to use simpler neuron activation functions (like ReLu) that, in addition to improve training, fit better in FPGA

resources. Moreover, the use of quantization techniques and custom data types (which is difficult to achieve, if possible at all, in devices with fixed architectures like ASICs and GPUs) can significantly reduce complexity and improve performance. In our opinion, the trends for neural network implementation in IoT devices in the following years can be summarized as follows:

- Training will rely on heavy-duty cloud-based GPUs. ASICs like the new Google´s TPU (optimized for both inference and training, with impressive performance) will have a piece of the pie here, but with the limitation posed by their lack of flexibility.
- The simplest IoT devices will use CPUs and ASICs for inference, to reduce cost and power consumption, respectively. Larger devices will use FPGAs/FPSoCs for inference because of their balanced flexibility and computer power. For heavy-duty inference, the same considerations as for training apply.

FPSoCs are an excellent alternative for Evolutionary Computing, because they allow the algorithm itself to be executed in software while the objective function can be computed in parallel in hardware for all individuals. However, their efficiency in this context greatly depends on whether or not the specific operations involved in the computation of the objective function fit available hardware resources. It can be concluded that, thanks to the availability of hard processors with floating point units, FPSoCs are very suitable for implementing evolutionary computing algorithms. In the case of particle swarm, it has been discussed how the same performance as a desktop computer can be achieved with FPSoCs with a fraction of the size, cost, and power consumption.

In our opinion, the implementation in FPSoCs of IoT devices with machine learning capabilities will be boosted by the availability of increasingly efficient high level synthesis tools based on widely known and used languages, such as OpenCL, C/C++, or Matlab, enabling software designers to take advantage of the excellent characteristics of FPSoC devices.

REFERENCES

[1] M. Jaffe, "IoT Won't Work Without Artificial Intelligence | WIRED." [Online]. Available: https://www.wired.com/insights/2014/11/iot-wont-work-without-artificial-intelligence/. [Accessed: 22-Mar-2018].

[2] E. Sappin, "How AI and IoT must work together | VentureBeat." [Online]. Available: https://venturebeat.com/2017/06/28/how-ai-and-iot-must-work-together/. [Accessed: 22-Mar-2018].

[3] E. Ahmed *et al.*, "The role of big data analytics in Internet of Things," *Comput. Networks*, vol. 129, pp. 459–471, Dec. 2017.

[4] B. Del Monte and R. Prodan, "A scalable GPU-enabled framework for training deep neural networks," *2016 2nd Int. Conf. Green High Perform. Comput. (ICGHPC 2016)*, 2016.

[5] M. D. Valdés Peña, J. J. Rodriguez-Andina, and M. Manic, "The Internet of Things: The Role of Reconfigurable Platforms," *IEEE Ind. Electron. Mag.*, vol. 11, no. 3, pp. 6–19, 2017.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," 2017.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[9] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data Mining for the Internet of Things: Literature Review and Challenges," *Int. J. Distrib. Sens. Networks*, vol. 11, no. 8, p. 431047, Aug. 2015.

[10] C. Perera, S. Member, A. Zaslavsky, and P. Christen, "Context Aware Computing for The Internet of Things : A Survey," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 1, pp. 1–41, 2014.

[11] J. Tang, D. Sun, S. Liu, and J. L. Gaudiot, "Enabling Deep Learning on IoT Devices," *Computer (Long. Beach. Calif).*, vol. 50, no. 10, pp. 92–96, 2017.

[12] K. Panetta, "Gartner's Top 10 Strategic Technology Trends for 2017 - Smarter With Gartner." [Online]. Available: https://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/. [Accessed: 14-Mar-2018].

[13] Xue-Wen Chen and Xiaotong Lin, "Big Data Deep Learning: Challenges and Perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[14] M. Hilbert, "Big Data for Development: A Review of Promises and Challenges," *Dev. Policy Rev.*, vol. 34, no. 1, pp. 135–174, Jan. 2016.

[15] H. Hu, Y. Wen, T. S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[16] A. Akbar, A. Khan, F. Carrez, and K. Moessner, "Predictive Analytics for Complex IoT Data Streams," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1571–1582, Oct. 2017.

[17] D. Nallaperuma, D. De Silva, D. Alahakoon, and X. Yu, "A cognitive data stream mining technique for context-aware IoT systems," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 4777–4782.

[18] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," Springer, Berlin, Heidelberg, 1999, pp. 304–307.

[19] I. Goodfellow, B. Yoshua, and A. Courville, *Deep Learning*. MIT Press, 2016.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[21] K. Xu *et al.*, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," in *International Conference on Machine Learning*, 2015.

[22] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating Videos to Natural Language Using Deep Recurrent Neural Networks," Dec. 2014.

[23] S. Wang and J. Jiang, "Learning Natural Language Inference with LSTM," Dec. 2015.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[25] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sep. 2014.

[26] J. Walker, "Smart City Artificial Intelligence Applications and Trends -." [Online]. Available: https://www.techemergence.com/smart-city-artificial-intelligence-applications-trends/. [Accessed: 22-Mar-2018].

[27] J. Chin, V. Callaghan, and I. Lam, "Understanding and personalising smart city services using machine learning,

The Internet-of-Things and Big Data," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 2050–2055.
[28] M. Paukova, "A smarter smart city - MIT Technology Review." [Online]. Available: https://www.technologyreview.com/s/610249/a-smarter-smart-city/. [Accessed: 22-Mar-2018].
[29] D. L. Marino, K. Amarasinghe, and M. Manic, "Building Energy Load Forecasting using Deep Neural Networks," Oct. 2016.
[30] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *IEEE International Symposium on Industrial Electronics*, 2017.
[31] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Networks*, vol. 6, pp. 91–99, 2016.
[32] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger, "Intelligent Buildings of the Future: Cyberaware, Deep Learning Powered, and Human Interacting," *IEEE Ind. Electron. Mag.*, vol. 10, no. 4, 2016.
[33] R. J. F. Rossetti, "Traffic Control & Management Systems in Smart Cities," *Readings on Smart Cities*, vol. 2, no. 3, 2016.
[34] M. Bojarski *et al.*, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car," Apr. 2017.
[35] C.-R. Yu, C.-L. Wu, C.-H. Lu, and L.-C. Fu, "Human Localization via Multi-Cameras and Floor Sensors in Smart Home," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, 2006, pp. 3822–3827.
[36] A. H. M. Amin, N. M. Ahmad, and A. M. M. Ali, "Decentralized face recognition scheme for distributed video surveillance in IoT-cloud infrastructure," *Proc. - 2016 IEEE Reg. 10 Symp. TENSYMP 2016*, pp. 119–124, 2016.
[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.
[38] J. H. Ko *et al.*, "Energy-efficient neural image processing for Internet-of-Things edge devices," *Midwest Symp. Circuits Syst.*, vol. 2017–Augus, pp. 1069–1072, 2017.
[39] H. Fang and C. Hu, "Recognizing human activity in smart home using deep learning algorithm," in *Proceedings of the 33rd Chinese Control Conference*, 2014, pp. 4716–4720.
[40] "What makes a Nest thermostat a Nest thermostat? | Nest." [Online]. Available: https://nest.com/thermostats/. [Accessed: 23-Mar-2018].
[41] R. Muradov, "'Alexa, Understand Me' - MIT Technology Review." [Online]. Available: https://www.technologyreview.com/s/608571/alexa-understand-me/. [Accessed: 23-Mar-2018].
[42] B. Marsh, "The intelligent pacemaker that can talk to your doctor | Daily Mail Online," *Daily Mail - UK*. [Online]. Available: http://www.dailymail.co.uk/health/article-122444/The-intelligent-pacemaker-talk-doctor.html. [Accessed: 02-Apr-2018].
[43] W. R. DASSEN, K. DULK, and H. J. WELLENS, "Modern Pacemakers: Implantable Artificial Intelligence?," *Pacing Clin. Electrophysiol.*, vol. 11, no. 11, pp. 2114–2120, Nov. 1988.
[44] J. Pacheco and S. Hariri, "IoT Security Framework for Smart Cyber Infrastructures," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 242–247.
[45] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15*, 2015, pp. 1–6.
[46] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.
[47] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," *Futur. Gener. Comput. Syst.*, Mar. 2018.
[48] J. Canedo and A. Skjellum, "Using machine learning to secure IoT systems," *2016 14th Annu. Conf. Privacy, Secur. Trust. PST 2016*, pp. 219–222, 2016.
[49] "Artificial Intelligence Architecture | NVIDIA Volta." [Online]. Available: https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/. [Accessed: 22-Mar-2018].
[50] Y. Pu *et al.*, "A 9-mm^2 Ultra-Low-Power Highly Integrated 28-nm CMOS SoC for Internet of Things," *IEEE J. Solid-State Circuits*, vol. 53, no. 3, pp. 1–13, 2018.
[51] C. Trigas, "Design challenges for system-in-package vs system-on-chip," pp. 663–666, 2003.
[52] J. J. Rodríguez Andina, E. de la Torre-Arnanz, and M. D. Valdes Peña, *FPGAs : Fundamentals, advanced features, and applications in industrial electronics*. .
[53] S. S. Iyer, "Heterogeneous Integration for Performance and Scaling," *IEEE Trans. Components, Packag. Manuf. Technol.*, vol. 6, no. 7, pp. 973–982, Jul. 2016.
[54] M. Sadri, C. Weis, N. Wehn, and L. Benini, "Energy and performance exploration of accelerator coherency port using Xilinx ZYNQ," in *Proceedings of the 10th FPGAworld Conference on - FPGAworld '13*, 2013, pp. 1–8.
[55] L. Costas, R. Fernandez-Molanes, J. J. Rodriguez-Andina, and J. Farina, "Characterization of FPGA-master

ARM communication delays in zynq devices," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, 2017, pp. 942–947.
[56] R. F. Molanes, J. J. Rodriguez-Andina, and J. Farina, "Performance Characterization and Design Guidelines for Efficient Processor–FPGA Communication in Cyclone V FPSoCs," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4368–4377, May 2018.
[57] B. Fort *et al.*, "Automating the Design of Processor/Accelerator Embedded Systems with LegUp High-Level Synthesis," in *2014 12th IEEE International Conference on Embedded and Ubiquitous Computing*, 2014, pp. 120–129.
[58] Altera Corporation, "Implementing FPGA Design with the OpenCL Standard," 2013.
[59] N. Cardoso *et al.*, "Multi-Camera Home Appliance Network: Handling device interoperability," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 69–74.
[60] Arm, "ARM Security Technology Building a Secure System using TrustZone Technology," 2009.
[61] Y. Liu, J. Briones, R. Zhou, and N. Magotra, "Study of secure boot with a FPGA-based IoT device," *Midwest Symp. Circuits Syst.*, vol. 2017–Augus, pp. 1053–1056, 2017.
[62] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 97–109, Jan. 2018.
[63] "TrustZone – Arm." [Online]. Available: https://www.arm.com/products/security-on-arm/trustzone. [Accessed: 28-Mar-2018].
[64] R. F. Molanes *et al.*, "Implementation of Particle Swarm Optimization in FPSoC devices," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1274–1279.
[65] S. Sridharan, P. Durante, C. Faerber, and N. Neufeld, "Accelerating particle identification for high-speed data-filtering using OpenCL on FPGAs and other architectures," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–7.
[66] D. Mahajan *et al.*, "TABLA: A unified template-based framework for accelerating statistical machine learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 14–26.
[67] K. Guo *et al.*, "Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.
[68] A. X. M. Chang and E. Culurciello, "Hardware accelerators for recurrent neural networks on FPGA," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.
[69] J. Lachmair, T. Mieth, R. Griessl, J. Hagemeyer, and M. Porrmann, "From CPU to FPGA - Acceleration of self-organizing maps for data mining," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017–May, pp. 4299–4308, 2017.
[70] W. Fang, Y. Zhang, B. Yu, and S. Liu, "FPGA-based ORB Feature Extraction for Real-Time Visual SLAM," Oct. 2017.
[71] T. Mekonnen *et al.*, "Energy Consumption Analysis of Edge Orchestrated Virtualized Wireless Multimedia Sensor Networks," *IEEE Access*, vol. 6, pp. 5090–5100, 2018.
[72] Xiaoyin Ma, W. A. Najjar, and A. K. Roy-Chowdhury, "Evaluation and Acceleration of High-Throughput Fixed-Point Object Detection on FPGAs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 1051–1062, Jun. 2015.
[73] M. Sit, R. Kazami, and H. Amano, "FPGA-based accelerator for losslessly quantized convolutional neural networks," in *2017 International Conference on Field Programmable Technology (ICFPT)*, 2017, pp. 295–298.
[74] H. Nakahara, A. Jinguji, T. Fujii, and S. Sato, "An acceleration of a random forest classification using Altera SDK for OpenCL," in *2016 International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 289–292.
[75] Altera, "Floating-Point IP Cores User Guide Subscribe Send Feedback," 2016.
[76] R. Domingo *et al.*, "High-level design using Intel FPGA OpenCL: A hyperspectral imaging spatial-spectral classifier," in *2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2017, pp. 1–8.
[77] R. Finker, J. Echanobe, I. del Campo, and K. Basterretxea, "Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons," *Electron. Lett.*, vol. 49, no. 25, pp. 1598–1600, Dec. 2013.
[78] S. Gomar, M. Mirhassani, and M. Ahmadi, "Precise digital implementations of hyperbolic tanh and sigmoid function," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, 2016, pp. 1586–1589.
[79] F. Ertam and G. Aydin, "Data classification with deep learning using Tensorflow," in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 755–758.
[80] S. Shin, K. Hwang, and W. Sung, "Fixed-Point Performance Analysis of Recurrent Neural Networks," in *IEEE Int conf Acoustics, Speech and Signal Processing 2016*, 2016, no. 1, pp. 976–980.

[81] M. Shah, J. Wang, D. Blaauw, D. Sylvester, H.-S. Kim, and C. Chakrabarti, “A fixed-point neural network for keyword detection on resource constrained hardware,” in *2015 IEEE Workshop on Signal Processing Systems (SiPS)*, 2015, pp. 1–6.
[82] X. Zhang *et al.*, “Machine Learning on FPGAs to Face the IoT Revolution,” pp. 894–901, 2017.
[83] R. Doshi, K.-W. Hung, L. Liang, and K.-H. Chiu, “Deep learning neural networks optimization using hardware cost penalty,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1954–1957.
[84] J. Qiu *et al.*, “Going Deeper with Embedded FPGA Platform for Convolutional Neural Network,” in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA ’16*, 2016, pp. 26–35.
[85] N. Sugimoto *et al.*, “Trax solver on Zynq with Deep Q-Network,” in *2015 International Conference on Field Programmable Technology (FPT)*, 2015, pp. 272–275.
[86] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, “DeepBurning: Automatic Generaion of FPGA-based Learning Accelerators for the Neural Network Family,” in *Design Automation Conference (DAC)*, 2016.
[87] Chia-Feng Juang, Chun-Ming Lu, Chiang Lo, and Chi-Yen Wang, “Ant Colony Optimization Algorithm for Fuzzy Controller Design and Its FPGA Implementation,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1453–1462, Mar. 2008.
[88] W. Wang, A. C.-F. Liu, H. S.-H. Chung, R. W.-H. Lau, J. Zhang, and A. W.-L. Lo, “Fault Diagnosis of Photovoltaic Panels Using Dynamic Current–Voltage Characteristics,” *IEEE Trans. Power Electron.*, vol. 31, no. 2, pp. 1588–1599, Feb. 2016.
[89] T. M. Chan, K. F. Man, K. S. Tang, and S. Kwong, “A Jumping Gene Paradigm for Evolutionary Multiobjective Optimization,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 143–159, Apr. 2008.
[90] N. N. Morsi, M. B. Abdelhalim, and K. A. Shehata, “Efficient hardware implementation of PSO-based object tracking system,” in *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, 2013, pp. 155–158.
[91] Shih-An Li, Ching-Chang Wong, Chia-Jun Yu, and Chen-Chien Hsu, “Hardware/software co-design for particle swarm optimization algorithm,” in *2010 IEEE International Conference on Systems, Man and Cybernetics*, 2010, pp. 3762–3767.
[92] R. Courtland, “Gordon Moore: The Man Whose Name Means Progress - IEEE Spectrum.” [Online]. Available: https://spectrum.ieee.org/computing/hardware/gordon-moore-the-man-whose-name-means-progress. [Accessed: 22-Mar-2018].

Figure 1: IoT devices (adapted from [3]).

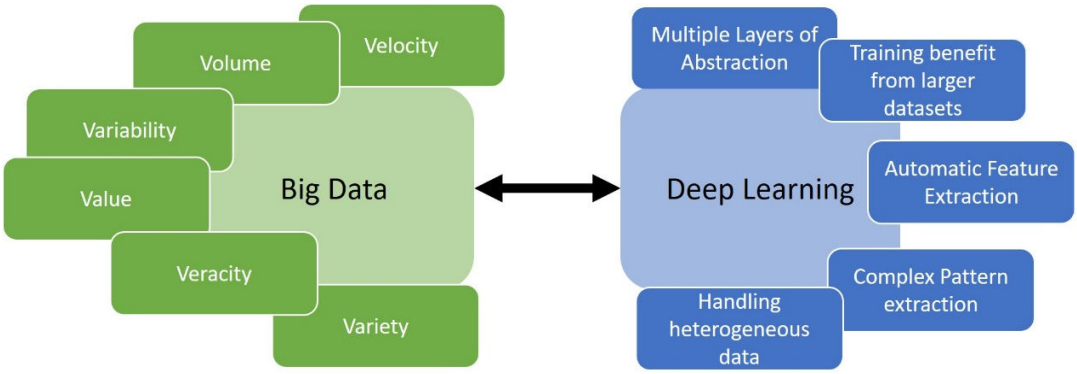

Figure 2: Big data 6 V's and connection with deep learning.

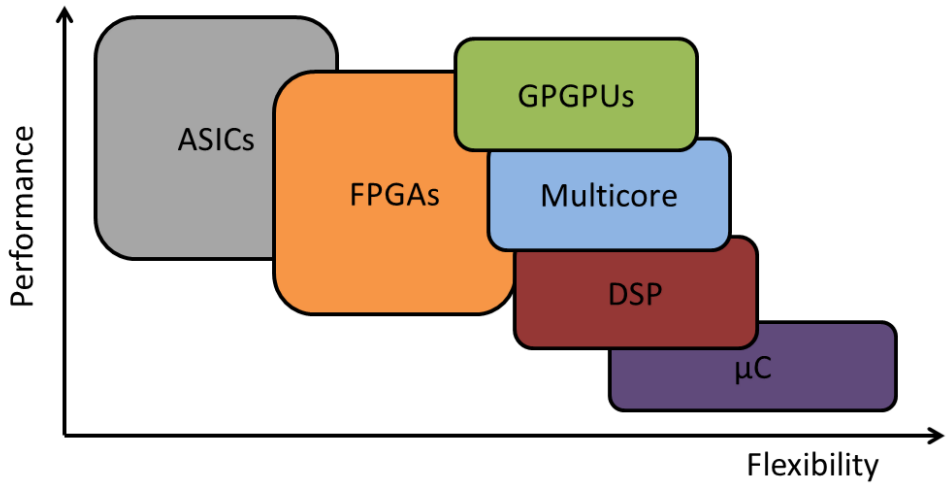

Figure 3: Performance vs Flexibility of digital processing platforms (adapted from [52]).

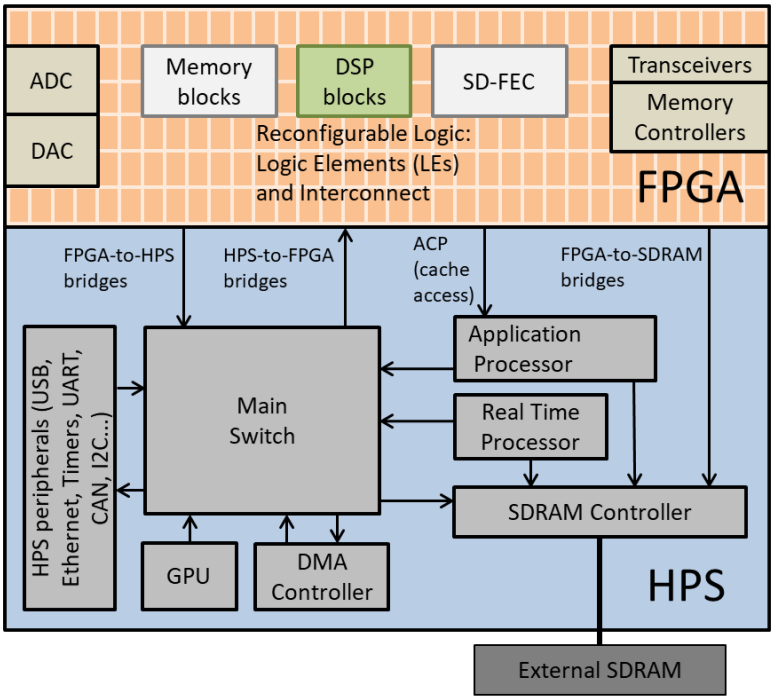

Figure 4: Block diagram of a modern FPSoC.

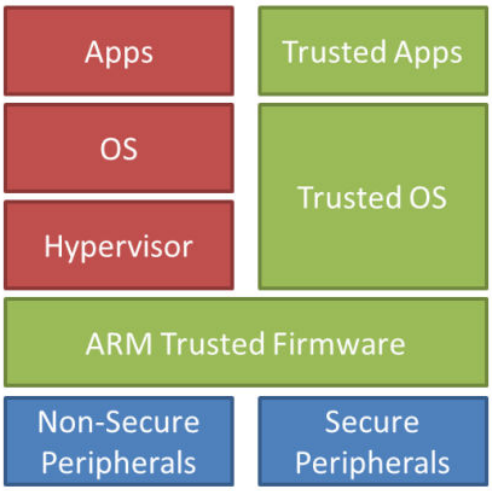

Figure 5: ARM Trustzone Security (adapted from [63]).

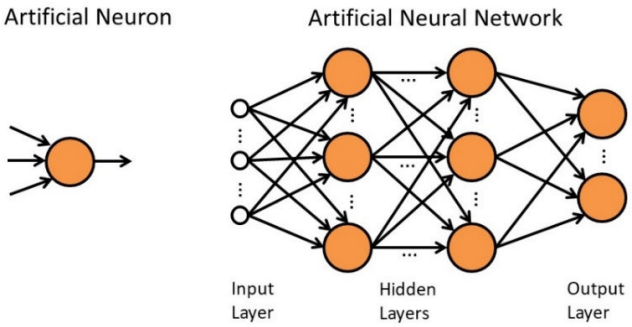

Figure 6: Graphical representation of a single neuron and an artificial neural network.

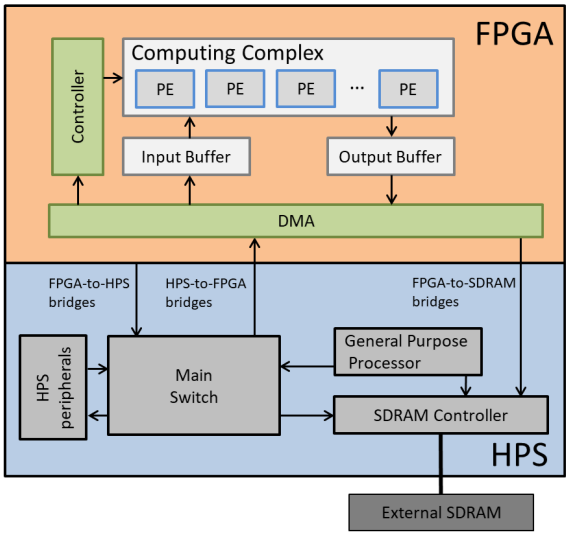

Figure 7: Implementation of a deep convolutional neural network on Zynq-7000.

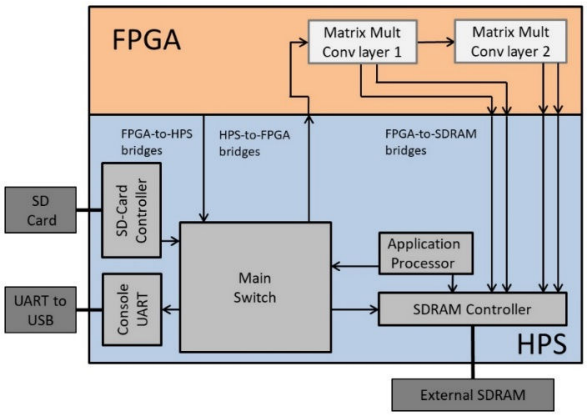

Figure 8: Implementation of a Deep-Q network on Zynq-7000.

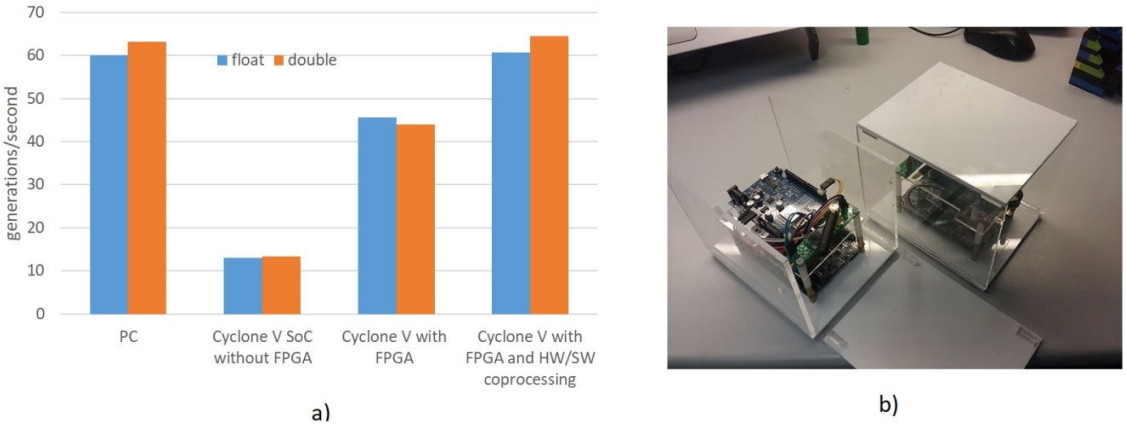

Figure 9: a) Performance comparison of particle swarm optimization algorithm for different Cyclone V SoC implementations and a desktop computer, b) The system based on a Cyclone V SoC board.

View publication stats