Adaptive approximate computing in edge AI and IoT applications

A review

Damsgaard, Hans Jakob; Grenier, Antoine; Katare, Dewant; Taufique, Zain; Shakibhamedan, Salar; Troccoli, Tiago; Chatzitsompanis, Georgios; Kanduri, Anil; Ding, Aaron Yi; More Authors

DOI
10.1016/j.sysarc.2024.103114

Publication date
2024

Document Version
Final published version

Published in
Journal of Systems Architecture

Citation (APA)
Damsgaard, H. J., Grenier, A., Katare, D., Taufique, Z., Shakibhamedan, S., Troccoli, T., Chatzitsompanis, G., Kanduri, A., Ding, A. Y., & More Authors (2024). Adaptive approximate computing in edge AI and IoT applications: A review. *Journal of Systems Architecture*, *150*, Article 103114. https://doi.org/10.1016/j.sysarc.2024.103114

Journal of Systems Architecture 150 (2024) 103114

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Adaptive approximate computing in edge AI and IoT applications: A review

Hans Jakob Damsgaard [a,*], Antoine Grenier [a], Dewant Katare [b], Zain Taufique [c], Salar Shakibhamedan [d], Tiago Troccoli [e], Georgios Chatzitsompanis [f], Anil Kanduri [c], Aleksandr Ometov [a], Aaron Yi Ding [b], Nima Taherinejad [g,d], Georgios Karakonstantis [f], Roger Woods [f], Jari Nurmi [a]

[a] Electrical Engineering Unit, Tampere University, Tampere, 33720, Finland
[b] Information and Communication Technology Unit, TU Delft, Delft, 2628, Netherlands
[c] Department of Computing, University of Turku, Turku, 20500, Finland
[d] Institute of Computer Technology, TU Wien, Vienna, 1040, Austria
[e] Wirepas Ltd, Tampere, 33720, Finland
[f] Institute of Electronics, Communications & Information Technology, Queen's University Belfast, Belfast, BT7 1NN, Northern Ireland, UK
[g] Institute for Computer Engineering, Heidelberg University, Heidelberg, 69120, Germany

ARTICLE INFO

Keywords:
Approximate computing
Autonomous driving
Edge computing
Positioning
Smart sensing

ABSTRACT

Recent advancements in hardware and software systems have been driven by the deployment of emerging smart health and mobility applications. These developments have modernized the traditional approaches by replacing conventional computing systems with cyber–physical and intelligent systems combining the Internet of Things (IoT) with Edge Artificial Intelligence. Despite the many advantages and opportunities of these systems within various application domains, the scarcity of energy, extensive computing needs, and limited communication must be considered when orchestrating their deployment. Inducing savings in these directions is central to the Approximate Computing (AxC) paradigm, in which the accuracy of some operations is traded off with energy, latency, and/or communication reductions. Unfortunately, the dynamics of the environments in which AxC-equipped IoT systems operate have been paid little attention. We bridge this gap by surveying adaptive AxC techniques applied to three emerging application domains, namely autonomous driving, smart sensing and wearables, and positioning, paying special attention to hardware acceleration. We discuss the challenges of such applications, how adaptive AxC can aid their deployment, and which savings it can bring based on traits of the data and devices involved. Insights arising thereof may serve as inspiration to researchers, engineers, and students active within the considered domains.

1. Introduction

Computing contributes significantly to the world's rising energy consumption. In 2018, data centers in the EU accounted for 76.8 TWh or 2.7% of the total electricity demand, and this number is predicted to increase to 98.5 TWh (a 28% increase) or 3.2% of the total demand by 2030 [1]; others predicting the total energy spent on computing will exceed world energy production by 2040 [2], as illustrated in Fig. 1. This increase is dictated by the rising number of data centers in the Cloud (e.g., computing, storage), and it only adds fuel to the current world energy crisis [3]. In particular, the growing number of Internet of Things (IoT) devices is predicted to exceed 30 billion by 2027, challenging the scalability of Cloud computing [4], as these devices rely on offloading data for processing, incurring communication latency and energy consumption, thereby compounding the aforementioned compute energy. Not only does this increase force the data centers themselves to suddenly manage many more user requests, but it also burdens the network backhaul that communicates data back and forth between various processing elements [5].

The Edge and Fog computing paradigms were introduced to deal with the increased network bandwidth [25], aiming at limiting the

* Corresponding author.
E-mail addresses: hans.damsgaard@tuni.fi (H.J. Damsgaard), antoine.grenier@tuni.fi (A. Grenier), d.katare@tudelft.nl (D. Katare), zatauf@utu.fi (Z. Taufique), salar.shakibhamedan@tuwien.ac.at (S. Shakibhamedan), tiago.troccoli@wirepas.com (T. Troccoli), georgios.chatzitsompanis@qub.ac.uk (G. Chatzitsompanis), spakan@utu.fi (A. Kanduri), aleksandr.ometov@tuni.fi (A. Ometov), aaron.ding@tudelft.nl (A.Y. Ding), nima.taherinejad@ziti.uni-heidelberg.de (N. Taherinejad), g.karakonstantis@qub.ac.uk (G. Karakonstantis), r.woods@qub.ac.uk (R. Woods), jari.nurmi@tuni.fi (J. Nurmi).

https://doi.org/10.1016/j.sysarc.2024.103114
Received 21 December 2023; Received in revised form 12 March 2024; Accepted 13 March 2024
Available online 17 March 2024

Table 1
Overview and comparison of related surveys.

Authors	Reference	Year	Technologies				Applications		
			AxC	Edge	ML/AI	Comms.	Auto. driving	Smart sensing	Positioning
Han et al.	[7]	2013	✓						
Xu et al.	[8]	2015	✓						
Mittal	[9]	2016	✓						
Shi et al.	[10]	2016		✓		✓			
Betzel et al.	[11]	2018	✓			✓			
Ibrahim et al.	[12]	2018	✓		✓				
Yousefpour et al.	[13]	2019		✓					
Ma et al.	[14]	2019		✓	✓	✓		✓	✓
Cococcioni et al.	[15]	2020	✓		✓		✓		
Shi et al.	[16]	2020		✓	✓	✓			
Pascacio et al.	[17]	2021				✓			✓
Kiran et al.	[18]	2021			✓		✓		
Ometov et al.	[19]	2021		✓		✓		✓	✓
Ding et al.	[20]	2022		✓	✓	✓			
Damsgaard et al.	[21]	2022	✓	✓					
Badran et al.	[22]	2023	✓			✓			
Katare et al.	[23]	2023		✓	✓	✓	✓		
Grenier et al.	[24]	2023				✓			✓
Our work	—	2024	✓	✓	✓		✓	✓	✓

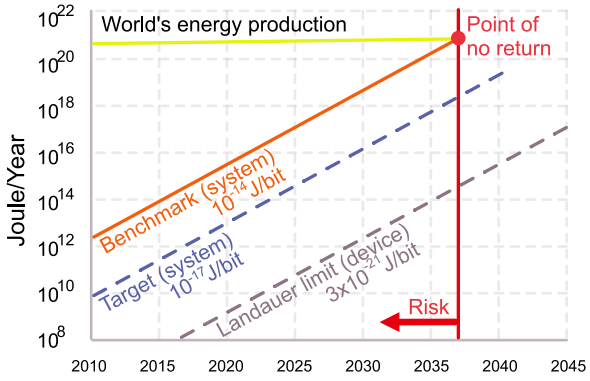


Fig. 1. Historic and predicted energy consumption of computing systems compared to the world energy production (reproduced from [6]).

communicated data and increasing the data processing in the network Edge devices [26]. Recent technological advances, such as the increased popularity of smartphones, have enabled this form of processing [13], which has been shown to potentially improve energy efficiency [27] and improve the overall communications quality balance [28]. While the Edge and Fog domains differ in some details, as will be seen later, for ease of explanation they are considered as one mid-tier computing layer that facilitates offloading with reduced overheads here. Executing applications closer to the user end devices can reduce communication latency and backhaul contention, opening research avenues focusing on the design of new intelligent applications that can be executed on low-power, but enhanced Edge devices, commonly referred to as Edge Artificial Intelligence (AI) [10,16]. The intersection of IoT and Edge AI is particularly interesting as it enables responsiveness and privacy at low power consumption [20].

Designing applications for low power consumption involves optimizations at different levels: (1) at the *device* level, as embedded devices' access to energy is often constrained; (2) at the *communication* level, as the energy required to offload data needs to be compared against onboard processing; and (3) at the *Cloud* level, with accelerators tailored for specific applications to optimize efficiency. The emerging Approximate Computing (AxC) domain that has been increasingly reviewed over the past decade spans all these levels and involves trading off numerical accuracy or functional correctness for lower energy consumption, communication latency, circuit area, etc. [8,9,21]. In contrast to conventional precision-oriented developments, it exploits the observation that many applications are error-resilient and have user requirements that can be satisfied with a lower-grade system or imperfect models [21,29,30]. Such applications employ algorithms that aggregate large (redundant) data sets, iteratively refine outputs thereby attenuating errors, or produce ranges of outputs considered to be equally acceptable [31]. These characteristics are present in, among others, image and video processing, positioning, analytics, and especially in Machine Learning (ML), for which even drastic approximations (e.g., binarization) have limited quality implications on the results [32].

1.1. Focus areas

In this paper, we focus on adaptive AxC as a technological enabler of more energy-efficient, smarter applications at the intersection of IoT and Edge AI. For this purpose, we consider adaptive AxC as a collection of techniques for improving the performance and energy efficiency of computing systems by optimizing their energy-latency-accuracy trade-offs dynamically with respect to their instantaneous quality constraints. We divide computing systems into two layers: **L1** comprising applications and algorithms and **L2** comprising hardware architectures and devices. Each layer possesses its requirements and opportunities for making effective use of AxC: at **L1**, mathematical models, and algorithmic understanding are needed to highlight areas where approximations may be applied, and at **L2**, hardware should implement arithmetic and logic circuitry to support preceding layer extensions of software. Binding the two together requires support for managing approximations according to error resilience characteristics for performance enhancement, power and energy savings, reliability, lifetime, or other parameters at run time with low overhead [21].

As this field of survey is large, we narrow down the scope to focus on three IoT and Edge AI-related application domains in which we believe AxC can make a difference: **autonomous driving**, **smart sensing and wearables**, and **positioning**; and the adaptive AxC techniques relevant to these. These domains are popular and have been subject to considerable research effort for some years now; yet significant challenges remain to satisfy their energy requirements, as we will see later. Fig. 2 highlights our appreciation of the landscape of these applications and the hardware on which they are executed by mapping the **L1** to the *Applications* and *Algorithms in Software* classes and **L2** to the *Algorithms in Hardware* and *Hardware* classes. Naturally, the separation of algorithms and techniques across these classes is not trivial, and some exist at the intersections; for example, edge detection algorithms may just as well be executed purely in software as in hardware.

1.2. Contributions

There exist several surveys and reviews in the fields of AxC and the three covered application domains. Two papers, for example, focus

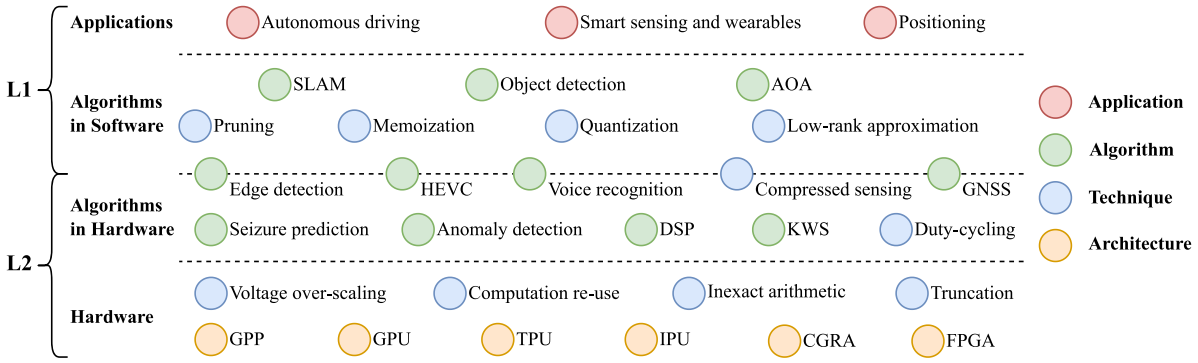


Fig. 2. Overview of AxC techniques throughout layers and their application domains.

on circuit- and architecture-level AxC techniques [7,8]; another gives a broader but mostly application-agnostic survey of AxC techniques [9]; and yet another considers only techniques for Edge computing hardware, providing examples of applications that can benefit from AxC but without covering their application-level approximations [21]. This particular emphasis on use-case-agnostic techniques limits the existing works' articulation on exploiting application-specific characteristics. Similarly, existing surveys on Edge computing or Edge AI do not consider AxC in detail [10,13,16,20]. We summarize the differences between the present survey and the existing literature in Table 1.

This survey bridges these gaps in the existing literature by considering the intersection of AxC with three emerging Edge AI and IoT-related application domains. We describe application-, architecture-, and circuit-level AxC techniques and survey and highlight recent publications on their use in the three domains. Our main contributions are (1) a discussion of the challenges of applications within these emerging domains; (2) a description of how AxC techniques can be applied to these applications; and (3) an outline of the potential benefits that arise from using AxC based on the properties of the processed data and the devices involved in these application domains. With this, we aim to showcase the vast amount of existing work on AxC in the three domains and to highlight the importance of combining AxC techniques across the system levels to maximize its benefits. Moreover, we hope to instigate further research in three directions: (1) application of adaptive AxC to new application domains, (2) development of new techniques suitable for application-specific or generic use, and (3) implementation of these techniques in software and hardware.

1.3. Methodology

As the topic of the present survey is rather broad, conducting a comprehensive, systematic review of its related literature is infeasible. Instead, we performed an *integrative review* [33] by collating papers within each of the authors' research domains and filtering them to avoid multiple references to the same topic or use case. The resulting set of publications was reviewed by the senior authors and adjusted as needed to sufficiently cover the surveyed techniques and domains.

1.4. Paper structure

The paper is structured as follows. Section 2 provides a brief background of Edge computing and ML needed to follow the next sections. Section 3 covers circuit-level AxC techniques and architectures at **L2** and application- or algorithm-level techniques at **L1** relevant in the present context. Next, Section 4 presents the three application domains, outlines relevant algorithms, and describes their approximation opportunities. We put little emphasis on techniques for binding **L1** and **L2** together, considering them as mere communication links between applications and hardware. Section 5 discusses observations made in the paper and suggests directions for future work. Section 6 concludes the survey.

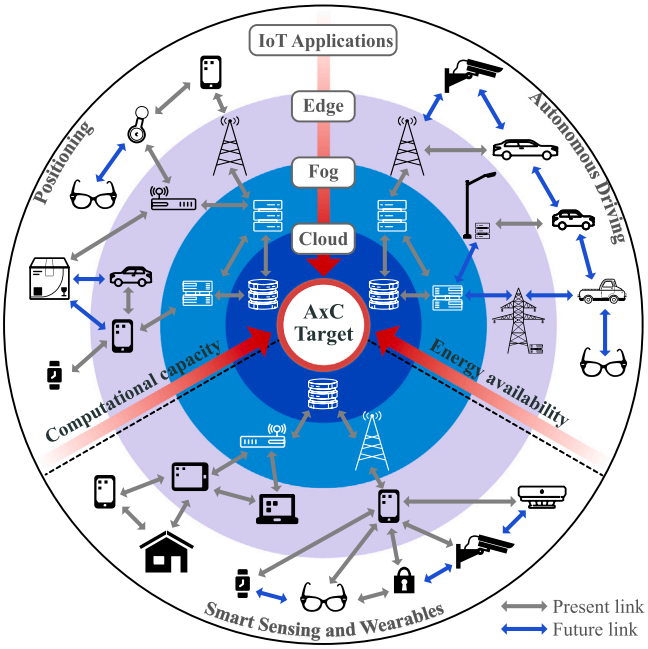


Fig. 3. Overview of the main computing paradigms and their devices and communication links relevant within the three application domains considered.

2. Background

Before surveying state-of-the-art publications related to the aforementioned three considered application domains, we provide a brief motivation and introduction to related general technologies and trends. Specifically, we cover Edge computing and ML. We aim to maintain a high abstraction level for better accessibility to a broader readership, referring interested readers to consider the following sections for more details.

2.1. Edge computing

The number of connected devices has exploded over several decades [4,25]. These devices drastically increase network bandwidth requirements as more data are produced at the end devices but *offloaded* for storage and processing in Cloud datacenters [13]. Moreover, growing interest in the IoT indicates the continuation of this trend beyond network requirements sustainable by existing infrastructure [13,25,27].

The Edge and Fog computing paradigms were introduced to address these challenges. Both paradigms are centered around sinking Cloud computing capabilities from the network core to the geographical edges (e.g., base stations, routers, access points, smartphones, etc.),

Table 2
List of popular ML models by learning domain.

Type	Model description
Supervised	**Regression** is a model trained to predict the value of some (continuous) variable given a number of (continuous or discrete) features [34].
	Nearest Neighbor models are non-parametric and do not require training. Instead, at inference time, the model's output is the value or the class of the known example nearest to the input by some distance metric [34].
	***k*-Nearest Neighbors (k-NN)** implements k nearest neighbor models in ensemble, merging their outputs by majority voting or averaging. With large training sets, k-NN models can achieve good capacity but do so at high computational costs [34].
	Decision Trees are, like the name hints at, tree-like models that comprise decision nodes and classification labels. During training, the model assigns features and thresholds to its decision nodes and labels to its leaf nodes. At inference time, the resulting class is identified by following decision nodes from the root to a leaf [35]. Unconstrained decision trees may grow arbitrarily large and, thus, become unwieldy. This may be avoided with special training algorithms [34].
	Random Forests implement several decision trees in ensemble, merging their outputs by majority voting or averaging. Individual trees can be trained either on different data subsets or only on poorly-classified data (i.e., boosting) to improve overall accuracy [35].
	Support Vector Machines (SVMs) are trained to map input data to an n-dimensional hyperspace–n being the number of features–to maximize the distance between disparate categories, essentially computing a distinguishing hyperplane. An input example is classified by which side of the hyperplane it is at [34,36]. SVMs inherently perform binary classification but may be used in ensemble to form multi-class classifiers [37].
	Bayesian Models are statistical models based on Bayes' theorem. Given a prior distribution and an observation, the corresponding posterior distribution can be computed [38].
	Neural Networks mimic biological brains, implementing networks of artificial, non-linear neurons and weighted synapses. Typical models comprise several *layers* of neurons: an *input* layer, a number of optional *hidden* layers, and an *output* layer. Non-linearity is introduced by passing the accumulated values of each neuron through an *activation function* like sigmoid, softmax, or ReLU. NN architectures show great variety from DNNs with multiple hidden layers, CNNs with neurons organized in convolutional filters, RNNs with integrated memory elements, to transformers with *attention* modules that assign context-specific *soft* weights to their inputs [39]. During training, the synaptic weights are updated, while they remain frozen during inference [34,40].
Unsupervised	**Clustering** algorithms group unlabeled data such that grouped data points are more similar to each other than to other groups according to some metric [34,41].
	Association Rules are frequently used in data analysis for understanding relations between features according to some metric of interest [41,42].
	Dimension reduction techniques aim at reducing data dimensionality with minimal information loss, essentially implementing a lossy compression of inputs [43–45]. Recently popular methods are based on *autoencoders*, i.e., models that comprise mirrored, but otherwise often identical, encoder and decoder NNs to learn a low-dimensional intermediate representation from which they can reproduce input examples [34,46].
Reinforcement	**Markov Decision Processes** are sequential decision-making models comprising a present state from which an action can be performed. The action's quality is evaluated based either on a model of the environment or on samples gathered from a physical/virtual environment directly. During training, high quality actions are rewarded (*reinforced*) and low quality actions are penalized. This ensures the model converges towards an optimal behavior within its environment [40].

performing computations closer to the end devices [32,47,48]. As such, they are intermediate to traditional local and Cloud computing, differing in where computation and storage are performed. CISCO coined the Fog computing term, referring to a paradigm in which *Fog nodes*–small-scale servers capable of managing tasks for many users simultaneously–are distributed around the internet [49]. Edge computing refers to a paradigm in which computational nodes are more numerous, smaller, and further distributed close to the end users [10]. Both paradigms revert to Cloud computing when tasks cannot be performed in their distributed devices. Owing to their similar distributed nature, we consider the two paradigms as one and provide an overview of them in Fig. 3.

Understanding the difference between the Cloud and Edge paradigms is crucial. In the present survey, we focus on compute-capable devices belonging to the bottom two categories – IoT and Edge – and consider resource-constrained IoT devices with little-to-no computational capabilities outside our scope. Our focus on Edge AI necessitates this distinction as related ML algorithms tend to be compute-heavy. Moreover, like prior work [21], we expect the benefits of AxC to be more pronounced in the related devices – autonomous cars, smart sensors and wearables, and positioning systems – than in the upper network layers' devices. Yet, despite vast amounts of Edge systems research, only little work focuses on its practical implementation. Supporting frameworks must implement primitives for local data collection and processing, wireless and secure data transmission, and task offloading, as well as a flexible Edge–Cloud server backend to manage the processed data [50,51]. Everything needs to be interconnected by well-defined Application Programming Interfaces (APIs) [52]. We assume the existence of such a framework with minimal overheads. Now, as a basic understanding of ML is necessary to follow the survey's technical sections, we briefly introduce it.

2.2. Machine learning for edge AI

The autonomous driving, smart sensing, and positioning systems that we consider in this survey are expected to grow increasingly intelligent, aggregating data from multiple sensors to navigate traffic, detect seizures, or accurately locate a device [53]. This kind of AI is commonly implemented using ML algorithms that can learn patterns – *probability distributions* – from data or actions and later be used to infer information from new data or act in new environments [34]. Despite being widely known and applied, we find it suitable to provide a brief introduction to the field's components and refer readers interested in details to more comprehensive texts [34,38,53].

ML algorithms are typically used following a two-stage flow consisting of an (offline) *training* phase, during which their weights are updated according to training data, followed by an (online) *deployment* phase, during which their weights are frozen and the models are used only for inference [34]. Table 2 outlines and summarizes popular ML techniques, of which especially variations of Neural Networks (NNs) are popular in the literature, as we will see later. We relate these models to their learning models: Supervised, Unsupervised, Semi-supervised, and Reinforcement Learning (RL), described below. In addition to these, we cover Federated Learning (FL), which is a distributed, online learning model popularized by advances in Edge computing and the growing need for adaptability [53].

Supervised Learning is an approach to train ML models with labeled data to correctly detect, classify, or predict its labels. During training, a model is adjusted to minimize a pre-determined *loss function* most often using some variant of gradient descent, for example, backpropagation in NNs. Backpropagation works by estimating the contribution of each neuron to the current loss and subsequently adjusting the weights according to the loss function's gradient, gradually approaching one of its minima [34,38].

Unsupervised Learning can be applied to synthesize new information from unlabeled data, useful to discover patterns and groupings by similarity or difference in examples without needing human intervention. Depending on the use case, models are adjusted during training according to a pre-determined metric of similarity or interest [34,42]. In systems with high-dimensional inputs, unsupervised techniques like autoencoding [44,45] can be used to pre-process data before they are passed to a supervised model.

Viewing ML models as ways to learn probability distributions over a dataset renders the lines between supervised and unsupervised learning rather blurry [34]. The fact that some ML models, like NNs, may be used both for supervised and unsupervised tasks further adds to this blur. The **Semi-supervised Learning** hybrid model also originates from this. It is motivated by the time-consuming, cost-intensive need for expert supervision to maintain the labeling quality of datasets [40]. Semi-supervised techniques combine characteristics from both supervised and unsupervised techniques, even extending upon their functionality enabling dynamic performance maintenance or adaptation to new data, for example through pseudo-labeling [40,54].

Reinforcement Learning distinguishes itself from the above techniques by being environment-driven rather than data-driven. An agent will attempt to learn the optimal behavior by being rewarded or penalized based on actions performed to its environment, i.e., its beneficial actions will be *reinforced*. It is particularly useful in complex robotics and autonomous driving scenarios [40].

Federated Learning is a relatively new field in ML, specifically designed for collaborative or joint learning in the distributed computing paradigms of today. Its related models are closely linked to those of supervised and unsupervised learning, but its training algorithms are federated, and a centralized server aggregates model updates rather than training data. This preserves system privacy but requires powerful end devices, e.g., autonomous cars and wearables [53].

Datasets used for ML tasks are often split (randomly) into two or three subsets: a *training set*, a *test set*, and optionally a *validation set* [34,38]. A model is trained on the largest of these subsets: the training set, and evaluated periodically during training on the validation set. Its eventual performance, however, is measured on the test set, meaning the model needs *capacity* not only to minimize its training error but also the gap between its training and test errors [34]. Too low capacity may lead a model to *underfit* and fail to minimize its training error, while too high capacity can make it *overfit* and fail to generalize to the test set [34]. Designing an ML algorithm, thus, means deciding on or determining an adequate capacity as well as a suitable set of hypotheses about the underlying statistical processes of the dataset [34].

AxC opportunities in ML are numerous. In addition to operating on noisy input data, most of the models outlined in Table 2 rely on compute- and memory-heavy linear algebra and non-linear activation functions. In many cases, such models are designed with a larger capacity than their targeted task requires [55,56]. The combination of these operations and the over-provisioning of capacity, particularly in NNs, makes models inherently resilient to small computational errors [57]. This trait can be exploited to reduce their demands, rendering them more suitable for execution at the Edge, as we will see later in the paper.

3. Approximate computing

In parallel with the development in the number of connected devices, the applications they execute have also changed. The vast amounts of produced data require massive computational efforts to process and aggregate. Such recent applications are user-centric and can provide results evaluated on their *acceptability* rather than their *correctness* [21]; illustrated in our context by, e.g., how well an ML model performs on the road, the data quality from a smart sensor, or the precision of a positioning device. As a result, these applications show inherent error resilience that can be exploited using approximation

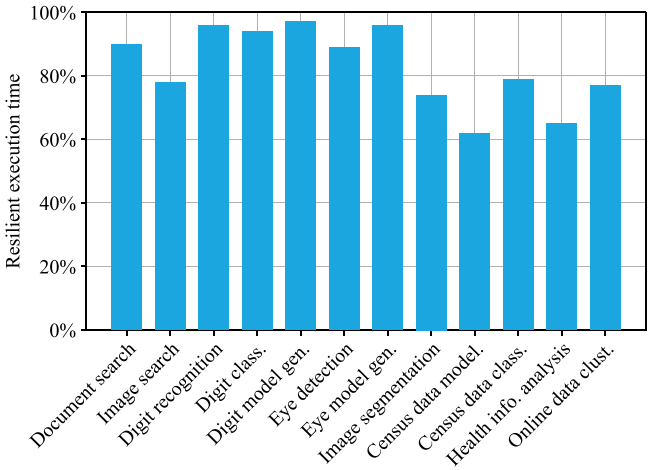


Fig. 4. Percentage of execution time spent on resilient computations (i.e., candidates for approximation) in some ML applications. Numerical values extracted for visualization from [31].

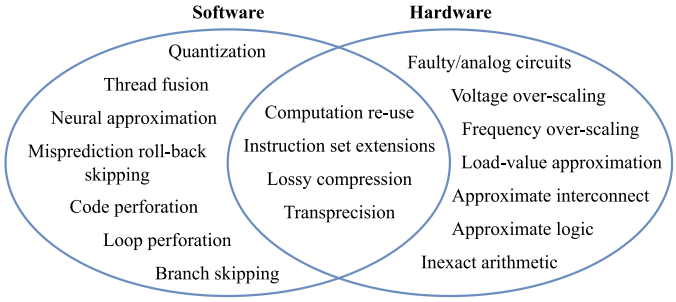


Fig. 5. Highlighted AxC software and hardware techniques. Techniques extracted from [9,21].

to achieve interesting trade-offs between energy, latency, and output quality. Recent research has revealed that many applications spend most of their execution time performing resilient computations that, with care, can be approximated with little-to-no quality degradation, as highlighted for some ML applications, in Fig. 4. Some of which may be executed in a smart sensor.

In the AxC domain, approximations are used to reduce applications' computational complexity, memory demands, or communication bandwidth [30]. Techniques range from generic circuit-level ones to application-specific algorithm- or software-level ones, the latter often bringing greater benefits. Fig. 5 outlines some generic techniques. We return to select hardware and software techniques later. At the software level, a developer can either introduce approximations that are hardware-agnostic (e.g., loop/code perforation, quantization, pruning) or ones that require hardware support (e.g., branch misprediction roll-back skipping) [9]. Similarly, a hardware designer can introduce static application-agnostic (e.g., inexact arithmetic) or application-driven approximations (e.g., instruction set extensions) [8,9], or dynamic approximations to facilitate Voltage Over-Scaling (VOS) and avoid any resultant timing or memory errors for more aggressive energy savings [58–60].

Managing approximations at run time is essential to satisfy quality constraints. Blindly applying AxC is simple but may lead to under-utilization or inadequate results [8,29]. Being too cautious leads to the former and results in sub-optimal savings, while being too generous leads to the latter and results in too large quality degradation. Striking a balance between the two is difficult and demands either developer intervention or run-time quality control in hardware [61,62]. Alternatively, the error impact of certain techniques can be established

Table 3
Comparison of the papers on inexact arithmetic.

Type	Architecture	Adaptive?	References
Adder	Non-segmented	No	[67–70]
		Yes	[71,72]
	Segmented	No	[73–76]
		Yes	[77]
Multiplier[a]	Logarithmic	No	[78]
	Tree	No	[79,80]
	Array	No	[67,81–86]
		Yes	[87,88]
MAC		No	[89–91]
		Yes	[92]

[a] We exclude three works [93–95] that present libraries of multipliers rather than individual designs.

Table 4
Comparison of the papers on circuit-level techniques.

Type	Parameter	Adaptive?	References
Precision scaling[a]	Truncation	Yes	[58,98–100]
	Voltage	Yes	[100,101]
	Refresh Rate	Yes	[59]
	Strategy	Level	
Approximate synthesis[b]	Pruning	HLS	[102–104]
		Gate	[105,106]
	Inexact arithmetic	HLS	[107]
		RTL	[108,109]

[a] We exclude two papers [60,110] that survey and study precision scaling in general terms.
[b] Circuits generated through approximate synthesis are by default not adaptive, so we use a different third column to categorize the associated work.

$$s = (\bar{a} + b) \cdot cin$$
$$cout = a$$

cin \ ab	00	01	11	10
0	00	00	10	10
1	01	01	11	10

(a) The 4th approximate mirror adder of [67]

$$p_0 = a_0 \cdot b_0$$
$$p_1 = (a_0 \cdot b_1) + (a_1 \cdot b_0)$$
$$p_2 = a_1 \cdot b_1$$
$$p_3 = 0$$

a_1a_0 \ b_1b_0	00	01	11	10
00	0000	0000	0000	0000
01	0000	0001	0011	0010
11	0000	0011	0111	0110
10	0000	0010	0110	0100

(b) The approximate 2 × 2 multiplier of [70]

Fig. 6. Boolean equations and Karnaugh maps. Red digits indicate errors. Outputs are in order of significance.

with formal or probabilistic model checking tools preemptively [63]. Formal tools can be used to compute characteristics like worst-case error [64], while probabilistic tools use randomized simulation to infer with some degree of confidence the presence of some given properties in a system [63,65,66]. Both these types of analyses are relevant at design time when various approximate operating points of a system must be determined. Our focus on adaptive AxC also involves a particular interest in controlling approximations according to changing application requirements [29] and varying operands during execution [58]. We present some techniques for this later.

As a broad spectrum of techniques exists within the AxC paradigm, we limit this introduction to the commonly applied ones relevant to our surveyed application domains. We follow a bottom-up approach: first, we consider circuit-level techniques; then, their integration into various computing architectures; and last, some application- or algorithm-level techniques.

3.1. Circuit-level techniques

We first focus on two prevalent categories of techniques: inexact arithmetic and circuit-level approximations. Depending on their adaptability and the target application's quality requirements, these techniques are generally applicable. More details on them are available in other surveys [8,9,21]. We summarize the covered work in Tables 3 and 4.

3.1.1. Inexact arithmetic

Functional approximation of arithmetic units is a particularly popular and active area of research. This branch mainly deals with the design of adders and multipliers at different abstraction levels, i.e., transistor, gate, or register-transfer level. Some notable approximate adders include speculative adders, segmented adders, and approximate Full Adders (FAs). Significant research effort has also been spent on approximating multipliers – the most power-hungry components in ML accelerators. We focus on these two units and their combination: the Multiply-Accumulate (MAC) unit that is essential for NN accelerators.

Adders play a crucial role in computing systems. Demands for high speed and energy efficiency have promoted the design of approximate adders that save area and power consumption and increase performance at the expense of accuracy [69]. Adders calculate the sum of two binary numbers and come in different layouts, the two most common are the Ripple-Carry Adder (RCA) and the Carry-Lookahead Adder (CLA) [96,97]. Briefly, an n-bit RCA cascades n FAs, propagating the carry from each FA to the next, giving it a linear delay. A CLA instead computes and propagates carries in slices of bits, operating these modules in parallel to produce a sum, giving them logarithmic delay yet a considerably greater area than an RCA.

Many approximate adders have been proposed. The simplest ones employ approximate FAs in the Least-Significant Bits (LSBs) of an RCA to reduce the carry chain's length and area [7,67,68]. Examples of such include substituting FAs with simple OR-gates [67] and various logically inexact FAs [73] or so-called *mirror adders* [69], illustrated by the Karnaugh map in Fig. 6(a). Alternatively, an adder can be segmented into several smaller adders that operate in parallel. Exact Carry-Select Adders and CLAs already integrate this technique, and their approximate counterparts cut their carry chains [74]. Speculative adders generalize this segmentation to predict each sum bit from its $k < n$ less significant bits [73,75]. The majority of these adders are designed for Application-Specific Integrated Circuit implementation and integrating them on Field-Programmable Gate Array (FPGA) does not necessarily bring comparable savings [111]. There are, however, also FPGA-specific designs [70,76]. These adders reduce latency by cutting their carry chains and later reducing the arising errors by feeding Look-Up Tables (LUTs) with duplicated inputs.

Within our scope of adaptive AxC, some authors have explored complementing the aforementioned adders with extra logic to select the degree of approximation. This concept may be integrated into an RCA by inserting multiplexers at each carry, enabling fine-grained quality control at the expense of high overheads [71]. Reduced-overhead alternatives to this design include the RCA of [72], which dynamically selects the number of bits used in carry prediction, and the CLA of [77], which approximates every fourth carry through power gating.

Multipliers are another crucial component in computing systems. As for adders, several different approximate designs exist, including ones that apply the speculative adders described above [75]. However, using adders directly to implement multipliers may be inefficient in trading off accuracy for area and energy savings. Instead, multiplication is often

implemented by a cascaded array of adders that reduce partial products into the final results. Approximations often target reducing the critical path of this array by truncating some number of LSBs [67,81], by approximating the adders [95], or by reducing the number of partial products through (hybrid) high-radix encodings [82]. Some authors have also explored using genetic algorithms to generate libraries of inexact multipliers [93].

Other designs implement combinations of the above techniques. One work notices that rounding operands to their nearest power-of-two means multiplication turns into simple shift operations [78], while another performs multiplication recursively based on simple inexact 2×2 multipliers [79], whose logical behavior is given in Fig. 6(b). Some designs integrate error compensation [67,81] or input pre-processing [83] to reduce output errors, while others aggressively approximate both partial product generation and reduction to reduce energy consumption further [84,85]. Again, implementing these multipliers on FPGA does not guarantee any savings [94]. In recognition of this, some work constructs partial product reduction with approximate compressors [86], while others utilize inexact 4×4 multipliers or partial product generation circuits that map well to the fabric's primitives [80,94]. The latter is even collected in a library similar to that of [93].

In addition to the above, there also exist run-time configurable inexact multipliers. Two such designs are proposed in [87] and [88]. The multiplier of [87] can perform either one wide multiplication or two narrow multiplications, both of whose results are inexactly compressed, while that of [88] combines inexact compression with dynamic input truncation to enable greater control of output quality.

MACs are usually constructed by a multiplier and an adder fused to maintain precision. Therefore, integer designs can be approximated both in their adders and multipliers using units like the above, while floating-point ones require being conservative, particularly concerning the exponent logic. Four publications explore integer MACs: one that approximates accumulation by early termination [112], one focused on customizability in FPGAs [89], one using sign prediction and a special input encoding [90], and another that supports integer operations by dynamically disabling the exponent logic of a floating-point MAC [92]. Another paper simplifies a floating-point MAC by removing its overflow and underflow circuits and approximating its mantissa multiplier [91]. Most papers in this category target CNN acceleration.

3.1.2. Circuit-level approximations

For some applications, *static precision scaling*, like quantization, has already been applied in commercial hardware such as [113]. Though the benefits of this are clear, one must be conservative to guarantee output quality when applying static techniques. This has brought attention to adaptive *dynamic precision scaling* techniques that can tailor computational precision to the temporal changes in an application's error resilience [98,114]. As outlined above, this style of adaptability is possible to implement in inexact arithmetic, e.g., with selective error compensation [67,73,81], disabled carry propagation [71,72,77], or early-terminated accumulation [112]. However, these techniques fail to exploit the benefits of low-level circuit characteristics: voltage and frequency.

The voltage and frequency knobs are commonly adjusted in Dynamic Voltage and Frequency Scaling to maximize energy efficiency when the computational load is low and performance when it is high [115,116]. Still, they may analogously be used for approximation to adjust circuit precision [98]. Over-scaling the voltage or frequency will significantly increase the risk of timing failures in a circuit – some of which other approximations can counter – while potentially leading to vast energy savings [58,60,117].

Several papers explore VOS. Their proofs-of-concept are mainly floating-point arithmetic units compared with statically truncated ones to highlight potential benefits [58,98]. The approach appears to be mostly used for NN acceleration, for which it involves keeping track of *good-enough* precision and adjusting the word length of weights and activations [99,100]. In some instances, the approach is taken to the extreme near-threshold case at which significant reductions in static *and* dynamic power can be achieved at the cost of longer computation time [101,117]. Unlike inexact arithmetic, voltage over-scaling can be applied to FPGAs, though its effects may be difficult to predict and vary greatly across a chip. Errors may be mitigated by mapping sensitive logic to the most fault-resilient parts of the FPGA [110]; a strategy similar to that, which is often applied to reduced refresh-rate Dynamic RAM (DRAM) [59].

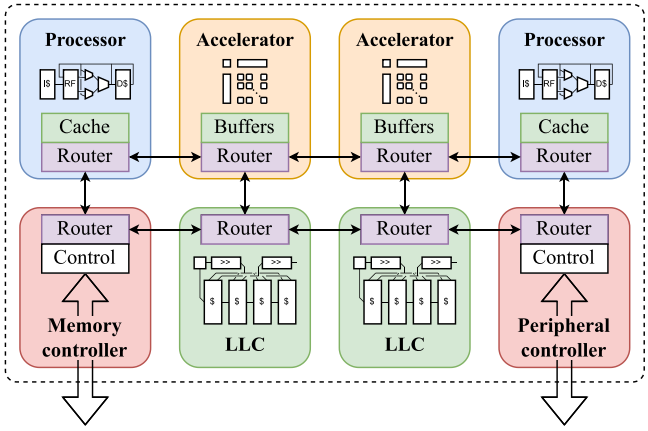


Fig. 7. Abstract illustration of a heterogeneous SoC with two processor cores, two accelerator cores, a distributed last-level cache, a memory controller, and a peripheral controller; interconnected by a mesh-style NoC.

Earlier we described how, e.g., inexact arithmetic units can decrease the area and power consumption of a system. However, in some cases, it is unnecessary to limit the Design Space Exploration (DSE) only to these units. Instead, to broaden their scope and avoid costly iterative simulation and synthesis runs, a large pool of work considers introducing approximations during synthesis [107]. Some authors propose integrating approximate circuits characterized at design time into High-Level Synthesis (HLS) flows [102–104], while others introduce approximations at the Register-Transfer Level [108,109] and even at the gate level [105,106]. While a more fine-grained granularity can lead to greater savings, it has synthesis time overheads [106].

3.2. Architectures

Having introduced circuit-level AxC techniques, we turn our attention to compute architectures. We distinguish between general-purpose and application-specific architectures, though the techniques described in Section 3.1 are often equally applicable in either [21]. The application domains we consider often demand a high energy efficiency that can be only achieved with bespoke accelerators. This is particularly true in ML and image and video processing, which are crucial in autonomous driving and smart sensing, and in Digital Signal Processing (DSP) tasks, which are the cornerstone of positioning. Therefore, we focus mainly on architectures relevant to these domains. We again summarize our findings in Tables 5 and 6.

3.2.1. General-purpose architectures

As described in Section 2.1, extremely constrained Edge devices may not permit the implementation of several different application-specific accelerators [13]. Instead, they implement only the absolute essentials: a highly optimized General-Purpose Processor (GPP) flanked by a few accelerators [118]. Such systems are often collected as a System on Chip (SoC) and interconnected by a Network on Chip (NoC), see Fig. 7. AxC may be applied to all these parts, as we will now detail.

Table 5
Comparison of the papers on general-purpose architectures.

Type	Target	Technique	References
Processors	Arithmetic	Truncation	[119]
	Cache	Cache line overlapping	[121]
	Core	Speculation	[124]
		Memoization	[120]
		Neural approximation	[122,123]
Generic accelerators	PE	Inexact arithmetic	[126,127]
		Voltage over-scaling	[29]
Networks-on-chip	Router	Voltage over-scaling	[128]
		Dropping, compression	[129]
	Network interface	Dropping, compression, etc.	[130]

Table 6
Comparison of the papers on application-specific architectures.

Sub-domain	Application	Technique	References
Machine learning[a]	NNs[b]	Truncation	[57]
		Inexact arithmetic	[136]
	CNNs[b]	Voltage over-scaling	[100]
		Dot product encoding	[137]
		Early termination	[138]
	Hyperdimensional computing	Approximate similarity	[139,140]
	SVMs	Inexact arithmetic	[141]
	k-NN	Partitioning	[142]
Multimedia	DCT	Inexact arithmetic	[143]
		Voltage over-scaling	[144,145]
	HEVC	Inexact arithmetic	[146]
		Early termination	[147]
	Edge detection	Inexact arithmetic	[148]
Signal processing	General	Voltage over-scaling	[118]
	WT	Distributed arithmetic	[149]

[a] Four works on k-NN and decision tree accelerators do not explicitly apply any AxC techniques but are included in the survey for completeness [150–153].
[b] All works on NN and CNN accelerators utilize quantization.

GPPs generally require some form of instruction set changes to support AxC. Yet with such changes implemented, developers can utilize inexact arithmetic units and approximate loads/stores, as explored by Ndour et al. [119]. Other work proposes using *memoization* and approximate caches [120,121]. A memoization module may keep track of inputs and outputs of blocks of instructions to skip their execution if similar inputs re-appear, while the cache may exploit the similarity between cache lines to effectively increase cache size. Others suggest approximating compute-heavy kernels by small NNs in custom accelerators [122,123]. Lastly, it is possible to approximate the processor's control flow by, e.g., selectively disabling roll-back on a branch or load-value misprediction in out-of-order cores [124]. The common notion is that approximations should target several instructions, memory operations, or control flow to be effective; in line with observations in [125].

Generic accelerators may also integrate approximations while focusing on striking a good balance between reconfigurability and its overheads. Most surveyed designs resemble Coarse-Grained Reconfigurable Arrays (CGRAs) suitable for accelerating compute-heavy kernels. Yet, despite many similarities, they vary in how approximations are applied: one applies *effort scaling* by combining VOS and clock gating-based truncation [29], and another applies dynamic operand truncation [58]. Others integrate inexact arithmetic units and either adjust the error correction applied to one or select between multiple units at run time [126,127]. Both techniques can induce great energy savings and offer a wide range of options for run-time adaptation to be established at design time.

NoCs interconnect processors, accelerators, and various controllers (see Fig. 7), carrying packets of data or synchronization messages between pairs of nodes. Being either wired or wireless, they also offer opportunities for approximation, including selective VOS of links [128], adaptive packet truncation or dropping [129], approximate locks or lock coarsening, and skipping low-impact updates to shared memory locations [130]. This not only reduces network traffic (and contention), but it can also significantly speed up particularly parallel applications at the expense of reduced synchronization with varying degrees of output error [130]. Similar strategies can be applied to networks on a larger scale, though we do not cover such techniques in the present survey [11].

3.2.2. Application-specific architectures

Accelerator architectures are often costly in area and power consumption, meaning a certain level of utilization is demanded to justify their integration. However, if well-utilized, they offer much higher energy efficiency and exploit approximation opportunities in their target applications better than general-purpose architectures [21].

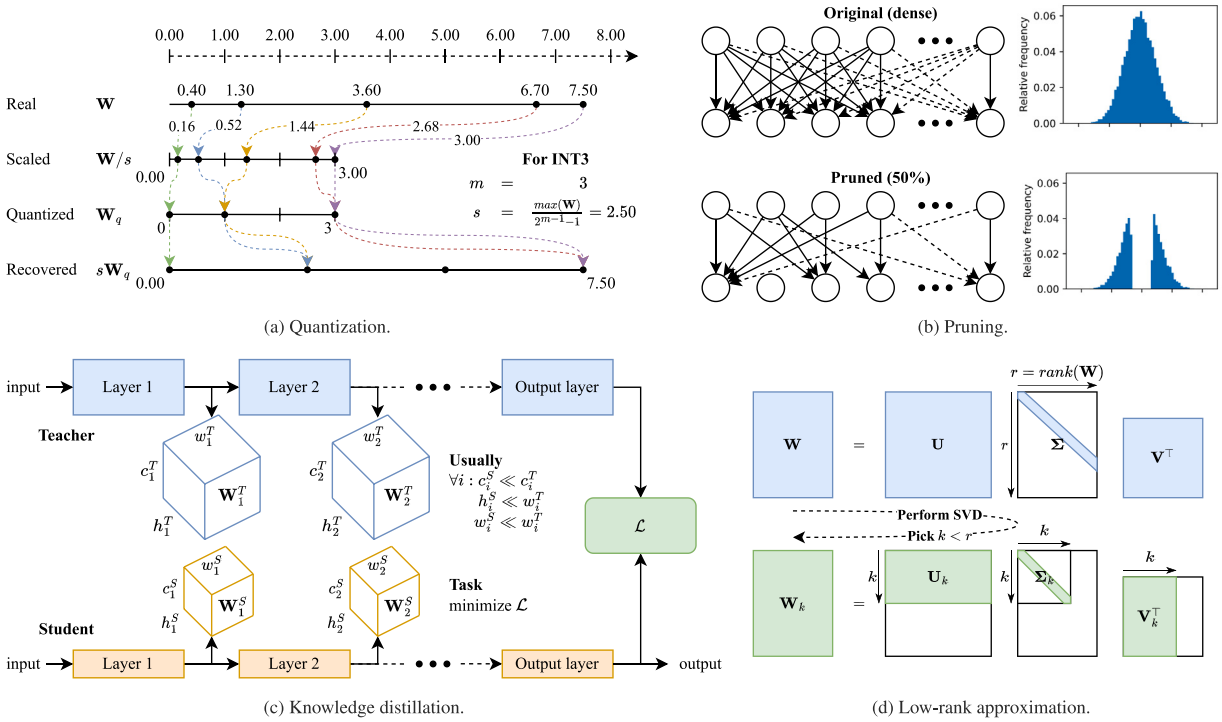
ML is, by far, the most popular application for which approximate architectures are implemented. This is highlighted by work such as [20] and its relevance to both autonomous driving (e.g., in Simultaneous Localization and Mapping (SLAM) [131]) and smart sensing [132] (e.g., in emotion detection [133]). Due to its computational demands, most ML research is done using floating-point arithmetic on GPP or Graphics Processing Unit architectures but quantized and implemented with, e.g., 8-bit integer operations in resource-constrained devices [134,135].

NNs are the main focus of many papers. For example, one work [136] proposes a highly efficient keyword spotting accelerator employing binary-weighted NNs and custom delay-based analog multipliers. Another [57] describes the entirety of IBM's research and development of a general DNN accelerator (like [29]) with corresponding tool flows, which enable both software- and hardware-level approximations. Another three papers propose accelerator architectures for CNNs, exploiting kernel size reductions and inexact arithmetic [138], a custom bit-level dot product implementation [137], and layer-wise quantization combined with VOS [100]. The latter motivates its design by targeting resource-constrained Edge devices.

Despite their prominence in related work, NNs are not the main focus of all papers: some instead implement SVMs or *hyperdimensional* computing with inexact arithmetic units or in-memory architectures [139–141]. To minimize accuracy degradation arising from approximations, some propose re-training networks with approximation awareness [57,137].

When ultra-low power operation is required, NNs and even SVMs may be too complex. Therefore, another set of papers focuses on low-complexity implementations of k-NN and decision tree models on FPGA. In [150], the authors propose two highly parallel k-NN accelerator cores, targeting models with many narrow example vectors and ones with few wide vectors, respectively. The former of these designs is made dynamically reconfigurable in [151], while the accelerator of [142] integrates search optimizations specific to a Light Detection and Ranging (LiDAR) localization application. The accelerator of [152] stores many parallel per-class decision trees in block RAM internal to designated cores and iterates over them before outputting final inference results. The authors of [153] instead optimize the decision trees themselves to be shallow and store them in distributed RAM within the FPGA's LUTs.

Image and video processing and other multimedia applications are frequently found in autonomous driving and smart sensing systems [95,132,154] and are often approximated in custom hardware. Example designs include high-efficiency Discrete Cosine Transform implementations for which Almurib et al. [143] describe three design steps: (1) selecting a low-complexity algorithm, (2) filtering out high-frequency components, and (3) employing approximations in, e.g., arithmetic. Others employ significance-driven approximations, showing that it is possible to facilitate VOS with minor quality loss but ensuring that significant operations are executed correctly, while the rest are approximated [60,144,145]. Others implement various edge detection



(a) Quantization. (b) Pruning.

(c) Knowledge distillation. (d) Low-rank approximation.

Fig. 8. Overview of the four ML-focused AxC techniques: (a) weight quantization to 3-bit signed integer with real-valued scaling factor (only positive numbers shown), (b) pruning the 50% smallest absolute value weights, (c) knowledge distillation from a teacher to a student model, and (d) low-rank approximation using SVD. The illustrations are simplified and do not take retraining into account.

algorithms with reduced-complexity filters [148]. For video processing, most publications explore architectures for approximating encoding and decoding steps to and from the HEVC (or H.265) format, particularly using inexact arithmetic units [146,147].

DSP tasks, as we have seen, are relevant both in wearables and in positioning. Keeping our focus on bio-related applications, examples of such architectures include a SoC-style design that incorporates a CGRA for vector operations with an array of small GPP cores [118] and different implementations of the discrete Wavelet Transform (WT) with error compensation [149]. At the time of writing, we are not aware of any architectures for positioning that exploit AxC techniques, yet we expect many hardware techniques to be portable to these algorithms too.

3.3. Application- and algorithm-level techniques

Though the hardware AxC techniques of Sections 3.1 and 3.2 can bring significant benefits, the greatest gains may often be achieved by applying high-level approximations tailored to a particular application. With the prevalence of NN models in both autonomous driving and smart sensing, as we will see later, we focus on techniques relevant to these algorithms here, while we cover algorithmic approximations specific to particular applications in the following sections.

As we will see later, NNs suffer from scaling issues that may prevent their efficient implementation and execution in resource-constrained Edge devices. Motivated by this, we consider model compression or *sparsification* techniques that can reduce model size [155]. Four frequent examples are quantization, pruning, knowledge distillation, and low-rank approximation. We illustrate their functionality in simplified form in Fig. 8 and briefly introduce them below. We summarize our findings in Tables 7, 8, 9, and 10.

3.3.1. Quantization

Quantization is likely the most commonly applied AxC technique for NNs. It involves reducing the bit-width of weights and/or activations and transforming floating-point operations to fixed-point equivalents in NNs. First shown to be a viable strategy in [159], subsequent work has demonstrated reductions of weights and/or activations to 8-bit [165], 4-bit [165], 2-bit [156,157], and even binary [158] widths. Different strategies either uniformly assign the same number format to all network components of the same type (e.g., weights) [45,160] or non-uniformly select optimal formats for different components [169,170]. Alternative strategies apply different quantization levels to individual layers in DNNs [171], individual channels in CNNs [166,167], or groups of network components [162–164].

Fig. 8(a) illustrates weights-only quantization to 3-bit signed integers. First, the real-valued weight matrix is scaled by a factor s to bring its elements within the target format's range. Second, the scaled elements are rounded, or quantized, to the precision of the target format. This process can be applied after training – *post-training quantization* – but doing so may degrade model performance greatly as errors accumulate in the forward pass [161]. Instead, the quantization effects can be modeled [55,161] or even trained [168] during *quantization-aware training*. Models generally retain a higher accuracy

Table 7
Comparison of the papers on quantization.

Type	Components	Precision[a]	References
Uniform	Weights only	2-bit	[156,157]
		1-bit	[158]
	Weights and activations	8-bit	[159]
		4-bit	[160,161]
		2-bit	[45,55]
Non-uniform	Weights only	4-bit	[162]
		1-bit	[163,164]
	Weights and activations	4-bit	[165–168]
		2-bit	[169,170]
		1-bit	[171]

[a] We report the minimal permissible precision for non-zero model components in the surveyed works.

Table 8
Comparison of the papers on pruning.

Type	Model type	Target	References
Structured	CNNs	Filters	[56,172,173]
		Filter weights	[32]
		Neurons	[174]
Unstructured	CNNs	Weights	[174]
	Transformers	Weight rows	[39]

Table 9
Comparison of the papers on knowledge distillation.

Optimization	References
Maximum mutual information in activations	[175]
Minimum Frobenius norm on quantized weights	[162]
Minimum squared norm on international correlation	[176]

We exclude one work [177] that presents a comparison of knowledge distillation strategies.

Table 10
Comparison of the papers on low-rank approximation.

Strategy	References
Truncated singular value decomposition	[178]
Depthwise convolution decomposition	[179]
Learned sparse "sketches"	[180]
Tucker decomposition	[181]
Canonical polyadic decomposition	[43]

when quantization is applied in a non-uniform, fine-grained style, but identifying such optimal configurations is computationally demanding, albeit performed only once during training [163].

3.3.2. Pruning

Pruning is a common technique applied in DNNs and CNNs involving the removal of low-significance or redundant model components, i.e., either weights [39] or neurons [56,172,173], known as unstructured and structured pruning [32,174]. Fig. 8(b) exemplifies unstructured pruning of the 50% least significant weights of a fully-connected layer. We use the absolute value as our metric of significance and show the effects of pruning in a histogram. The degree of approximation correlates well with how aggressively these components are removed.

Fully exploiting the benefits of pruning may require dedicated hardware support for sparse matrix operations. In the first category, Yang et al. [56] propose a method for CNNs in which filters are pruned *softly* thus enabling them to be updated until the model converges on some filters being consistently near-zero. Their scheme achieves greater reductions in computations incurring lower accuracy loss than competing schemes, including that in [172]. The work of Xiao et al. [174] is in the second category and the authors propose an automatic scheme for pruning neurons based on *auxiliary variables*, or indicators, that help select the most appropriate network structure during training like Jin et al. [168] do for quantization. This allows them to achieve high compression ratios and fewer computations than other state-of-the-art schemes.

3.3.3. Knowledge distillation

The core idea of knowledge distillation is to transfer knowledge from one model (the *teacher*) to a smaller model (the *student*) [162,175]. The approach is orthogonal to reducing a model's size by cutting away redundant information through quantization or pruning. It is a two-step process: first, training a large model over a complete dataset; and second, training a small model over a subset of the data while minimizing some metric of difference in knowledge between pairs of layers in the two models. This metric may be the correlation [176] or mutual information [175].

Fig. 8(c) shows an abstract representation of this concept applied to a pair of CNNs of which the student is significantly smaller than the teacher. In practical scenarios, there may be a metric of difference ℓ per layer or even channel [176], but for simplicity, we show only one between the networks' outputs. Extreme versions of knowledge distillation aim to directly transfer soft probabilities from one model to another, reducing the time required to retrain the small model to achieve satisfactory accuracy. Such schemes, however, have yet to achieve similar compression ratios as the above techniques [177].

3.3.4. Low-rank approximation

The large matrices that constitute NNs can also be reduced in size through factorization [178] and decomposition [179,180]. In practice, this may be implemented through Singular-Value Decomposition (SVD) [179,180], Tucker decomposition [181], and Canonical Polyadic Decomposition [43], suitable for reducing matrix size by removing low-significance sub-matrices not only in NNs. Without diving into mathematics, Fig. 8(d) illustrates this technique using the SVD on a weight matrix. Guo et al. [179] focus on CNNs and propose an algorithm that transforms models with regular convolutions to equivalents with simpler, depth-wise convolutions that need no re-training. The resulting models incur some accuracy loss but require much fewer computations. Part of this accuracy loss can be restored or prevented through sparse regularization during training [182].

Applying these techniques statically may bring some benefits but might also lead to insufficient model accuracy. This can be mitigated to some extent through approximation-aware training [161] or simple iterative re-training after approximation [183]. State-of-the-art schemes combine several of the above techniques; e.g., pruning, quantization, and compressive coding in Han et al.'s *Deep Compression* [48].

As can be seen from the above, there are vast opportunities for applying different AxC techniques across the system stack. Moreover, these techniques can be utilized in ways that enable adaptivity with relatively limited overheads through, e.g., error compensation in inexact adders [68,72,77], dynamic run time truncation in inexact multipliers [87,88], or run time tuning in various architectures [29,126] or applications [164,184]. In the following discussion, we will refer back to these sections when relevant. In tables, we will also only report the nine most commonly applied AxC techniques: quantization, pruning, knowledge distillation, low-rank approximation, lossy compression,[1] various algorithmic approximations, inexact arithmetic, voltage over-scaling, and approximate synthesis.

4. Applications and algorithms

With the fundamentals of AxC in place, we now turn our attention to the three selected, emerging application domains: autonomous driving, smart sensing and wearables, and positioning. All three domains comprise algorithms whose energy efficiency may be improved through approximation. Contrary to other work [21], we do not consider AI or ML as standalone application domains but rather as enablers of, especially, autonomous driving and smart sensing and wearables. Despite apparent overlaps between the three domains, we cover them separately in this section. For each domain, we describe relations to our overall focus on IoT and Edge AI, outline underlying algorithms, and survey papers that apply AxC to them. When relevant, we provide references to background (Section 2) and AxC techniques (Section 3).

[1] Though we do not explicitly introduce lossy compression as an AxC technique, we find that it is frequently applied in combination with other techniques.

Table 11
AxC techniques applied in the included papers that explicitly list autonomous driving as a use case. References sorted by publication year.

AxC Tech.	Quantization	Pruning	Knowledge Distillation	Low-rank Approx.	Compression	Algorithmic Approx.	Inexact Arithmetic	Voltage Over-Scaling	Approx. Synthesis
[48]	✓	✓			✓				
[179]		✓		✓					
[174]		✓							
[76]							✓		
[16]	✓	✓		✓	✓			unexplored	unexplored
[142]						✓			
[131]						✓			
[176]			✓						
[177]			✓						
[20]	✓								
[88]							✓		

4.1. Autonomous driving

As vehicles are extended with several simple or complex services to assist drivers and easier mobility, they gradually become more like other IoT devices [185], requiring either local or low-latency offloaded data processing. Services such as braking assistance, lane departure warning, adaptive cruise control, and Global Navigation Satellite Systems (GNSS)-based navigation have seen wide adoption by automotive manufacturers [186], some are even already instances of Edge AI, relying on ML techniques [154]. Other services, such as High-definition Map and onboard internet-supported audio and video streaming applications (e.g., Spotify and BMW's iDrive), are still undergoing development [187]. Combining these services and extending them further is expected to enable fully autonomous driving. We start by reviewing the levels of autonomy laid out by the Society of Automotive Engineers:

- **L1 No Automation**: The driver performs all driving tasks.
- **L2 Driver Assistance**: Driving tasks are performed by the driver with little input from vehicle sensors and driving assistance features.
- **L3 Partial Automation**: An in-vehicle compute unit can perform some driving tasks – adaptive cruise control, emergency braking, etc. – based on environment sensing, but the driver is required to maintain control and monitor vehicle surroundings.
- **L4 Conditional Automation**: An in-vehicle compute unit can perform all driving tasks, but the driver must be able to take control of the vehicle on demand.
- **L5 High Automation**: An in-vehicle compute unit can perform all driving tasks and negotiate with other vehicles and infrastructure *under certain conditions*. The driver can take control of the vehicle.
- **L6 Full Automation**: An in-vehicle compute unit can perform all driving tasks and negotiate with other vehicles and infrastructure *under all conditions*. The driver can, potentially, take control of the vehicle.

As expected, the computation and networking requirements grow with increasing autonomy. At present, Level 3 autonomy exists and has been regularly tested by manufacturers such as Tesla, Volvo, and Volkswagen [186]. The progression of these manufacturers' systems is towards the integration of real-time sensing with statistical learning algorithms to replicate or imitate the driving process carried out by humans, aiming either to assist human drivers or replace them altogether [186,187]. Commonly, the systems rely on sensor data originating from cameras, LiDARs, radars, GNSS receivers, and networking devices, which must be processed to enable intelligent decision-making, illustrated in [23, Fig. 13]. As shown, the autonomous vehicle data-flow consists of three modules: a SENSE module comprising the aforementioned sensors, a THINK module implementing the algorithms necessary to process sensor data to enable intelligent decisions on which the ACT module can base its actuation [188,189].

According to several reports, the sensor data required for autonomous driving will constitute 30 to 40 terabytes per day [190]. This mere volume puts great pressure on onboard data management services. It also implies a need for powerful processing pipelines. Moreover, connected vehicles are projected to rely on frequent communication to share data with other vehicles. The driving services might also incur competition over computing and networking resources with advanced, connected multimedia services [191]. In general, infotainment systems have evolved from simple radios to Cloud-driven multimedia platforms that allow for audio and video streaming. This evolution is likely to continue as vehicles become more autonomous, keeping their drivers less occupied. Maintaining a low energy consumption of these systems is crucial for maximizing the primary goal when designing electric cars: driving range. Maximizing the range also ensures minimal carbon emissions and energy costs. However, current predictions point in another direction: energy consumption may increase from 750 Wh to 2000 Wh per 100 km driven in the minimal, efficient networking scenario that assumes short-range communication and prioritizes essential services over onboard multimedia processing [191].

4.1.1. Driving services at the edge

Naturally, the increasing amounts of data and computations required for autonomous driving raise the question of whether processing is more efficiently done in the vehicle or offloaded to external servers. This avenue enables interesting trade-offs between available computing resources, communication demands, and achievable latency. While the Cloud provides vast resources, it is infeasible to execute sensitive services within it due to its bandwidth and latency constraints [192,193]. Executing these services at the Edge appears as a reasonable compromise retaining the benefits of offloading while providing fewer computational resources. This concept is explored by Zhou et al. [53] who list its many benefits, e.g., low latency, privacy preservation, and energy efficiency. Computations can be distributed between all participating devices, including other vehicles that have under-utilized resources [194,195].

Regardless of the platform targeted, offloading introduces some requirements on the systems involved. Firstly, the vehicle needs to implement wireless networking devices with suitable protocols, as highlighted by Shi et al. [16]. Secondly, the distributed nature of autonomous vehicles demands an equally distributed, i.e., decentralized control structure. Several frameworks already implement this based on function virtualization: Feng et al.'s AVE [196] and Tang et al.'s π-Edge [197] are initial examples of such, implementing real-time task scheduling and resource allocation algorithms. More recently, Tang et al. proposed LoPECS [198] that extends the communication functionalities of π-Edge. Santa et al. describe another two frameworks that virtualize available computing resources, including other vehicles, and consider them in a multi-access Edge computing layer [194,195]. They report improved device access latency and overall system speedup. Ibn-Khedher et al. [199] consider a similar scenario but

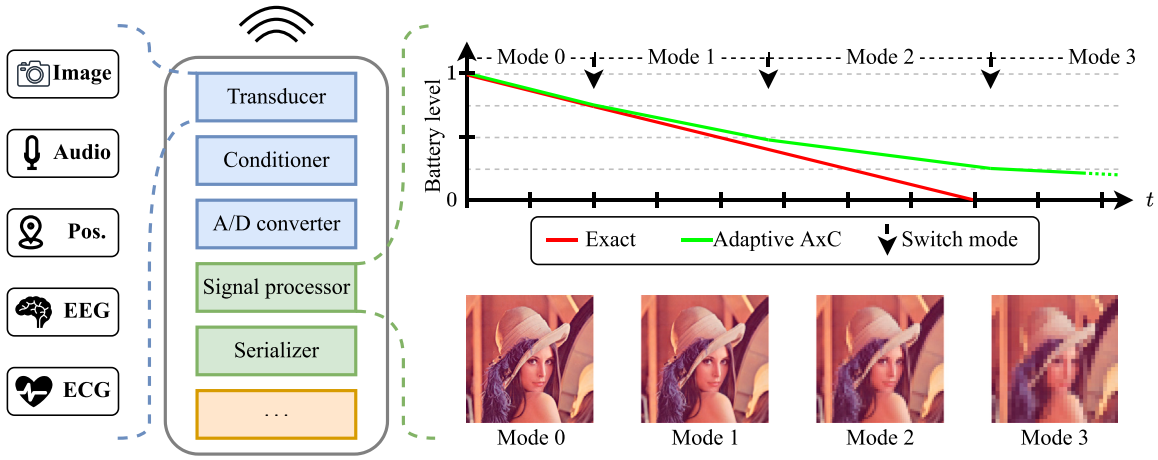


Fig. 9. Illustration of (left) a generic smart sensor's internals, some transducers, and potential built-in actuator, and (right) an example of how adaptive AxC enables trading off output quality for energy savings.

offload non-critical services to the Cloud, using reinforcement learning for offloading decision-making, as described in Section 2.2. As energy consumption constraints remain tight, some services may benefit from AxC; we consider this direction later.

4.1.2. Enabling autonomy

In addition to SLAM algorithms for environment modeling [154], the SENSE-THINK-ACT model indicates that modern autonomous driving mostly relies on AI, particularly models from ML and deep learning. Several such models have been proposed as replacements for the traditionally used algorithms for vision-based detection [167], forward collision warning [200], and path planning and control [201]. Various types of NNs – including DNNs [200], CNNs [41,167], and RNNs [201] – are most frequently used in the reviewed literature and are already implemented in existing **Level 3**-autonomous vehicles, such as Tesla's [201]. These models are expected to be key to achieving full autonomy, but their computational requirements pose a significant challenge if they are to be performed at the Edge [167].

Depending on the ML models used and available computing and energy resources, the autonomous driving models may be trained *in-vehicle* in a federated style. Enabling such features requires a deep understanding of, e.g., driving patterns such that model training does not take away from the vehicle's main purpose: driving. Adaptive approximation could be an interesting strategy to achieve a good balance in this aspect, for example, by allocating fewer resources to driving tasks in simple scenarios, such as highway driving in sunny weather. Moreover, an FL system is complex not only because its models may vary across vendors or vehicle models with different sensors and compute resources available, but also because of unpredictable networking conditions for communicating models [191,202]. Nonetheless, the concept of dynamic model adaptation is appealing.

Given the prevalence of NN models and anticipating the possibility of in-vehicle training of such, the model compression techniques described in Section 3.3 are highly relevant for autonomous driving applications. This observation resonates well with the literature, as shown in Table 11, which highlights the techniques that have been applied explicitly to autonomous driving-related applications in the included publications. The columns list the nine AxC techniques selected in Section 3.

Quantization and pruning are the most popular techniques. This is likely due to them being well-established in both research and production environments (even Google's TPU and Tesla's FSD chip execute quantized NNs [113,134,203]) as they are known for bringing very high savings with relatively little impact on model accuracy, as we will see later. This latter point may be particularly important for autonomous driving systems whose decision-making may be fatal for human beings.

Safety is of utmost importance in autonomous driving systems [204], as laid out in international standards [205], and cannot be compromised with any sort of approximation, adaptive or not. Yet, the applications outlined above have vastly different impacts on driving safety: SLAM is crucial for the vehicle's environmental understanding, while automatic windshield wipers represent more of a convenience to the driver and passengers. Hence, it may be difficult to convince manufacturers to integrate potentially non-deterministic techniques like VOS and approximate synthesis (see Section 3.1), despite initial academic efforts showing good results [88,106,108]. On the other hand, some established techniques have been shown to improve system reliability [22], but using these would in any case require vast testing to ensure the intended, safe functionality in all scenarios.

Moreover, despite not yet being explicitly covered in the surveyed literature, we envision that compression techniques as well as AxC-equipped architectures for audio and video processing (see Section 3.2) may also prove useful in reducing the energy consumption of infotainment systems. There are also several non-critical services in cars whose processing of noisy sensor data can be more aggressively approximated for greater energy savings. Examples of such include automatic windshield wipers and high-beam headlight control.

4.2. Smart sensing and wearables

Having reviewed publications relevant to autonomous driving – a domain that has some energy headroom owing to the mere size of its platforms – we shift our attention to smart sensing and wearables for which energy is even more scarce. Smart sensors are ubiquitous, energy-constrained, often mobile devices that implement sensors and low-power computing and networking hardware [206,207]. Wearables are instances of such devices. We focus particularly on biomedical wearables that are the fundamental building blocks of modern health technologies, characterized by energy and memory constraints yet expected to perform compute-heavy detection and classification tasks [208,209]. As a result, the focus is on developing ultra-low-power solutions that efficiently execute these tasks, providing sufficiently timely and accurate results despite their limited computing resources.

Fig. 9 illustrates the functionality of a smart sensor that integrates adaptive AxC. An incoming signal is first sensed and converted to an analog, electrical signal that is conditioned, or pre-processed, before it is converted to digital. Next, a signal processing unit, typically comprising standard processor cores, memory, and potentially some accelerators [118,210], performs the desired detection and classification tasks on the signal. Finally, the results are serialized to a

Table 12
AxC techniques applied in the included papers that explicitly list smart sensing as a use case. References sorted by publication year.

AxC Tech.	Quantization	Pruning	Knowledge distillation	Low-rank approx.	Compression	Algorithmic approx.	Inexact arithmetic	Voltage Over-Scaling	Approx. synthesis
[215]			Unexplored	Unexplored		✓	✓		
[216]						✓			
[209]						✓			
[217]								✓	
[218]						✓			
[219]						✓			
[172]		✓							
[108]									✓
[118]								✓	
[136]	✓	✓			✓		✓		
[127]							✓		
[220]					✓	✓			
[221]						✓			
[222]	✓	✓					✓		
[101]								✓	
[212]							✓		
[208]						✓			
[214]						✓			
[20]	✓								
[223]	✓								

local store or offloaded [206,211]. As shown in the figure, an AxC-equipped smart sensor extends upon this functionality by applying and managing adaptive AxC techniques itself [212], for example, by monitoring computational load, battery level, or network latency and selecting (approximate) operating points thereupon. We show the concept's energy-saving effects for a system with four modes resulting in different output qualities, visualized with images of reduced quality.

The achievable performance of a smart sensor is mainly limited by its power constraints, affecting the signal processing step's performance [213], and the complexity and dynamics of its environment, affecting the algorithms that it is to execute [214]. Attaining satisfactory energy consumption, and thus lifetime, means balancing these. The sensor's signal processing step dominates the power consumption when the sensor is operating as a standalone device, i.e., performing tasks without communicating with external devices. Reducing this step's power consumption may be achieved in mostly the same ways as described in the previous section, i.e., by maximally utilizing available computing resources and lower-level (cache) memories, minimizing power consumption from idle hardware and operations reaching costly higher-level storage that consumes two-three orders of magnitude more power per operation than arithmetic [125].

4.2.1. Smart sensing at the edge

Motivated by many of the same reasons as for autonomous driving, when signals require too much processing to be carried out locally, a smart sensor can instead choose to offload all or parts of the processing to an external device. Yet, while networking poses a challenge for autonomous vehicles due mostly to its latency, reducing its power consumption is more challenging for low-power, embedded devices such as smart sensors and wearables [220], especially if long range is required. Balancing these effects is difficult but can be rewarding for the device's lifetime [224].

Contrary to autonomous driving, traditional smart sensing applications, such as biomedical wearables, have little risk of impacting their surroundings and, thus, a lower criticality in terms of latency. Until now, this has motivated offloading of processing in the Cloud whose drawbacks we have already outlined in Section 2.1. Thus, Edge processing once again represents an interesting alternative for meeting the privacy and reliability demands of smart sensing applications [132]. In this domain, a sensor is considered smart when it is capable of acquiring, processing, and interpreting data to perform decision-making without relying on Cloud offloading [210].

The adaptive AxC techniques described above represent one way of decreasing reliance on the Cloud. Related methods have already been used to reduce the effects of network latency variations [225], and approximate aggregation of data from many distributed sources [226,227], but they are equally applicable to healthcare monitoring [228] and seizure detection [212]. For both these applications, approximation can reduce the amounts of data needing to be transferred to the Cloud and thereby extend battery life.

4.2.2. Effective processing of sensor data

Two components of a smart sensor or wearable alter the input signals: a conditioner may apply filters and amplification, doing so directly in hardware with minimal overhead, and a signal processor can perform more complex algorithms on the signals [211]. Due to the complex, dynamic environments in which these devices operate, ML algorithms are particularly popular in this domain [132]. As described earlier, ML algorithms are inherently insensitive to noisy input data (and approximations), rendering them suitable for systems with analog sensors and digital processing engines [229].

While autonomous driving relies on DNNs, the energy and latency constraints of smart sensors and wearables prohibit the use of such large models. Therefore, although DNNs remain relevant [132], this motivates the use of alternative, lower-complexity ML models such as k-NN or SVM [229], as introduced in Section 2.2. Regardless of the model applied, using ML imposes some requirements on the pre-processing, or *extraction*, steps required to bring sensor data into a format suitable for digital processing [229]. For biomedical applications, some example extraction techniques are Electroencephalogram (EEG), Electrocardiogram (ECG), Somatosensory Evoked Potential, and Visual Evoked Potentials.

Once pre-processed, the data undergoes a *feature extraction* process to extract statistics relevant to a particular application and to reduce their size. Examples of such extraction techniques include the following:

- **Principal Component Analysis** extracts *principal components* – linearly uncorrelated vectors – by orthogonal transformation of correlated vectors in an observation set [34]. For EEG enhancement and seizure detection, it is useful for extracting frequency features [216,230].
- **Independent Component Analysis** is an artifact removal strategy that decomposes signals to separate data and noise [231]. It is also useful for EEG processing [216], yet, despite being effective, its computational complexity renders low-power realizations difficult.

- **Fast Fourier Transform (FFT)** is likely the most well-known algorithm for converting time-domain signals to the frequency domain. It is useful for example in Chronic Neurological Disorder (CND) [218]. The FFT cannot be applied directly to non-stationary signals like EEG, but assuming that the signal is locally stationary, the windowed Short-Time Fourier Transform [212] or derived, FFT-based methods [221] may be applied instead.
- **Wavelet Transform** applies variably sized windows generated by a wavelet to a signal to extract both time- and frequency-domain features. The windowing feature makes it suitable for applications with non-stationary signals like EEG [215,219] and ECG [217].

While we avert further descriptions of the aforementioned ML models, we find it relevant to exemplify their applicability. Demanding ultra-low complexity, k-NN is frequently applied in this domain, e.g., for CND detection [232], despite suffering from inaccuracy when processing redundant data. SVMs perform well in Electrocardiogram classification [229], being especially good at handling multi-dimensional inputs when a clear boundary exists between distinct classes. Unfortunately, SVMs scale poorly with dataset size [37]. Various NNs are often used for Keyword Spotting or general speech recognition, typically being implemented in custom hardware accelerators [136].

For biomedical smart sensing applications, it is often infeasible to compute fully exact models in real-time when the battery level is low and safety considerations permit so. In such cases, adaptive AxC techniques can relieve the system of some of its demands at the expense of some errors [217,223], for example, by switching to an approximate mode when the battery level is low [212,228], as shown in Fig. 9. The safety requirements of the particular application targeted by the wearable play a significant role in determining how far such approximations can be applied: simple fitness monitoring tasks may, for example, be of lesser importance than cardiac arrhythmia detection [229] or various workplace safety tasks [233].

To maximize their energy efficiency, wearables and smart sensors often implement application-specific accelerator hardware. This enables them to, for example, benefit fully from the model compression techniques described in Section 3.3 when executing NN-based applications. As was outlined in Section 3.2.2, such architectures may also make more broad use of circuit-level AxC techniques than general-purpose ones; again, assuming compliance with the applications' safety requirements.

We highlight the AxC techniques applied to smart sensing and wearable use cases in Table 12. As for autonomous driving, NN-based designs frequently make use of either quantization or pruning to reduce module size [136,222], while they have yet to explore the more advanced techniques of knowledge distillation and low-rank approximation. Regardless, algorithmic approximations are vastly more popular in this domain. These approximations target the pre-processing and feature extraction stages, whose hardware implementations may also be approximated [149,222]. We see that, contrary to autonomous driving, VOS and approximate synthesis have been applied to smart sensing and wearable applications, albeit only to a limited extent thus far.

For adaptability, other work has also proposed letting the smart sensor or wearable execute light-weight models of its hardware to dynamically select between (inexact) arithmetic units (or entire ML models [234]) at run-time as per the application requirements; an example being low-complexity decision trees [95]. In dynamic environments, such systems may even implement RL techniques to better adapt to changes in input data [132]. This is explored in two general schemes by Maity et al. [235] for managing approximate memory architectures and by Lin et al. [184] for dynamically pruning convolutional filters. Both techniques achieve significant improvements in power consumption and execution time.

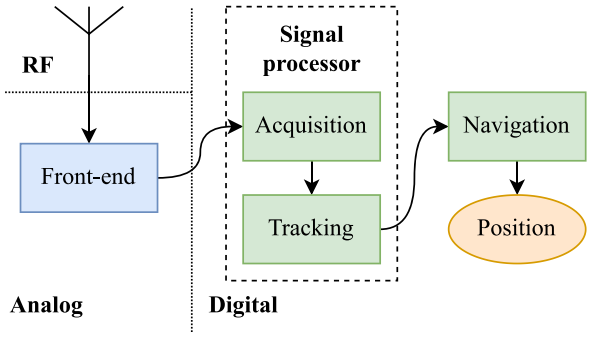


Fig. 10. Processing chain inside a GNSS receiver. Notice the similarity with the generic smart sensor shown in Fig. 9.

4.3. Positioning

An application relevant to both previously surveyed domains is positioning. For autonomous vehicles, navigation requires both precision and reliability, achieved today by the fusion of many sensors [236]. Additionally, though indoor and outdoor positioning sensors also fall into the scope of smart sensing, we detail them as standalone applications. Miniaturization of electronics has led to new use cases in positioning, e.g., navigation in consumer products (e.g., smartphones) using satellite-based positioning systems. Yet, receivers for such systems, as illustrated in Fig. 10, are known to be energy-hungry [237] and, thus, challenging to integrate into energy-constrained devices (e.g., IoT). Additionally, they do not offer positioning in certain environments such as indoors. As a result, we focus on two main scenarios: satellite-based outdoor positioning and mesh network-based indoor positioning.

4.3.1. Outdoor positioning

Outdoor positioning can include a multitude of positioning sensors and techniques. The most widely used technique today is satellite-based positioning, as it offers global coverage with no additional infrastructure required while providing accurate absolute positioning. Satellite positioning systems are also referred to as Global Navigation Satellite Systems (GNSS), specifically one or more global constellations: GPS (USA), GLONASS (Russia), Galileo (EU), and BeiDou (China), or regional constellations: IRNSS (India) and QZSS (Japan). GPS and GLONASS may be considered legacy systems, the others having been developed to gain independence in satellite positioning, as such systems were originally designed for military purposes. All these systems, while different in their implementations, are built around the same principle: a very precise clock orbiting the Earth, continuously transmitting its current time over a radio signal. A receiver located on Earth receiving this signal will decode its time of transmission and compare it to its time of reception to obtain *pseudoranges*. By receiving pseudoranges from at least four satellites and using the principles of *trilateration*, the receiver can compute its position [238].

Since their launches, the legacy systems and their new global or regional counterparts have been equipped with modernized signals designed to increase performance in challenging environments and answer the current challenges of satellite-based positioning. These signals enable higher resistance to intentional and unintentional interference, Safety-of-Life services, positioning in complex signal environments, and increased positioning accuracy and precision [238]. Yet, decoding them comes with a computational cost, expected due to their higher complexity. Significant research efforts over the last decade have been spent on mitigating these costs, yet they remain more energy-intensive than legacy GPS signals [239]. The integration of GNSS receivers in even more energy-constrained devices necessitates exploring new ways of processing the signals with less energy. As we will review later, AxC could be a suitable approach to answer such problems.

Table 13
AxC techniques applied in the included papers that explicitly list positioning as a use case. References sorted by publication year.

AxC Tech.	Quantization	Pruning	Knowledge distillation	Low-rank approx.	Compression	Algorithmic approx.	Inexact arithmetic	Voltage Over-Scaling	Approx. synthesis
[242]						✓			
[243]					✓				
[244]						✓			
[245]						✓			
[246]					✓				
[247]	Unexplored	Unexplored	Unexplored	Unexplored		✓	Unexplored	Unexplored	Unexplored
[236]						✓			
[248]						✓			
[249]						✓			
[250]						✓			
[251]						✓			
[252]						✓			

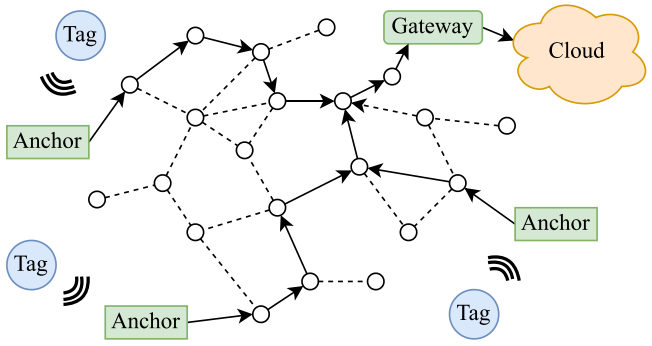


Fig. 11. IoT mesh network for indoor positioning.

4.3.2. Indoor positioning

Where GNSS-based solutions cannot provide sufficient accuracy, due to their limited reception power or difficult environment (e.g., urban canyoning, inside buildings, etc.), other techniques are needed. Indoors is one of the most tedious environments to navigate due to its complex geometry [17]. As such, many techniques have been proposed to provide a reliable and precise positioning solution [240]. Yet, contrary to outdoor systems that are well-developed and integrated into many existing devices, indoor systems have yet to see broad application. Nonetheless, the number of applications is tremendous – including, e.g., efficient warehouse maintenance – which explains research interest in it [241].

As indoor systems operate on much shorter distances than their satellite-based counterparts, they also apply other techniques, for example, Received Signal Strength Indicator (RSSI) and Angle-of-Arrival (AOA) that estimate the position of (battery-powered) *tags* relative to known positions of (mains-connected) *anchors* [17]. As transmitted signals are attenuated by the air and obstacles they pass, measuring RSSI allows for a receiver to estimate its distance to the transmitter. AOA-based localization systems instead employ arrays of antennas that enable them to estimate the angle of arrival of an incoming signal. AOA techniques have gained popularity in the IoT industry since 2019 as devices supporting Bluetooth Low-Energy were shown to achieve centimeter-level precision [253]. Such results are a great improvement over RSSI-based solutions [254].

Recently, AOA methods have also found application in IoT (mesh) networks, illustrated in Fig. 11. Such networks function as described above, but may need to rely on multi-hop links to pass a packet from an anchor to the network gateway due to low transmission range of individual nodes [255]. In such networks, AOA techniques can be prohibitively expensive, especially if they are executed outside the Cloud, i.e., in the anchors [252]. For this case, numerical algorithms and AxC may be combined to reduce the algorithmic complexity [251].

4.3.3. Positioning at the edge

While outdoor and indoor positioning applications face different challenges for signal processing, the devices involved closely resemble smart sensors (see Fig. 9) and the energy available on-board is similarly limited. This leads to common solutions like offloading the processing operations at the Edge. Thus, understanding the factors that increase energy consumption is paramount to designing solutions adapted to application needs and constraints. Even though GNSS receivers are generally implemented in battery-powered, embedded platforms, many applications (e.g., surveying and construction) do not prioritize energy consumption reductions. While these applications cannot afford reduced precision, they constitute a decreasing part of a market set to be dominated by low-power IoT devices [249]. Consequently, low-power GNSS is increasingly being used in low-cost applications, where the quality of positioning constraints can be relaxed: meter-level positioning is sufficient, e.g., for smartphones. The algorithms should be adapted to answer this goal to the accuracy level required [24].

Offloaded processing has been considered in both applications with varying degrees of acceptance. For indoor positioning, transferring raw signal measurements to, e.g., the Cloud for processing would rapidly drain the mesh nodes' batteries and delay measurements unacceptably [252]. Therefore, the preferred solution is to perform processing in the anchor nodes, further reducing energy consumption by disabling the otherwise periodic transmissions from stationary tags. A similar possibility exists in GNSS-based systems that require a high sampling rate to capture signal bandwidth. This means that near-sensor (or Edge) processing often is cheaper than re-transmission, making offloading suitable only for applications that need infrequent position updates [256]. We note that the energy consumption effects of the channel used, the distance from the remote server, and interference linked to the environment are yet to be evaluated.

4.3.4. Efficient signal processing

Regardless of the point of computation, both indoor and outdoor positioning applications rely on DSP algorithms. Specifically, GNSS processing involves the steps outlined in Fig. 10. The *acquisition* and *tracking* steps require the most computations. A GNSS receiver extracts the aforementioned pseudoranges from tracked signals to compute a position. Most often, this processing is implemented in a dedicated hardware accelerator whose real energy impact is caused by the required tracking time. Many studies have looked into reducing the acquisition and tracking complexity, specifically for modernized signals [250,257].

Some work also considers offloading for GNSS, though naïvely applying such an architecture would require large amounts of data transfers for uploading raw Radio Frequency signals. As the data rate of GNSS signals is quite low (e.g., 50 bps for GPS L1 Coarse/Acquisition Code [238]), several seconds are usually required to successfully extract the transmission time, and since these signals must be sampled at the

order of MHz to be successfully tracked, this constitutes too large an amount of data to be efficiently uploaded by an IoT device [246].

Instead, offloaded GNSS only partially integrates the steps in Fig. 10 onboard. Reducing the data transfers can be done in two ways: Coarse-Time Navigation (CTN) and Compressive Sensing (CS). CTN as introduced by [243] requires only a few milliseconds of data to produce a position at much degraded precision. This is explored by Ramos et al. [258] who offload the results of acquisition and tracking, and by Liu et al. [245] who offload the raw data. Unfortunately, neither study considers the energy consumption of transmitting the data. CS, which is not related specifically to GNSS, aggressively subsamples a signal well below the Nyquist rate before reconstructing it using well-defined mathematics. Misra et al. [246] combine this with CTN after signal reconstruction, successfully computing a position with minimal data transfers required. However, details on the experiment's parameters are missing, so an exhaustive analysis of this technique cannot be undertaken.

Whilst several acquisition and tracking algorithms exist for GNSS, there are also many AOA methods suitable for indoor positioning use cases. Examples include SAGE [259], MVDR [260], ESPRIT [261], and MUSIC [262], of which we focus on the latter two. These methods rely on In-Phase and Quadrature samples from Uniform Linear Arrays and are known to attain great accuracy [263], outperforming even *beamforming* alternatives such as MVDR [244].

Roy and Kailath's ESPRIT and Schmidt's MUSIC require computation of Eigenvectors of the measured signals' covariance [261,262], a mathematically complex and computationally heavy task that should be carried out only when necessary. Thus, while it is possible to extend these algorithms to a dynamic case, i.e., for tracking moving objects rather than localizing static ones, naïve repeated application rarely leads to good results. Instead, most proposed implementations complement skipping executions by so-called *subspace tracking* that involves an unbounded optimization problem [242].

When an application's accuracy requirements permit, AxC techniques may be employed to reduce the energy consumption of both outdoor and indoor positioning systems. Unlike autonomous driving and smart sensing applications, the positioning systems we have surveyed do not currently employ ML techniques. This is reflected in the AxC techniques applied to positioning use cases, highlighted in Table 13. We see that, apart from compression (including CS), only algorithmic approximations are applied in the current literature. Notably, however, we only cover positioning algorithms and, therefore, may have overseen studies on advancements in wireless communication systems and relevant hardware enabled by AxC. Nevertheless, comparing Table 13 to Tables 11 and 12 shows a clear lack of research in applying AxC to positioning systems.

Examples of such techniques are plenty. For GNSS-based systems, these include [247,256]: adapting measurement frequency and quality to *just* meet application needs [247]; reducing acquisition/tracking algorithm complexity, even in the context of modernized signals [250,257]; implementing duty cycling, for example, in the tracking step [247]; and reducing the sampling rate to the Nyquist rate (or below) based on band-pass theory [248], though modernized signals have a wider frequency spread, requiring higher sampling rates for successful tracking. All these strategies naturally lend themselves to adaptation dependent on, for example, momentary signal conditions.

While some of these may be adapted to indoor positioning algorithms, other relevant techniques are more involved and closely linked to the aforementioned algorithms, ESPRIT and MUSIC. As these are based on linear algebra, reducing matrix size [244], converting complex matrices to real ones with unitary transforms [264], and substituting the Singular-Value Decomposition of one matrix with the cheaper Eigendecomposition of another are examples of such. For MUSIC in particular, approximations target its compute-heavy *peak finding* step by using more coarse-grained angle intervals or by replacing the step with polynomial root finding [265].

5. Discussion and open directions

We have now surveyed a wide range of papers in Sections 3.1, 3.2, and 3.3 and outlined the use of AxC within the three application domains in Sections 4.1, 4.2, and 4.3. While non-comprehensive, this survey has led us to observe several directions for future work, which we highlight here. We consider open directions in each of the three application domains individually and refer to AxC techniques when applicable. Finally, we suggest research into cross-layer approximations. We summarize our findings in Table 14.

Before diving into the application domains, however, we highlight the potential gains of applying the surveyed AxC techniques. These savings are highly dependent on the application or algorithm the AxC techniques are applied to, its error tolerance, and even its implementation. We notice that relatively few papers state clear energy and area savings over an exact baseline, reporting instead reductions in model size (in ML-focused papers) or algorithmic complexity (in positioning-related papers). We combine these two metrics in the compound *computational cost* metric. Moreover, as many of the surveyed papers are application-agnostic by nature, their reported savings may not translate 1-to-1 into our considered domains.

Despite the substantial number of papers surveyed herein, only relatively few report savings in terms of the same, or similar, metrics. For example, just three covered papers that use low-rank approximation alone describe computational cost savings [43,178,181], and a mere four papers that use inexact arithmetic alone detail energy savings [69,78,79,212]. This limited number of experiments instills little confidence in estimates of the effectiveness of each technique in general as well as within any particular application domain, and it renders a comprehensive quantitative comparison of the results reported in the surveyed papers statistically infeasible. Therefore, we restrain this discussion to a qualitative comparison in Table 15. We notice that relatively few papers state clear energy and area savings over an exact baseline, reporting instead reductions in model size (in ML-focused papers) or algorithmic complexity (in positioning-related papers). Table 15 illustrates a point highlighted throughout our survey: ML algorithms common to both autonomous driving and smart sensing are highly tolerant even to aggressive approximations. As mentioned earlier, the negative effects thereof on accuracy can often be mitigated through re-training [155].

5.1. Autonomous driving

5.1.1. Insights

Despite autonomous vehicles having come a long way over the last decade and several car manufacturers offering Level 3 autonomy [186], the underlying technologies need further improvements to mature and allow for safe operation at Level 4 and, eventually, Level 5 [41]. This development will likely involve – if not require – intelligent connectivity (so-called vehicle-to-X or *V2X*) and external computing, for example in infrastructure, to satisfy rising compute demands [187], e.g., in ML and SLAM applications [185,236]. Such a system will need new networking and computing architectures [199], new intelligent automobile frameworks [154,197,198], in addition to new algorithms for scheduling and resource allocation [196]. Additionally, as new devices are connected, the system heterogeneity grows and new algorithms are needed to constrain the inevitable overheads of such to sustain real-time operation [187].

5.1.2. Future directions

We forecast autonomous driving to increasingly rely on Edge AI and expect its underlying ML algorithms to remain popular and, thus, obvious targets of AxC. There is plenty of potential for future work in this direction: quantization algorithms can be taken to the extreme, binarized case [156] or applied to smaller network elements than layers [171] with more modeling and experiments to follow [162,169].

Table 14
Summarized directions of future work extracted from surveyed works.

Applications and algorithms			Architectures and circuits		
Domain	References	Challenges	Domain	References	Challenges
Autonomous Driving	[154,185,187, 196–199,236]	New networking and computing architectures, automobile frameworks, and scheduling and resource allocation algorithms to support intelligent (*V2X*) connectivity.	Architectures	[8,21,65,119, 121,124,128]	Better sensitivity analysis and probabilistic model checking tools to enable easier AxC integration into GPPs and other architectures with guaranteed effects.
	[32,43,56,156, 171,175,178, 180]	Novel quantization, pruning, knowledge distillation, and low-rank approximation techniques and their combination with AxC.		[126,127]	Greater architectural reconfigurability to enable energy-efficient acceleration of various applications with low overheads, for example in FPGAs or CGRAs.
Smart Sensing and Wearables	[211,234]	Offloading-enabled approximations in data transmission and Edge execution of multiple sensing models in parallel.	Circuits	[21,71,79,141]	Development of inexact arithmetic units, including the use of under-design and accuracy-configurability.
	[218,220]	Generalization of ML models and supporting hardware for AI-capable IoT across different applications.		[59,60,115,117]	Understanding and mitigating negative effects of voltage over-scaling, or exploiting its combination with non-volatile memories for efficient parallel computing.
	[118,212]	Low-power signal processing with intrinsic approximate effects or reconfigurability.		[102,103,105, 106,108,109, 266]	Scalable inexact (high-level) synthesis algorithms, for example using genetic mutation or heuristics.
Positioning	[244,247]	Porting of AxC techniques used in smart sensing applications and their combination with reduced sampling rate, algorithmic approximations, etc.	Cross-layer	[21,30,57,60–62, 144,145]	Cross-layer AxC and low-overhead quality management techniques to support efficient runtime adaptation with quality guaranteed.
	[17,241,267, 268]	Development and sharing of common, heterogeneous datasets, testing environments, and performance metrics, for example as simulations or SDRs.		[10,13,16,50,53, 192]	Secure hardware technologies, resource management algorithms, programming and software platforms, and real-world frameworks for Edge computing management for scalable Edge AI and IoT.
	[246,252,268]	Design space exploration in combining different algorithms, signals, processing platforms, and AxC techniques for low-power operation, for example with SDRs.		[20,53,57]	Further development of Edge-suitable AI models and their synergies with AxC.
				Our work	Comprehensive and systematic establishment of the impact of different AxC techniques on the power consumption, performance, safety, and reliability of various applications.

Similarly, pruning and knowledge distillation algorithms can be altered to better utilize the information of network elements [32,175] and allow for a theoretical understanding of their effects [56] prior to experiments [173]. And despite quantization and pruning leading to some reduction in model size, further research into compression by low-rank approximation is needed [43,178,180]. Lastly, it is relevant to quantify the effects of combining these techniques [56].

This broad dependence on large-scale ML algorithms will require further research into application-specific accelerators like the ones described in Section 3.2.2. These accelerators must take AxC techniques that are well-established in the field, i.e., the ones marked in Table 11, into account. However, even the unmarked techniques are relevant so long as their use complies with the restrictions set out in Section 4.1.2. The autonomous driving use case allows for some headroom in terms of circuit area, meaning the accelerators may not need but can benefit from various circuit-level techniques beyond inexact arithmetic. There are plenty of open research directions for such techniques [9,21,31], including adapting existing under-design and accuracy-configurability techniques applied in adders and multipliers to other arithmetic units [71,79]. With the safety constraints of the application in mind, AxC techniques can also be used to reduce the overheads of modular redundancy used to ensure proper functionality even under (partial) system failure [21,269,270].

5.2. Smart sensing and wearables

5.2.1. Insights

We have described how smart sensors and wearables are rapidly evolving and gaining traction in the public. As a result, they have received significant research attention and must be expected to keep receiving such. The range of applications these types of devices are expected to execute is very broad and includes DSP and ML tasks within the health information domain [212,228]. To match the energy and latency constraints of these tasks, the devices must increasingly rely on Edge offloading [132,220].

5.2.2. Future directions

In addition to the AxC techniques pertinent to ML models outlined above, offloading enables new trade-offs in data transmissions [211] and in executing multiple models in parallel in Edge hardware [234] that need further exploration. This may be necessary to satisfy scalability and latency constraints [132], which also require further analysis [223]. Adoption of AI-capable IoT will require a generalization of devices and algorithms, e.g., using common ML models in similar hardware for different applications, be it EEG classification [218] or neurological disease detection [220]. Such devices can benefit from mixed-signal implementations that naturally operate approximately [212] or architectural reconfigurability [118]. The algorithms, on the other hand, may benefit from currently unmarked techniques in Table 12, knowledge distillation and low-rank approximation, especially in combination with quantization and pruning.

Smart sensors' sizes prohibit the implementation of several application-specific accelerators. This implies a need for general-purpose programmability or hardware reconfigurability. Programmability is a common trait of traditional general-purpose processors, possibly modified to support approximate instructions like those described in Section 3.2.1. Future research in this direction may explore approximating other instruction classes than arithmetic and load/store [119], better index functions and replacement policies for approximate caches [121], and in general easier integration of AxC into GPP architectures [8, 21]. Several authors highlight a need for better sensitivity analysis tools for understanding the effects of and selection of approximations with guaranteed impact [124,128]. Reconfigurability can, as we have seen, be provided using CGRAs [126,127], but the functionality is also available in FPGAs [70,94,131,153]. Identifying the sweet spot of

Table 15
Qualitative savings arising from using the surveyed AxC techniques on their own.

AxC Tech.	Quantization[a]	Pruning[a]	Knowledge Distillation[a]	Low-rank Approx.[a]	Compression[b]	Algorithmic Approx.	Inexact Arithmetic[c]	Voltage Over-Scaling[c]	Approx. Synthesis[c]
Comp. cost savings					—		—	—	—
Energy savings	—	—	—						
Area savings	—	—	—	—	—	—		—	
Error									
References	[156–158,160, 163,165–167, 170,171]	[39,56,125, 172–174,184]	[175]	[43,178,181]	[220,246]	[221,245,247–249,252,257]	[69,78,79,95, 212]	[118,217,235]	[103–105,108, 109]
Legend			Low 	Medium 	High 	Very high 			

[a] AxC techniques, like quantization, pruning, knowledge distillation, and low-rank approximation, rarely directly affect the hardware area but rather permit fitting more computation into the budgeted resources.

[b] Compression is often complementary to computational offloading [246]; it can decrease data transmissions at the expense of increased local computation.

[c] Inexact arithmetic, VOS, and approximate hardware synthesis techniques do not directly affect run-time computational costs.

reconfigurability and its related overheads while exploiting AxC is an interesting research avenue.

In the above, we have seen how VOS has already been applied to several smart sensing systems [101,118,217]. While it may find application in general-purpose processors [8], its benefits are likely better exploited in application-specific circuits. There are plenty of opportunities for future research in this direction [8,59,60], including a need for a better understanding of system-wide effects, especially of volatile memories, e.g., caches [115]. This has led some to suggest using non-volatile, emerging technology-based memories both for storage and parallel computing [117], thus inherently integrating AxC.

5.3. Positioning

5.3.1. Insights

We have already highlighted the close similarity between devices used for positioning and those used for generic smart sensing applications. With this in mind, many AxC techniques relevant in this domain can be carried over to positioning as well, especially the ones used in DSP algorithms, e.g., reduced sampling rate, inexact arithmetic, and various algorithmic alterations [244,247].

Despite the domain split between indoor and outdoor positioning algorithms as described in Section 4.3, papers on both outline a need for more standardized experiments to ensure comparability and reproducibility. Such schemes involve the creation and sharing of repositories of heterogeneous datasets [17], the development of common testing environments as simulations [241] or Software-Defined Radios (SDRs) [267,268], and the selection of common performance metrics. In practice, this implies sharing code and simulation environments in open source [268].

5.3.2. Future directions

As outlined in Section 4.3.4 and shown in Table 13, there is little work applying AxC to positioning applications. Moreover, despite low-power operation being a goal for next-generation Galileo-capable chips, no approximation is mentioned as a potential direction for future research [249]. Similarly, publications on indoor positioning suggest only developing new algorithms [241,252]. We argue that some AxC techniques can help improve energy efficiency in positioning applications, but that using them involves identifying the elements of positioning systems that consume the most energy – an aspect that SDRs and more real-world experiments can help solve [246,252,268]. The interesting points in this direction include algorithms, signal selection, processing platform (onboard or offloaded), and AxC techniques; combinations of which are bound to have interesting trade-offs. We find that techniques related particularly to the ML domain are irrelevant to positioning applications so long as they do not rely on such algorithms. Conversely, circuit-level techniques can be highly relevant, especially in hardened DSP algorithms.

Considering the lifetime requirements of positioning systems, focusing solely on inexact arithmetic and VOS when pursuing the aforementioned direction may unnecessarily narrow the scope of approximations and give rise to unsatisfactory savings. Instead, AxC may be considered in a broader scope and applied to general hardware designs through inexact (high-level) synthesis [266]. Existing work introduces and combines various AxC techniques algorithmically, showing promising results but also describing many directions for future work. These include new genetic mutation algorithms and heuristics for optimization during DSE [103,108,109] and the inclusion of new AxC techniques into existing flows [102,109]. The general notion in these papers is that existing algorithms suffer from poor scalability and must be adapted to fit common design toolchains [105,106]. The latter is a particularly relevant research direction [266] as a well-integrated inexact synthesis flow may find application also when designing smart sensors or even accelerators for autonomous driving.

5.4. Cross-layer research

5.4.1. Insights

With the many open challenges within the application domains in mind, it is worth mentioning that combining AxC techniques can lead to even greater gains than they can individually. Implementing optimizations or approximations across layers from software to circuits can drastically improve overall system performance [30,57]. This is especially interesting when accounting for the distributed nature of the three application domains, we consider. As they rely on offloading, new approximation opportunities arise both in the data-producing devices, the data-processing devices, as well as their communication. Nevertheless, existing work only combines relatively few AxC techniques, as illustrated in Table 16. As noted before, ML applications are particularly tolerant and most frequently approximated with more than one technique, e.g., using quantization, pruning, compression, *and* inexact arithmetic [136]. Positioning algorithms, on the other hand, have so far only been approximated algorithmically.

Table 16
Cross-product of the surveyed AxC techniques. Tick marks indicate which techniques have been applied concurrently.

AxC tech.	Q	P	KD	LA	C	AA	IA	VOS	AS
Q		✓	✓	✓	✓	✓	✓	✓	
P				✓	✓		✓		
KD									
LA					✓				
C						✓	✓		
AA							✓		
IA								✓	✓
VOS									✓
AS									

Abbreviations: *Q*uantization, *P*runing, *K*nowledge *D*istillation, *L*ow-Rank *A*pproximation, *C*ompression, *A*lgorithmic *A*approximation, *I*nexact *A*rithmetic, *V*oltage *O*ver-*S*caling, and *A*pproximate *S*ynthesis.

5.4.2. Future directions

Introducing and managing approximations across system layers requires the development of hardware/software co-design tools and frameworks for DSE [21], for example with HLS as described above. Unfortunately, such tools are not enough; to safely apply approximations within an application requires intimate knowledge of its tolerance or resilience to errors. This metric varies across domains just as much as between individual applications within a domain. While sensitivity analysis may help in estimating numerical tolerances, establishing higher-level, domain-specific constraints may require the involvement of legal and ethical considerations: what classification accuracy is sufficient for an automatic emergency braking system in a car, for example? How precise should a heart rate monitor wearable be? And what positioning precision should a smartphone provide? Unfortunately, such safety constraints are not well-established in the literature, yet we find them to be crucial for the broader adoption of AxC and, in fact, even for the use of AxC within a particular application.

Returning to numerical constraints, it is especially challenging to ensure output quality under environmental variations. We have tried to highlight adaptive AxC as a potential solution to this throughout the paper. Unfortunately, its adoption thus far is limited by existing sensitivity analysis tools [29,31,124,128] and overheads from run-time adaptation [61,62], implementations of which mostly fail to provide quality guarantees [21].

In parallel with the development of new adaptive AxC solutions, researchers must focus on maturing the Edge computing domain further. Scalable Edge AI and IoT demand new hardware technologies [13,16], novel algorithms for resource management to minimize energy and task execution time [53,192], and programming and software platforms for managing them [10,16,53], continuously maintaining security, privacy, and reliability [10,53]. More progress is needed in Edge computing framework development [50] as well as in resource-friendly, Edge-suitable AI models [20,53] and their synergies with (adaptive) AxC [57]. The wide range of functionalities to be offered by an Edge computing framework means plenty of opportunities for adaptive approximation with interesting trade-offs to follow but implies a need for practical experimentation. Once all these elements are well-established, they will enable much easier integration of future connected applications.

6. Conclusion

The number of connected IoT devices and the computational demands needed to intelligently process the data they produce are rising exponentially. This trend renders traditional Cloud offloading and existing communication technologies insufficient as they fail to meet the devices' and applications' latency constraints. The solution is to perform more processing at the Edge of the Internet, where energy and compute power are sparse and must be optimally utilized. This is highly relevant in the three application domains of autonomous driving, smart sensing and wearables, and positioning. In these domains, it is necessary to exploit the applications' inherent error resilience through adaptive AxC to save energy and efficiently perform processing at the Edge.

In this paper, we have presented a survey of literature spanning these three application domains and AxC techniques relevant to them. We have covered these topics bottom-up, first describing AxC techniques from the circuit level to the application level, and later describing their applicability in each domain. Our discussion has focused on nine different AxC techniques: quantization, pruning, knowledge distillation, low-rank approximation, lossy compression, various algorithmic approximations, inexact arithmetic, voltage over-scaling, and approximate synthesis.

Our key findings are that all three application domains offer opportunities for exploiting AxC, particularly the ML-based applications in autonomous driving and smart sensing, but also the DSP-based tasks in positioning; that not every surveyed AxC technique has been applied in all the application domains or in combination with other techniques, and that the reasons therefore not always are clear; that each application domain's sensitivity to approximations is not well established, rendering the evaluation of proposed AxC techniques difficult or impossible to carry out; and that adaptive AxC can be a powerful tool for improving energy efficiency, but that it faces several challenges concerning low-overhead quality management and sensitivity analysis.

Achieving scalable Edge AI and IoT means many challenges must be solved and existing techniques must be improved. We expect adaptive AxC to be crucial in this development. To spark further initiative in this direction, we have highlighted many open challenges, including novel algorithms and application-level AxC techniques for autonomous driving; low-power signal processing and ML algorithms for offloading-capable IoT devices; adaptation and experimentation with various AxC techniques in positioning algorithms; further experimentation with combinations of AxC techniques in all three covered application domains; and greater reconfigurability and low-overhead quality management of architecture- and circuit-level AxC techniques. Additionally, we wish to emphasize the importance of establishing the quality or safety constraints within the three application domains. Without such, one may blindly approximate critical elements of such systems, causing unnecessary and avoidable harm.

Acronyms

AI	Artificial Intelligence
AOA	Angle-of-Arrival
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
AxC	Approximate Computing
BLE	Bluetooth Low-Energy
C/A	Coarse/Acquisition Code
CNN	Convolutional Neural Network
CGRA	Coarse-Grained Reconfigurable Array
CLA	Carry-Lookahead Adder
CND	Chronic Neurological Disorder
CPD	Canonical Polyadic Decomposition
CS	Compressive Sensing
CSA	Carry-Select Adder
CTN	Coarse-Time Navigation
DNN	Deep Neural Network
DRAM	Dynamic RAM
DCT	Discrete Cosine Transform
DSE	Design Space Exploration
DSP	Digital Signal Processing
DVFS	Dynamic Voltage and Frequency Scaling
ECG	Electrocardiogram
EEG	Electroencephalogram

ESPRIT	Estimation of Signal Parameters via Rotational Invariant Techniques
EVD	Eigendecomposition
FA	Full Adder
FFT	Fast Fourier Transform
FL	Federated Learning
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite Systems
GPP	General-Purpose Processor
GPS	Global Positioning System
HD Map	High-definition Map
HEVC	High-Efficiency Video Coding
HLS	High-Level Synthesis
IC	Integrated Circuit
ICA	Independent Component Analysis
IoT	Internet of Things
IQ	In-Phase and Quadrature
IRNSS	Indian Regional Navigation Satellite System
k-NN	*k*-Nearest Neighbors
KWS	Keyword Spotting
LiDAR	Light Detection and Ranging
LSB	Least-Significant Bit
LUT	Look-Up Table
MAC	Multiply-Accumulate
ML	Machine Learning
MUSIC	Multiple Signal Classification
MVDR	Minimum Variance Distortion-less Response
NN	Neural Network
NoC	Network on Chip
PCA	Principal Component Analysis
PE	Processing Element
PU	Processing Unit
QZSS	Quasi-Zenith Satellite System
RNN	Recurrent Neural Network
RAM	Random Access Memory
RCA	Ripple-Carry Adder
ReLU	Rectified Linear Unit
RF	Radio Frequency
RL	Reinforcement Learning
RSSI	Received Signal Strength Indicator
RTL	Register-Transfer Level
SAE	Society of Automotive Engineers
SAGE	Space Alternating Generalized Expectation-Maximization
SDR	Software-Defined Radio
SEP	Somatosensory Evoked Potential
SLAM	Simultaneous Localization and Mapping
SoC	System on Chip
STFT	Short-Time Fourier Transform
SVD	Singular-Value Decomposition
SVM	Support Vector Machine
TPU	Tensor Processing Unit
ULA	Uniform Linear Array
VEP	Visual Evoked Potentials
VOS	Voltage Over-Scaling
WT	Wavelet Transform

CRediT authorship contribution statement

Hans Jakob Damsgaard: Writing – original draft, Visualization, Data curation, Conceptualization. **Antoine Grenier:** Writing – original draft, Data curation. **Dewant Katare:** Writing – original draft, Data curation. **Zain Taufique:** Writing – original draft, Data curation. **Salar Shakibhamedan:** Writing – original draft, Data curation. **Tiago Troccoli:** Writing – original draft, Data curation. **Georgios Chatzitsompanis:** Writing – original draft, Data curation. **Anil Kanduri:** Writing – review & editing, Methodology. **Aleksandr Ometov:** Writing – review & editing, Supervision, Methodology. **Aaron Yi Ding:** Writing – review & editing, Methodology. **Nima Taherinejad:** Writing – review & editing, Methodology. **Georgios Karakonstantis:** Writing – review & editing, Methodology. **Roger Woods:** Writing – review & editing. **Jari Nurmi:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Hans Jakob Damsgaard reports financial support was provided by EU Framework Programme for Research and Innovation Marie Skłodowska-Curie Actions.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors gratefully acknowledge funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska Curie grant agreement No. 956090 (APROPOS, http://www.apropos-itn.eu/).

References

[1] F. Montevecchi, T. Stickler, R. Hintemann, S. Hinterholzer, Energy-efficient Cloud Computing Technologies and Policies for an Eco-friendly Cloud Market, Tech. Rep. KK-03-20-210-EN-N, European Commission, 2020, URL https://digital-strategy.ec.europa.eu/en/library/energy-efficient-cloud-computing-technologies-and-policies-eco-friendly-cloud-market.

[2] Semiconductor Industry Association and/or its affiliates, Rebooting the IT Revolution: A Call to Action, Tech. Rep., Semiconductor Industry Association and Semiconductor Research Corporation, 2015, URL https://www.semiconductors.org/wp-content/uploads/2018/06/RITR-WEB-version-FINAL.pdf.

[3] P.F. Borowski, Mitigating climate change and the development of green energy versus a return to fossil fuels due to the energy crisis in 2022, Energies 15 (24) (2022) 9289.

[4] P. Cerwal, et al., Ericsson Mobility Report, Tech. Rep. EAB-22:010742, Ericsson, 2021, URL https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2022.

[5] H. Feng, S. Guo, L. Yang, Y. Yang, Collaborative data caching and computation offloading for multi-service mobile edge computing, IEEE Trans. Veh. Technol. 70 (9) (2021) 9408–9422.

[6] A. Ometov, J. Nurmi, Towards approximate computing for achieving energy vs. Accuracy trade-offs, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2022, pp. 632–635.

[7] J. Han, M. Orshansky, Approximate computing: An emerging paradigm for energy-efficient design, in: 18th IEEE European Test Symposium, IEEE, 2013, pp. 1–6.

[8] Q. Xu, T. Mytkowicz, N.S. Kim, Approximate computing: A survey, IEEE Des. Test 33 (1) (2015) 8–22.

[9] S. Mittal, A survey of techniques for approximate computing, ACM Comput. Surv. 48 (4) (2016) 1–33.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.

[11] F. Betzel, K. Khatamifard, H. Suresh, D.J. Lilja, J. Sartori, U. Karpuzcu, Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems, ACM Comput. Surv. 51 (1) (2018) 1–32.

[12] A. Ibrahim, M. Osta, M. Alameh, M. Saleh, H. Chible, M. Valle, Approximate computing methods for embedded machine learning, in: 25th IEEE International Conference on Electronics, Circuits and Systems, IEEE, 2018, pp. 845–848.

[13] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330.

[14] D. Ma, G. Lan, M. Hassan, W. Hu, S.K. Das, Sensing, computing, and communications for energy harvesting IoTs: A survey, IEEE Commun. Surv. Tutor. 22 (2) (2019) 1222–1250.

[15] M. Cococcioni, F. Rossi, E. Ruffaldi, S. Saponara, B.D. de Dinechin, Novel arithmetics in deep neural networks signal processing for autonomous driving: Challenges and opportunities, IEEE Signal Process. Mag. 38 (1) (2020) 97–110.

[16] Y. Shi, K. Yang, T. Jiang, J. Zhang, K.B. Letaief, Communication-efficient edge AI: Algorithms and systems, IEEE Commun. Surv. Tutor. 22 (4) (2020) 2167–2191.

[17] P. Pascacio, S. Casteleyn, J. Torres-Sospedra, E.S. Lohan, J. Nurmi, Collaborative indoor positioning systems: A systematic review, Sensors 21 (3) (2021) 1002.

[18] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A. Al Sallab, S. Yogamani, P. Pérez, Deep reinforcement learning for autonomous driving: A survey, IEEE Trans. Intell. Transp. Syst. 23 (6) (2021) 4909–4926.

[19] A. Ometov, V. Shubina, L. Klus, J. Skibińska, S. Saafi, P. Pascacio, L. Flueratoru, D.Q. Gaibor, N. Chukhno, O. Chukhno, et al., A survey on wearable technology: History, state-of-the-art and current challenges, Comput. Netw. 193 (2021) 108074.

[20] A.Y. Ding, E. Peltonen, T. Meuser, A. Aral, C. Becker, S. Dustdar, T. Hiessl, D. Kranzlmüller, M. Liyanage, S. Maghsudi, et al., Roadmap for edge AI: A dagstuhl perspective, ACM SIGCOMM Comput. Commun. Rev. 52 (1) (2022) 28–33.

[21] H.J. Damsgaard, A. Ometov, J. Nurmi, Approximation opportunities in edge computing hardware: A systematic literature review, ACM Comput. Surv. 55 (12) (2022) 1–49.

[22] A. Badran, S. Sadeghi-Kohan, J.D. Reimer, S. Hellebrand, Approximate communication: Balancing performance, power, reliability, and safety, in: 28th IEEE European Test Symposium, IEEE, 2023, pp. 1–6.

[23] D. Katare, D. Perino, J. Nurmi, M. Warnier, M. Janssen, A.Y. Ding, A survey on approximate edge AI for energy efficient autonomous driving services, IEEE Commun. Surv. Tutor. (2023).

[24] A. Grenier, E.S. Lohan, A. Ometov, J. Nurmi, A survey on low-power GNSS, IEEE Commun. Surv. Tutor. (2023).

[25] G. Karakonstantis, C.J. Gillan, Computing at the EDGE: New Challenges for Service Provision, Springer Nature, 2022.

[26] N. Mäkitalo, T. Aaltonen, M. Raatikainen, A. Ometov, S. Andreev, Y. Koucheryavy, T. Mikkonen, Action-oriented programming model: Collective executions and interactions in the fog, J. Syst. Softw. 157 (2019) 110391.

[27] P. Nikolaou, Y. Sazeides, A. Lampropulos, D. Guilhot, A. Bartoli, G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, K. Tovletoglou, L. Mukhanov, et al., On the evaluation of the total-cost-of-ownership trade-offs in edge vs cloud deployments: A wireless-denial-of-service case study, IEEE Trans. Sustain. Comput. 7 (2) (2019) 334–345.

[28] H.J. Damsgaard, A. Ometov, M.M. Mowla, A. Flizikowski, J. Nurmi, Approximate computing in B5G and 6G wireless systems: A survey and future outlook, Comput. Netw. (2023) 109872.

[29] V.K. Chippa, S. Venkataramani, S.T. Chakradhar, K. Roy, A. Raghunathan, Approximate computing: An integrated hardware approach, in: Asilomar Conference on Signals, Systems and Computers, IEEE, 2013, pp. 111–117.

[30] S. Venkataramani, S.T. Chakradhar, K. Roy, A. Raghunathan, Approximate computing and the quest for computing efficiency, in: 52nd Annual Design Automation Conference, ACM, 2015, pp. 1–6.

[31] V.K. Chippa, S.T. Chakradhar, K. Roy, A. Raghunathan, Analysis and characterization of inherent application resilience for approximate computing, in: 50th Annual Design Automation Conference, ACM, 2013, pp. 1–9.

[32] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, ACM J. Emerg. Technol. Comput. Syst. 13 (3) (2017) 1–18.

[33] H. Snyder, Literature review as a research methodology: An overview and guidelines, J. Bus. Res. 104 (2019) 333–339.

[34] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[35] T.M. Hehn, J.F. Kooij, F.A. Hamprecht, End-to-end learning of decision trees and forests, Int. J. Comput. Vis. 128 (4) (2020) 997–1011.

[36] V. Blanco, A. Japón, J. Puerto, Optimal arrangements of hyperplanes for SVM-based multiclass classification, Adv. Data Anal. Classif. 14 (1) (2020) 175–199.

[37] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing 408 (2020) 189–215.

[38] S. Theodoridis, Machine Learning: A Bayesian and Optimization Perspective, Academic Press, 2015.

[39] J. Mao, H. Yang, A. Li, H. Li, Y. Chen, TPrune: Efficient transformer pruning for mobile devices, ACM Trans. Cyber-Phys. Syst. 5 (3) (2021) 1–22.

[40] I.H. Sarker, Machine learning: Algorithms, real-world applications and research directions, SN Comput. Sci. 2 (3) (2021) 1–21.

[41] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, 2016, p. 9, arXiv preprint arXiv:1604.07316.

[42] M. Injadat, A. Moubayed, A.B. Nassif, A. Shami, Machine learning towards intelligent systems: Applications, challenges, and opportunities, Artif. Intell. Rev. 54 (5) (2021) 3299–3348.

[43] M. Astrid, S.-I. Lee, CP-decomposition with tensor power method for convolutional neural networks compression, in: IEEE International Conference on Big Data and Smart Computing, IEEE, 2017, pp. 115–118.

[44] J. Tee, D.P. Taylor, A quantized representation of probability in the brain, IEEE Trans. Mol. Biol. Multi-Scale Commun. 5 (1) (2019) 19–29.

[45] L. Fangxin, Z. Wenbo, W. Yanzhi, D. Changzhi, J. Li, AUSN: Approximately uniform quantization by adaptively superimposing non-uniform distribution for deep neural networks, 2020, p. 17, arXiv preprint arXiv:2007.03903.

[46] M. Tschannen, O. Bachem, M. Lucic, Recent advances in autoencoder-based representation learning, 2018, arXiv preprint arXiv:1812.05069.

[47] A. Marchisio, M.A. Hanif, M. Martina, M. Shafique, PruNet: Class-blind pruning method for deep neural networks, in: International Joint Conference on Neural Networks, IEEE, 2018, pp. 1–8.

[48] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding, 2015, p. 14, arXiv preprint arXiv:1510.00149.

[49] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: 1st Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.

[50] R. Berta, F. Bellotti, A. De Gloria, L. Lazzaroni, Assessing versatility of a generic end-to-end platform for IoT ecosystem applications, Sensors 22 (3) (2022) 713.

[51] M.A. Rahman, M.S. Hossain, G. Loukas, E. Hassanain, S.S. Rahman, M.F. Alhamid, M. Guizani, Blockchain-based mobile edge computing framework for secure therapy applications, IEEE Access 6 (2018) 72469–72478.

[52] T. Subramanya, L. Goratti, S.N. Khan, E. Kafetzakis, I. Giannoulakis, R. Riggio, A practical architecture for mobile edge computing, in: IEEE Conference on Network Function Virtualization and Software Defined Networks, IEEE, 2017, pp. 1–4.

[53] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE 107 (8) (2019) 1738–1762.

[54] X. Yang, Z. Song, I. King, Z. Xu, A survey on deep semi-supervised learning, IEEE Trans. Knowl. Data Eng. (2022).

[55] Q. Jin, L. Yang, Z. Liao, Towards efficient training for neural network quantization, 2019, arXiv preprint arXiv:1912.10207.

[56] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, Y. Yang, Asymptotic soft filter pruning for deep convolutional neural networks, IEEE Trans. Cybern. 50 (8) (2019) 3594–3604.

[57] S. Venkataramani, X. Sun, N. Wang, C.-Y. Chen, J. Choi, M. Kang, A. Agarwal, J. Oh, S. Jain, T. Babinsky, et al., Efficient AI system design with cross-layer approximate computing, Proc. IEEE 108 (12) (2020) 2232–2250.

[58] I. Tsiokanos, L. Mukhanov, G. Karakonstantis, Low-power variation-aware cores based on dynamic data-dependent bitwidth truncation, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2019, pp. 698–703.

[59] K. Tovletoglou, L. Mukhanov, D.S. Nikolopoulos, G. Karakonstantis, HaRMony: Heterogeneous-reliability memory and QoS-aware energy management on virtualized servers, in: 25th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2020, pp. 575–590.

[60] G. Karakonstantis, A. Chatterjee, K. Roy, Containing the nanometer "Pandora-Box": Cross-layer design techniques for variation aware low power systems, IEEE J. Emerg. Sel. Top. Circuits Syst. 1 (1) (2011) 19–29.

[61] D.S. Khudia, B. Zamirai, M. Samadi, S. Mahlke, RUMBA: An online quality management system for approximate computing, in: 42nd International Symposium on Computer Architecture, ACM, 2015, pp. 554–566.

[62] T. Kemp, Y. Yao, Y. Kim, MIPAC: Dynamic input-aware accuracy control for dynamic auto-tuning of iterative approximate computing, in: 26th Asia and South Pacific Design Automation Conference, IEEE, 2021, pp. 248–253.

[63] J. Strnadel, Statistical model checking of approximate circuits: Challenges and opportunities, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2020, pp. 1574–1577.

[64] R. Venkatesan, A. Agarwal, K. Roy, A. Raghunathan, MACACO: Modeling and analysis of circuits for approximate computing, in: 30th International Conference on Computer-Aided Design, IEEE, 2011, pp. 667–673.

[65] A.K. Mishra, R. Barik, S. Paul, iACT: A software-hardware framework for understanding the scope of approximate computing, in: Workshop on Approximate Computing Across the System Stack, vol. 52, 2014.

[66] S.K. Lahiri, A. Haran, S. He, Z. Rakamaric, Automated Differential Program Verification for Approximate Computing, Tech. Rep., Microsoft Research, 2015, URL https://citeseerx.ist.psu.edu/document?doi=1be736d0e01199a8c759467491875cb88a32d3e6.

[67] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications, IEEE Trans. Circuits Syst. I. Regul. Pap. 57 (4) (2009) 850–862.

[68] B. Liu, Y. Li, L. Huang, H. Cai, W. Zhu, S. Guo, Y. Gong, Z. Wang, A background noise self-adaptive VAD using SNR prediction based precision dynamic reconfigurable approximate computing, in: Great Lakes Symposium on VLSI, ACM, 2020, pp. 271–275.

[69] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, Low-power digital signal processing using approximate adders, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 32 (1) (2012) 124–137.

[70] J. Echavarria, S. Wildermann, A. Becher, J. Teich, D. Ziener, FAU: Fast and error-optimized approximate adder units on LUT-based FPGAs, in: International Conference on Field-Programmable Technology, IEEE, 2016, pp. 213–216.

[71] A.B. Kahng, S. Kang, Accuracy-configurable adder for approximate arithmetic designs, in: 49th Annual Design Automation Conference, IEEE, 2012, pp. 820–825.

[72] R. Ye, T. Wang, F. Yuan, R. Kumar, Q. Xu, On reconfiguration-oriented approximate adder design and its application, in: 32nd International Conference on Computer-Aided Design, IEEE, 2013, pp. 48–54.

[73] A.K. Verma, P. Brisk, P. Ienne, Variable latency speculative addition: A new paradigm for arithmetic circuit design, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2008, pp. 1250–1255.

[74] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, A low latency generic accuracy configurable adder, in: 52nd Annual Design Automation Conference, IEEE, 2015, pp. 1–6.

[75] S.-L. Lu, Speeding up processing with approximation circuits, IEEE Comput. 37 (3) (2004) 67–73.

[76] T. Nomani, M. Mohsin, Z. Pervaiz, M. Shafique, xUAVs: Towards efficient approximate computing for UAVs—Low power approximate adders with single LUT delay for FPGA-based aerial imaging optimization, IEEE Access 8 (2020) 102982–102996.

[77] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, RAP-CLA: A reconfigurable approximate carry look-ahead adder, IEEE Trans. Circuits Syst. II 65 (8) (2016) 1089–1093.

[78] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, M. Pedram, Roba multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (2) (2016) 393–401.

[79] P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an underdesigned multiplier architecture, in: 24th Internatioal Conference on VLSI Design, IEEE, 2011, pp. 346–351.

[80] S. Ullah, S. Rehman, B.S. Prabakaran, F. Kriebel, M.A. Hanif, M. Shafique, A. Kumar, Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators, in: 55th Annual Design Automation Conference, IEEE, 2018, pp. 1–6.

[81] K.Y. Kyaw, W.L. Goh, K.S. Yeo, Low-power high-speed multiplier for error-tolerant application, in: IEEE International Conference of Electron Devices and Solid-State Circuits, IEEE, 2010, pp. 1–4.

[82] V. Leon, G. Zervakis, D. Soudris, K. Pekmestzi, Approximate hybrid high radix encoding for energy-efficient inexact multipliers, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 26 (3) (2017) 421–430.

[83] T. Yang, T. Ukezono, T. Sato, Design of a low-power and small-area approximate multiplier using first the approximate and then the accurate compression method, in: Great Lakes Symposium on VLSI, ACM, 2019, pp. 39–44.

[84] H. Jiang, J. Han, F. Qiao, F. Lombardi, Approximate radix-8 booth multipliers for low-power and high-performance operation, IEEE Trans. Comput. 65 (8) (2015) 2638–2644.

[85] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, F. Lombardi, Design of approximate radix-4 booth multipliers for error-tolerant computing, IEEE Trans. Comput. 66 (8) (2017) 1435–1441.

[86] S. Boroumand, P. Brisk, Approximate adder tree synthesis for FPGAs, in: International Conference on ReConFigurable Computing and FPGAs, IEEE, 2019, pp. 1–8.

[87] C. Guo, L. Zhang, X. Zhou, W. Qian, C. Zhuo, A reconfigurable approximate multiplier for quantized CNN applications, in: 25th Asia and South Pacific Design Automation Conference, IEEE, 2020, pp. 235–240.

[88] F.-Y. Gu, C. Lin, J.-W. Lin, A low-power and high-accuracy approximate multiplier with reconfigurable truncation, IEEE Access 10 (2022) 60447–60458.

[89] H.O. Ahmed, M. Ghoneima, M. Dessouky, Concurrent MAC unit design using VHDL for deep learning networks on FPGA, in: IEEE Symposium on Computer Applications & Industrial Electronics, IEEE, 2018, pp. 31–36.

[90] J. Chang, Y. Choi, T. Lee, J. Cho, Reducing MAC operation in convolutional neural network with sign prediction, in: International Conference on Information and Communication Technology Convergence, IEEE, 2018, pp. 177–182.

[91] H.J. Lee, C.H. Kim, S.W. Kim, Design of floating-point MAC unit for computing DNN applications in PIM, in: International Conference on Electronics, Information, and Communication, IEEE, 2020, pp. 1–7.

[92] H. Zhang, D. Chen, S.-B. Ko, New flexible multiple-precision multiply-accumulate unit for deep neural network training and inference, IEEE Trans. Comput. 69 (1) (2019) 26–38.

[93] V. Mrazek, L. Sekanina, Z. Vasicek, Libraries of approximate circuits: Automated design and application in CNN accelerators, IEEE J. Emerg. Sel. Top. Circuits Syst. 10 (4) (2020) 406–418.

[94] S. Ullah, S.S. Murthy, A. Kumar, SmApproxLib: Library of FPGA-based approximate multipliers, in: 55th Annual Design Automation Conference, IEEE, 2018, pp. 1–6.

[95] M. Masadeh, O. Hasan, S. Tahar, Machine-learning-based self-tunable design of approximate computing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 29 (4) (2021) 800–813.

[96] I. Koren, Computer Arithmetic Algorithms, AK Peters/CRC Press, 2001.

[97] B. Parhami, Computer Arithmetic: Algorithms and Hardware Design, Oxford University Press, 2010.

[98] S. Yesil, I. Akturk, U.R. Karpuzcu, Toward dynamic precision scaling, IEEE Micro 38 (04) (2018) 30–39.

[99] T. Na, S. Mukhopadhyay, Speeding up convolutional neural network training with dynamic precision scaling and flexible multiplier-accumulator, in: International Symposium on Low Power Electronics and Design, ACM, 2016, pp. 58–63.

[100] B. Moons, M. Verhelst, An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS, IEEE J. Solid-state Circuits 52 (4) (2016) 903–914.

[101] J. Nunez-Yanez, N. Howard, Energy-efficient neural networks with near-threshold processors and hardware accelerators, J. Syst. Archit. 116 (2021) 102062.

[102] S. Lee, L.K. John, A. Gerstlauer, High-level synthesis of approximate hardware under joint precision and voltage scaling, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2017, pp. 187–192.

[103] M.T. Leipnitz, G.L. Nazar, High-level synthesis of approximate designs under real-time constraints, ACM Trans. Embedded Comput. Syst. 18 (5) (2019) 1–21.

[104] Y. Dou, C. Wang, R. Woods, W. Liu, ENAP: An efficient number-aware pruning framework for design space exploration of approximate configurations, IEEE Trans. Circuits Syst. I. Regul. Pap. (2023) 1–12.

[105] R. Hrbacek, V. Mrazek, Z. Vasicek, Automatic design of approximate circuits by means of multi-objective evolutionary algorithms, in: International Conference on Design and Technology of Integrated Systems in Nanoscale Era, IEEE, 2016, pp. 1–6.

[106] I. Scarabottolo, G. Ansaloni, L. Pozzi, Circuit carving: A methodology for the design of approximate hardware, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2018, pp. 545–550.

[107] J. Castro-Godínez, J. Mateus-Vargas, M. Shafique, J. Henkel, AxHLS: Design space exploration and high-level synthesis of approximate accelerators using approximate functional units and analytical models, in: IEEE/ACM International Conference on Computer Aided Design, IEEE, 2020, pp. 1–9.

[108] M. Awais, H.G. Mohammadi, M. Platzner, An MCTS-based framework for synthesis of approximate circuits, in: IFIP/IEEE International Conference on Very Large Scale Integration, IEEE, 2018, pp. 219–224.

[109] K. Nepal, Y. Li, R.I. Bahar, S. Reda, ABACUS: A technique for automated behavioral synthesis of approximate computing circuits, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2014, pp. 1–6.

[110] B. Salami, O.S. Unsal, A.C. Kestelman, Comprehensive evaluation of supply voltage underscaling in FPGA on-chip memories, in: 51st International Symposium on Microarchitecture, IEEE, 2018, pp. 724–736.

[111] J. Echavarria, K. Schütz, A. Becher, S. Wildermann, J. Teich, Can approximate computing reduce power consumption on FPGAs? in: 25th IEEE International Conference on Electronics, Circuits and Systems, IEEE, 2018, pp. 841–844.

[112] Y.-H. Seo, D.-W. Kim, A new VLSI architecture of parallel multiplier–accumulator based on radix-2 modified booth algorithm, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 18 (2) (2009) 201–208.

[113] N.P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al., In-datacenter performance analysis of a tensor processing unit, in: 44th Annual International Symposium on Computer Architecture, IEEE, 2017, pp. 1–12.

[114] S. Venkataramani, V.K. Chippa, S.T. Chakradhar, K. Roy, A. Raghunathan, Quality programmable vector processors for approximate computing, in: 46th International Symposium on Microarchitecture, IEEE, 2013, pp. 1–12.

[115] K.D. Vogeleer, G. Memmi, P. Jouvelot, F. Coelho, The energy/frequency convexity rule: Modeling and experimental validation on mobile devices, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2013, pp. 793–803.

[116] A. Weissel, F. Bellosa, Process cruise control: Event-driven clock scaling for dynamic power management, in: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, ACM, 2002, pp. 238–246.

[117] S. Amanollahi, M. Kamal, A. Afzali-Kusha, M. Pedram, Circuit-level techniques for logic and memory blocks in approximate computing systemsx, Proc. IEEE 108 (12) (2020) 2150–2177.

[118] S. Basu, L. Duch, M. Peón-Quirós, D. Atienza, G. Ansaloni, L. Pozzi, Heterogeneous and inexact: Maximizing power efficiency of edge computing sensors for health monitoring applications, in: IEEE International Symposium on Circuits and Systems, IEEE, 2018, pp. 1–5.

[119] G. Ndour, T.T. Jost, A. Molnos, Y. Durand, A. Tisserand, Evaluation of variable bit-width units in a RISC-V processor for approximate computing, in: 16th International Conference on Computing Frontiers, ACM, 2019, pp. 344–349.

[120] X. He, S. Jiang, W. Lu, G. Yan, Y. Han, X. Li, Exploiting the potential of computation reuse through approximate computing, IEEE Trans. Multi-Scale Comput. Syst. 3 (3) (2016) 152–165.

[121] J.S. Miguel, J. Albericio, A. Moshovos, N.E. Jerger, Doppelgänger: A cache for approximate computing, in: 48th International Symposium on Microarchitecture, ACM, 2015, pp. 50–61.

[122] T. Moreau, M. Wyse, J. Nelson, A. Sampson, H. Esmaeilzadeh, L. Ceze, M. Oskin, SNNAP: Approximate computing on programmable SoCs via neural acceleration, in: International Symposium on High Performance Computer Architecture, IEEE, 2015, pp. 603–614.

[123] H. Song, C. Xu, Q. Xu, Z. Song, N. Jing, X. Liang, L. Jiang, Invocation-driven neural approximate computing with a multiclass-classifier and multiple approximators, in: 37th International Conference on Computer-Aided Design, ACM, 2018, pp. 1–8.

[124] B. Nongpoh, R. Ray, M. Das, A. Banerjee, Enhancing speculative execution with selective approximate computing, ACM Trans. Des. Autom. Electron. Syst. 24 (2) (2019) 1–29.

[125] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. 28 (2015).

[126] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, M. Shafique, X-CGRA: An energy-efficient approximate coarse-grained reconfigurable architecture, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (10) (2019) 2558–2571.

[127] J. Dickerson, I. Galanis, Z.-G. Tasoulas, L. Kinley, I. Anagnostopoulos, Adaptive approximate computing on hardware accelerators targeting Internet-of-Things, in: 6th World Forum on Internet of Things, IEEE, 2020, pp. 1–6.

[128] G. Ascia, V. Catania, S. Monteleone, M. Palesi, D. Patti, J. Jose, V.M. Salerno, Exploiting data resilience in wireless network-on-chip architectures, ACM J. Emerg. Technol. Comput. Syst. 16 (2) (2020) 1–27.

[129] S. Xiao, X. Wang, M. Palesi, A.K. Singh, L. Wang, T. Mak, On performance optimization and quality control for approximate-communication-enabled networks-on-chip, IEEE Trans. Comput. 70 (11) (2020) 1817–1830.

[130] V. Fernando, A. Franques, S. Abadal, S. Misailovic, J. Torrellas, Replica: A wireless manycore for communication-intensive and approximate data, in: 24th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2019, pp. 849–863.

[131] M.R. Gkeka, A. Patras, C. Antonopoulos, S. Lalis, N. Bellas, FPGA architectures for approximate dense SLAM computing, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2021, pp. 828–833.

[132] S.C. Mukhopadhyay, S.K.S. Tyagi, N.K. Suryadevara, V. Piuri, F. Scotti, S. Zeadally, Artificial intelligence-based sensors for next generation IoT applications: A review, IEEE Sens. J. 21 (22) (2021) 24920–24932.

[133] A.R. Aslam, T. Iqbal, M. Aftab, W. Saadeh, M.A.B. Altaf, A $10.13\mu J$/classification 2-channel deep neural network-based SoC for emotion detection of autistic children, in: IEEE Custom Integrated Circuits Conference, IEEE, 2020, pp. 1–4.

[134] Google and/or its affiliates, Edge TPU, 2019, https://cloud.google.com/edge-tpu.

[135] Xilinx and/or its affiliates, DPU for convolutional neural network, 2021, https://www.xilinx.com/products/intellectual-property/dpu.html.

[136] B. Liu, H. Qin, Y. Gong, W. Ge, M. Xia, L. Shi, EERA-ASR: An energy-efficient reconfigurable architecture for automatic speech recognition with hybrid DNN and approximate computing, IEEE Access 6 (2018) 52227–52237.

[137] L. Klemmer, S. Froehlich, R. Drechsler, D. Große, XbNN: Enabling CNNs on edge devices by approximate on-chip dot product encoding, in: IEEE International Symposium on Circuits and Systems, IEEE, 2021, pp. 1–5.

[138] Y. Gong, B. Liu, W. Ge, L. Shi, ARA: Cross-layer approximate computing framework based reconfigurable architecture for CNNs, Microelectron. J. 87 (2019) 33–44.

[139] M. Imani, A. Rahimi, D. Kong, T. Rosing, J. Rabaey, Exploring hyperdimensional associative memory, in: 23rd International Symposium on High Performance Computer Architecture, IEEE, 2017, pp. 445–456.

[140] B. Khaleghi, S. Salamat, A. Thomas, F. Asgarinejad, Y. Kim, T. Rosing, SHEAR er: Highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation, in: International Symposium on Low Power Electronics and Design, ACM, 2020, pp. 241–246.

[141] Y. Zhou, J. Lin, Z. Wang, Energy efficient SVM classifier using approximate computing, in: 12th International Conference on ASIC, IEEE, 2017, pp. 1045–1048.

[142] H. Sun, X. Liu, Q. Deng, W. Jiang, S. Luo, Y. Ha, Efficient FPGA implementation of K-nearest-neighbor search algorithm for 3D LIDAR localization and mapping in smart vehicles, IEEE Trans. Circuits Syst. II 67 (9) (2020) 1644–1648.

[143] H. Almurib, T.N. Kumar, F. Lombardi, Approximate DCT image compression using inexact computing, IEEE Trans. Comput. 67 (2) (2017) 149–159.

[144] G. Karakonstantis, N. Banerjee, K. Roy, Process-variation resilient and voltage-scalable DCT architecture for robust low-power computing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 18 (10) (2009) 1461–1470.

[145] G. Karakonstantis, D. Mohapatra, K. Roy, Logic and memory design based on unequal error protection for voltage-scalable, robust and adaptive DSP systems, J. Signal Process. Syst. 68 (2012) 415–431.

[146] W. El-Harouni, S. Rehman, B.S. Prabakaran, A. Kumar, R. Hafiz, M. Shafique, Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2017, pp. 1384–1389.

[147] C. Sau, F. Palumbo, M. Pelcat, J. Heulot, E. Nogues, D. Menard, P. Meloni, L. Raffo, Challenging the best HEVC fractional pixel FPGA interpolators with reconfigurable and multifrequency approximate computing, IEEE Embedded Syst. Lett. 9 (3) (2017) 65–68.

[148] L.B. Soares, J. Oliveira, E.A.C. da Costa, S. Bampi, An energy-efficient and approximate accelerator design for real-time canny edge detection, Circuits Systems Signal Process. 39 (2020) 6098–6120.

[149] M. Martina, G. Masera, M.R. Roch, G. Piccinini, Result-biased distributed-arithmetic-based filter architectures for approximately computing the DWT, IEEE Trans. Circuits Syst. I. Regul. Pap. 62 (8) (2015) 2103–2113.

[150] E.S. Manolakos, I. Stamoulias, IP-cores design for the kNN classifier, in: IEEE International Symposium on Circuits and Systems, IEEE, 2010, pp. 4133–4136.

[151] H.M. Hussain, K. Benkrid, H. Seker, An adaptive implementation of a dynamically reconfigurable K-Nearest neighbour classifier on FPGA, in: NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, 2012, pp. 205–212.

[152] D. Tong, Y.R. Qu, V.K. Prasanna, Accelerating decision tree based traffic classification on FPGA and multicore platforms, IEEE Trans. Parallel Distrib. Syst. 28 (11) (2017) 3046–3059.

[153] A. Alcolea, J. Resano, FPGA accelerator for gradient boosting decision trees, Electronics 10 (3) (2021) 314.

[154] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, T. Azumi, Autoware on board: Enabling autonomous vehicles with embedded systems, in: ACM/IEEE 9th International Conference on Cyber-Physical Systems, IEEE, 2018, pp. 287–296.

[155] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, A. Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, J. Mach. Learn. Res. 22 (1) (2021) 10882–11005.

[156] Z. Lin, M. Courbariaux, R. Memisevic, Y. Bengio, Neural networks with few multiplications, 2015, p. 9, arXiv preprint arXiv:1510.03009.

[157] G. Venkatesh, E. Nurvitadhi, D. Marr, Accelerating deep convolutional networks using low-precision and sparsity, in: IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2017, pp. 2861–2865.

[158] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-Net: ImageNet classification using binary convolutional neural networks, in: European Conference on Computer Vision, Springer, 2016, pp. 525–542.

[159] V. Vanhoucke, A. Senior, M.Z. Mao, Improving the speed of neural networks on CPUs, in: Deep Learning and Unsupervised Feature Learning Workshop, 2011.

[160] J. Choi, Z. Wang, S. Venkataramani, P.I.-J. Chuang, V. Srinivasan, K. Gopalakrishnan, PACT: Parameterized clipping activation for quantized neural networks, 2018, p. 15, arXiv preprint arXiv:1805.06085.

[161] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, A. Joulin, Training with quantization noise for extreme model compression, 2020, p. 20, arXiv preprint arXiv:2004.07320.

[162] Y. Yuan, C. Chen, X. Hu, S. Peng, Towards low-bit quantization of deep neural networks with limited data, in: 25th International Conference on Pattern Recognition, IEEE, 2021, pp. 4377–4384.

[163] Y. Yuan, C. Chen, X. Hu, S. Peng, EvoQ: Mixed precision quantization of DNNs via sensitivity guided evolutionary search, in: International Joint Conference on Neural Networks, IEEE, 2020, pp. 1–8.

[164] H.-T. Kung, B. McDanel, S.Q. Zhang, Term quantization: Furthering quantization at run time, in: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2020, pp. 1–16.

[165] D.D. Lin, S.S. Talathi, V.S. Annapureddy, Fixed point quantization of deep convolutional networks, 2015, p. 10, arXiv preprint arXiv:1511.06393.

[166] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, R. Fan, Fully quantized network for object detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 2810–2819.

[167] Q. Huang, D. Wang, Z. Dong, Y. Gao, Y. Cai, T. Li, B. Wu, K. Keutzer, J. Wawrzynek, Codenet: Efficient deployment of input-adaptive object detection on embedded FPGAs, in: ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ACM, 2021, pp. 206–216.

[168] Q. Jin, L. Yang, Z. Liao, Adabits: Neural network quantization with adaptive bit-widths, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2146–2156.

[169] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S.J. Hwang, C. Choi, Learning to quantize deep networks by optimizing quantization intervals with task loss, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 4350–4359.

[170] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, X.-s. Hua, Quantization networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 7308–7316.

[171] X. Zhu, W. Zhou, H. Li, Adaptive layerwise quantization for deep neural network compression, in: IEEE International Conference on Multimedia and Expo, IEEE, 2018, pp. 1–6.

[172] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, 2016, p. 13, arXiv preprint arXiv:1608.08710.

[173] J.-H. Luo, J. Wu, W. Lin, ThiNet: A filter level pruning method for deep neural network compression, in: IEEE International Conference on Computer Vision, IEEE, 2017, pp. 5068–5076.

[174] X. Xiao, Z. Wang, S. Rajasekaran, AutoPrune: Automatic network pruning by regularizing auxiliary parameters, Adv. Neural Inf. Process. Syst. 32 (2019).

[175] S. Ahn, S.X. Hu, A. Damianou, N.D. Lawrence, Z. Dai, Variational information distillation for knowledge transfer, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 9163–9171.

[176] L. Liu, Q. Huang, S. Lin, H. Xie, B. Wang, X. Chang, X. Liang, Exploring inter-channel correlation for diversity-preserved knowledge distillation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2021, pp. 8271–8280.

[177] F. Sarfraz, E. Arani, B. Zonooz, Knowledge distillation beyond model compression, in: 25th International Conference on Pattern Recognition, IEEE, 2021, pp. 6136–6143.

[178] S. Swaminathan, D. Garg, R. Kannan, F. Andres, Sparse low rank factorization for deep neural network compression, Neurocomputing 398 (2020) 185–196.

[179] J. Guo, Y. Li, W. Lin, Y. Chen, J. Li, Network decoupling: From regular to depthwise separable convolutions, 2018, p. 12, arXiv preprint arXiv:1808.05517.

[180] P. Indyk, A. Vakilian, Y. Yuan, Learning-based low-rank approximations, Adv. Neural Inf. Process. Syst. 32 (2019).

[181] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, 2015, p. 16, arXiv preprint arXiv:1511.06530.

[182] K. Mitsuno, J. Miyao, T. Kurita, Hierarchical group sparse regularization for deep convolutional neural networks, in: International Joint Conference on Neural Networks, IEEE, 2020, pp. 1–8.

[183] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, J. Xin, Understanding straight-through estimator in training activation quantized neural nets, 2019, p. 30, arXiv preprint arXiv:1903.05662.

[184] J. Lin, Y. Rao, J. Lu, J. Zhou, Runtime neural pruning, Adv. Neural Inf. Process. Syst. 30 (2017).

[185] J. Czarnowski, T. Laidlow, R. Clark, A.J. Davison, Deepfactors: Real-time probabilistic dense monocular SLAM, IEEE Robot. Autom. Lett. 5 (2) (2020) 721–728.

[186] SAE International and/or its affiliates, Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, Tech. Rep. J3016, SAE International, 2021, URL https://www.sae.org/standards/content/j3016/.

[187] P. Coppola, F. Silvestri, Autonomous vehicles and future mobility solutions, in: Autonomous Vehicles and Future Mobility, Elsevier, 2019, pp. 1–15.

[188] Bosch Mobility Solutions, Sense, think, act: What automated vehicles need to be capable of, 2023, https://www.bosch-mobility-solutions.com/en/mobility-topics/automated-driving-sense-think-act/.

[189] Siemens Digital Industries Software and/or its affiliates, The Sense-Think-Act Model, Tech. Rep. 81278-C4, Siemens Digital Industries Software, 2020, URL https://www.plm.automation.siemens.com/media/global/de/Siemens-SW-The%20sense-think-act-model-White%20Paper_tcm53-81439.pdf.

[190] B. Krzanich, Data is the New Oil in the Future of Automated Driving, Tech. Rep., Intel Corporation, 2016, URL https://download.intel.com/newsroom/2021/archive/2016-11-15-editorials-krzanich-the-future-of-automated-driving.pdf.

[191] M. Krail, On autopilot to a more efficient future? How data processing by connected and autonomous vehicles will impact energy consumption, Tech. Rep. 53-2021-EN, Agora Verkehrswende, 2021, URL https://www.agora-verkehrswende.de/en/publications/on-autopilot-to-a-more-efficient-future/.

[192] X. Long, J. Wu, L. Chen, Energy-efficient offloading in mobile edge computing with edge-cloud collaboration, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2018, pp. 460–475.

[193] J. Shao, H. Zhang, Y. Mao, J. Zhang, Branchy-GNN: A device-edge co-inference framework for efficient point cloud processing, in: IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2021, pp. 8488–8492.

[194] J. Santa, P.J. Fernández, J. Ortiz, R. Sanchez-Iborra, A.F. Skarmeta, SURROGATES: Virtual OBUs to foster 5G vehicular services, Electronics 8 (2) (2019) 117.

[195] J. Santa, J. Ortiz, P.J. Fernandez, M. Luis, C. Gomes, J. Oliveira, D. Gomes, R. Sanchez-Iborra, S. Sargento, A.F. Skarmeta, MIGRATE: Mobile device virtualisation through state transfer, IEEE Access 8 (2020) 25848–25862.

[196] J. Feng, Z. Liu, C. Wu, Y. Ji, AVE: Autonomous vehicular edge computing framework with ACO-based scheduling, IEEE Trans. Veh. Technol. 66 (12) (2017) 10660–10675.

[197] J. Tang, S. Liu, B. Yu, W. Shi, PI-edge: A low-power edge computing system for real-time autonomous driving services, 2018, p. 13, arXiv preprint arXiv:1901.04978.

[198] J. Tang, S. Liu, L. Liu, B. Yu, W. Shi, LoPECS: A low-power edge computing system for real-time autonomous driving services, IEEE Access 8 (2020) 30467–30479.

[199] H. Ibn-Khedher, M. Laroui, M.B. Mabrouk, H. Moungla, H. Afifi, A.N. Oleari, A.E. Kamal, Edge computing assisted autonomous driving using artificial intelligence, in: International Wireless Communications and Mobile Computing, IEEE, 2021, pp. 254–259.

[200] D. Katare, M. El-Sharkawy, Embedded system enabled vehicle collision detection: An ANN classifier, in: IEEE 9th Annual Computing and Communication Workshop and Conference, IEEE, 2019, pp. 284–289.

[201] M. Lechner, R. Hasani, A. Amini, T.A. Henzinger, D. Rus, R. Grosu, Neural circuit policies enabling auditable autonomy, Nat. Mach. Intell. 2 (10) (2020) 642–652.

[202] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.

[203] E. Talpes, D.D. Sarma, G. Venkataramanan, P. Bannon, B. McGee, B. Floering, A. Jalote, C. Hsiong, S. Arora, A. Gorti, et al., Compute solution for Tesla's full self-driving computer, IEEE Micro 40 (2) (2020) 25–35.

[204] M. Koschuch, W. Sebron, Z. Szalay, Á. Török, H. Tschiürtz, I. Wahl, Safety & security in the context of autonomous driving, in: IEEE International Conference on Connected Vehicles and Expo, IEEE, 2019, pp. 1–7.

[205] A. Ramamoorthy, Automotive Megatrends and Their Impact on Memory and Storage, Tech. Rep., Micron Technology, 2023, URL https://media-www.micron.com/-/media/client/global/documents/solutions/automotive/automotive_megatrends_white_paper.pdf.

[206] J. Gehrke, S. Madden, Query processing in sensor networks, IEEE Pervasive Comput. 3 (1) (2004) 46–55.

[207] F. Pereira, R. Correia, P. Pinho, S.I. Lopes, N.B. Carvalho, Challenges in resource-constrained IoT devices: Energy and communication as critical success factors for future IoT deployment, Sensors 20 (22) (2020) 6420.

[208] Z. Taufique, B. Zhu, G. Coppola, M. Shoaran, M.A.B. Altaf, A low power multi-class migraine detection processor based on somatosensory evoked potentials, IEEE Trans. Circuits Syst. II 68 (5) (2021) 1720–1724.

[209] J. Yoo, L. Yan, D. El-Damak, M.B. Altaf, A. Shoeb, H.-J. Yoo, A. Chandrakasan, An 8-channel scalable EEG acquisition SoC with fully integrated patient-specific seizure classification and recording processor, in: IEEE International Solid-State Circuits Conference, IEEE, 2012, pp. 292–294.

[210] X. Liu, O. Baiocchi, A comparison of the definitions for smart sensors, smart objects and things in IoT, in: IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference, IEEE, 2016, pp. 1–4.

[211] D.J. Pagliari, M. Poncino, On the impact of smart sensor approximations on the accuracy of machine learning tasks, Heliyon 6 (12) (2020) e05750.

[212] Z. Taufique, A. Kanduri, M.A.B. Altaf, P. Liljeberg, Approximate feature extraction for low power epileptic seizure prediction in wearable devices, in: IEEE Nordic Circuits and Systems Conference, IEEE, 2021, pp. 1–7.

[213] K. Bregar, T. Krištofelc, M. Depolli, V. Avbelj, A. Rashkovska, Power autonomy estimation of low-power sensor for long-term ECG monitoring, Sensors 22 (14) (2022) 5070.

[214] H. Zong, P. Brimblecombe, L. Sun, P. Wei, K.-F. Ho, Q. Zhang, J. Cai, H. Kan, M. Chu, W. Che, et al., Reducing the influence of environmental factors on performance of a diffusion-based personal exposure kit, Sensors 21 (14) (2021) 4637.

[215] H. Markandeya, G. Karakonstantis, S. Raghunathan, P. Irazoqui, K. Roy, Low-power DWT-based quasi-averaging algorithm and architecture for epileptic seizure detection, in: 16th ACM/IEEE International Symposium on Low Power Electronics and Design, ACM, 2010, pp. 301–306.

[216] A. Subasi, M.I. Gursoy, EEG signal classification using PCA, ICA, LDA and support vector machines, Expert Syst. Appl. 37 (12) (2010) 8659–8666.

[217] G. Karakonstantis, A. Sankaranarayanan, M.M. Sabry, D. Atienza, A. Burg, A quality-scalable and energy-efficient approach for spectral analysis of heart rate variability, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE, 2014, pp. 1–6.

[218] G. Zhu, Y. Li, P.P. Wen, Epileptic seizure detection in EEGs signals using a fast weighted horizontal visibility algorithm, Comput. Methods Programs Biomed. 115 (2) (2014) 64–75.

[219] O. Faust, U.R. Acharya, H. Adeli, A. Adeli, Wavelet-based EEG processing for computer-aided seizure detection and epilepsy diagnosis, Seizure 26 (2015) 56–64.

[220] A. Ghosh, A. Raha, A. Mukherjee, Energy-efficient IoT-health monitoring system using approximate computing, Internet Things 9 (2020) 100166.

[221] C. Eleftheriadis, G. Karakonstantis, Fast and accurate power spectral analysis of heart rate variability using fast Gaussian gridding, Comput. Cardiol. 48 (2021) 1–4.

[222] B. Liu, X. Ding, H. Cai, W. Zhu, Z. Wang, W. Liu, J. Yang, Precision adaptive MFCC based on R2SDF-FFT and approximate computing for low-power speech keywords recognition, IEEE Circuits Syst. Mag. 21 (4) (2021) 24–39.

[223] A. Muneeb, M. Ali, M.A.B. Altaf, A 2.7 μJ/classification machine-learning based approximate computing seizure detection SoC, in: IEEE International Symposium on Circuits and Systems, IEEE, 2022, pp. 55–59.

[224] W.B. Qaim, A. Ometov, C. Campolo, A. Molinaro, E.S. Lohan, J. Nurmi, Understanding the performance of task offloading for wearables in a two-tier edge architecture, in: 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, IEEE, 2021, pp. 1–9.

[225] A. George, A. Ravindran, Scalable approximate computing techniques for latency and bandwidth constrained IoT edge, in: International Summit Smart City 360°, Springer, 2020, pp. 274–292.

[226] Z. Wen, P. Bhatotia, R. Chen, M. Lee, et al., ApproxIoT: Approximate analytics for edge computing, in: IEEE 38th International Conference on Distributed Computing Systems, IEEE, 2018, pp. 411–421.

[227] A.R. Zamani, I. Petri, J. Diaz-Montes, O. Rana, M. Parashar, Edge-supported approximate analysis for long running computations, in: IEEE 5th International Conference on Future Internet of Things and Cloud, IEEE, 2017, pp. 321–328.

[228] M.A. Scrugli, D. Loi, L. Raffo, P. Meloni, A runtime-adaptive cognitive IoT node for healthcare monitoring, in: 16th ACM International Conference on Computing Frontiers, ACM, 2019, pp. 350–357.

[229] M.A. Sohail, Z. Taufique, S.M. Abubakar, W. Saadeh, M.A.B. Altaf, An ECG processor for the detection of eight cardiac arrhythmias with minimum false alarms, in: IEEE Biomedical Circuits and Systems Conference, BioCAS, IEEE, 2019, pp. 1–4.

[230] X. Yu, P. Chum, K.-B. Sim, Analysis the effect of PCA for feature reduction in non-stationary EEG based motor imagery of BCI system, Optik 125 (3) (2014) 1498–1502.
[231] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, John Wiley & Sons, Inc., 2001.
[232] H. Rajaguru, S.K. Prabhakar, Power spectral density and KNN based adaboost classifier for epilepsy classification from EEG, in: International Conference of Electronics, Communication and Aerospace Technology, IEEE, 2017, pp. 441–444.
[233] E. Svertoka, S. Saafi, A. Rusu-Casandra, R. Burget, I. Marghescu, J. Hosek, A. Ometov, Wearables for industrial work safety: A survey, Sensors 21 (11) (2021) 3844.
[234] J. Lee, B. Varghese, R. Woods, H. Vandierendonck, TOD: Transprecise object detection to maximise real-time accuracy on the edge, in: IEEE 5th International Conference on Fog and Edge Computing, IEEE, 2021, pp. 53–60.
[235] B. Maity, B. Donyanavard, A. Surhonne, A. Rahmani, A. Herkersdorf, N. Dutt, SEAMS: Self-optimizing runtime manager for approximate memory hierarchies, ACM Trans. Embed. Comput. Syst. 20 (5) (2021) 1–26.
[236] X. Meng, H. Wang, B. Liu, A robust vehicle localization approach based on GNSS/IMU/DMI/LiDAR sensor fusion for autonomous vehicles, Sensors 17 (9) (2017) 2140.
[237] S. Narayana, R.V. Prasad, V. Rao, L. Mottola, T.V. Prabhakar, Hummingbird: Energy efficient GPS receiver for small satellites, in: 26th Annual International Conference on Mobile Computing and Networking, ACM, 2020, pp. 1–13.
[238] E.D. Kaplan, C.J. Hegarty, Understanding GPS, Principles and Applications, third ed, Artech House, 2017.
[239] J. Leclère, R. Landry Jr., C. Botteron, Comparison of L1 and L5 bands GNSS signals acquisition, Sensors 18 (9) (2018) 2779.
[240] J. Torres-Sospedra, I. Silva, L. Klus, D. Quezada-Gaibor, A. Crivello, P. Barsocchi, C. Pendão, E.S. Lohan, J. Nurmi, A. Moreira, Towards ubiquitous indoor positioning: Comparing systems across heterogeneous datasets, in: International Conference on Indoor Positioning and Indoor Navigation, IEEE, 2021, pp. 1–8.
[241] Y. Lu, M. Gerasimenko, R. Kovalchukov, M. Stusek, J. Urama, J. Hosek, M. Valkama, E.S. Lohan, Feasibility of location-aware handover for autonomous vehicles in industrial multi-radio environments, Sensors 20 (21) (2020) 6290.
[242] B. Yang, Projection approximation subspace tracking, IEEE Trans. Signal Process. 43 (1) (1995) 95–107.
[243] F. Van Diggelen, A-GPS: Assisted GPS, GNSS, and SBAS, Artech House, 2009.
[244] Z. Chen, G. Gokeda, Y. Yu, Introduction to Direction-of-Arrival Estimation, Artech House, 2010.
[245] J. Liu, B. Priyantha, T. Hart, H.S. Ramos, A.A. Loureiro, Q. Wang, Energy efficient GPS sensing with cloud offloading, in: 10th ACM Conference on Embedded Network Sensor Systems, ACM, 2012, pp. 85–98.
[246] P. Misra, W. Hu, Y. Jin, J. Liu, A.S. de Paula, N. Wirström, T. Voigt, Energy efficient GPS acquisition with sparse-GPS, in: 13th International Symposium on Information Processing in Sensor Networks, ACM, 2014, pp. 155–166.
[247] V. Bellad, Intermittent GNSS Signal Tracking for Improved Receiver Power Performance (Ph.D. thesis), University of Calgary, 2015, URL https://prism.ucalgary.ca/handle/11023/2667.
[248] Y. Zhang, M. Wang, Y. Li, Low computational signal acquisition for GNSS receivers using a resampling strategy and variable circular correlation time, Sensors 18 (2) (2018) 678.
[249] European GNSS Agency and/or its affiliates, Power-Efficient Positioning for the Internet of Things, Tech. Rep. TS-02-20-382-EN-N, European GNSS Agency, 2020, URL https://www.euspa.europa.eu/newsroom/news/power-efficient-positioning-iot.
[250] J. Svatoň, F. Vejražka, P. Kubalík, J. Schmidt, J. Boreckỳ, Novel partial correlation method algorithm for acquisition of GNSS tiered signals, NAVIGATION: J. Inst. Navig. 67 (4) (2020) 745–762.
[251] Q. Jie, X. Wang, Y. Chen, F. Shu, X. Zhan, P. Zhang, A rapid power-iterative root-MUSIC estimator for massive/ultra-massive MIMO receiver, 2022, p. 6, arXiv preprint arXiv:2205.03269.
[252] T. Troccoli, J. Pirskanen, A. Ometov, J. Nurmi, V. Kaseva, Implementation of embedded multiple signal classification algorithm for mesh IoT networks, in: International Conference on Localization and GNSS, IEEE, 2022, pp. 1–7.
[253] G. Pau, F. Arena, Y.E. Gebremariam, I. You, Bluetooth 5.1: An analysis of direction finding capability for high-precision location services, Sensors 21 (11) (2021) 3589.
[254] R. Bembenik, K. Falcman, BLE indoor positioning system using RSSI-based trilateration, J. Wirel. Mob. Netw. Ubiquitous Comput. Depend. Appl. 11 (3) (2020) 50–69.
[255] T. Troccoli, J. Pirskanen, J. Nurmi, A. Ometov, J. Morte, E.S. Lohan, V. Kaseva, Direction of arrival method for L-shaped array with RF switch: An embedded implementation perspective, Sensors 23 (6) (2023) 3356.
[256] B. Heidtmann, Low-Power GNSS for Tracking Applications, Tech. Rep., Ublox, 2021, URL https://www.u-blox.com/en/publication/white-paper/low-power-gps-tracking-applications.
[257] T. Feng, Decimation double-phase estimator: An efficient and unambiguous high-order binary offset carrier tracking algorithm, IEEE Signal Process. Lett. 23 (7) (2016) 905–909.
[258] H.S. Ramos, T. Zhang, J. Liu, N.B. Priyantha, A. Kansal, LEAP: A low energy assisted GPS for trajectory-based services, in: 13th International Conference on Ubiquitous Computing, ACM, 2011, pp. 335–344.
[259] J.A. Fessler, A.O. Hero, Space-alternating generalized expectation-maximization algorithm, IEEE Trans. Signal Process. 42 (10) (1994) 2664–2677.
[260] S.A. Vorobyov, Principles of minimum variance robust adaptive beamforming design, Signal Process. 93 (12) (2013) 3264–3277.
[261] R. Roy, T. Kailath, ESPRIT-estimation of signal parameters via rotational invariance techniques, IEEE Trans. Acoust. Speech Signal Process. 37 (7) (1989) 984–995.
[262] R. Schmidt, Multiple emitter location and signal parameter estimation, IEEE Trans. Antennas and Propagation 34 (3) (1986) 276–280.
[263] O.A. Oumar, M.F. Siyau, T.P. Sattar, Comparison between MUSIC and ESPRIT direction of arrival estimation algorithms for wireless communication systems, in: The First International Conference on Future Generation Communication Technologies, IEEE, 2012, pp. 99–103.
[264] R.M. Gray, L.D. Davisson, An Introduction to Statistical Signal Processing, Cambridge University Press, 2004.
[265] A. Barabell, Improving the resolution performance of eigenstructure-based direction-finding algorithms, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE, 1983, pp. 336–339.
[266] I. Scarabottolo, G. Ansaloni, G.A. Constantinides, L. Pozzi, S. Reda, Approximate logic synthesis: A survey, Proc. IEEE 108 (12) (2020) 2195–2213.
[267] C. Fernandez-Prades, J. Arribas, P. Closas, C. Aviles, L. Esteve, GNSS-SDR: An open source tool for researchers and developers, in: 24th International Technical Meeting of the Satellite Division of the Institute of Navigation, 2011, pp. 780–794.
[268] Z. Bhuiyan, et al., The FGI-gsrx software defined GNSS receiver goes open source, 2022, https://www.maanmittauslaitos.fi/en/topical_issues/fgi-gsrx-software-defined-gnss-receiver-goes-open-source.
[269] K. Chen, J. Han, F. Lombardi, Two approximate voting schemes for reliable computing, IEEE Trans. Comput. 66 (7) (2017) 1227–1239.
[270] J.H. Anajemba, J.A. Ansere, F. Sam, C. Iwendi, G. Srivastava, Optimal soft error mitigation in wireless communication using approximate logic circuits, Sustain. Comput.: Inform. Syst. 30 (2021) 100521.

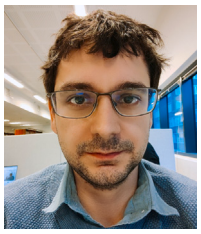
Hans Jakob Damsgaard received the B.Sc. degree in electrical engineering in 2019 and the M.Sc. degree in computer science and engineering in 2021 as part of the Honours programme at the Technical University of Denmark, DTU. He is currently pursuing a Ph.D. in reconfigurable approximate accelerators for secure edge computing at Tampere University, TAU. He received the best paper award at NorCAS'21. His current research interests include hardware accelerators, networks-on-chip, computer architecture, and hardware verification.

Antoine Grenier received his M.E. in geodesy and geomatics from the Ecole Supèrieure des Géométres et Topographes (France) in 2019. He then worked at the European Space Agency (Netherlands) in the Galileo System Engineering section from 2019 to 2021 as a Young Graduate Trainee. He is currently pursuing a Ph.D. in approximation techniques for low-cost GNSS receivers at Tampere University (TAU). His current research interests include low-cost GNSS receivers, GNSS processing algorithms, Software-Defined Radios and GNSS measurements on Android devices.

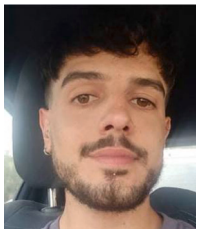
Dewant Katare received his B.Sc. degree in electrical and electronic engineering in 2016 at The Netherlands, and M.S. degree in electrical and computer engineering in 2019 from Purdue University, USA. He is currently a Ph.D. candidate, researching on the topics of energy efficient approximate Edge-AI for automated driving services at the department of engineering systems and services, Delft University of Technology, The Netherlands. His current research interests include autonomous driving, distributed ML, Edge-AI and model approximations.

Zain Taufique received his B.Sc. degree in electrical engineering in 2015 at University of Engineering and Technology, Lahore UET. He received his M.Sc. degree in Electronics and Embedded Systems in 2020 from Lahore University of Management and Sciences, LUMS. He is currently pursuing a Ph.D. in approximate computing for resource constraint devices at University of Turku, UTU. His current research interests include multi-core computer architectures, low power logic designs and edge computing.

Salar Shakibhamedan received his B.Sc. and M.Sc. degrees in electrical engineering in 2015 and 2018 respectively, from K. N. Toosi University of Technology in Iran. He is currently a Ph.D. student, researching on the Approximate Computing for smart sensors and embedded machine learning. In the institute of computer technology at the Vienna university of technology in Austria. His current research interests, include deep learning, Edge-AI, autonomous driving, and approximate computing.



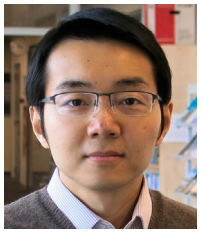
Tiago Troccoli obtained his B.Sc. and M.Sc. degrees in Computer Science from the University of São Paulo and the University of Campinas, respectively, in Brazil. He is currently a Doctorate Researcher responsible for pioneering the novel generation of Indoor localization systems based on direction of arrival (DOA). His research project involves overcoming the challenges of implementing complex numerical methods in Internet of Things networks composed of constrained embedded systems. His research is a collaborative effort between the industry (Wirepas) and academia (University of Tampere) which is supported by the European Union's Horizon 2020 program through the Marie Skodowska-Curie grant agreement.



Georgios Chatzitsompanis received his Diploma M.Eng degree in Electrical Engineering from the National Technical University of Athens (NTUA) in 2020. He is currently working towards his Ph.D. degree in the School of Electronics, Electrical Engineering and Computer Science at the Queen's University of Belfast. His research interests include machine learning, approximate hardware VLSI design for low-power and investigation of process variation tolerant circuits.

Anil Kanduri received the MSc (Tech) degree in embedded computing, in 2014, and the Ph.D. (Tech) degree in computer systems, in 2018, from the University of Turku, Finland. He is currently a senior researcher with the Department of Future Technologies, University of Turku, Finland. His research interests are in on-chip resource management, run-time power and performance modeling and analysis, system software for heterogeneous processors, and alternative computing methods and architectures.

Aleksandr Ometov received the M.Sc. degree in Information Technology and the D.Sc. (Tech.) degree in Telecommunications from the Tampere University of Technology (TUT), Finland, in 2016 and 2018, respectively. He also holds the Specialist degree in Information Security from the Saint Petersburg State University of Aerospace Instrumentation (SUAI) from 2013. He is a Postdoctoral Research Fellow at Tampere University (TAU), Finland, and the coordinator of the CONVERGENCE of Humans and Machines research field funded by Jane and Aatos Erkko Foundation, managing EU H2020 MCSA A-WEAR and APROPOS ITN projects. His research interests include wireless communications, information security, computing paradigms, blockchain technology, and wearable applications.



Aaron Yi Ding is a tenured Associate Professor and leading the Cyber–Physical Intelligence (CPI) Laboratory at TU Delft, Netherlands. He has over 16 years of research and development experience across EU, U.K., and USA, including TU Munich, University of Cambridge, and Columbia University. His research focuses on edge AI solutions for cyber–physical systems in smart health, mobility, and energy domains. He has 80+ peer reviewed publications, receiving best paper awards and recognition from ACM SIGCOMM, ACM EdgeSys, ACM SenSys CCIoT, IEEE INFOCOM, and the esteemed Nokia Foundation Scholarships. He is the scientific director and coordinator for the EU Horizon project SPATIAL. He is an Associate Editor of *ACM Transactions on Internet of Things* (TIOT) and IEEE Open Journal of the Intelligent Transportation Systems.

Nima Taherinejad 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, as the PI of two projects. 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 80 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).

Georgios Karakonstantis is an Associate Professor at the School of Electronics, Electrical Engineering and Computer Science of Queen's University Belfast, United Kingdom. He has published more than 85 papers in peer reviewed journals and conferences, and he is inventor of a US patent and author of two book chapters. He is the recipient of two HiPEAC paper awards, two best paperawards at DATE, and of a prize at the Altera Innovate Design Contest. He received the M.Sc. and Ph.D. degree in Electrical and Computer Engineering from Purdue University, West-Lafayette, USA. His research focuses on energy-efficient and error-resilient computing and storage architectures for embedded and high-performance applications.

Roger Woods received the B.Sc. and Ph.D. degrees from Queen's University Belfast, U.K., in 1985 and 1990. He is currently a Professor of Digital Systems at the School of Electronics, Electrical Engineering and Computer Science and the Dean of Research in the Faculty of Engineering and Physical Sciences, Queen's University Belfast. He is also a Fellow of the Royal Academy of Engineering. His research interests include heterogeneous programmable systems for data analytics and embedded systems for medical and smart city applications. He has published two research books and over 245 leading journals and conference papers. In 2007, he co-founded the spin-off company, Analytics Engines Ltd. with several of his Ph.D. students, and now acts as their Chief Scientist. He has supervised 30 Ph.D. theses. He is one of the technical program committee members for a number of IEEE conferences, including FPL, FPT and SAMOS.

Jari Nurmi received the D.Sc. degree in technology in 1994. He works as a Professor at the Electrical Engineering Unit, Tampere University, TAU (formerly Tampere University of Technology, TUT), Finland, since 1999. He is working on embedded computing systems, System-on-Chip, approximate computing, wireless localization, positioning receiver prototyping, and software-defined radio and software-defined networks. He held various research, education, and management positions at TUT since 1987, (e.g., an Acting Associate Professor from 1991 to 1994) and was the Vice President of the SME VLSI Solution Oy from 1995 to 1998. He has supervised 32 Ph.D. and about 150 M.Sc. theses, and been an opponent or a reviewer of over 50 Ph.D. theses for other universities worldwide. He is a member of the Technical Committee on VLSI Systems and Applications at IEEE CASS. In 2004, he was one of the recipients of the Nokia Educational Award and a recipient of the Tampere Congress Award in 2005. In 2011, he received the IIDA Innovation Award and in 2013 the Scientific Congress Award and the HiPEAC Technology Transfer Award. He is

a steering committee member of four international conferences (chairman in two). He has edited five Springer books and has published over 400 international conference and journal articles and book chapters. He is also an associate editor of three international journals. He is the Director of the national DELTA doctoral training network of about 200 Ph.D. students, the Coordinator of the European doctoral training network APROPOS, and the Head of the A-WEAR European joint Ph.D. Degree Program at TAU. He was also nominated as Tampere Congress Ambassador in 2023 and serves in the board of Visit Tampere Ltd.