EASE: Energy Optimization through Adaptation — A Review of Runtime Energy-Aware Approximate Deep Learning Algorithms

Salar Shakibhamedan*, Amin Aminifar+, Nima TaheriNejad+*, Axel Jantsch*
*Technische Universität Wien (TU Wien), Austria, +Heidelberg University, Germany
*{salar.shakibhamedan, axel.jantsch}@tuwien.ac.at, +{nima.taherinejad, amin.aminifar}@ziti.uni-heidelberg.de

***Abstract*—This survey provides an overview of the state-of-the-art in runtime adaptive Approximate Computing (AxC) for Deep Learning (DL) algorithms, highlighting the challenges and opportunities in the field. The survey covers a broad spectrum of applications, including medical applications, computer vision, and natural language processing. Various power-constrained platforms, such as System-on-Chips (SoCs), Application Specific Integrated Circuits (ASICs), and Field Programmable Gate Arrays (FPGAs), are explored for their utilization in implementing runtime adaptive AxC. The survey explores various techniques, such as dynamic quantization, adaptive pruning, and low-rank approximations, offering a detailed discussion of their advantages and disadvantages. Specifically, in some surveyed research works, the runtime approximation is achieved through the utilization of machine learning algorithms, with a notable emphasis on Reinforcement Learning (RL). These approaches aim to realize runtime conditions and exploit them appropriately. By providing insights into the advancements and trends in runtime adaptive AxC, this survey serves as a valuable resource for researchers and practitioners interested in this rapidly evolving area of computing. This survey conducts an in-depth investigation into the application, challenges, and scope of runtime adaptive AxC techniques, aiming to mitigate energy consumption while preserving acceptable levels of accuracy in DL models. Our primary focus lies on Convolutional Neural Networks (CNNs), with an emphasis on their application in diverse domains. In striving for comprehensiveness, the survey encompasses selected research works that extend beyond CNNs, including alternative DL models like Recurrent Neural Networks (RNNs). our scope of applications, focuses on CNNs; however, to make a comprehensive survey, we cover some surveyed research works that contain other DL models, such as RNNs. It also highlights the importance of considering specific application requirements and available resources when choosing the appropriate technique.**

***Index Terms*—Energy-Aware, Runtime Approximate Computing, Adaptation, Machine Learning, Deep Learning**

I. Introduction

Recent research illustrates that the energy consumption of computers would be larger than the energy that world energy resources can generate by 2040 [1]. Internet of Things (IoT) and edge devices, such as cameras, cellphones, sensors, and bio-sensors have a pervasive presence in our daily lives. For instance, smartphones containing many different sensors are utilized for many applications in many different domains. They could be used as a health-monitoring device, voice assistance, real-time translation, and many other applications. IoT and edge devices are generally power-constrained. They are usually supplied by batteries and have limited functionality compared to compared to devices that connect to electrical outlets and rely on a continuous and stable power supply. Recently a hot trend has been to run and use Deep Learning (DL) models on the IoT [2] and edge devices [3] have given a dramatic increase in demand. For instance, nowadays, many smartphones have the ability to conduct face recognition using their cameras, speech recognition using their microphones, and health monitoring using their bio-sensors.

Aoccdrnig to a rscheearch at Cmabridge Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be at the rghit pclae. And we spnet hlaf our lfie larennig how to splel wrods. Amzanig, no!

Fig. 1: An example of approximation in the human brain, which allows a habitual or native English speaker to read the text despite its many spelling errors [9].

In recent years, the field of DL has seen significant growth. However, the computational demands of these algorithms can be extremely high, making them challenging to implement on resource-constrained devices. The energy consumption of DL models is a significant concern as it directly impacts the battery life of the device and the performance of the system [4]–[7]. For instance, the power consumption of Nvidia Tesla V100 Graphics Processing Unit (GPU), in a usual computation unit for the DL algorithms, could reach up to 250 watts [8]. The high computational complexity of these algorithms also result in the need for powerful and expensive hardware, which is not always feasible for resource-constrained devices. Approximate Computing (AxC) offers a potential solution to this problem by allowing for the use of approximate versions of these algorithms that still produce acceptable (accurate enough) results while reducing computational demands. Among Machine Learning (ML) algorithms, DL models try to mimic the human brain and biological neuron functions. The human brain makes approximations in its routine tasks and performance. An example of approximation in the human brain is presented in Figure 1.

As mentioned above, ML and DL algorithms inherently need enormous energy to be implemented on power-constrained devices. To have energy consumption improvement, we need to deal with this trade-off. We can consider

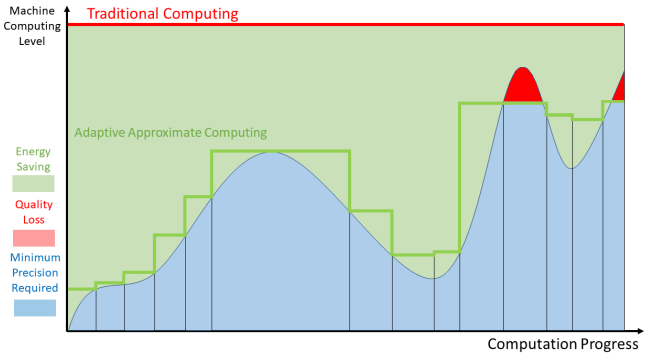


Fig. 2: Temporal dynamics of computation precision for energy-efficient computing.

(a) A clear image of a dog **(b)** An opaque image of a dog

Fig. 3: Complexity difference in recognizing the same class of objects, in this case, a dog.

accuracy-speed-energy as a space, which one needs to find a point that meets the requirements.

Runtime adaptive AxC is a specific approach in AxC that allows the precision of computations to be adjusted dynamically at runtime, based on the specific requirements of the application and the available resources. As illustrated in Figure 2, adaptiveness allows us to land on different points of this space, based on the runtime conditions. The figure shows that the minimum computation precision necessary for exact computations may vary over time and across different time intervals. This observation motivates the exploration of energy-saving opportunities by compromising computation precision compared to conventional computing approaches. However, it is important to note that in certain scenarios, our adaptive approaches may not satisfy the minimum requirement due to the maximum allowable approximate precision. As a result, a compromise in quality may be observed. Therefore, the adoption of adaptive approximate computing introduces a trade-off between energy conservation and computational precision, which can be dynamically managed over time based on prevailing conditions, specific requirements, and other pertinent factors. This approach can be particularly useful for DL algorithms, as it allows for the optimization of the energy-accuracy the trade-off for these algorithms. The use of runtime adaptive AxC in DL algorithms is driven by the need to reduce the computational demands of these algorithms in order to enable their implementation on resource-constrained devices. By using runtime adaptive AxC, it is possible to significantly reduce the energy consumption of these devices while still maintaining acceptable levels of accuracy.

Additionally, some of the computations could be conducted inexactly and still lead to correct outputs. For instance, ResNet 50 [10] is a well-known DL model for object recognition and consists of 50 layers. It is trained by images captured in different situations, such as in low light, sunny, and shadow conditions. Making the decision for some of them is not challenging because the subject is perfectly recognizable (e.g., the light condition is great and the object is completely in the frame). For some others, decision-making is more difficult (e.g., in low-light conditions). Nevertheless, the ResNet 50 conducts all the computations in all 50 layers for both situations to make the decision. For the images with better conditions, some computations are redundant, and the decision could be made using much fewer computations compared to complex images.

As shown in Figure 3, both images contain a dog that should be recognized by the ResNet 50. However, the decision making for the left image, Figure 3a, could be done using less computations due to better data and conditions (good lighting in the image, distinguished background, and location of the subject). Accordingly, the accuracy of the algorithms could be modified during the time and based on such (or other) reasons [11]. Runtime adaptive AxC can also be used to adapt the precision of computations based on the situational requirements of the application. For example, in a self-driving car application, the precision of computations might need to be higher when the car is in a busy urban area compared to when it is on a highway. By using runtime adaptive AxC, it is possible to adjust the precision of computations in real-time, in order to optimize the energy-accuracy trade-off for the specific requirements of the application.

There are a number of techniques that have been proposed for runtime adaptive AxC in DL. These include dynamic quantization [12], adaptive pruning [13], and low-rank approximations [14]. Such AxC methods could be applied at software, hardware, and architecture levels. The level of AxC could be adjusted based on some criteria and conditions such that we can efficiently perform inherently intensive computations on power-constraint devices and platforms.

In this article, we provide an overview of the state of the art in runtime adaptive AxC for DL models and algorithms, highlighting the challenges and opportunities in the field, to serve as a useful resource for researchers and practitioners interested in this rapidly-evolving area of computing. The works surveyed in this paper cover a broad spectrum of applications, such as medical applications, computer vision, and natural language processing. Moreover, various power-constrained platforms such System-on-Chips (SoCs), Application Specific Integrated Circuits (ASICs), and Field Programmable Gate Arrays (FPGAs) have been used in these works. Each of these techniques has its own set of advantages and disadvantages, and the choice of technique will depend on the specific requirements of the application and the available

resources.

The rest of this paper is organized as follows. In Section II, we introduce the taxonomy used in this survey. We elaborate on the surveyed research works in Section III. In Section IV, we discuss what we learned in our survey and analyze them. Finally, in Section V, we provide our conclusions about the issues, hot topics, and trends in this topic.

II. Taxonomy

In this survey, we study runtime adaptive approximate DL algorithms with the following criteria in mind:

- whether they use a ML approach for the approximation,
- awareness level of the algorithm,
- approximation level,
- implementation platform,
- data dependency, and
- application.

We have classified the surveyed research works into the categories mentioned above to highlight their main contribution. Additionally, we use a subset of categorization for AxC that is more fit for runtime situations and based on DL applications. Similar to traditional AxC taxonomy, approaches and techniques are categorized into three main groups, i.e., 1) Algorithmic (Program/Software), 2) Hardware (Circuit), and 3) Architecture.

Each of these categories incorporates several sub-categories. This taxonomy is illustrated in Figure 4, and we briefly review its classes in the rest of this section.

1) Algorithm: DL models have demonstrated their effectiveness in various domains. In return for this outstanding performance, they need heavy computational power inherently. Usually, DL models consist of several different layers. For instance, VGG19 [15] consists of sixteen convolution layers, three fully connected layers, five MaxPool layers, and one SoftMax layer. VGG19 needs 19.6 billion FLOPs for each inference. To reduce these complexities, we can use software (algorithmic) approaches.

Pruning: Pruning is the elimination of weights and connections in a DL model that have low-significance and low-importance [16]–[20]. The criteria for importance detection are various such as the value of connection [21], importance [22], and filter correlation [23]. Moreover, the magnitude of pruning or the pruning rate is a trade-off between accuracy and speed.

Quantization: The ML and DL models' parameters are presented by the float32 format in many ML frameworks such as TensorFlow [24] and PyTorch [25]. It means that for each parameter, we need 4 bytes in the memory. For instance, for the VGG19 with 138 million parameters, about 500 megabytes are needed in the memory [15]. By Quantization, the bit-width of the parameter representation (32 bit) could be reduced [12], [26]–[28]. By this reduction, we can reduce the memory demand for saving and computing the parameters and also the computation power demand for them. There are different degrees of Quantization [29] such as float16, float8, int8, 4-bit representation, 2-bit representation, and even binary (1-bit) neural network (BNN), which is an extreme application of Quantization in the Convolutional Neural Networks (CNNs) [30].

Weight Sharing: Weight sharing gathers the weights and parameters into sets to reduce the size of the networks. By weight sharing, each parameter is assigned to a specific value from a definite set [31]–[33]. Moreover, weight sharing can enable multiplications to be conducted by lookup tables and lower power consumption [34].

Weight and Activation Sparsification: Weight and activation sparsification is a general form of pruning that is applied to both weights and activation values [35]–[37]. Typically, the sparsification approaches are classified into two general groups [38]: i) non-structured and, ii) structured. Each of these approaches has its own advantages and disadvantages. For instance, non-structured sparsification leads to random connection removal between the neurons, whereas structured sparsification can reduce computation resources significantly for the convolution layers.

Approximating Networks: By this approach, the main components of a neural network are approximated, replaced, or (and) removed [39]. For instance, to reach lower computation demands, the number of the hidden layers in a neural network can be reduced, or neurons can be replaced with the approximate ones [40].

Approximation of processes: In approximation of processes, certain steps and disciplines in exact and standard processing are skipped to reduce the consumption of power and computational resources. A well-known approach here is skipping several rows in the weight matrix of a neural network to reduce the number of the needed operations [41].

Knowledge Distillation: Knowledge distillation is a technique for neural network compression and size reduction based on the information and knowledge that is gained by a trained network. In this technique, a smaller neural network (sub-neural network) is trained by a bigger neural network [42]. In some types of this technique, an ensemble of models (teachers) train the simple and/or shallow model (student) to distillate the knowledge from the teachers to the student [43], [44]. In this procedure, the main target is that the student networks can mimic and imitate the teachers' behavior as much as possible with the least accuracy reduction.

Loop Perforation: Loop perforation is a well-known approximation technique that can be adopted in DL training processes. In this technique, a percentage of iterations are skipped to reduce the computational overhead [45], [46].

Tiling and Data Reuse: Using the tiling technique, massive data can be decomposed to smaller tiles that can be stored in the on-chip cash memory to reduce the Dynamic Random Access Memory (DRAM) read accesses [47]–[49]. DRAM or off-chip memory reads need much more power and cause longer delay compared to Static Random Access Memory (SRAM) or cache memory reads. Besides, tiling provides scalability to the entire architecture to make the different components compatible with on-chip memory storage.

Input-Dependent Computation: Different parts and regions of input data may make different contributions to the output of a Deep Neural Network (DNN). By utilizing the input-dependency computation concept, and analyzing the

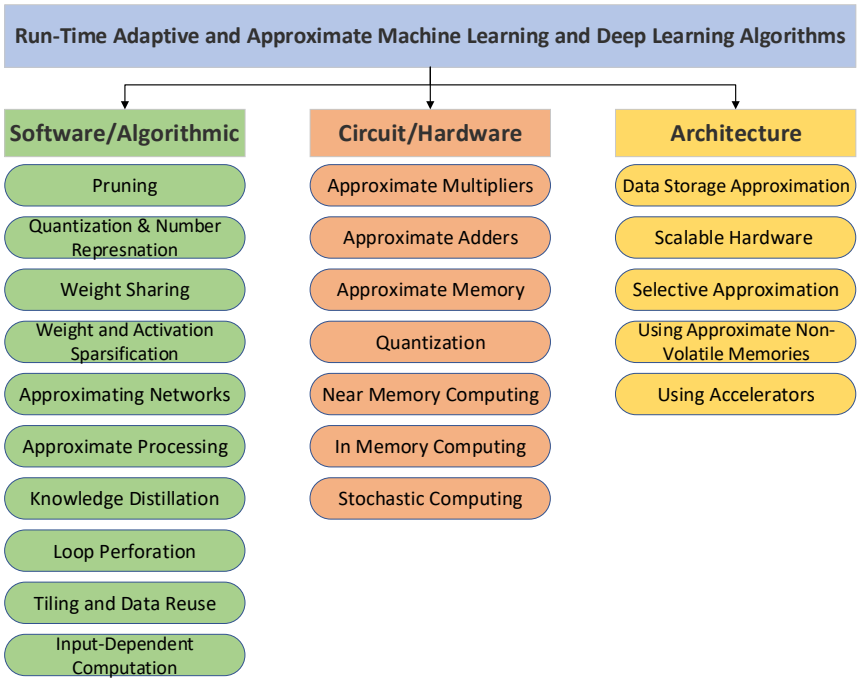


Fig. 4: Taxonomy of runtime adaptive and approximate DL algorithms

contribution of each data part to the output, the importance of each part can be illustrated [50]–[52]. Therefore, the needed computation for the redundant or low-importance part can be recognized and approximated, replaced, or removed to have more compression, higher execution time, and lower power consumption [52], [53].

2) Hardware: The techniques and approaches in this category are implemented and conducted at the hardware level. AxC at the hardware level involves designing accelerators and circuits that bear inaccuracies in their computations, prioritizing efficiency over precise results to make them particularly useful in certain applications where minor errors are tolerable. Many techniques jointly consider software and hardware to improve the effectiveness, implementation, and approximation. The main sub-categories of hardware approximation are presented as follows:

Approximate Adders: Adder is the primary arithmetic operator for the computations [54]–[56]. Indeed, other arithmetic operators such as multipliers, subtractors, and dividers are derived from the adder [57]. Approximate adders are adders that are not entirely accurate. In these kinds of adders, some logical units are removed or replaced by simpler ones to reduce the area, power consumption, or delay of the adder. In approximate adders, it is often tried to sacrifice the Least-Significant Bit (LSB) as much as possible to reduce the error due to approximation [58].

Approximate Multipliers: Multiplication is one of the most used arithmetic operations in computations, especially for DL. [59]–[61]. Similar to adders, approximate multipliers are inaccurate and low-latency, energy-efficient multipliers with fewer units and areas than the exact ones.

Approximate Memory: Reading and writing are two energy-intensive operations that occur many times for DL applications [62]–[64]. To reduce the energy consumption of these operations, the approximation concept can be applied to them [65]. There are two main memory categories in the computation units: SRAM and DRAM. For having the approximate memory, the operation voltage of SRAMs and the refresh rate of the DRAMs can be modified [66]–[68]. By these regulations, we can save energy, while achieving sufficient accuracy considering the application constraints [66].

Quantization: Quantization can also be applied at the hardware level [69], [70]. The standard representation bit-width for the weights, activations, arithmetic operations, and memory read and write is float32. The bit-width can be decreased to reduce resource usage, such as 8-bit, 4-bit, and 1-bit representation in the most sparse representation. [69], [71]. These quantized units can be directly implemented at the hardware level [72], [73]. For instance, the 8-bit buffers can be dedicated to the weights and activations values [69]. Furthermore, for the arithmetic computation, low-bit adders and multipliers can be used to reduce the energy consumption of the hardware implementation and deployment of the DL models [73].

Stochastic Computing: stochastic computing is inherently different from conventional computing approaches and usually demands much less power to be conducted [74]–[76]. There are many stochastic approaches at the hardware level for DL applications. Several examples are provided as follows: i) weight representation by using stochastic rounding [69], ii) training using stochastic gradient descent (SGD) [77] or stochastic variance reduction gradient (SVRG) [78],

iii) using Binary Interface Stochastic Computing (BISC) for implementation [79], iv) using stochastic discrete neurons [80], v) network binarization to reduce its size by using stochastic binarization [81], and vi) using stochastic times smooth units mask in CNNs to have facilitation in conditional computation which leads to massive compression for CNN networks [82].

3) Architecture: These approaches try to achieve approximation goals via a suitable design and implementation of different system components to improve the collaboration of these components and the system's holistic performance according to the application.

Data Storage Approximation: This is a method to store data by using memory access operations that are performed on the unreliable cache memories and are not protected against read/write errors [83]. Data storage approximation would have various impacts on the performance depending on the applications and their input characteristics [84]. There could also be awareness and dependency on the input data [85]. By using this technique, low-latency, and low-energy overhead can be reached by having a partially-protected cache architecture compared to the fully-protected cache memory.

Selective Approximation: Here, the processes are conducted with the approximate processing elements that have formed a hierarchy structure that provides distinct points on energy vs. quality trade-off [86]. This hierarchy structure has three primary levels, Approximate Processing Elements (APE), Mixed Accuracy Processing Elements (MAPE), and Completely Accurate Processing Elements (CAPE). This selection method is conducted by a hardware mechanism based on precision scaling and error monitoring.

Using Approximate Non-Volatile Memories: This technique can explore the energy-quality tradeoff in the non-volatile memories such as STT-RAM [87], MRAM [88], and PCM [89], [90] for the probability and availability for minor errors in read/write operations to gain extensive improvements in energy efficiency and saving. Many mechanisms have been used for this purpose, including i) lowering the sensing current [91], ii) lowering the sensing period and simultaneously increasing the reading current [92], iii) lowering the writing period or current [93], and iv) modifying the reading voltage [83]. With these techniques, there will be an adaptivity that can modify the performance of the systems based on the desired constraints and the application. In summary, this approach aims to reduce the cost of data movement in the processing.

Using Accelerators: Although general-purpose computation units such as Central Processing Units (CPUs) are ubiquitous nowadays, their performance is not much splendid given the compute-intensive nature of the DL algorithms [94]. To address this problem, hardware accelerators with more appropriate structures have been developed for this purpose, such as GPUs, FPGAs, and ASICs [95]–[97]. Many software and toolkits such as TensorFlow [24] and PyTorch [25] have been developed for this hardware. Certain DL frameworks such as TensorFlow Lite [98] have been developed for resource-constraint, embedded, edge, and IoT devices.

In this survey, 23 recent adaptive runtime AxC approaches in DL have been studied and classified using our taxonomy.

III. Approximate Computing in Action

Inspired by properties of adaptive AxC, we acquire approximation techniques in computer science applications, including DL applications. Such techniques can be categorized into two main groups, namely A. ML-based approximation approaches, and B. non-ML-based approximation approaches. To investigate the performance and accuracy, the survey proposes the utilization of certain criteria and metrics, depending on the application, such as accuracy, recall, etc. In the following, we evaluate the potential, space, and possibilities of the application for approximation based on the sophistication of the algorithms, the application's complexity, demands from the implementation.

A. ML-based approximation approaches

Due to the abilities of the Reinforcement Learning (RL) methods in handling the runtime situations and uncertainties and also conducting adaptation, we divided the ML-based approaches category into two sub-categories, i.e., 1) the ML-based approaches that use RL, and 2) the ML-based approaches that use other methods.

1) RL-based approaches : The first study focuses on memory, as memory usage involves the most power-intensive operations [66]. To reduce the power consumption of read/write operations, they designed an approximation framework with two approximation knobs. One of them controls the system's main memory (DRAM), and the other controls and conducts the approximation on the on-chip cache memory (SRAM). For the DRAM, the approximation is conducted by modifying the refresh period, and for the SRAM, it is done by regulating the operating voltage. The framework, similar to all the surveyed works, is at runtime and also is input-dependent. The SEAMS (Self-Optimizing Runtime Manager for Approximate Memory Hierarchies) is implemented by using RL to consider the features of input data, the complexity of input data, and the Quality of Service (QoS) or the approximation and learning policies. They have evaluated their framework in three different applications, image processing, ML, and financial analysis.

The authors reported that they reached up to 37% power saving while considering the QoS of the application. Moreover, they reported the superiority of the SEAMS as a runtime algorithm compared to the DART [99] as a design-time state-of-art. The overall schematic of SEAMS is presented in Figure 5.

In [18], the authors utilize RL to apply a runtime pruning approximation on the weights of VGG16 [15]. To the runtime property of the proposed framework, in contrast with classical pruning methods, the framework applies the pruning on the VGG16 dynamically and according to the structure and complexity of the input images. For instance, if the subject is located in a good place with great low-noise and light conditions, much lower computation is needed to classify this image.

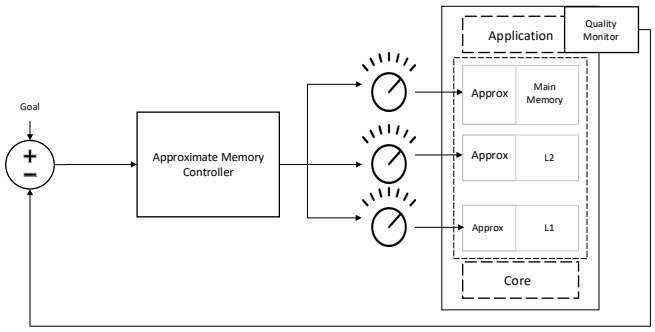


Fig. 5: The schematic of SEAMS [66]

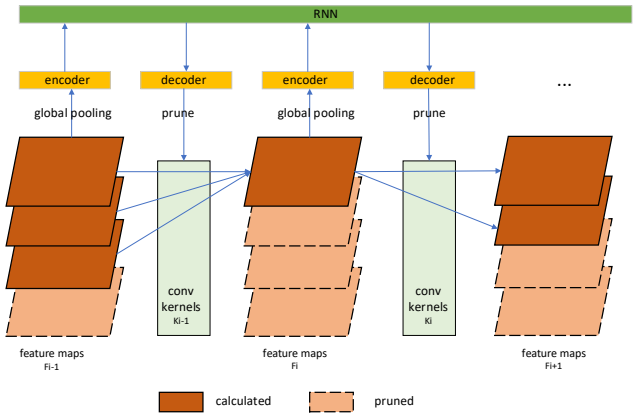


Fig. 6: The overall schematic of runtime neural pruning (RNP) framework [18]

In DL models, the first layers learn the simple features of the input data, e.g., edges and corners in an image, while the deep layers learn more complex features, e.g., patterns and objects in the input image data [18]. Thus, pruning can be adopted based on the complexity of the input data. In this study, RL is applied by a Recurrent Neural Network (RNN) with an encoder and decoder for each layer of the VGG16 to have appropriate learning from the behavior of the network and also the circumstantiality of the input image data. The authors introduce certain parameters in their method to offer a tradeoff between speed (complexity) and accuracy (performance). They demonstrate the outperformance of their method against the state of the art. Their framework achieved up to 5.9 times speed improvement with negligible deterioration in the model performance. The overall schematic of the framework is shown in Figure 6.

In [100], Rao et al. propose the Runtime Network Routing (RNR) framework that focuses on the algorithm and is the extension of RNP (Runtime Network Pruning) introduced in [18]. RNR selects certain paths from the input layer to the output layer based on the input data features, the complexity of the input data, and available resources. This is performed dynamically and in the runtime. They achieve significantly lower error rates for VGG16 in various levels of speedup (up to 10 times reduction in FLOPs) compared to the state-of-the-art approaches. They also achieve up to 5.9 and 1.51 times speed up in inference time for VGG16 and ResNet-50 [10], respectively. The other novelty of RNR is that each path in a DNN behaves like an individual decision-making algorithm. The authors showed that aggregating the results of these paths, similar to ensemble algorithms, can produce better results compared to traditional DNNs. Figure 7 shows the overall schematic of the RNR framework.

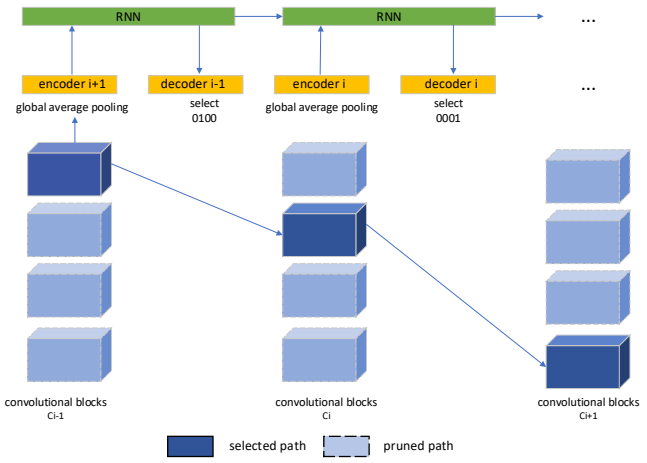


Fig. 7: The overall schematic of runtime neural routing framework [100]

2) Classical ML-based approaches : In [27], Taylor et al. propose an adaptive runtime DNN model selection to have a software approximation for the embedded systems. They use k-nearest neighbour (KNN), decision tree, Support Vector Machine (SVM), and CNNs to achieve adaptivity and reduce the required resources and energy based on the input data. Their aim is to run an image classification algorithm on an NVIDIA Jetson TX2. They performed feature extraction to extract properties such as brightness, aspect ratio, contrast, and edge lengths and employed model selection to select the best DL model for the input image. The inference would be performed based on the DL model that is selected by the ML method in the model selection stage. The existing Neural Networks (NNs) for the selection are diverse versions of Inception [101], ResNet [10], and MobileNet [102]. On average, the latency of inference based on their approach is less than a second and 1.8 times faster than the Inception model, which achieves the highest accuracy. Furthermore, their approach uses only 25% of available RAM in the worst case. The energy footprint of the methodology is 4x and 24x lower than MobileNet and ResNet, respectively. For the improvement of top-1 and top-5 accuracy criteria, the proposed methodology reach 16.6% and 6% for MobileNet, 7.6%, and 0.34% for Inception, and 10.7% (just top-1) for ResNet models. In summary, their approach achieves up to 96% of the performance of the best possible case called Oracle. The overall schematic of the proposed methodology is depicted in Figure 8.

In [26], Taylor et al. extended the adaptive model selection of DNN inference on embedded systems. In their work, they considered machine translation applications and RNNs algorithm that is usually employed for sequential, time-series,

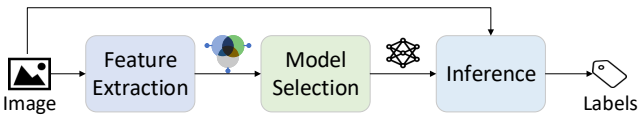
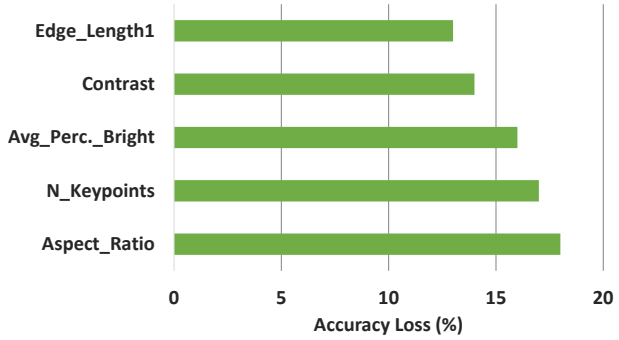
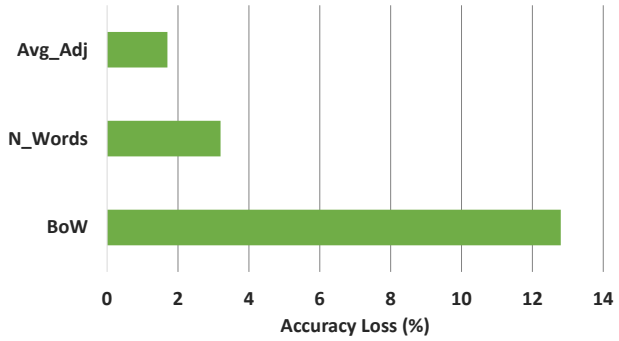


Fig. 8: The overall figure of the proposed methodology [27]



(a) Image classification task



(b) Machine translation task

Fig. 9: Feature engineering and selection for image classification Figure 9a, and machine translation Figure 9b [26].

and temporal data. They also consider the naïve Bayes, in addition to previous algorithms, as ML approach to select the appropriate DNN based on the character of the input data.

Their approach achieved an overall top-1 accuracy above 87.44% for image classification, which means a 7.5% improvement in average accuracy and a 1.8 times reduction in inference time compared to the most-accurate single DNN models. Furthermore, their approach reduced the inference time by 1.34 times compared to the single most capable model without a significant accuracy drop for machine translation. The feature engineering process is illustrated in Figure 9.

In [28], the authors introduce an AxC framework at the hardware level and aim to use 8-bit and 16-bit approximate multipliers with 20 different designs, approximate degrees, and settings. Awareness of the input data and considering the Quality of Results (QoR) is mandatory for adaptation, and to this end, they use a decision tree and a shallow neural network as two lightweight ML-based approaches. Their application is in the image processing domain. The approximate hardware components are the multipliers based on the Approximate Mirror Adder (AMA) with five different designs and four different approximate degrees. The training procedure of the ML-based adaptive runtime controlling frameworks is conducted by the data generating based on the input structure. Through these approaches, they reached input dependency. So, when the proposed framework faces the inputs for multiplication, the framework selects the most appropriate approximate multiplier (degree and type) component based on what it "learned" in the training procedure. The hardware platform used in this research work is an FPGA from the Virtex-6 family. Based on the selected approximate multiplier component on the runtime, they achieved different improvement levels in energy consumption. The power reduction reached from 27.2% to 93.4%, with an average of 60.8%. Regarding the energy reduction, the authors reported a reduction ranging from 23.2% to 99.4%, with 65.5% on average. The area reduction and delay reduction range between 10% and 90% and -12% and 92%, respectively. The adaption time is reported to be negligible compared to the total execution time. The proposed framework performance meets the Target Output Quality (TOQ) by an appropriate margin in all the applications. The specification of each of the approximate multipliers that are used in this work is illustrated in Table I.

TABLE I: Specification of approximate multipliers [28]

Design		Dynamic Power (mW)	Slice LUTs	Occupied Slices	Period (ns)	Frequency (MHz)	Energy (pj)
Type	Degree						
AMA1	D1	306	79	23	8	129	2376
	D2	253	76	29	8	128	1977
	D3	196	77	29	9	105	1860
	D4	38	75	27	7	136	279
AMA2	D1	271	69	23	9	107	2522
	D2	207	63	29	9	107	1940
	D3	165	57	23	10	102	1613
	D4	29	46	18	10	103	281
AMA3	D1	322	67	23	9	108	2970
	D2	262	58	20	7	134	1962
	D3	189	55	21	9	109	1727
	D4	36	32	14	3	323	111
AMA4	D1	263	56	29	6	163	1610
	D2	210	47	19	7	148	1416
	D3	124	40	16	6	170	730
	D4	31	13	7	1	723	43
AMA5	D1	242	53	26	7	143	1698
	D2	183	41	19	6	176	1042
	D3	113	31	11	5	216	523
	D4	30	6	6	1	1416	21
Exact		**442**	**85**	**33**	**9**	**114**	**3866**

In [11] , the authors proposed hardware-efficient approximate accelerators to implement DL algorithms on the FPGAs as an error-resilient application. The authors proposed AxC at the hardware level based on the Dynamic Partial Reconfiguration (DPR) for FPGA implementation. Similar to their previous research work, their approximated hardware core is the approximate multipliers implemented by the DPR feature and on the FPGAs in this research work. In other words, the authors integrated the DPR and AxC.

According to their study on the suitable ML-based approach for adaptive runtime AxC implementation, they selected the decision tree algorithm for this purpose called "design selector". Using the design selector based on the decision tree algorithm allows modifications on the runtime without any reset. Furthermore, they evaluated the energy consumption aspect with more in-depth studies by conducting informative investigations on area, execution time, energy, throughput, accuracy, etc.

Finally, their proposed framework reached 81.82%, 80.4%, and 89.4% of the exact execution time (by considering the user-

given TOQ) for image blending, audio blending, and image filtering applications, respectively.

Leroux et al. [103] introduce an innovative adaptive multi-branch model selection for automated anomaly detection in surveillance videos, emphasizing adaptiveness, awareness of real-world challenges (dynamics), and energy-efficient performance. Existing approaches excel on clean datasets but struggle in adverse weather conditions and evolving scenes. The proposed adaptive model incorporates a trainable preprocessing step, efficiently adapting to the changing and dynamics of environments. Moreover, multiple branches selection shows enhancement in foreground feature exploration, improving anomaly detection accuracy during day-night and sunny-rainy (adversarial) conditions in addition to power efficiency. Experimental validation on distorted datasets and real-world surveillance data demonstrates superior performance compared to existing methods.

In [104], the authors address the challenge of Global Channel Pruning (GCP) for multitask model compression to reduce the computation and memory costs on mobile platforms. Existing works face difficulties when handling multi-task pruning due to task mismatch and inter-layer filter interactions. To tackle this, the authors propose the Performance-Aware Global Channel Pruning (PAGCP) framework. PAGCP optimizes the joint saliency of filters from intra and inter-layers, preserving globally task-related filters. A sequentially adaptive classification-based pruning strategy is developed with a performance-aware oracle criterion to evaluate filter sensitivity to each task. Experimental results on multiple multitask datasets show that PAGCP achieves over 60% reduction in FLOPs and parameters with minor performance drops. Additionally, the proposed framework achieves 1.2x to 3.3x acceleration on cloud and mobile platforms. Real-time application potential on mobile devices is also showcased, making PAGCP a promising solution for efficient multitask model compression.

B. Non-ML-based approximation approaches

1) Approximation by awareness-based concepts :

The runtime adaptive AxC approaches are not conducted and implemented only by ML-based approaches. The other work which has conducted AxC based on the memory is [105]. Galijaard et al. have proposed a “memory awareness” framework for execution and inference on various DNN models in different applications. They aim to improve the runtime performance of the DL model’s inference without any additional computational resources on edge. Authors have designed a novel memory-aware policy called “MEMA” to manage and handle the loading and execution tasks in the inference procedure. Their framework is layer-wise and decomposes the inference on each layer of the network to the three main subtasks, including i) initialization ii) loading iii) execution. Subsequently, by utilizing their novel memory-aware policy, the memory usage is scheduled by order of the layers, the dependency between the layers, the complexity of each layer, constraints of the application, and some other criteria to have the best memory usage and footprint. The overall schematic of MEMA is illustrated in Figure 10.

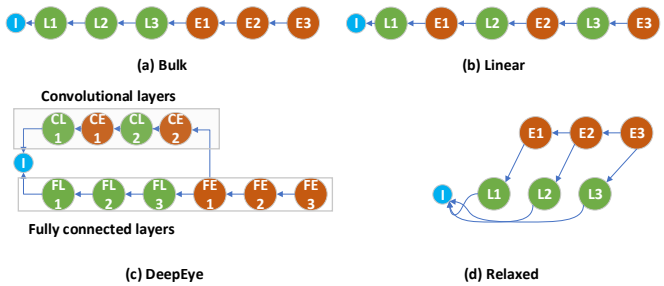


Fig. 10: Specification of DNNs in MEMA evolution [105]

Moreover, in the evaluation section, many applications and CNNs have been covered, such as AgeNet [106], GenderNet [106], FaceNet [107], and SoS-GoogleNet [108]. By the power consumption aspect, they achieved between 40% to 60% improvement in memory demand and up to 5x in execution time in a memory space-constrained platform.

In [109], the authors present a novel ResNet-based architecture with a focus on adaptive computation, awareness of computational cost, and energy consumption. The network leverages the shortcut connections in ResNets, enabling deep structures without suffering from vanishing gradients. By incorporating parameter sharing and adaptive computation time, the proposed model achieves a significant 90% reduction in model size, emphasizing its energy-efficient nature. Moreover, the adaptive computation time feature allows the network to dynamically adjust its computational cost based on the complexity of the input image, resulting in substantially lower energy consumption compared to conventional ResNets.

In [110], the authors improved their previous work [105] (MEMA) by expanding their cases and covering DNNs, parallel computing, model profiler, measuring the dependency between the tasks, and using workers for their implementation. Moreover, Cox et al., in their new framework called (MASA) have implemented the multi-inference, meaning the execution on several DNNs simultaneously that can be used in many cases such as health monitoring. In addition to the MEMA in MASA, the peak memory demand of each layer has been considered in the policies, as mentioned in combination with the dependency measurements between the layers and tasks. MASA aims to minimize the average response time of multi-DL inferences by balancing tasks over the available resources by considering constraints.

The task is a multi-DL inference over DNNs: AgeNet, GenderNet, FaceNet, SoS, GoogleNet, TinyYOLO [111], Emotion Net [112], and Scene Net [113] in the lifelogging application. The specifications of the networks have been illustrated in Table II.

Finally, by proposing the MASA, they achieved up to 79% reduction in response time and 7% improvement in power consumption in different limited memory storage and two defined scenarios.

Meloni et al. proposed an architectural-aware framework for this purpose in the edge DL domain [114]. The authors pro-

TABLE II: Specification of deep neural networks in MASA evolution [110]

Network	Inference	Storage (MB)	Memory Usage (MB)	#Conv Layer	#FC Layer	# Total Layer
AgeNet	Age	44	183	3	3	19
GenderNet	Gender	44	186	3	3	19
FaceNet	Face	217	875	5	3	23
SoS	Saliency	218	875	5	3	21
GoogleNet	Saliency	23	404	22	2	151
TinyYOLO	Object	62	263	9	-	39
EmotionNet	Emotion	378	761	5	3	22
MemNet	Memorability	217	880	5	3	22
SceneNet	Object	221	892	5	3	23

posed a framework which is called "ALOHA". As the authors mentioned, they considered three main objects for ALOHA, i) security: to avoid the unsecured connection between the edge devices and server for computation, ii) power efficiency: which is the main target of any AxC approaches, and iii) adaptiveness: to have the best modification and optimization based on the dynamic of the systems, input data, environment, etc.
ALOHA contains three phases. In the first phase, the optimal algorithm configuration suitable for the specific application (use-case) is selected. This selection is made by using different tools and components, such as the genetic algorithm, a Design Space Exploration (DSE) engine, training configuration (in case of training a DNN), NN configuration (e.g., pruning or quantization), security evaluation and performance/power evaluation.
The second phase's task is optimization and mapping on a heterogeneous low-energy target processing platform. This phase also has different components, such as a system-level DSE engine and Architecture Optimization Workbench (AOW).
Finally, the last phase is the optimization of power and energy saving during the deployment, which is conducted using middleware generation and code customization. They considered the VGG16 as their primitive DNN and then derived two custom DNNs from it for training and also testing on the mentioned hardware platform. They achieved a 48.31% reduction in required FLOPs and a 4x compression compared to the reference model only by their quantization method by considering the compression aspect.

In [115], Meloni et al. introduced a use-case and an implementation of the CNN algorithms by using the ALOHA. The authors aimed to propose a software framework to provide automation for optimal algorithm configuration selection and optimized implementation by considering the given hardware and architectural constraints.

In addition to the proposed methodology (ALOHA) in [114], they mentioned that the components of the ALOHA communicate with each other by using HTTP/REST APIs. But on the other hand, in this recent research work [115], they put their focus on quantization and memory usage (as on the most energy-intensive operations in the computations).

Indeed they declared that using higher bit formats poses that pressure on the memory bandwidth, which would lead to more power usage because of data transfer from DDR to on-chip memory. They reported reaching approximately 2x in performance (i.e., GOPs/s – Giga operation per second) in which 8-bit format representation (the quarter precision representation format) is selected for both weights and activations in the VGG16 model without significant degradation in the accuracy. In the noise-based attack case, the provided robustness by the ALOHA improves the accuracy of the CNN up to 55%. The overall schematic of ALOHA is illustrated in Figure 11.

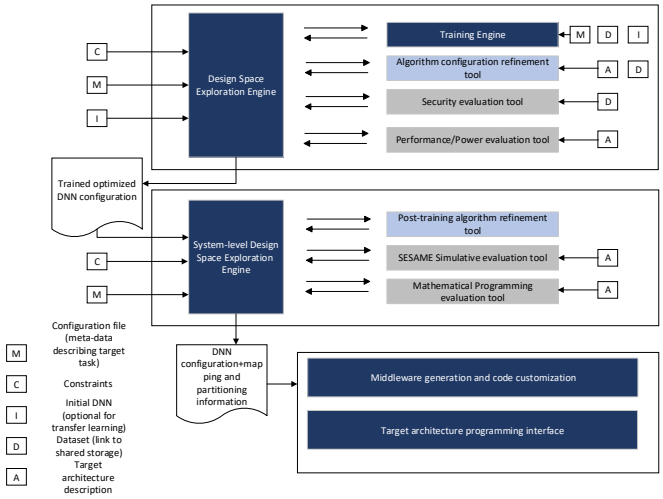


Fig. 11: The overall schematic of ALOHA [114]

In [116], Parra et al. propose a two-stage hardware-aware adaptive approximation methodology for CNNs. It exploits kernel-wise resilience, unlike traditional low-power accelerators that use layer-wise quantization and approximation. In the first stage, quantization and approximation are applied in a generic manner, using 8-bit for activations and 4-bit for weights. The second stage applies kernel-wise approximation and optimization, identifying relevant features in the CNN through back-propagation. The methodology was implemented in ResNet-8, ResNet-14, and ResNet-20, trained on the CIFAR10 dataset. Evaluations showed a 28% power saving in the first stage and up to 34% energy saving in the second stage, with no accuracy drop. The kernel-wise approach is found to result in the most available approximation, with accuracy maintenance as the main constraint. The proposed optimization is shown in Figure 12.

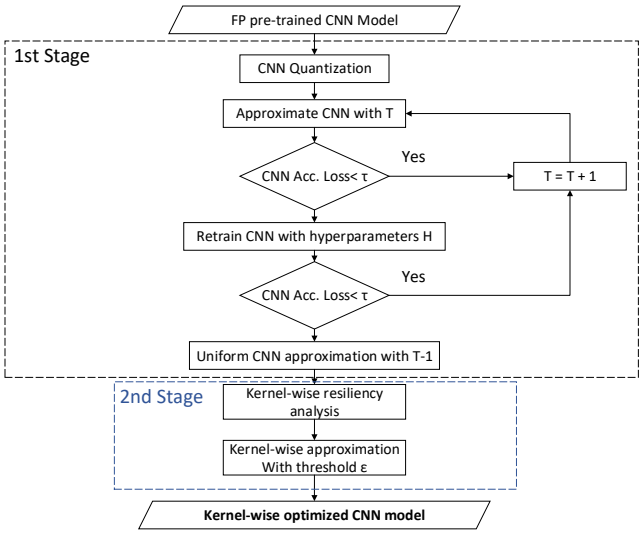


Fig. 12: The overall schematic of hardware-aware optimization methodology [116]

In [117], the authors focus on Quantization-Aware Training (QAT), a model compression technique that leverages weight and activation redundancy. Existing QAT methods require end-to-end training on the entire dataset, leading to lengthy training times and high energy costs. To improve training efficiency, the paper introduces a novel perspective through coreset selection, which aims to enhance data efficiency by exploiting training data redundancy.

The authors introduce a quantization-aware adaptive coreset selection (ACS) method, which intelligently selects data for each training epoch. Evaluation of the proposed method is performed on various networks (ResNet-18, MobileNetV2) and datasets (CIFAR-100, ImageNet-1K) under different quantization settings. Notably, the method achieves an impressive 68.39% accuracy for 4-bit quantized ResNet-18 on the ImageNet-1K dataset using only a 10% subset, resulting in an absolute gain of 4.24% compared to the baseline.

In [118], the authors propose Adaptive Sharpness-Aware Pruning (AdaSAP), a method that optimizes the loss landscape to create robust sparse networks. AdaSAP combines adaptive weight perturbation for pruning and consistent regularization towards flatter regions, unifying the goals of generalization across domains and domain-specific compression. The method demonstrates strong performance across various experiments, outperforming state-of-the-art pruning techniques significantly in image classification and object detection tasks on ImageNet and Pascal VOC datasets. AdaSAP excels in both clean and robust performance, showcasing superior results in image classification and object detection tasks compared to other pruning techniques. Additionally, the authors believe AdaSAP could also benefit unstructured pruning, despite the paper's focus on structured channel pruning.

2) Approximation by other concepts, : In addition to dealing with the accuracy-performance trade-off, AxC can be utilized for other applications such as detecting and defending against adversarial attacks in DL models [119]. DL models are susceptible to adversarial attacks that can alter inputs to produce incorrect results that can harm the victim or benefit the attacker. However, these approaches are insufficient for stronger, high-confidence adversarial attacks.

To address this, a hardware-accelerated defense called DNN-SHIELD is proposed [119], which adapts the strength of the response to the confidence of the adversarial input. This approach employs dynamic and random sparsification of the DL model to achieve approximation efficiently with fine-grain control over the approximation error.

Adversarial inputs are detected by comparing the output distribution characteristics of sparsified inference to a dense reference. An adversarial detection rate of 86% was achieved when applied to VGG16 and 88% when applied to ResNet50 [10], exceeding the detection rate of state-of-the-art approaches with lower overhead. Experiments indicate that an FPGA-based accelerator implementation of software/hardware co-design reduces the performance impact of DNN-SHIELD compared to CPU and GPU software-only implementations. With the increasing prevalence of IoT and DL, their usage is becoming increasingly widespread, especially for medical purposes [120]–[126]. Health monitoring is critical for ensuring the well-being of patients and early detection of any health issues.

Scrugli et al. [127] proposed a runtime adaptive IoT algorithm (model) for health monitoring. The authors aimed to adapt the operating point and make it selective in accordance with the computational load, to minimize power consumption in runtime. The monitoring system consists of a sensor node that detects and senses patients' EEG signals. The sensed data is then processed at the edge and sent to the cloud. The hardware platform has five different operating modes with varying operating frequencies and power usage. To reduce energy consumption during data transmission, the authors considered two scenarios for runtime adaptation: sending raw data and sending processed data.

The second scenario is further divided into two modes, one in which raw data is processed using peak detection to compute the heartbeat rate and the other uses a CNN-based processing unit to detect patterns and recognize anomalous events. The evaluations showed power savings ranging from 7% to 43%, with detailed formalized power consumption modeling for each case. Memory usage and battery life were also reported, with battery life ranging from 10 to 22 days depending on the operation mode and the system using around 31% of the total memory available. The overall schematic of the proposed method is shown in Figure 13.

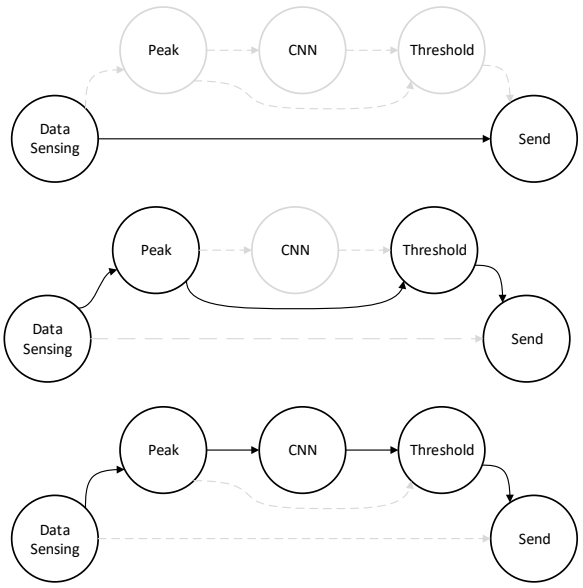


Fig. 13: The overall schematic of runtime adaptation in IoT node for healthcare monitoring [127]

In [128], an adaptable automated approximate multiplier optimization method is proposed, driven by input distribution and polarity. Partial products are compressed through logic and shift operations, leading to low hardware costs and little to no performance loss.

The proposed method was tested on three different-scale quantized DL models, including LeNet [129] on MNIST [130], AlexNet [131], and VGG16 on CIFAR-10, resulting

in 26.4% to 47.6% gains in PDA compared to state-of-the-art approximate multipliers, with accuracy losses of no more than 0.01%. The impact of input polarity on the accuracy of asymmetric approximate multipliers was also demonstrated through results obtained from AlexNet and VGG16. The feasibility of producing a multiplier for similar applications with similar data distributions was verified. The effectiveness of the generated 16-bit signed multipliers was demonstrated in an adaptive LMS-FIR filter, providing up to 27.1% savings in PDA with negligible PSNR loss, compared to the state-of-the-art approximate multipliers.

In [132], Lee et al. conducted an adaptive real-time DNN model selection methodology for real-time object recognition on resource-constrained hardware platforms. They considered four versions of the YOLOv4 [111] (YOLOv4-416, YOLOv4-tiny-416, YOLOv4-288, and YOLOv4-tiny-288) as DNNs models that are used in their proposed methodology, which is called transparencies object detection (TOD) to maximize real-time accuracy on edge. Moreover, Lee et al. mentioned that their proposed methodology would be for video streams and real-time object detection, so the network should not be expensive to evaluate each frame, and also the temporal information of the frame should also be considered in the methodology.

The authors mentioned that due to their studies, lightweight NNs would generally be appropriate for large and fast objects. On the other hand, the heavyweight ones are suitable for small and slow objects. These illustrations provide evidence for implementing runtime adaptive model selection to maintain real-time inference accuracy. The main criteria which are used to implement the runtime adaptive model selection policy are the median of bounding box sizes (MBBs). As they mentioned, mislocalization and false bounding is the main reason for the frame drop in the real-time inference. So, they defined the controller criteria based on the bounding boxes. Furthermore, they have shown that the TOD has the best performance and accuracy for almost all scenarios. Besides, they illustrated the deployment and usage frequency, memory usage, power consumption, and inference latency of each DL model and the TOD methodology. The results showed that the power consumption of the TOD (on average) is lower than the individual usage of each DL model in real-time scenarios, especially for the heavyweight DNNs. The overall schematic of TOD is shown in Figure 14.

In [133], the authors present a design for layer-wise approximate computation at varying approximation levels, aimed at reducing the computational demands of DL models inference while retaining accuracy. The optimal approximation level for each layer of the DL model is determined through a sensitivity-based high-dimensional search. Through extensive evaluations of various DL model benchmarks for medium- and large-scale image classification with CIFAR10 [134], CIFAR100 [134], and ImageNet [135], the methodology demonstrates high flexibility and an optimal balance between accuracy and throughput. An average speedup of 5x and a maximum of 8x without accuracy loss is achieved through a novel approach, which targets in-memory architectures as well as accelerators based on bit decomposition of MAC operation.

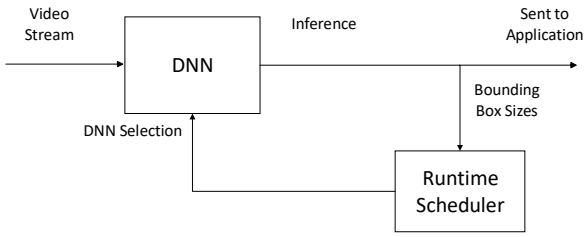


Fig. 14: The overall schematic of TOD. TOD selects the appropriate DNN adaptivity at runtime due to the properties of sequential frames. [132]

In [136], Sharif et al. propose ApproxTuner, a comprehensive approximation and optimization framework for tensor-based domains. It conducts approximation and optimization in three phases: development-time, install-time, and runtime, using both hardware-independent and hardware-specific methods. ApproxTuner also includes a novel predictive approximation-tuning system for end-to-end accuracy prediction. The framework has five software/hardware approximation types and can adapt to changing system conditions for meeting application goals. Evaluations showed improvements in speedup and power saving for 11 benchmarks, including 10 CNNs. The overall schematic of ApproxTuner is illustrated in Figure 15.

In [137], Neda et al. declare that a reduction in the computational load of NNs is promised by the approach of quantization, where the minimum bit-width that maintains the original accuracy can vary greatly among different NNs and even among different layers of a single network. Over-provisioning of NN accelerators with sufficient bit-width is a characteristic of most existing designs in order to preserve the required accuracy across a wide range of NNs. The authors present mpDNN, a multi-precision multiplier with dynamically adjustable bit-width for DNN acceleration. The design supports the run-time splitting of an arithmetic operator into multiple independent operators with smaller bit-width, thereby increasing throughput when lower precision is needed. The proposed architecture, optimized for the LUT-based structure of FPGAs, is designed for FPGAs. The improvement in throughput of 3-15x by enabling run-time precision adjustment is demonstrated through experimental results.

IV. Discussion

In this section, we discuss what we surveyed, and deduce about our studies. Table III summarizes the results of a comprehensive survey of recent research works in the field of runtime adaptive approximation methods. The table indicates that a substantial proportion (70%) of the surveyed works were published in recent years (2021 to 2023), with a majority (74%) of the works being published in the last four years. Figure 16 is a visual representation of the surveyed works outlining the employed approaches and techniques. Notably, a majority of the investigated research endeavors employing

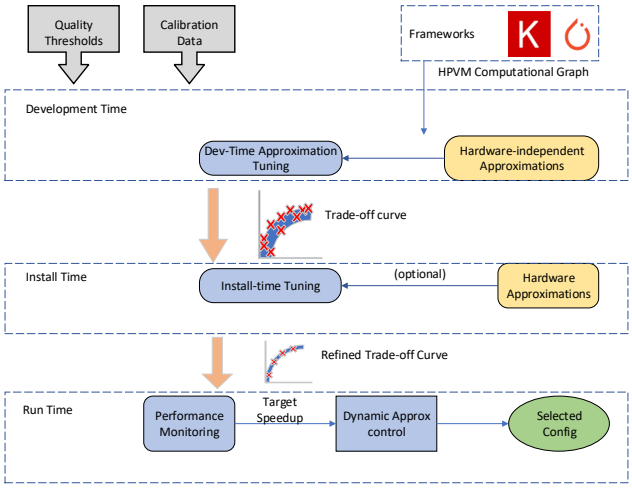


Fig. 15: The overall schematic of ApproxTuner [136]

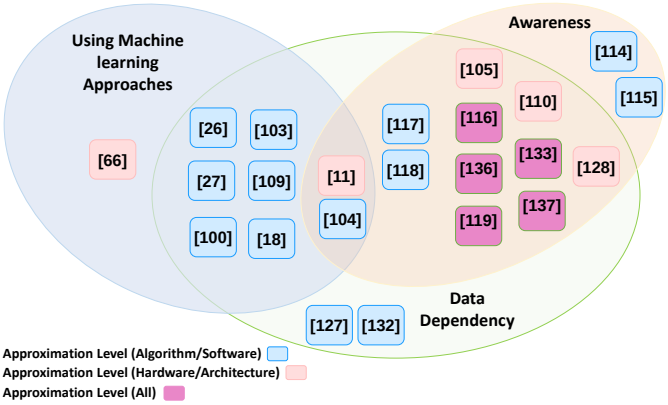


Fig. 16: Venn diagram visualization of surveyed research works.

TABLE III: Summary of the surveyed research works in adaptive runtime AxC approaches and techniques for DL models.

Name	Year	Using ML Approaches	Awareness	Approximation Level	Platform	Dependency on Nurture Of Data
[105]	2021	No	Yes	Hardware	SoC	Yes
[110]	2021	No	Yes	Hardware	SoC	Yes
[26]	2019	Yes	No	Software	TPU	Yes
[27]	2018	Yes	No	Software	TPU	Yes
[115]	2018	No	Yes	Software	SoC	No
[114]	2018	No	Yes	Software	SoC	No
[127]	2019	No	No	Software	SoC	Yes
[11]	2021	Yes	Yes	Hardware/ Architecture	CPU	Yes
[18]	2017	Yes	No	Software	GPU	Yes
[66]	2021	Yes	No	Hardware	FPGA	No
[132]	2021	No	No	Software	TPU	Yes
[100]	2018	Yes	No	Software	GPU	Yes
[116]	2021	No	Yes	Software/ Hardware	GPU	Yes
[136]	2021	No	Yes	Software/ Hardware/ Architecture	TPU	Yes
[119]	2022	No	Yes	Software/ Hardware/ Architecture	FPGA/ GPU/ CPU	Yes
[128]	2022	No	Yes	Hardware	TPU	Yes
[133]	2022	No	Yes	Software/ Hardware/	GPU	Yes
[137]	2022	No	Yes	Software/ Hardware	FPGA	Yes
[109]	2018	Yes	No	Software	TPU	Yes
[103]	2022	Yes	No	Software	GPU	Yes
[104]	2023	Yes	Yes	Software	SoC/GPU	Yes
[117]	2023	No	Yes	Software	GPU	Yes
[118]	2023	No	Yes	Software	GPU	Yes

software-level approximations demonstrate a preference for adopting ML-based approaches to achieve adaptivity. Conversely, approximate techniques characterized by hardware-level approximations predominantly incorporate awareness as a means to attain adaptiveness.

A. Statistical Analysis

One of the key findings of this survey is that runtime approximation (online) is more suitable for addressing unseen and unpredictable conditions, situations, and dynamics, compared to design-time (offline) approximation. Design-time approximation only considers specific, default, and predicted situations and dynamics, and designers use assumptions and limited options to address unseen conditions. In contrast, runtime adaptivity enables designers to handle a wide range of diverse situations. However, the search space and the number of possible states and configurations in approximation and optimization are often extremely large which can hinder the practical implementation of approximation algorithms and methods in real-time applications. To overcome this challenge, designers frequently employ ML-based approaches that are appropriate for complex situations. Figure 16 shows that 35% (8 of 23) of the surveyed works have utilized at least one ML approach to address the complex search space and dynamics.

Another approach to optimizing the search space is to consider the nature and structure of the input data (data dependency). By taking into account data dependency, many points in the search space can be eliminated, as they would not have the capability to support approximation and optimization due to the nature of the data. This results in a tighter search space, as shown in Table III, where 91% of the surveyed research works have utilized at least one of these concepts.
In addition to ML and data dependency, awareness is another important concept that has the potential to be used in dynamic environments, such as runtime adaptive approximation and optimization. As the survey results illustrated in Table III, many aspects of awareness have been used in the surveyed research works, including input awareness, accuracy awareness, hardware awareness, memory awareness, and architecture awareness. These awareness types can be categorized into software/algorithm (input awareness and accuracy awareness), hardware/circuit (memory awareness and hardware awareness), and architecture awareness. Using awareness is another approach that can reduce the search space and the number of possibilities for approximation and optimization. In the surveyed research works, approximately 61% have used some form of awareness in their proposed approximation approaches. Some works have combined awareness with either ML-based approaches or data dependency to further narrow the search space for approximation and optimization. Table III provides a summary of the used approaches.

B. Approximation Approaches

As demonstrated in Figure 17, all the studies included in the survey have employed at least one of the methods outlined (utilizing ML-based approaches, awareness concepts, and data dependency concepts) to address the challenge posed

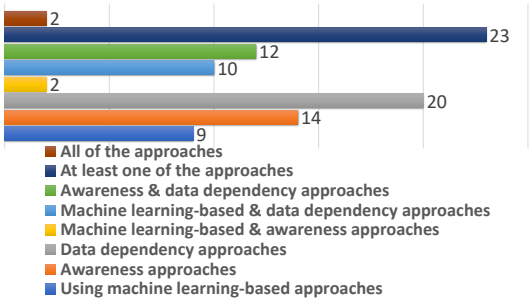


Fig. 17: Different used approaches in the surveyed research work

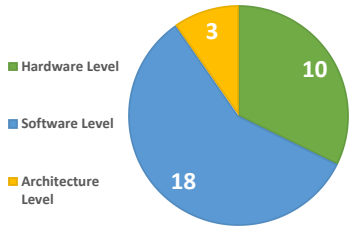


Fig. 18: Different employed approximate level in the surveyed research works

by the large search space and the potential for approximation and optimization. Handling the large search space in the approximation domain is an important aspect that should be considered in research. Furthermore, 87% of the studies have utilized data dependency concepts, highlighting the significance of data dependency in approximation. Redundancy in designs, computations, and configurations can be eliminated by utilizing data dependency. Thus, the computational components and structure should not be uniform for all input data, as some data may be less complex and less informative compared to others, and therefore, complex computations or representations may not be necessary.

The second most commonly used method for reducing the size of the search space and the number of approximate possibilities is through ML-based approaches. These methods have demonstrated their capability and potential in dealing with sophisticated systems, dynamics, and environments. They are mainly data-driven, capable of solving complex issues and problems, derived directly (or indirectly in some cases) from the data and training process. Some of these methods include supervised algorithms such as SVM and decision trees, which extract information and knowledge from the data in the training process. The trained algorithms can process new and unseen data by leveraging the knowledge acquired and finding similarities with the training data, thereby making appropriate decisions. In the surveyed studies, these decisions were focused on defining the search space, configuration possibilities, and approximation knobs. RL was also utilized in several studies. In this method, the algorithm acquires knowledge and learns directly from data through observations within the environment, employing diverse strategies and policies, thereby enabling it to manage unfamiliar data and cope with uncertainties.

The distribution of approximation levels among the surveyed research works is shown in Figure 18. Our survey found that the software level is the most commonly used approximation level, with approximately 78% usage. There are several reasons for this. Firstly, the approximation approaches are applied easier to the software (algorithm) level than the hardware and architecture levels, due to lower costs, lower time consumption, lower complexity, etc. Secondly, as discussed above, many research works have utilized the data dependency approaches. Due to the dynamics of the data dependency approaches and the adaptivity concepts, the best level for a high dynamic nature is the software (algorithm) level. Indeed, in about 89% of the research work with software-level approximation, the data dependency concept has been utilized. Thirdly, according to the applications of the surveyed research works (which will be discussed in the following), DL is the most used case as an application. The DL approaches are inherently conducted at the software (algorithm) level. Indeed, they are algorithms that can make decisions, estimate and classify, predict, etc., using the learning procedures and available data. Therefore, the software level approximation would be the best option for these kinds of computations and approaches due to implementing facility and data-driven-oriented essence of software (algorithm) level.

C. Apprxomation in Applications

The application of 100% of the research works that use approximation and optimization at the software level focus DL algorithms. 83% of the works with DL model as their applications utilize software-level approximation. 43% of the surveyed work used approximation and optimization at the hardware (circuit) level. Indeed, all (100%) the surveyed works with the hardware level approximation are relatively recent (after 2019).

Due to the increasing utilization of SoCs in the pervasive IoT landscape, there is a growing trend toward adopting resource-constrained hardware platforms for the implementation of DL applications. This is particularly relevant in light of recent advancements in in-memory and near-memory processing techniques.

Based on our survey, 91% of the works with hardware-level approximation and optimization focus on DL algorithms. Upon analysis and investigation, we found that 87% of the approximation approaches in the DL domain employed data dependency approaches, and 78% of these approaches were at the software level approximation. Additionally, 61% of the approaches utilized the awareness concept.

D. Hardware Platforms

With regards to the hardware implementation platform, Figure 19 indicates that 57% of the hardware platforms are resource-constrained, such as SoCs, TPUs, and FPGAs, while only 8% are general-purpose platforms, such as CPUs. Due to the nature of resource-constrained platforms, approximation and optimization approaches are more critical for these types of platforms than for power-intensive ones like CPUs and GPUs. This is particularly relevant given the proliferation of edge-AI devices and services in recent years, particularly

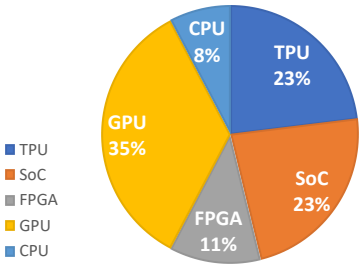


Fig. 19: Statistical distribution of operated hardware platform

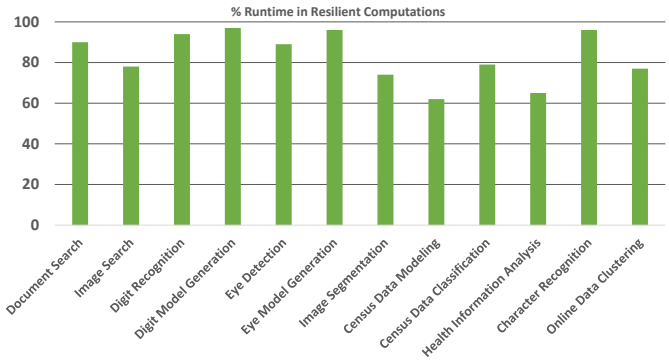


Fig. 20: Percentage of runtime resilient computations in different DL applications [139]

with the growing use of IoT and AI algorithms. As a result, the energy consumption of these devices is becoming a major concern in the years ahead, especially considering one recent study which found that by 2040, the power consumption of computers could surpass the power generation of all world resources [138].

V. Conclusion

Many applications, including DL, and other data-intensive and data-driven applications, are often inherently error-resilient and can tolerate some errors (in computations, Processes, etc.) and a certain degree of approximation without significant degradation in performance and accuracy. Many redundancies (in various aspects) exist in these applications, which can be reduced by applying approximation and optimization. These redundancies provide ample room and capacity for approximation approaches when considering edge-device implementations as the primary target, particularly with regard to runtime and inference execution time, where time is a significant and critical parameter. According to [139], 83% of the runtime is spent on computations that can be approximated in DL applications (on average).

Figure 20 illustrates the percentages of the computations with approximation capability in different DL applications. As shown, ML and DL algorithms have much room for applying approximation and optimization. Based on the findings from our survey, we outline current challenges and future research directions.

TABLE IV: Energy consumption of various arithmetic components (In 45nm Technology) [140]

Operation:	Energy (pJ)	Area (µm2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A

A. Trends

1) More Hardware Approximation: Our survey reveals that all the works that have used the approximation at the hardware level are from 2021 and 2022. In 2021 and 2022, 85% of the surveyed works utilized hardware-level approximation. This is due to two main reasons. First, the advantages and superiority of using quantized number representation systems compared to the standard 32-bit float number representation while keeping accuracy. Using quantization, many principle computation components, such as adders and multipliers, can be replaced with lower energy-intensive and lower bitwidth ones. Table IV depicts the energy consumption of each arithmetic operation with different bit-width.

As shown, lower bit-width arithmetic computation units consume less energy compared to 16-bit and 32-bit operations. The quantization technique and low bitwidth operators have been utilized in 85% of the surveyed works that used approximation methods at the hardware level. Moreover, quantization leads to model compression. As shown, the energy consumption of "memory" operations is much higher than arithmetic operations. According to these results, the second main reason for using approximation approaches at the hardware level is memory usage (footprint). The distance of the memory location directly affects its energy cost. As illustrated in Table IV, the energy usage of DRAM reading (as an off-chip memory) is 128x more than SRAM reading (as an on-chip memory). As a result, researchers have focused on the memories and their impact on computation costs, performance, and speed. Indeed, 70% of the surveyed works utilizing hardware approximation approaches considered memory as their primary concern for hardware-level approximation. Some have tried to make computations and data fit to put them on cache memory (on-chip memory) instead of an off-chip memory like DRAMs.

2) In- and Near-Memory Computing: The near-memory and in-memory computation techniques have been gaining significant attention in recent times. In [141], Iandola et al. achieved up to 510x model compression without sacrificing accuracy through employing pruning, quantization, and other approximation methods. The results of their study are summarized in Table V. In [136] and [133], an in-memory AI accelerator is used as one of the hardware approximation methods.

TABLE V: Comparison of Model Compression [141]

CNN Architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.20%	80.30%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.00%	79.40%
AlexNet	Network Pruning (Han et al., 2015)	32 bit	240MB → 27MB	9x	57.20%	80.30%
AlexNet	Deep Compression (Han et al., 2015)	5-8 bit	240MB → 6.9MB	35x	57.20%	80.30%
SqueezeNet (Iandola et al.)	None (baseline)	32 bit	4.8MB	50x	57.50%	80.30%
SqueezeNet (Iandola et al.)	Deep Compression	8 bit	4.8MB → 0.66MB	363x	57.50%	80.30%
SqueezeNet (Iandola et al.)	Deep Compression	6 bit	4.8MB → 0.47MB	510x	57.50%	80.30%

3) Informativeness: As many surveyed works have shown, the computation components and approaches cannot be the same for all input data, especially for runtime adaptive approximation algorithms where the dynamics of the system are high, and latency and accuracy are critical. 86% of the surveyed research works have used data dependency concepts to address these concerns, and 45% of them have also used ML-based methods. However, the data is dealt with using elements that are not related to the informativeness and explainability of the input data.

In [27] and [26], the authors use ML-based approaches for extracted features (from input data) that have shown their abilities in tasks such as classification and regression, but are not related to the "amount of information" in the corresponding data. In other cases, [114], [115], [18], and [100], the pruning is performed by considering general constraints like accuracy degradation, sparsity, and value of the weights and connections, but the importance of the weights is determined by their values or by the approaches, which are not related to the informativeness of the input data. The key point is that the computation components and approaches must be adapted to the specific parts of input data and their characteristics, taking into account the informativeness and explainability of the data.

B. Future Works

1) Need for New Metrics: Many DNNs outperform human performance and perception, which can be considered redundant in many applications [142]–[145]. Several examples are provided in Figure 21. Therefore, we have the possibility of making approximations based on the redundancies present in the algorithms to achieve an optimized solution.

Our belief is that the current approximation approaches do not fully consider the information flow in DNNs and the informative parts of the input data. These crucial concepts cannot be involved in approximation methods through generic data-dependent criteria such as accuracy degradation, the use of DSP-based features, or ML-based approaches with no related loss functions. We believe that the next steps in runtime adaptive approximation for DL model implementations on resource-constrained devices should involve all surveyed approximation approaches at the software, hardware, and architecture levels. This should be achieved by considering criteria and metrics that determine the information flow from the input data and recognizing the informative parts of it.

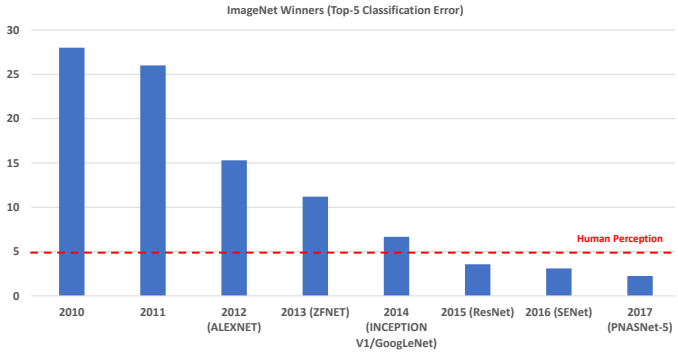


Fig. 21: The accuracy of the image classification algorithms compared with human perception [146]

This will be even more critical and vital in real-time scenarios when dynamics are high, power and resources are limited, and latency is a concern.

2) Multi-Level Approximation: Another aspect that is not sufficiently present in current research is the full utilization of ML-based approaches for multi-level approximation. Although ML has proven its ability in solving complex problems, none of the surveyed works have used its full capabilities for multi-level approximation. In the current research works with ML-based approaches, the approximation is only applied at one level, ignoring the potential benefits of applying approximation at multiple levels, especially at the hardware-software level. The utilization of ML-based approaches would provide more potential and capabilities for approximation.

3) Temporal Approximation: Another point with potential for improvement in the current research space is the consideration of temporal information in runtime approximation approaches. Many DL algorithms in computer vision applications, such as CNNs, U-Nets, GANs, etc., deal with multi-dimensional data such as 2D or 3D images that contain spatial information. The order of the pixels is significant for the algorithms and systems. In certain cases, the algorithms also deal with spectral information, such as audio data or hyperspectral images. Temporal information, which is directly based on time, can also provide benefits, if used in runtime approximation approaches. When DL algorithms are implemented on resource-constrained platforms, they are mainly used for real-world applications. Temporal information will play a significant role in these real-world applications, especially when it comes to adaptation. Each new situation observed in adaptation can be compared to the previous state in terms of changes over time, providing meaningful information for the adaptation approach. Unfortunately, none of the surveyed research works have considered or involved temporal information in approximation.

4) Comprehensive Approximation: The final thoughts on the subject of runtime approximation for DL algorithms on power-constrained devices revolve around the idea that the

approximation should be a comprehensive process. The ultimate goal of this process should be to move from signals and data, which are often redundant, to information, knowledge, and wisdom, which are abstract and informative without redundancy [147]–[149].

For DL applications, concepts such as attention mechanisms [150] and transformers [150] can be utilized to minimize the redundant and non-informative parts of the data. While the approximation process may result in a loss of accuracy, this trade-off can be compensated through the use of multimodality and data fusion, making the approaches more comprehensive. In essence, the aim of the runtime approximation should be to transform the data and signals into meaningful information that can be used for optimal computations and decision-making to address energy consumption reasonably.

ACKNOWLEDGEMENT

The authors gratefully acknowledge funding from European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska Curie grant agreement No. 956090 (APROPOS: Approximate Computing for Power and Energy Optimisation, http://www.apropos.eu/).

REFERENCES

[1] R. Chirgwin. By 2040, computers will need more electricity than the world can generate, 2016.

[2] N. N. Alajlan and D. M. Ibrahim. Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications. *Micromachines*, 13(6):851, 2022.

[3] M. B. Alazzam *et al.* Federated deep learning approaches for the privacy and security of iot systems. *Wireless Communications and Mobile Computing*, 2022:1–7, 2022.

[4] A. Anzanpour *et al.* Self-awareness in remote health monitoring systems using wearable electronics. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 1056–1061. IEEE, 2017.

[5] D. Sopic *et al.* Real-time classification technique for early detection and prevention of myocardial infarction on wearable devices. In *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4. IEEE.

[6] D. Sopic *et al.* Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems. 12(5):982–992. Publisher: IEEE.

[7] F. Forooghifar *et al.* Self-aware anomaly-detection for epilepsy monitoring on low-power wearable electrocardiographic devices. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4. IEEE.

[8] Nvidia tesla v100. https://www.nvidia.com/en-gb/data-center/tesla-v100/#:~:text=NVIDIA\%C2\%AE\%20Tesla\%C2\%AE\%20V100,CPUs\%20in\%20a\%20single\%20GPU. Accessed: 2023-06-19.

[9] G. Rawlinson. The significance of letter position in word recognition. *Aerospace and Electronic Systems Magazine, IEEE*, 22:26 – 27, 02 2007.

[10] K. He *et al.* Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] M. Masadeh *et al.* A quality-assured approximate hardware accelerators–based on machine learning and dynamic partial reconfiguration. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(4):1–19, 2021.

[12] A. Gholami *et al.* A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[13] Z. Yang *et al.* Data-aware adaptive pruning model compression algorithm based on a group attention mechanism and reinforcement learning. *IEEE Access*, 10:82396–82406, 2022.

[14] Z. Chen *et al.* Deep neural network acceleration based on low-rank approximated channel pruning. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(4):1232–1244, 2020.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[16] L. Liebenwein *et al.* Lost in pruning: The effects of pruning neural networks beyond test accuracy, 03 2021.

[17] G. Li *et al.* Fusion-catalyzed pruning for optimizing deep learning on intelligent edge devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39:1–1, 11 2020.

[18] J. Lin *et al.* Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 2178–2188, Red Hook, NY, USA, 2017. Curran Associates Inc.

[19] S.-K. Yeom *et al.* Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021.

[20] M. Ganesh *et al.* Mint: Deep network compression via mutual information-based neuron trimming, 03 2020.

[21] H. Li *et al.* Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[22] P. Molchanov *et al.* Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

[23] P. Singh *et al.* Leveraging filter correlations for deep model compression. In *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, pp. 835–844, 2020.

[24] M. Abadi *et al.* Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[25] A. Paszke *et al.* Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[26] V. S. Marco *et al.* Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1):1–28, 2020.

[27] B. Taylor *et al.* Adaptive deep learning model selection on embedded systems. In *Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, LCTES 2018, pp. 31–43, New York, NY, USA, 2018. Association for Computing Machinery.

[28] M. Masadeh *et al.* Machine-learning-based self-tunable design of approximate computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(4):800–813, 2021.

[29] M. N and S. A. Approximation computing techniques to accelerate cnn based image processing applications – a survey in hardware/software perspective. *International Journal of Advanced Trends in Computer Science and Engineering*, 9:3828–3846, 06 2020.

[30] H. Qin *et al.* Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

[31] A. Brock *et al.* Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017.

[32] G. I. Parisi *et al.* Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[33] W. Roth and F. Pernkopf. Bayesian neural networks with weight sharing using dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):246–252, 2020.

[34] H. Bagherinezhad *et al.* Lcnn: Lookup-based convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7120–7129, 2017.

[35] T. Hoefler *et al.* Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[36] M. Barlaud and F. Guyard. Learning sparse deep neural networks using efficient structured projections on convex constraints for green ai. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 1566–1573, 2021.

[37] M. Kurtz *et al.* Inducing and exploiting activation sparsity for fast inference on deep neural networks. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5533–5543. PMLR, 13–18 Jul 2020.

[38] M. A. Raihan and T. M. Aamodt. Sparse weight activation training, 2020.

[39] S. Venkataramani *et al.* Axnn: Energy-efficient neuromorphic systems using approximate computing. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 27–32, 2014.

[40] S. S. Sarwar *et al.* Energy efficient neural computing: A study of cross-layer approximations. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):796–809, 2018.

[41] Q. Zhang *et al.* Approxann: An approximate computing framework for artificial neural network. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 701–706, 2015.

[42] G. Hinton *et al.* Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[43] J. Gou *et al.* Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[44] A. Alkhulaifi *et al.* Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 7, 2021.

[45] C.-Y. Chen *et al.* Exploiting approximate computing for deep learning acceleration. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 821–826, 2018.

[46] S. Sidiroglou-Douskos *et al.* Managing performance vs. accuracy trade-offs with loop perforation. ESEC/FSE '11, pp. 124–134, New York, NY, USA, 2011. Association for Computing Machinery.

[47] C. Wang *et al.* Dlau: A scalable deep learning accelerator unit on fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(3):513–517, 2017.

[48] C. Zhang *et al.* Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pp. 161–170, New York, NY, USA, 2015. Association for Computing Machinery.

[49] 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, pp. 26–35, New York, NY, USA, 2016. Association for Computing Machinery.

[50] A. Almahairi *et al.* Dynamic capacity networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2549–2558, New York, New York, USA, 20–22 Jun 2016. PMLR.

[51] C. Gao *et al.* Deltarnn: A power-efficient recurrent neural network accelerator. In *26th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA' 18)*, pp. 21–30, New York, NY, USA, February 2018. ACM Digital Library.

[52] A. Karpathy *et al.* Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[53] B. D. Rouhani *et al.* Deep3: Leveraging three levels of parallelism for efficient deep learning. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[54] M. Shafique *et al.* Adaptive and energy-efficient architectures for machine learning: Challenges, opportunities, and research roadmap. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 627–632, 2017.

[55] N.-C. Huang *et al.* Sensor-based approximate adder design for accelerating error-tolerant and deep-learning applications. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 692–697, 2019.

[56] V. Mrazek *et al.* Using libraries of approximate circuits in design of hardware accelerators of deep neural networks. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 243–247. IEEE, 2020.

[57] V. Mrázek *et al.* Evoapproxsb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 258–261, 2017.

[58] J. Lee *et al.* A novel approximate adder design using error reduced carry prediction and constant truncation. *IEEE Access*, 9:119939–119953, 2021.

[59] I. Hammad and K. El-Sankary. Impact of approximate multipliers on vgg deep learning network. *IEEE Access*, 6:60438–60444, 2018.

[60] U. Lotrič and P. Bulić. Applicability of approximate multipliers in hardware neural networks. *Neurocomputing*, 96:57–65, 2012. Adaptive and Natural Computing Algorithms.

[61] V. Mrazek *et al.* Design of power-efficient approximate multipliers for approximate artificial neural networks. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2016.

[62] A. Ranjan *et al.* Approximate memory compression for energy-efficiency. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2017.

[63] A. Ranjan *et al.* Approximate storage for energy efficient spintronic memories. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2015.

[64] S. S. Sarwar *et al.* Energy efficient neural computing: A study of cross-layer approximations. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8:796–809, 2018.

[65] J. He and J. Callenes-Sloan. A software-defined hybrid cache with reduced energy: Poster. Middleware '17, pp. 1–2, New York, NY, USA, 2017. Association for Computing Machinery.

[66] B. Maity *et al.* Seams: Self-optimizing runtime manager for approximate memory hierarchies. *ACM Trans. Embed. Comput. Syst.*, 20(5), jul 2021.

[67] S. Koppula *et al.* Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 166–181, 2019.

[68] M. Shoushtari *et al.* Exploiting partially-forgetful memories for approximate computing. 7(1):19–22, mar 2015.

[69] S. Gupta *et al.* Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

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

[71] G. Venkatesh *et al.* Accelerating deep convolutional networks using low-precision and sparsity. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2861–2865. IEEE, 2017.

[72] H. Sharma *et al.* Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 764–775, 2018.

[73] K. Wang *et al.* Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

[74] S. Mandt *et al.* Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

[75] G. Huang *et al.* Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

[76] J. Li *et al.* Normalization and dropout for stochastic computing-based deep convolutional neural networks. *Integration*, 65:395–403, 2019.

[77] G. Yang *et al.* Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pp. 7015–7024. PMLR, 2019.

[78] W.-Y. Zhao *et al.* Stochastic variance reduction for deep q-learning. *arXiv preprint arXiv:1905.08152*, 2019.

[79] M. Courbariaux *et al.* Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

[80] E. O. Neftci *et al.* Stochastic synapses enable efficient brain-inspired learning machines. *Frontiers in Neuroscience*, 10, 2016.

[81] Z. Lin *et al.* Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

[82] F. J. Bragman *et al.* Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1385–1394, 2019.

[83] F. Sampaio *et al.* Approximation-aware multi-level cells stt-ram cache architecture. In *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '15, pp. 79–88. IEEE Press, 2015.

[84] D. T. Nguyen *et al.* An approximate memory architecture for energy saving in deep learning applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(5):1588–1601, 2020.

[85] M. Abe *et al.* Computational approximate storage with neural network-based error patrol of 3d-tlc nand flash memory for machine learning applications. In *2020 IEEE International Memory Workshop (IMW)*, pp. 1–4, 2020.

[86] S. Venkataramani *et al.* Quality programmable vector processors for approximate computing. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, 2013.

[87] Y. Pan *et al.* A multilevel cell stt-mram-based computing in-memory accelerator for binary convolutional neural network. *IEEE Transactions on Magnetics*, 54:1–5, 2018.

[88] D. Fan and S. Angizi. Energy efficient in-memory binary deep neural network accelerator with dual-mode sot-mram. 11 2017.

[89] M. Gallo and A. Sebastian. An overview of phase-change memory device physics. *Journal of Physics D: Applied Physics*, 53, 02 2020.

[90] V. Joshi *et al.* Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1):1–13, 2020.

[91] A. Ranjan *et al.* Approximate storage for energy efficient spintronic memories. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, New York, NY, USA, 2015. Association for Computing Machinery.

[92] A. M. H. Monazzah *et al.* Quark: Quality-configurable approximate stt-mram cache by fine-grained tuning of reliability-energy knobs. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2017.

[93] C. W. Smullen *et al.* Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, HPCA '11, pp. 50–61, USA, 2011. IEEE Computer Society.

[94] A. Shahid and M. Mushtaq. A survey comparing specialized hardware and evolution in tpus for neural networks. In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pp. 1–6, 2020.

[95] R. Raina *et al.* Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 873–880, New York, NY, USA, 2009. Association for Computing Machinery.

[96] A. Shawahna *et al.* Fpga-based accelerators of deep learning networks for learning and classification: A review. *ieee Access*, 7:7823–7859, 2018.

[97] W. Dai and D. Berleant. Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 148–155. IEEE, 2019.

[98] R. David *et al.* Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems*, 3:800–811, 2021.

[99] R. Yarmand *et al.* Dart: A framework for determining approximation levels in an approximable memory hierarchy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP:1–14, 09 2019.

[100] Y. Rao *et al.* Runtime network routing for efficient image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2291–2304, 2019.

[101] C. Szegedy *et al.* Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[102] A. G. Howard *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[103] S. Leroux *et al.* Multi-branch neural networks for video anomaly detection in adverse lighting and weather conditions. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3027–3035, 2022.

[104] H. Ye *et al.* Performance-aware approximation of global channel pruning for multitask cnns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[105] J. Galjaard *et al.* Mema: Fast inference of multiple deep models. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 281–286, 2021.

[106] G. Levi and T. Hassncer. Age and gender classification using convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 34–42, 2015.

[107] S. S. Farfade *et al.* Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 643–650, 2015.

[108] J. Zhang *et al.* Salient object subitizing. *International Journal of Computer Vision*, 124, 09 2017.

[109] S. Leroux *et al.* Iamnn: Iterative and adaptive mobile neural network for efficient image classification. *arXiv preprint arXiv:1804.10123*, 2018.

[110] B. Cox *et al.* Masa: Responsive multi-dnn inference on the edge. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, 2021.

[111] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[112] G. Levi and T. Hassner. Emotion recognition in the wild via convolutional neural networks and mapped binary patterns. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, pp. 503–510, New York, NY, USA, 2015. Association for Computing Machinery.

[113] B. Zhou *et al.* Learning deep features for scene recognition using places database. In Z. Ghahramani *et al.*, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[114] P. Meloni *et al.* Aloha: An architectural-aware framework for deep learning at the edge. In *Proceedings of the Workshop on INTelligent Embedded Systems Architectures and Applications*, INTESA '18, pp. 19–26, New York, NY, USA, 2018. Association for Computing Machinery.

[115] P. Meloni *et al.* Architecture-aware design and implementation of cnn algorithms for embedded inference: the aloha project. In *2018 30th International Conference on Microelectronics (ICM)*, pp. 52–55, 2018.

[116] C. De la Parra *et al.* Exploiting resiliency for kernel-wise cnn approximation enabled by adaptive hardware design. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[117] X. Huang *et al.* Efficient quantization-aware training with adaptive coreset selection. *arXiv preprint arXiv:2306.07215*, 2023.

[118] A. Bair *et al.* Adaptive sharpness-aware pruning for robust sparse networks. *arXiv preprint arXiv:2306.14306*, 2023.

[119] M. H. Samavatian *et al.* Dnnshield: Dynamic randomized model sparsification, a defense against adversarial machine learning. *arXiv preprint arXiv:2208.00498*, 2022.

[120] J. Kelly *et al.* The internet of things: impact and implications for healthcare delivery (preprint). *Journal of Medical Internet Research*, 22, 05 2020.

[121] D. Sopic *et al.* e-glass: A wearable system for real-time detection of epileptic seizures. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2018.

[122] G. Surrel *et al.* Online obstructive sleep apnea detection on medical wearable sensors. *IEEE transactions on biomedical circuits and systems*, 12(4):762–773, 2018.

[123] F. Forooghifar *et al.* A self-aware epilepsy monitoring system for real-time epileptic seizure detection. *Mobile Networks and Applications*, pp. 1–14, 2019.

[124] F. Forooghifar *et al.* Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud. *IEEE transactions on biomedical circuits and systems*, 13(6):1338–1350, 2019.

[125] A. Aminifar *et al.* Monitoring motor activity data for detecting patients' depression using data augmentation and privacy-preserving distributed learning. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 2163–2169. IEEE, 2021.

[126] A. Aminifar *et al.* Extremely randomized trees with privacy preservation for distributed structured health data. *IEEE Access*, 10:6010–6027, 2022.

[127] M. A. Scrugli *et al.* A runtime-adaptive cognitive iot node for healthcare monitoring. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, pp. 350–357, New York, NY, USA, 2019. Association for Computing Machinery.

[128] Z. Li *et al.* Adaptable approximate multiplier design based on input distribution and polarity. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(12):1813–1826, 2022.

[129] Y. Lecun *et al.* Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[130] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[131] A. Krizhevsky *et al.* Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[132] J. Lee *et al.* Tod: Transprecise object detection to maximise real-time accuracy on the edge. In *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC): Proceedings*, pp. 53–60, June 2021. 5th IEEE International Conference on Fog and Edge Computing 2021, IEEE ICFEC ; Conference date: 10-05-2021 Through 13-05-2021.

[133] C. D. l. Parra *et al.* Increasing throughput of in-memory dnn accelerators by flexible layerwise dnn approximation. *IEEE Micro*, 42(6):17–24, 2022.

[134] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[135] J. Deng *et al.* Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[136] H. Sharif *et al.* Approxtuner: A compiler and runtime system for adaptive approximations. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '21, pp. 262–277, New York, NY, USA, 2021. Association for Computing Machinery.

[137] N. Neda *et al.* Multi-precision deep neural network acceleration on fpgas. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 454–459, 2022.

[138] By 2040, computers will need more electricity than the world can generate, howpublished = www.theregister.co.uk, note = Accessed: 2019-09-27.

[139] V. K. Chippa *et al.* Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, New York, NY, USA, 2013. Association for Computing Machinery.

[140] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

[141] F. N. Iandola *et al.* Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[142] R. Geirhos *et al.* Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[143] F. A. Wichmann and R. Geirhos. Are deep neural networks adequate behavioral models of human visual perception? *Annual Review of Vision Science*, 9, 2023.

[144] L. Gruber and A. Haruvi. Perceptual dominance in brief presentations of mixed images: human perception versus deep neural networks. *Frontiers in Computational Neuroscience*, 12, 07 2018.

[145] R. A. Jacobs and C. J. Bates. Comparing the visual representations and performance of humans and deep neural networks. *Current Directions in Psychological Science*, 28(1):34–39, 2019.

[146] C. P. Langlotz *et al.* A roadmap for foundational research on artificial intelligence in medical imaging: From the 2018 nih/rsna/acr/the academy workshop. *Radiology*, 291(3):781–791, 2019. PMID: 30990384.

[147] M. Hussain *et al.* Intelligent knowledge consolidation: From data to wisdom. *Knowledge-Based Systems*, 234:107578, 2021.

[148] P. Nagabhushan *et al.* Towards machine learning tonbsp;machine wisdom: A potential quest. In *Big Data Analytics: 9th International Conference, BDA 2021, Virtual Event, December 15-18, 2021, Proceedings*, pp. 261–275, Berlin, Heidelberg, 2021. Springer-Verlag.

[149] Z. Malik *et al.* Wisdom extraction in knowledge-based information systems. *J. Knowl. Manag.*, 23:23–45, 2019.

[150] A. Vaswani *et al.* Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Salar Shakibhamedan was born in Tehran, Iran, in 1992. He received the B.Sc. Degree in electrical engineering from the K. N. Toosi University of Technology, Tehran, Iran, in 2015 and the M.Sc. Degree in electrical engineering, focused on multimodal signal processing from the same university, in 2018. Currently, he is working as a Ph.D. Student at the APROPOS (EU-funded project) focuses on approximate computing in embedded machine learning and deep learning.

Amin Aminifar received his Ph.D. degree in computer science from the Western Norway University of Applied Sciences, Bergen, Norway, in 2022. He is currently a ZITI postdoctoral fellow at the Institute of Computer Engineering (ZITI), Heidelberg University, Germany. His current research interests include privacy-preserving federated learning and distributed machine learning with applications in the healthcare domain, as well as machine learning in resource-constrained devices in the healthcare domain.

Nima TaheriNejad (S'08-M'15) received his Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2015. He is currently a Full Professor at Heidelberg University, Heidelberg, Germany and affiliated with TU Wien (formerly known also as Vienna University of Technology), Vienna, Austria. His areas of work include in-memory computing, cyber-physical and embedded systems, systems on chip, memristor-based circuit and systems, self-* systems, and health-care. He has published three books, three patents, and more than 90 articles. Dr. Taherinejad has served as a reviewer and an editor of many journals and conferences. He has also been an organizer and a chair of various conferences and workshops. He has received several awards and scholarships from universities, conferences, and competitions he has attended. This includes the Best University Booth award at DATE 2021, First prize in the 15th Digilent Design Contest (2019) and in the Open-Source Hardware Competition at Eurolab4HPC (2019) as well as Best Teacher and Best Course awards at TU Wien (2020).

Axel Jantsch (Senior Member, IEEE) received the Dipl.Ing. and Ph.D. degrees in computer science from TU Wien, Vienna, Austria, in 1987 and 1992, respectively. From 1997 to 2002, he was an Associate Professor with the KTH Royal Institute of Technology, Stockholm. From 2002 to 2014, he was a Full Professor of electronic systems design at the KTH. Since 2014, he has been a Professor of systems on chips with the Institute of Computer Technology, TU Wien. He has published five books as an editor and one as the author and over 300 peer-reviewed contributions in journals, books, and conference proceedings. He has given over 100 invited presentations at conferences, universities, and companies. His current research interests include systems on chips, selfaware cyber-physical systems, and embedded machine learning.