Adaptive and Energy-Efficient Architectures for Machine Learning: Challenges, Opportunities, and Research Roadmap

Muhammad Shafique*, Rehan Hafiz†, Muhammad Usama Javed†, Sarmad Abbas†,
Lukas Sekanina‡, Zdenek Vasicek‡, Vojtech Mrazek‡

*Computer Architecture and Robust, Energy-Efficient Technologies (CARE-Tech.) Group, Vienna University of Technology, Wien, Austria
†Vision Processing (VISpro) Lab, Information Technology University (ITU), Lahore, Pakistan
‡Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic

***Abstract*—** Gigantic rates of data production in the era of Big Data, Internet of Thing (IoT) / Internet of Everything (IoE), and Cyber Physical Systems (CSP) pose incessantly escalating demands for massive data processing, storage, and transmission while continuously interacting with the physical world under unpredictable, harsh, and energy-/power-constrained scenarios. Therefore, such systems need to support not only the high performance capabilities at tight power/energy envelop, but also need to be intelligent/cognitive, self-learning, and robust. As a result, a hype in the artificial intelligence research (e.g., deep learning and other machine learning techniques) has surfaced in numerous communities. This paper discusses the challenges and opportunities for building energy-efficient and adaptive architectures for machine learning. In particular, we focus on brain-inspired emerging computing paradigms, such as approximate computing; that can further reduce the energy requirements of the system. First, we guide through an approximate computing based methodology for development of energy-efficient accelerators, specifically for convolutional Deep Neural Networks (DNNs). We show that in-depth analysis of datapaths of a DNN allows better selection of Approximate Computing modules for energy-efficient accelerators. Further, we show that a multi-objective evolutionary algorithm can be used to develop an adaptive machine learning system in hardware. At the end, we summarize the challenges and the associated research roadmap that can aid in developing energy-efficient and adaptable hardware accelerators for machine learning.

***Keywords*—**machine learning, approximate computing, deep learning, neural networks, energy efficiency, performance, low power, accelerators, architecture, memory, FPGA, CGRA, adaptive, roadmap.

I. INTRODUCTION

In recent years, Artificial Intelligence (AI) has achieved great applicability due to the advancements in its sub-domains of Machine Learning (ML), Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNN) and Deep Neural Networks (DNNs). Neural networks based ML algorithms give computers the ability to find a solution by learning through a training data. Starting from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2012), techniques based upon DNNs and specifically deep CNNs have witnessed groundbreaking success in various big data analytics and computer vision tasks such as: object classification and recognition, autonomous driving and handwritten digit recognition [1]-[3]. However, CNN operations are characterized by extensive convolution operations, enormous memory traffic, and storage requirements [5]. Deep CNNs are variants of DNNs that have further hidden layers that comprise of multiple layers of convolution, pooling (selection) and non-linear activation functions [3]. The weights that are employed in the convolution process are computed based upon a training phase that typically involves a back propagation based scheme to minimize the classification error [5]. GPUs are amenable to such machine learning kernels when compared against CPUs due to the availability of plenty of simple cores, the support of massive thread-level parallelism, and ease of parallel programmability. However, their computational superiority is coupled with a serious compromise: the enormous power consumption. Once such example is NVIDIA DGX-1 GPU that can train VGG-D CNN 58 times faster as compared to a CPU server with two Intel Xeon E5-2699v4 CPUs. However, as compared to commodity GPUs (typically 100 watts), DGX-1 can consume upto 32 times more power [4]. Therefore, besides for the energy-constrained embedded devices, it is also important to provide extreme energy efficiency in high-end DNN implementations and training to lower cooling costs, thus GPUs may potentially become a less-attractive option as more specialized, energy-efficient and configurable architectures evolve that support high degree of parallelism and ease of programmability.

With the development of mobile computing, wearable (healthcare) devices, Internet of Things (IoT) and their applications, there is an urgent need to provide capabilities for smart operations and artificial intelligence (or at least a part of it) in systems with limited energy and resources. *Thus, GPUs, that may be favored for compute-intensive training for time being, are still not suitable for inference on energy-constrained embedded and IoT devices that are typically battery operated and/or survive on harvested energy, and often have power/energy budget in sub-Watt range.* Therefore, the design of highly energy-efficient hardware accelerators for compute-intensive CNNs (or their kernels) has received a lot of interest [5]. Using FPGAs, ASICs, and other specialized compute fabrics like Application-Specific Instruction Set Processors (ASIPs) and Coarse-Grained Reconfigurable Processors (CGRAs) for CNN acceleration lowers the energy/power consumption while providing high throughput, and thereby reducing the overall cost of the operation. Power/energy consumption can further be decreased by using (brain-inspired) emerging computing paradigms such as approximate computing blocks [6] and neuromorphic computing [8].

Approximate Computing (aka Inexact Computing) relies on relaxing the bounds of precise computing to provide new opportunities for improving the area, power/energy, and performance efficiency of systems by orders of magnitude (when considering both memory and computations) at the cost of reduced output quality (typically within user-tolerable range) [6]. *Thus, there is a need to apply emerging computing paradigms (such as approximate computing)* ***systematically*** *to realize specialized hardware accelerators for machine learning with extreme energy efficiency.*

Our Contributions: This paper provides various insights for systematic development of energy-efficient accelerators for Machine Learning, specifically in the context of DNNs, where a high degree of adaptivity can be realized by employing quality-/energy-configurable approximate modules inside the accelerator datapath. Towards the end, we discuss various challenges and outline a roadmap for research in adaptive and energy-efficient architectures for machine learning.

Paper Organization: We provide a brief review of existing work, which can be related to different phases of our systematic approach. In Section III, we illustrate that in-depth analysis of datapaths of a DNN allows *better selection* of Approximate Computing modules for developing energy-efficient accelerators. Section IV provides an overview of an Evolutionary Algorithm based low power and adaptive architecture for Machine Learning. Section V outlines open research challenges and discusses the roadmap for further research in this area to enable extreme energy efficiency. Without loss of generality, throughout

978-1-5090-6762-6/17/$31.00 © 2017 IEEE

this paper, we make use of the MNIST dataset [36], which is a dataset of handwritten English characters.

II. Energy-Efficient Adaptive Hardware Accelerators for Neural Networks

Recent advancements in low-power architectures for DNNs apply optimization at various stages of accelerator development [7]. To further reduce the power/energy consumption, implementations of DNNs can significantly benefit from approximate computing based low-energy computing modules, as the applications of DNNs are inherently error resilient. This resilience can be attributed to the following reasons:

1. In DNN architecture, all input vectors are typically processed in the same way despite the fact that they vary greatly in their inherent difficulty for a particular classification application. There is a potential in identifying neurons whose contribution to the expected quality is low and hence simplifying their implementation (or even removing them) to reduce power consumption without affecting the quality of processing [8].
2. Training of DNN is an iterative process, which can be stopped when good enough results are obtained. Moreover, retraining can even mitigate the effects of approximations and lead to better results than with the accurate implementation [8].

A systematic application of such techniques can thus result in significant energy reduction. *Fig.1 provides our systematic methodology for realizing energy-efficient accelerators*, specifically for CNN and DNN based AI systems, which works in the following steps.

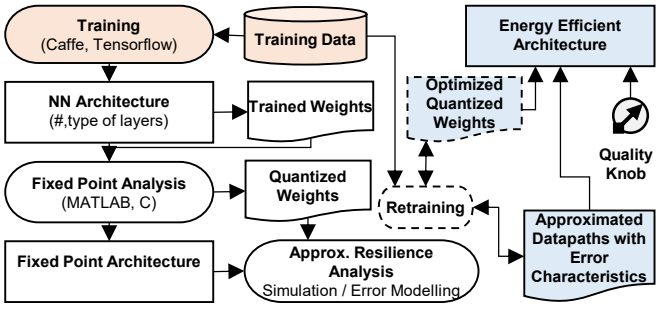

Fig. 1. Our methodology for designing energy-effcient and adaptive neural network accelerator-based architectures for Machine Learning and AI systems.

1) Libraries such as *TensorFlow* and *Caffe* are typically employed to customize pre-trained networks for a particular application [5]. The pre-trained synaptic weights, computed as a result, are then further tuned as per the application requirement. These trained weights are applied via convolutional kernels and hence "multiplication" constitutes the bulk of computational load for CNNs/DNNs.
2) The next step is to perform a fixed point analysis in order to select an appropriate fixed point format ($Q_{m.n}$) that keeps minimum number of bits for integer (*m* bits) and fraction (*n* bits), while keeping a minimum accuracy level.
3) The quantized weights and the architectures are than evaluated to explore their error resilience properties. This step can be performed either via *Monte-Carlo* based simulations where a certain randomly selected training data is used to evaluate the suitability of each datapaths for approximation or via analytical modeling of approximate modules [43]-[45].
4) Since, machine learning based systems can be employed in a variety of settings with application-specific requirements such as high accuracy and/or low power the underlying hardware accelerators should provide (re-)configurability. *Quality knobs* that give control over the accuracy-power tradeoff and allow selection of appropriate approximate hardware modules as per the requirements are thus desirable for such adaptive hardware accelerators.
5) The approximated datapaths may also be retrained for updating the quantized weights to further improve the classification accuracy.

In the following, we discuss works related to above steps.

A. Fixed Point Analysis

Courbariaux et al. [15] used floating-point, fixed-point and dynamic fixed-point formats to train neural networks and demonstrated that fixed-point weights are sufficient for training. Gupta et al. [9] presented a hardware based DNN accelerator for image classification in which they studied the impact of different fixed-point rounding schemes on the accuracy. Judd et al. [10] demonstrated that minimum precision requirement not only varies across different networks, but also across different layers of the same network. Lin et al. [11] identified the optimal data precision for all layers of a network based on fixed-point quantization methodology. Gysel et al. [12] presented a framework, *Ristretto*, for fixed-point quantization and re-training of CNNs based on *Caffe*. Lai et al. [13] proposed an approach based on floating-point representation for CNN weights and fixed-point for activations. They showed that the proposed approach can reduce storage requirements by up to 36% and multiplier energy by up to 50%. In [9], authors used a stochastic rounding scheme and demonstrated network training with 16-bit fixed-point weights. Hashemi et al. [14] presented the DNN based quantized hardware accelerator on *Ristretto*, in which they analyzed the effects of varying quantization bits on accuracy, area and energy for hardware based designs.

B. Approximation Resiliance Analysis

Use of approximate arithmetic modules has been vastly explored to develop energy-efficient and high performance hardware accelerators [16]-[18], [46]-[48]. However, this comes at the cost of loss of classification accuracy and hence requires careful analysis to select a particular approximate arithmetic configuration. In [16] authors evaluated the use of low-power imprecise multipliers using the UCI Machine Learning repository. They reported that inexactness could be overcome by retraining the network. They reported an energy savings of up to 62.49% over accurate face recognition. In [19], the authors proposed a Lower-Part-OR Adder, which computes least significant bits of an addition using approximate logic, using OR gates instead of accurate adders, along with a Broken Array approximate multiplier which omits cells of Carry Save Adder array circuit. The approach similar to [16] was followed in [20], where a multiplier-less neuron structure was presented. Furthermore, re-training of the network was carried out to compensate for the inaccuracies introduced due to approximations. Multiplication is implemented by shifting pre-computed alphabets, and then adding them. As pre-computed alphabets introduce area overhead, approximation in introduced by not calculating all the alphabets. To alleviate the accuracy loss incurred due to approximation, constrained re-training of neural network is done to get rid of undesired combinations. They reported an accuracy loss of 2.83% for 8-bit and ~0.25% for 12-bit implementations for handwritten digit recognition on MNIST dataset [36]. It was further shown that by incorporating a selective rule to apply approximation only to multiplication where both operands are non-zero, the power consumption can be significantly reduced for a small decrease in classification accuracy (Fig. 2) [21].

Memory operations are responsible for a considerable portion of energy consumption. In order to reduce this energy, one can optimize memory cells, memory hierarchy, memory access patterns, data storage, refresh rates of DRAMs, and other properties of the memory subsystem [25]. For example, a hybrid 8T-6T SRAM cell was proposed to store the synaptic weights, wherein the sensitive MSBs are stored in 8T bit-cells and resilient LSBs are stored in 6T bit-cells [22]. If voltage over-scaling is applied, errors can appear in 6T cells, while 8T cells remain operating correctly. In another approach, a memory compression scheme exploiting the error resilience of DNN was introduced with the aim of reducing overall memory traffic and memory energy [18]. Finally,

several non-conventional technologies were proposed such as spintronic neural computing platforms in which the neural and synaptic functionalities are approximated by the underlying device physics. It was demonstrated that such systems can achieve competitive classification accuracy in a large number of complex recognition problems. Such architectures achieve over 15x improvement in energy compared to well-optimized CMOS designs [23].

In the following, we provide a case study of approximate resilience analysis that selects a datapath within a particular DNN that is more suitable for approximation.

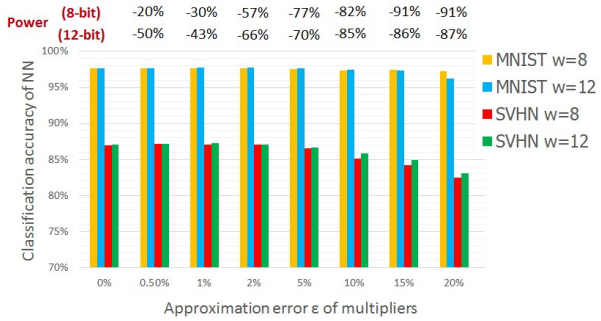

Fig. 2. Classification accuracy of NN with approximate multipliers for MNIST and SVHN datasets and 8/12 bit precision [21].

III. Error Resiliance Analysis: DNN-Specific Approximations for Low-Power Accelerators

We consider a design case where a reduced power/energy budget necessitates the application of low-power approximations. The problem then reduces to the selection of appropriate datapath of the DNN architecture where approximations can be employed (at application and/or hardware levels). We propose that *since the filters in a trained DNN architecture are either low-pass, high-pass or band-pass, their outputs shall have defined data distributions. Since, approximate arithmetic modules behave differently to different data distributions, a careful analysis of data distribution across various data-paths of neural network can aid in selecting an approximate circuit that provides minimum loss to classification accuracy. Data-unaware approximations may severely turn the benefits into loss, and therefore require additional training, or other kind of overhead, etc. as faced by existing approaches.*

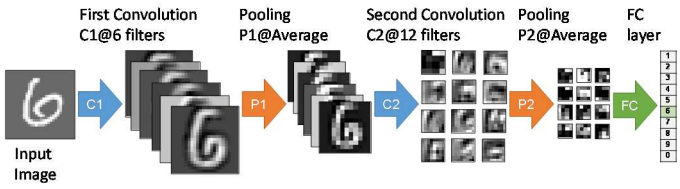

Fig. 3. Modified LeNet for Handwritten Digit Classification (MNIST dataset) used in our experiments. Figure illustrates the outputs of various layers.

For our analysis, we consider a deep CNN architecture based on the *LeNet-5* [2] as shown in Fig. 3 for classifying the MNIST dataset. Input image size is 28×28. The network has two convolution layers (C_1 and C_2), two pooling layers (P_1 and P_2), and finally, a Fully Connected layer (FC). C_1 has six different kernels, and produce six different output Feature Maps (FM_1, FM_2, …, FM_6) for a single input image. Then, P_1 applies average pooling for stride of one, and neighborhood of 2×2. Thus, the size of a feature map (FM) is reduced to half after passing through a pooling layer. Afterwards, C_2, which has 12 collections of six kernels each, operates on these six feature maps. In each collection, six feature maps from P_1 are convolved with six kernels, and resultant six feature maps are added together, to generate a final FM per collection. Finally, these twelve output feature maps are concatenated together in a vector, and the vector is passed onto FC layer. Output vector at the end of FC layer contains a prediction about the digit classified. Fixed point simulations with Q (6.7) bit format (6 integer and 7 fractional bits); provided a classification accuracy of 94%, when tested on 200 images.

As an illustrative case, *we analyze the data distribution at the output of 6 filters applied during C_1*. The analysis will then be employed to select an appropriate low-power approximate adder during the pooling layer P_1. Fig. 4 shows the Feature Maps at the output of six filters of C_1, for four randomly selected input images.

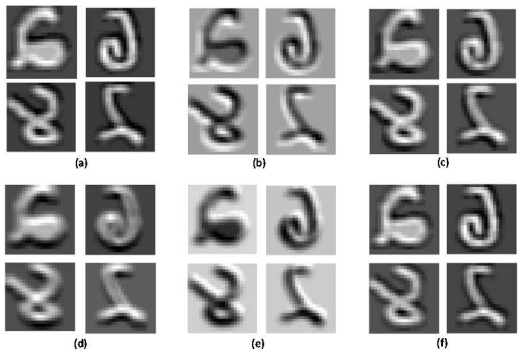

Fig. 4. (a) to (f) show the Feature Maps (FMs) at the output of six filters of C1, obtained for 4 randomly selected input images.

Fig. 5 provides the corresponding data distribution for the same feature maps. It can be observed that all the histograms within each part of Fig. 5 have strong correlation. Thus, *regardless of the input image, the data distribution of the output FM of a particular filter is identical*.

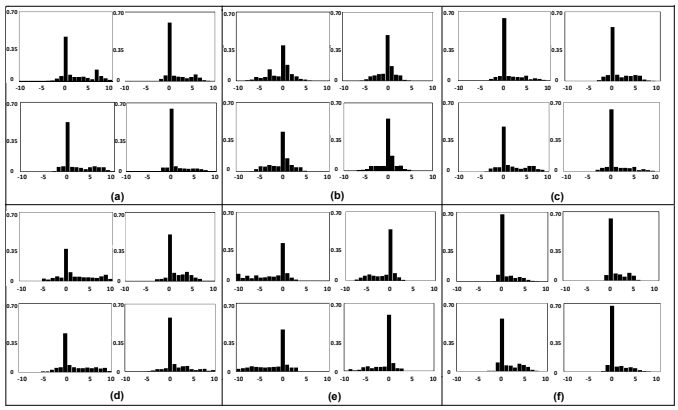

Fig. 5. Histograms within (a) to (f) provide the data distribution of the corresponding Feature Maps illustrated in Fig. 4.

Next, we evaluate the performance of five approximate adder configurations (α_1, α_2, α_3, α_4 and α_5), selected from the low-power IMPACT [17] adder configurations, for our Q *6.7* format fixed-point number. The IMPACT adders provide five imprecise alternate circuits (LP_1 to LP_5) for Full Adder. Thus, α_1 = LP_1:6, means that during approximate addition the 6 least significant bits (from Q6.7) shall be of the type LP_1. Similarly, α_2, α_3, α_4, α_5, were chosen to be LP_2:6, LP_3:6, LP_4:7 and LP_5:7, respectively. Afterwards, we populated the data for each of the six filter using 50 images selected randomly from the training set, and used it to evaluate the performance of each of these 5 adders for the six Feature Maps. As evident from Table 1, approximate adders have different tolerance for different data distributions. It can be observed that datapath that belongs to FM_2, FM_5 and FM_6 provide better resilience to approximate adders (when applied at the P_1 layer) as compared to FM_1, FM_3 and FM_4. *This is because regardless of the adder type, they provide a classification accuracy of at least 92%.* This can be attributed to the fact that the input data distribution resulted into adder operands that were less sensitive to error due to approximation. Based upon this analysis, we define a function **R(α)**, which ranks the approximation resilience of a particular feature map for each adder type. Thus, the FM that experiences

the lowest error for a particular adder ranks first in terms of error resilience. Thus, for the simulated data, **R(α_2)** = {FM_6, FM_5, FM_2, FM_4, FM_3, FM_1}. Accordingly, if there is a power constraint requirement to apply α_2 approximation to the P_2 layer of any one of the FMs, FM_1 shall be the least suitable. The table also highlights that in general α_1, α_4 and α_5 provide results with higher accuracy as compared to other adder variants for all the FMs. This is because, according to their truth tables [17], these adders produce outputs that are equivalent to an accurate adder in case both inputs are zero. As evident from the data distributions in Fig.5, these FMs contain a significant number of zero-valued or closer to zero data and hence, α_2 and α_3 are bound to generate erroneous outputs.

TABLE I. CLASSIFICATION ACCURACY OBTAINED BY EMPLOYING APPROXIMATE ADDERS (α_1, TO α_5) IN THE P_1 LAYER FOR VARIOUS DATAPATHS.

Adder Configuration	Approximation applied to individual Feature Maps						
	FM_1	FM_2	FM_3	FM_4	FM_5	FM_6	Average
α_1	92%	92%	92%	92%	90%	92%	92.00%
α_2	56%	92%	60%	84%	92%	94%	80.00%
α_3	56%	92%	60%	88%	93%	94%	80.67%
α_4	92%	92%	92%	92%	91%	92%	92.00%
α_5	94%	92%	92%	94%	92%	94%	93.34%
Average	78%	92%	79%	90%	92%	93%	87.60%

To evaluate our hypothesis, we consider a power constraint where it is required to apply approximation to at least two datapaths out of the total six. Table-1, in general, predicts highest classification accuracy to be achieved when datapaths comprising FM_5 and FM_6 are approximated. We thus compared the classification accuracy achieved when datapaths comprising FM_5 and FM_6 are approximated (i.e., the *best case*) as compared to the case when that of FM_1 and FM_3 are approximated (i.e., the *worst case*). We populated the results of Fig.6 by testing on 50 images, from the testing dataset. Fig.6 illustrates that, for $\alpha1$, $\alpha4$ and $\alpha5$, both the cases show comparable performance. However, for $\alpha2$ and $\alpha3$, *approximation of FM_1 and FM_3 leads to substantial accuracy loss in terms of classification accuracy*. This is in agreement with the predictions of Table-1. Thus, we demonstrate that due to the fixed architectural design of DNN, each datapath has a particular data distribution. *Since, error of approximate modules is also function of data distribution, prior analysis on training data can thus provide useful insight and thus guide the process of employing approximations.*

As the number of approximate configurations and number of layers increase, manual analysis can become non-trivial mandating the need for automated analysis and design tools. In the next section, we provide an Evolutionary Algorithm (EA)-guided approximation selection strategy to *automatically* develop adaptive and power/energy-efficient accelerators for machine learning.

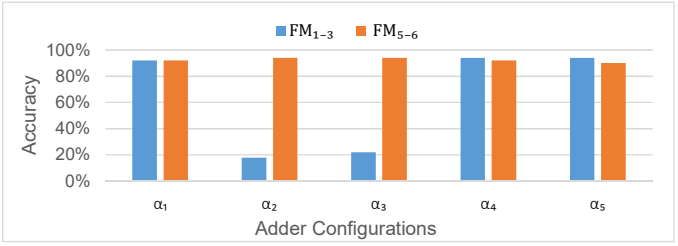

Fig. 6. Classification accuracy for two cases

IV. AN EA-BASED LOW-ENERGY ARCHITECTURE FOR MACHINE LEARNING

In this section, we will focus on how multi-objective evolutionary algorithms (EAs) can contribute to creating, optimizing and adapting machine learning systems in hardware. In Cartesian Genetic Programming (CGP), an n_i-input and n_o-output combinational circuit is modeled using an array of n_c x n_r programmable nodes forming a Cartesian grid (Fig. 7). The set of available node functions (G) contains n_a-input functions. No feedback is allowed in the basic version of CGP. The primary inputs and programmable nodes are uniquely numbered (from 0 to 8 in Fig. 7). CGP evolves a population (a set) of candidate circuits that are represented using the so-called chromosomes in the form of strings of integers. For each node the chromosome contains (n_a+1) values that represent (i) the node logic function and (ii) n_a addresses specifying the input connections. The chromosome also contains n_o values specifying the nodes connected to the primary outputs. The chromosome size is n_c x n_r (n_a + 1) + n_o integers. The chromosome can be understood as a simplified netlist.

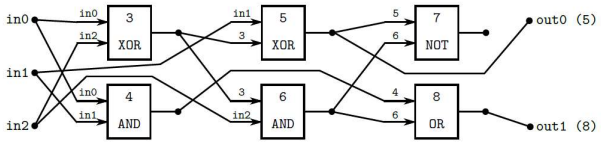

Fig. 7. Full adder represented by CGP with parameters: n_i = 3; n_o = 2; n_c = 3; n_r = 2; n_a = 2; G = {AND^0; OR^1; XOR^2; NOT^3}. Chromosome: (0, 2, 2) (0, 1, 0) (1, 3, 2)(3, 2, 0) (5, 6, 3) (4, 6, 1) (5, 8). Node functions are underlined.

The search is usually performed using a simple (1 + λ) evolutionary algorithm, where λ is between 1 and 20. In this algorithm, every new population consists of the best individual of the previous population and its λ offspring created using a mutation operator which modifies up to *h* genes (integers) of the chromosome.

1) *Evolutionary Circuit Approximation*

The approximate circuit design problem can be formulated as a multi-objective optimization problem in which the error, area, delay (or performance) and power consumption are conflicting design objectives. Designers expect obtaining a set of solutions which exhibit various trade-offs among key circuit parameters. The optimal set of such solutions is the so-called Pareto optimal set [24]. Available tools (such as [26]) typically approximate the Pareto front by multiple executions of approximation engines which are initialized using different parameters. The reasons for using an advanced evolutionary approach (contrasted to a simple greedy search or ad hoc heuristics) is that the population-based approach suits well in finding multiple solutions and its niche-preservation methods can be exploited to discover diverse solutions. We deal with evolutionary circuit approximation conducted with "gate-level" CGP as most evolutionary approaches developed in this area utilize CGP. However, EAs can be applied to approximate RTL or even behavioral descriptions. For example, ABACUS [40] tool creates an abstract syntax tree from the input behavioral RTL description of a circuit, and then applies variant operators to the AST to create acceptable approximate designs that are implemented using a standard design tool. The search is performed by means of a multi-objective genetic algorithm NSGA-II. ABACUS was evaluated in three different domains - machine learning, signal processing, and computer vision - with power savings reaching up to 40% [27]. In another application, approximate implementations of complex median networks that are useful in convolutional kernels and signal processing were evolved [28].

2) *Evaluation of Candidate Approximate Circuits*

CGP can naturally be extended for circuit approximation because the fitness function always includes a component measuring the circuit functionality. If the circuit under approximation is not complex, it is possible to evaluate its responses for all possible input combinations and compute its exact error according to an arbitrary chosen error metric (e.g., as implemented in [29]).

In the case of more complex circuits, candidate circuits are evaluated using a training set (i.e., a subset of all possible vectors) and the resulting error is thus only estimated (e.g., [26], [27]). This is a common practice used in the literature. If the exact error value is requested, more advanced approaches such as formal relaxed equivalence checking techniques

have to be employed. Depending on the particular error metric (e.g., the mean error or the worst-case error), the error calculation problem is transformed to a decision problem and solved by means of SAT solving or binary decision diagrams (BDD). However, approximate arithmetic circuits no more complex than 16-bit adders and 8-bit multipliers have been reported so far with the known (exact) error [30], [27], [31]. For example, it is unknown how to effectively compute the (exact) mean error for complex arithmetic circuits. Table III compares the evaluation time when optimized circuit simulator and BDD-based method are used in the case of 16-bit adder. Other parameters (delay, power, area) of candidate circuits are only estimated in the process of evolution. At the end of evolution, selected resulting circuits (especially those forming the Pareto front) are implemented for a given technology (or FPGA) using common circuit design tools to determine their final electrical parameters.

TABLE II. THE AVERAGE TIME NEEDED TO PERFORM THE WORST-CASE (ε_{MAX}) AND THE AVERAGE-CASE ERROR (ε_{AVG}) ANALYSIS FOR W-BIT ADDERS WITH SIMULATION AND BDD

bit-width	Inputs	optimized simulation	BDD-based method		speedup	
		ε_{max} + ε_{avg}	ε_{max}	ε_{avg}	ε_{max}	ε_{avg}
4-bit	8	4.5 us	10.3 us	14.0 us	0.43 ×	0.32 ×
8-bit	16	1.9 ms	3.5 ms	4.6 ms	0.54 ×	0.42 ×
12-bit	24	682.4 ms	127.9 ms	312.7 ms	5.33 ×	2.18 ×
16-bit	32	140.9 s	1.38 s	2.93 s	102.3 ×	48.09 ×

The EA-based approximation methods can be classified as, Error-oriented, multi-objective and evolutionary (Fig.8). Error-oriented EA tries to evolve a circuit showing a predefined error, and optimize circuit parameters without worsening this error [32]. Resources-oriented is one in which resources (e.g., the number of gates) are constrained and EA is used to minimize the error with available resources [36]. And finally the Multi-objective, in which all criteria are optimized together using a multi-objective EA such as NSGA-II [27],[29].

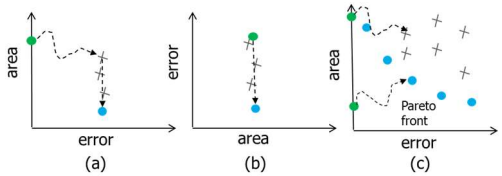

Fig. 8. Error-oriented (a), area-ariented (b) and multiobjective (c) evolutionary circuit approximation. Accurate circuits (green points) are evolved to resulting approximate circuits (blue points).

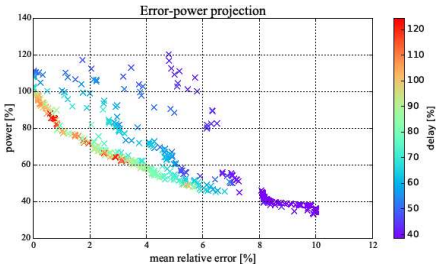

Fig. 9. Pareto front showing parameters of 8-bit approximate multipliers, where Ripple-Carry Array Multiplier represents 100%.

3) *Evaluating Automatically Generating Approximate Modules*

We provide a rich and well-focused library of approximate modules [29] (called *EvoApprox8b*) that can be immediately used for benchmarking of circuit approximation methods. The library consists of approximate 8-bit adders and 8-bit multipliers. In addition to circuit parameters (error, area, delay, power etc.) and 7 error metrics, the library contains Matlab, C and Verilog implementations for all components which allows the user not only to immediately use the components, but also to calculate the error for a new target error. There are 430 different approximate 8-bit adders and 471 different approximate 8-bit multipliers forming a Pareto front in a three-dimensional space defined using the mean relative error, delay, and power metrics (non-dominated multipliers are shown in Fig. 9). These implementations were obtained by a multi-objective CGP. As a starting point for the CGP-based approximation process, 13 different adders and 6 different multiplier conventional accurately-operating architectures were employed.

V. CHALLENGES, PERSPECTIVES AND RESEARCH ROADMAP FOR EFFICIENT MACHINE LEARNING ARCHTIECTURES

In the following, we highlight selected but important research challenges, and associated research opportunities and perspectives that will help the community in realizing highly adaptive and energy-efficient architectures for machine learning.

Optimize for Memory or Computations First: A Million-Dollar Question? ML is heavily deployed in data-intensive applications, and memory footprint of the DNNs is significant and further escalating. Some studies have shown that the memory energy contributions can even grow up to 50%-80% of the total energy. Therefore, reducing the memory energy should be one of the top-most priorities when designing specialized architectures for ML with extreme energy efficiency. It should be done throughout the memory hierarchy considering all the feasible/beneficial knobs and techniques like (weight) compression, efficient data storage, refresh rate control, optimizing memory cells and access patterns, devising fine-grained memory power management policies, and using non-volatile memories. Going one step ahead, deployment of emerging technologies like nano-wires based 3D-stacked architectures could provide a very attractive solution, where the sea of accelerators can be directly connected to the dense memory layers through highly energy-efficient and high bandwidth nano-wires. An example project in this direction is *N3XT1000x* [41]. Spintronic devices (e.g., see the works at the *C-Spin* center of excellence [39]), Memristors, and Graphene based design are other parallel emerging technologies. On the computing side, there is a huge potential of architectures based on sea of accelerators, which can highly specialized and context-adaptive, with the support of enhanced power-management features.

Accuracy vs. Energy Tradeoff: New computing paradigms such as Approximate/Stochastic and Neuromorphic computing add an additional dimension to the power/energy, area, performance, and accuracy design space, and thereby enabling design and run-time tradeoffs. Particularly, ML architectures may require different precision/accuracy requirement for different set of applications. Thus, an efficient hardware accelerator for ML should be able to adaptively control the accuracy of the accelerator such as to minimize the energy consumption without violating the user-tolerable quality constraints, or alternatively maximize the quality under a given energy budget. However, to achieve this, there is a need to carefully analyze the effects of approximations on the DNN architecture design and its efficiency. This requires development of efficient simulation frameworks, evaluation methodologies, as well as analytical error analysis techniques. Due to the layered nature of the ML architectures, it may be advantageous to distribute the power/energy budget across multiple layers in a stochastic manner while accounting for the sensitivity of each layer and/or sensitivity of different neurons inside layers. Furthermore, since analytical modeling becomes non-trivial for large-sized DNN architectures, a divide-and-conquer methodology may help in achieving a (near-)accurate error analysis in significantly less amount of time.

Adaptability, (Re-)Configurability, Flexibility, and Scalability: DNN architectures are characterized by repetitive arrangements of standard layers, e.g., convolution, pooling and activation functions. However, developing a generic DNN accelerator shall require a highly adaptive, configurable, flexible and scalable system since the order of layers, synaptic weights of convolutional layers and dimensions of input data vary across different applications. This can further lead to interesting design decisions, such as, time-shared vs. dedicated designs and choice of scratchpad vs. caches, activation control of different layers

while keeping the configuration overhead minimal or doing it in a transparent fashion without exposing the overhead to the user. Towards this end, selection of FPGA or CGRAs as the basic compute fabric is also an interesting research question, as both have their pros and cons in terms of reconfiguration granularity and associate overhead. Finally, the data bandwidth requirements also scale with input data dimensions. Efficient data storage, synaptic weight compression, and multiplier-less designs can allow for energy-efficient architectures.

Run-Time Evolutionary Algorithms for Designing and Optimizing DNN Architectures: General-purpose optimization algorithms such as EAs can also be used to optimize and approximate particular layers of DNN, the whole architecture, memory access patterns, or the training procedure, possibly in a holistic scenario in which errors introduced in some part of NN can be compensated by a suitable construction of other parts of DNN. In addition to the offline evolutionary design, there is a great potential in the online adaptation of components of DNNs with the goal of providing the best trade-off between energy/power consumption and error of processing anytime when deployed.

Correct Benchmarking with Fairness and High Fidelity: Circuit approximation methods and actual approximate circuits obtained by means of these methods have to be evaluated by comparing them with existing solutions. However, due to a rapid development of the field, a fair evaluation methodology has not been established yet, especially across different communities working on these challenging problems. Also, metrics to estimate the fidelity of results, standard test conditions, etc. are also required for fair validation. In order to compare circuit approximation methods, it would be requested for each method and each benchmark circuit to provide a Pareto-front containing the best tradeoffs achieved for key circuit parameters such as the error, area, delay and power/energy consumption (in a given fabrication technology), assuming that a fixed time budget is available for the approximation procedure. Several papers provided detailed comparison of different approximation approaches, for example, in the case of adders and multipliers [33], [34]. However, if the models of appropriate circuits, experimental platforms and test conditions are not available online, it is not only hard to reproduce their results, but also near-to-impossible to achieve fair comparisons for the newly proposed methods and solutions.

Open-Source Contributions: Besides the above point, also, as a community, to move faster and to address challenges in this rapidly growing field effectively, there is a strong need for researchers and developers to open-source their contributions along with their test configurations and settings. This way, other researchers and developers can not only reproduce their results, but can also use it for fair comparison and also as a baseline to rapidly progress forward without reinventing the wheel. Towards this end, as a first step, we contribute through our open-source libraries of approximate modules at: GeaR-DAC15 [42], [37], lpACLib-ICCAD16 [35], [38] and [29].

REFERENCES

[1] M. Bojarski et al."End to End Learning for Self-Driving Cars," arXiv:1604, pp. 1–9, 2016.

[2] Y. LeCun et al. , B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a backpropagation network," NIPS, 1989.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on CVPR*, Boston, MA, pp. 1-9, 2015.

[4] http://images.nvidia.com/content/technologies/deep-learning/pdf/Datasheet-DGX1.pdf Accessed on 03 May 2017.

[5] Y. H. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017.

[6] M. Shafique et al., "Invited: Cross-layer approximate computing: From logic to architectures," ACM/EDAC/IEEE 53rd Design Automation Conference (DAC), June 2016.

[7] C. Zhang, D. Wu, J. Sun, G. Sun, G. Luo, J. Cong, "Energy-efficient CNN implementation on a deeply pipelined FPGA cluster," ACM ISLPED, pp. 326–331, 2016

[8] P. Panda et al., "Invited – cross-layer approximations for neuromorphic computing: From devices to circuits and systems," ACM/EDAC/IEEE 53rd Design Automation Conference (DAC), June 2016.

[9] S. Gupta, A. Ankur, G. Kailash, N. Pritish, "Deep Learning with Limited Numerical Precision," ICML, pp. 1737-1746. 2015.

[10] P. Judd et al., "Reduced-precision strategies for bounded memory in deep neural nets," arXiv:1511.05236, 2015.

[11] D.D. Lin, S.S. Talathi, V.S. Annapureddy, "Fixed point quantization of deep convolutional networks," arXiv:1511.06393, 2015.

[12] P. Gysel, "Ristretto: Hardware-oriented approximation of convolutional neural networks," arXiv:1605.06402, 2016

[13] L. Lai, N. Suda, V. Chandra. "Deep Convolutional Neural Network Inference with Floating-point Weights and Fixed-point Activations," arXiv:1703.03073, 2017.

[14] S. Hashemi, N. Anthony, H. Tann, R. Bahar, S. Reda. "Understanding the Impact of Precision Quantization on the Accuracy and Energy of Neural Networks." arXiv:1612.03940, 2016.

[15] M. Courbariaux, Y. Bengio, JP. David, "Training deep neural networks with low precision multiplications," arXiv:1412.7024, 2014.

[16] Z. Du et al.,"Leveraging the Error Resilience of Machine-Learning Applications for Designing Highly Energy Efficient Accelerators," ASP-DAC, pp. 201–206, 2014.

[17] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," ISLPED, pp. 409–414, 2011.

[18] P. Judd et al.,"Proteus: Exploiting Numerical Precision Variability in Deep Neural Networks," ACM International Conference on Supercomputing (ICS), 2016,.

[19] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I Regul. Pap., vol. 57, no. 4, pp. 850–862, 2010.

[20] S. S. Sarwar, S. Venkataramani, A. Raghunathan, K. Roy, "Multiplier-less Artificial Neurons Exploiting Error Resiliency for Energy-Efficient Neural Computing," DATE, 2016.

[21] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," ICCAD, 2016.

[22] G. Srinivasan, P. Wijesinghe, S. S. Sarwar, A. Jaiswal, K. Roy, "Significance driven hybrid 8t-6t sram for energy-efficient synaptic storage in artificial neural networks," DATE, 2016.

[23] S. G. Ramasubramanian, R. Venkatesan, M. Sharad, K. Roy, A. Raghunathan, "Spindle: Spintronic deep learning engine for largescale neuromorphic computing," ISLPED, 2014.

[24] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[25] M. Shafique, F. Sampaio, B. Zatt, S. Bampi, J. Henkel, "Resilience-Driven STT-RAM Cache Architecture for Approximate Computing", Workshop on Approximate Computing (AC), Paderborn, Germany, Oct. 15-16, 2015.

[26] S. Venkataramani, K. Roy, A. Raghunathan, "Substitute-and simplify: a unified design paradigm for approximate and quality configurable circuits," DATE, 2013.

[27] K. Nepal, S. Hashemi, H. Tann, R. I. Bahar, S. Reda, "Automated high-level generation of low-power approximate computing circuits," IEEE TETC, 2017.

[28] Z. Vasicek and V. Mrazek, "Trading between quality and non-functional properties of median filter in embedded systems," Genetic Programming and Evolvable Machines, vol. 2017, no. 1, pp. 45–82, 2017.

[29] V. Mrazek, R. Hrbacek, Z. Vasicek, L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," DATE, pp. 258–261, 2017.

[30] A. Chandrasekharan, M. Soeken, D. Große, R. Drechsler, "Precise error determination of approximated components in sequential circuits with model checking," DAC, 2016.

[31] C. Yu and M. Ciesielski, "Analyzing imprecise adders using bdds – a case study," in 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 152-157, IEEE, 2016.

[32] Z. Vasicek and L. Sekanina, "Evolutionary design of complex approximate combinational circuits," Genetic Programming and Evolvable Machines, vol. 17, no. 2, pp. 1–24, 2016.

[33] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," 25th Great Lakes Symposium on VLSI, pp. 343–348, ACM, 2015.

[34] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, J. Han, "A comparative evaluation of approximate multipliers," IEEE/ACM Intr. Symposium on Nanoscale Architectures, 2016.

[35] lpACLib Library: https://sourceforge.net/projects/lpaclib/ Accessed on 17 May 2017

[36] yann.lecun.com/exdb/mnist Accessed on 03 May 2017.

[37] GeaR Library: https://sourceforge.net/projects/approxadderlib/ Accessed on 17 May 2017

[38] S. Rehman et al. "Architectural-space exploration of approximate multipliers," 2016 IEEE/ACM ICCAD, Austin, TX, 2016, pp. 1-8.

[39] C-SPIN: http://cspin.umn.edu/ Accessed on 17 May 2017

[40] K. Nepal, K., Y. Li., R. Bahar, and S. Reda, "Abacus: A technique for automated behavioral synthesis of approximate computing circuits", IEEE/ACM 17th DATE Conference March, 2014.

[41] M. M. Sabry Aly et al., "Energy-Efficient Abundant-Data Computing: The N3XT 1,000x," in IEEE Computer, vol. 48, no. 12, pp. 24-33, Dec. 2015.

[42] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, "A low latency Generic Accuracy Configurable Adder" ACM/EDAC/IEEE 52nd DAC, San Francisco, CA, USA, June 2015.

[43] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, "Probabilistic Error Modeling for Approximate Adders", in IEEE Transactions on Computers (TC), vol. 66, no. 3, pp. 515-530, 2017.

[44] M. A. Hanif, R. Hafiz, O. Hasan, M. Shafique, "QuAd: Design and Analysis of Quality-Area Optimal Low-Latency Approximate Adders", ACM/EDAC/IEEE 54th Design Automation Conference (DAC), June 2017.

[45] M. K. Ayub, O. Hasan, M. Shafique, "Statistical Error Analysis for Low Power Approximate Adders", ACM/EDAC/IEEE 54th Design Automation Conference (DAC), June 2017.

[46] W. El-Harouni, S. Rehman, B. S. Prabakaran, A. Kumar, R. Hafiz, M. Shafique, "Embracing Approximate Computing for Energy-Efficient Motion Estimation in High Efficiency Video Coding", IEEE/ACM 20th DATE Conference March, 2017.

[47] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, "An Area-Efficient Consolidated Configurable Error Correction for Approximate Hardware Accelerators", ACM/EDAC/IEEE 53rd Design Automation Conference (DAC), June 2016.

[48] D. Palomino, M. Shafique, A. Susin, J. Henkel, "Thermal Optimization using Adaptive Approximate Computing for Video Coding", IEEE/ACM 19th Design, Automation and Test in Europe Conference (DATE), Mar. 2016.