Doctoral Dissertation
Doctoral Program in Energy Engineering (36^{th}cycle)

Synthesis of Smart and Intelligent Sensors

By

Chen Xie

******

Supervisor(s):
Prof. Enrico Macii, Supervisor
Prof. Andrea Calimera, Co-Supervisor
Prof. Daniele Jahier Pagliari, Co-Supervisor

Doctoral Examination Committee:
Prof. Jose Ayala, Referee, Complutense University of Madrid
Prof. Leticia Bolzani, Referee, RWTH Aachen University
Prof. Donkyu Baek, Chungbuk National University
Prof. Francesco Regazzoni, Università della Svizzera italiana
Prof. Sara Vinco, Politecnico di Torino

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Chen Xie
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To my mother Liu Ping.

Acknowledgements

During the process of completing my Ph.D., many individuals and organizations have generously supported and assisted me. I would like to express my heartfelt gratitude to them.

First of all, I would like to thank Prof. Enrico Macii, Prof. Andrea Calimera, Prof. Massimo Poncino, and Prof. Daniele Jahier Pagliari for offering me the opportunity to pursue this Ph.D. When I walked into Prof. Massimo Poncino's office and discussed my Master's thesis with him and Prof. Daniele Jahier Pagliari, I had no idea that it would mark the beginning of all the stories. At that moment I only had some rough ideas and aspirations of integrating my electronics background with cutting-edge algorithms, and it was they who made me turn those thoughts into reality, and have become the main theme of my doctoral journey. It was unforgettable when I decided to start my Master's thesis focusing on hardware accelerators with Prof. Massimo Poncino, he commented "You won't regret your choice." He was right.

I would like to thank Prof. Daniele Jahier Pagliari for his thoughtful guidance and support throughout my doctoral research. Not only did he provide me with invaluable academic guidance, but he also offered encouragement and swift and reliable assistance whenever difficulties and challenges happened. He has always been a smart, kind and optimistic person, and his enthusiasm for research infects everyone in the team. He taught me everything to become a researcher by generously sharing his invaluable expertise from academic research to scientific paper writing, and more importantly, by an academically rigorous and proactive attitude. It is no doubt that I will benefit from them throughout my life.

I would like to express my gratitude to the sponsor of my Ph.D. STMicroelectronics, Marco Castellano, and Luca Gandolf, my supervisors in the company, and all of the people in their team. Throughout my research, they and their team have

selflessly assisted and supported me, engaging in discussions with me and sharing their expertise and experiences, which have greatly enriched my research life, and have achieved my original aspirations to closely collaborate with the industrial field and translate my research findings into practical applications. Thanks to Andrea Mineo, who provided me with several lessons related to synthesis. Thanks to Massimo Michele Antonio Sorbera, Sonia Andronaco, and Aflio Basile, they answered many of my questions and doubts during my work. Importantly, from every engineer from STMicroelectronics I met during my Ph.D. life, not only their rich professional experience greatly benefited me, but also their endless patience and kindness deeply influenced me.

I would like to thank the entire STMicroelectronics MEMS Goup R&D in Prague for my research exchange in the context of AMBEATion project, especially to my supervisors Dalibor Barri and Patrik Vacula during my exchange. Their excellent professional knowledge benefits me a lot and expands my research in the EDA field. Importantly, all the kindness and friendship offered by the entire group made the moments in Prague so treasurable. In addition, thanks to my friends in Prague, especially Giorgia. Our laughter once echoed through the streets and alleys of Prague.

I would like to thank everyone in our EDA group and Lab 4, for their support and friendships, making my research life not lonely. Thanks to Alessio Burrello, Francesco Daghero, and Matteo Risso, who were in close collaboration with me. Thanks to all members in the group meeting(or lunch), where we shared thoughts (and food). Thanks to STMicroelectronics Polito, especially Michelangelo Grosso, who offered me support and kindness. Thanks to my friend Dr. Walid, who discussed both techniques and lives with me, and gave me many suggestions. I sincerely give my best wishes for his research career in Pakistan.

Last but not least, thanks to my family. Although I cannot go home during COVID pandemic, they have always been my solid support. Thanks to all my friends, they listened to me, encouraged me, and enriched my life. It is so hard to count them all, but there is one thing with doubts: Without them, I cannot reach here. Thank you all.

Abstract

As the crucial intermediary between the real world and Information Communication Technology (ICT) infrastructures, smart and intelligent sensors play a vital role in extracting valuable information for a variety of tasks from human activity recognition, and people counting, to hand gesture recognition, etc, depending on the type of smart sensors.

The process involves translating raw physical data into electrical signals, and performing data analysis and processing to fulfill specific requirements. With advances in machine learning (ML) and deep learning (DL) algorithms in recent decades, the intelligence of smart sensors has undergone significant development. However, it poses challenges due to the memory and computation-intensive nature inherent in ML/DL models. Moreover, the sensing process involves the transmission of large amounts of data, which consume a significant portion of the energy budget, far beyond the availability of battery-operated smart sensors at the edge.

This Ph.D. thesis aims to devise innovative design solutions for efficient ML architectures, prioritizing energy conservation and minimizing data movement. This involves exploring novel implementation strategies and advanced tools to synthesize and optimize heterogeneous designs targeting smart sensors, especially where the digital and analog domains intersect. The research encompasses various aspects of the design hierarchy, including front-end tasks such as automatic design and optimization of new architectures, as well as back-end tasks such as circuit-level design techniques for optimizing area, power, and digital-to-analog components. These research endeavors are conducted in close collaboration with STMicroelectronics.

This thesis undertakes a comprehensive exploration of the optimization of smart sensor designs; the techniques involved can be categorized into three main aspects: algorithm level, hardware level, and the deployment of advanced EDA tools, targeting each phase in the design flow. All research activities are integrated with

practical applications in smart sensor design. In particular, ML models deployed in smart sensors are optimized through multiple techniques, including quantization and adaptive inference, while specific hardware features are studied to efficiently support such models, including a custom Central Processing Unit (CPU) and efficient approximate computing for I2C buses. In addition, topology recognition in layout placement automation targeting complex analog-mixed signal (AMS) IC design for smart sensors is studied.

The experiments in this thesis are performed on extensive datasets, and run on real-world devices, including both academic open-source platforms and commercial ones. In particular, a smart sensor prototype was realized to demonstrate the effectiveness of the proposed optimization techniques.

Contents

List of Figures **xii**

List of Tables **xiv**

1 Introduction **1**

1.1 Overview of Smart Sensor Design 1

1.1.1 Smart and Intelligent Sensors 2

1.1.2 Design Flow of Smart Sensors 4

1.2 Machine Learning-Enabled Smart Sensors 6

1.3 Energy Efficient Implementation and Optimization 8

1.3.1 Algorithm Employment 9

1.3.2 Hardware Implementation 10

1.3.3 Advanced EDA Tools 10

2 Background **12**

2.1 Algorithms for Efficient Smart Sensors 12

2.1.1 Compact Neural Networks 13

2.1.2 Decision Trees and Tree-based Ensembles 20

2.2 Hardware-level Design and Optimization 23

2.2.1 Hardware Platforms for Smart Sensors 24

2.2.2 Customized Microcontrollers 26

2.3 EDA Tools for Smart Sensors 26
2.3.1 Mixed-signal Integrated Circuit and Smart Sensors 26
2.3.2 Analog Mixed-Signal Back-End Design Automation 28

3 Related Work 31
3.1 Algorithms-Level Techniques 31
3.1.1 ML/DL Algorithms in Smart Sensors 31
3.1.2 Optimizations of Algorithms/System Techniques 34
3.2 Hardware-Level Techniques 36
3.2.1 Customized CPU 36
3.2.2 Approximate Bus Encoding 37
3.3 Electronic Automation Design with Advanced Techniques 38

4 Efficient Machine Learning Algorithms for Smart Sensors 40
4.1 Application 1: Person Counting and Social Distance Monitoring with Multipixel Infrared Sensors 42
4.1.1 Motivation and Context 42
4.1.2 Classic People Counting and Social Distance Monitoring Solutions 46
4.1.3 Limitations in the Existing Solutions 47
4.1.4 Efficient Deep Learning Models for Social Distancing . . . 50
4.1.5 Efficient Deep Learning Models for People Counting 54
4.1.6 Experimental Results 60
4.1.7 Conclusion and Discussion 80
4.2 Application 2: Human Activity Recognition with Acceleration Sensors 85
4.2.1 Motivation and Context 85
4.2.2 The State-of-the-art Human Activity Recognition Solutions . 86
4.2.3 Limitations in the Existing Solutions 86

4.2.4 Proposed RF Implementation 87
4.2.5 Experimental results 89
4.2.6 Conclusion 91
4.3 Application 3: Gesture Recognition with Surface Electromyography Sensors 92
4.3.1 Motivation and Context 92
4.3.2 The State-of-art Gesture Recognition Solutions 94
4.3.3 Proposed Method 95
4.3.4 Experimental Results 103
4.3.5 Conclusion 110

5 Hardware Acceleration for Smart Sensors 112
5.1 Motivation 113
5.2 Customized Core for Low-precision DNN Inference 114
5.2.1 Context and Motivation 114
5.2.2 Limitations in the Existing Solutions 115
5.2.3 MAUPITI System and Smart Sensor Prototype 115
5.2.4 Experimental Results 119
5.3 Energy-Efficient Approximate Bus Encoding for I2C 120
5.3.1 Motivation 120
5.3.2 Proposed Method 121
5.4 Conclusion and Discussion 128

6 Electronic Design Automation for Smart Sensors 131
6.1 Motivation 131
6.2 Analog Mixed-Signal Topology Recognition in a Real-World Industrial Flow 132
6.3 The State-of-the-art Topology Recognition Solutions 133

6.4 Proposed Method 134
6.4.1 Graph Representation and Subgraph Isomorphism 134
6.4.2 VF2-based Topology Recognition Approach 134
6.5 Experimental Results 136
6.6 Conclusion and Discussion 137

7 Conclusion and Future Work **139**

References **146**

List of Figures

1.1 Overview of smart sensor design. 6

2.1 The basic architecture of a CNN model. Cn, Sn, and FCn denotes n-th feature maps from Convolution, subsampling and fully-connected layers respectively. 15

2.2 Classic CNN architectures: LetNet-5 [1] and AlexNet [2]. 16

2.3 Overview of vanilla Transformer architecture. 18

2.4 One example of a DT. 21

2.5 One example of a RF. 22

2.6 The generic architecture of a smart sensor. 23

2.7 The workflow of such IC AMS placement automation. 29

4.1 Sensor positioning and IR frames examples. 43

4.2 People counting problem formulation based on IR array sensors. . . 44

4.3 Social distancing problem formulation based on IR array sensors. . . 45

4.4 CNN architecture and input generation. 52

4.5 Overview of the adaptive inference system. 53

4.6 DL model architectures. 56

4.7 An instance of an infrared frame sequence involves two individuals in motion as they approach each other closely. 57

4.8 Pareto frontier of Bal.Acc vs. model size and the number of operations. 66

4.9 Evaluation results of the adaptive system in terms of threshold vs Bal. Acc. (a), vs Acc. (b), vs F1 (c) and vs energy saving (d). 68

4.10 Evaluation Results in terms of Bal.Acc. vs. parameters number and operations number for considered DL models in people counting. The left figure shows all models, while the right shows the isolated Pareto curves. 72

4.11 A compact RF implementation. 88

4.12 Accuracy versus inference cycles and memory occupation of RFs and [3]. 90

4.13 Overview of the efficient gesture recognition solution. 96

4.14 Examples of sEMG signals from Ninapro DB6 [4] respect to three gesture classes. 97

4.15 The considered architecture template of bioformers [5]. 97

4.16 The results of architecture exploration of bioformers. 104

4.17 The results of patch size exploration in terms of complexity. 105

4.18 The comparison in dynamic inference results. 107

5.1 The architecture of customized IBEX RISC-V core. 116

5.2 The complete MAUPITI System. 118

5.3 I2C open-drain architecture and current flow at High/Low logic levels. 121

5.4 Hardware implementation of N-bit ACME. 124

5.5 The Pareto curves in terms of average error vs. energy saving obtained by ACME and comparison baselines on accelerometer and audio samples. 126

6.1 Topology recognition example on one current mirror. 135

List of Tables

4.1 LINAIGE Dataset information in social distancing. Viol = violations. 62

4.2 LINAIGE Dataset statistics and CV strategy in people counting. . . 63

4.3 Evaluation and deployment Results of three selected architectures on STM32L4 @ 80MHz (12.8 mW average power). 1/8F = 1-frame/8-frame input, Cn = Convolution with N channels, P = Max Pooling, FC = Fully-Connected. 67

4.4 Detailed evaluation and deployment results of models targeting social distancing on Quentin @ 205.1 MHz 70

4.5 Analysis of the characteristics of DL families for people counting based on IR sensors. 74

4.6 Deployment and evaluation results of considered architectures for people counting on STM32L4A6ZG @ 80MHz. 76

4.7 Comparison with the state-of-the-art ML/DL solutions for people counting. 78

4.8 Detailed deployment results of dynamic inference on GAP8. 108

5.1 Deployment results of three DL models from [6] on different HW platforms. 119

5.2 Comparison of synthesis results for the 12-bit encoder and decoder of ACME and ADE 128

6.1 Topology recognition validation results based on analog templates. . 136

Chapter 1

Introduction

This chapter introduces smart and intelligent sensors, detailing their design flow and definitions as used in this thesis. It overviews smart sensor design, emphasizing three main pillars for ML-enabled sensors: algorithm-level assessments and optimization, hardware-level implementation and optimization, and the development of advanced EDA tools in the industrial flow.

1.1 Overview of Smart Sensor Design

The design of smart sensors is approached by integrating advanced technologies to create devices capable of not only sensing but also processing and transmitting data efficiently. In this section, a comprehensive overview of the fundamental concepts and methodologies behind smart and intelligent sensors is provided. Definitions of smart sensors and intelligent sensors are clarified, highlighting their roles and capabilities within various applications. Subsequently, the design flow of smart sensors is outlined, detailing the key stages and considerations in developing these sophisticated devices, from conceptualization and algorithm optimization to hardware implementation and the development of advanced EDA tools in the industrial flow.

1.1.1 Smart and Intelligent Sensors

Smart and intelligent sensors are the pivotal link between the physical realm and Information Communication Technology (ICT) frameworks, taking on a rapidly growing role in the modern technological world over the past few years. Typically, the fundamental function of a sensor is to detect, measure, or sense its target [7]. Based on the sensemaking process, raw data are initially captured as physical quantities and then translated into electrical signals. Subsequently, these signals are continuous analog data and undergo discretization into digital entities before being processed by computing cores to align with task requirements [8].

The emergence of the concept of smart sensors was considered along with the Internet of Things (IoT) [8], which takes advantage of the significant improvement in the computing power of microprocessors in the last several decades. Among the ongoing evolution of technology, the concept of smart sensors has expanded and deepened. Initially, smart sensors were basic sensor devices enhanced with additional modules such as memory and communication interfaces. However, the progression has led to the integration of local computing resources within these sensors, enabling them to perform specific tasks efficiently [9, 8].

Moreover, there exists a slight difference between the definitions of smart sensors and intelligent sensors in the literature. However, the distinction between these two terms is not always clear-cut, and they are frequently used interchangeably in the literature. In [10], the definition of smart sensors and intelligent sensors were provided: *smart* sensor can be defined as a combination of a sensor with multiple extra modules, while an *intelligent* sensor highlights integrated intelligent functionality.

More recently, the definition of smart and intelligent sensors has been further enriched. The breakthrough of Machine Learning (ML), especially Deep Learning (DL) has ushered in a new era, showcasing their immense potential in the field of sensing. These technologies have shown notable improvements in both quality and response speed for signal processing [11]. As a result, more new opportunities have emerged to implement ML-enabled sensor systems integrated with advanced computational modules.

Based on the review of the literature, in this thesis, the broadly used concept of smart and intelligent sensors is considered as follows:

Different from conventional sensors, which primarily capture physical inputs and translate them into electrical signals without inherent data processing capabilities, smart and intelligent sensors are devices infused with computational resources to execute predefined functions on collected data and extract invaluable information through robust data processing. The resulting informative outcomes are then used to deduce environmental conditions or perform specific tasks such as Human Activity Recognition (HAR), Object Detection (OD), and many others.

In essence, the distinctive features of a smart sensor and intelligent system can be summarized as follows [7, 8, 10];

- **Local Computing Capability**: Capability to process and analyze the collected data locally. Compared to conventional sensors with threshold limit functions, smart sensors are equipped with smart models and advanced computing units;
- **Self-determination Capability**: Capability to make decisions independently for specific tasks locally;
- **Enhanced Communication Capability**: Capability to interact with the environment and other devices in real-time. The sensor module may be easily adaptable to the system following "plug and play" principle;
- **Adaptability**: Self-adaptive performance and functionality according to the variation of environment and specific tasks;
- **Power Efficiency**: Improved power efficiency due to enhanced power management and reduced data transfer compared to conventional sensors.

There exists a wide range of smart sensors spanning a variety of practical applications from computer vision [12–14] to health care [15, 16, 3], among others. Typically, the type of smart sensor can be divided by its sensing target such as an accelerometer, humidity sensor, infrared sensor, etc. [17–22]. Therefore, the type of smart sensors considered in this thesis can be summarized into two categories [7]: 1) physical and chemical sensors that detect the physical and chemical parameters of the target; and 2) visual sensors that capture pixel-level frames.

In this thesis, both types of sensors are considered. In particular, for the former, smart systems based on accelerometers, gyroscopes, and electromyography sensors are studied, spanning applications from HAR to gesture recognition; for the latter,

infrared smart sensors are investigated, targeting widely used applications from social distancing to people counting.

The study of these specific smart sensors is relevant and well motivated. First, these sensors encompass two distinct major categories and various fields, thanks to their capacity for measurement or visualization. This breadth contributes to the comprehensiveness of this research. Second, these sensors are popular due to their wide range of applications. Their versatile use cases, spanning from embedded computer vision to wearable sensing, render them applicable to various tasks in a practical and relevant manner. Hence, directing attention towards the intelligent design of these sensor systems holds great promise.

1.1.2 Design Flow of Smart Sensors

As sensor technologies continue to integrate with an expanding array of "smart" modules and achieve the benefits of increasingly advanced capabilities, the enlarged scale and increasingly complex systems have also posed new challenges for smart sensor design.

In fact, unlike conventional sensors that typically measure a single quantity, modern systems equipped with multiple built-in sensors are becoming increasingly prevalent [7]. Furthermore, the newly introduced intelligent features increase the complexity of the design of smart sensor systems. In particular, ML and DL models are widely utilized as smart models, which are crucial components in such smart sensor systems. Despite demonstrating their superior performance, ML/DL algorithms are in need of significant energy and demand high computational complexity. Last but not least, the numerous and frequent data transmissions in sensor systems also contribute significantly to the total energy budget.

On the other hand, these "all-in-one" sensor systems often need integration into IoT devices, such as smartphones or wearable devices. However, IoT nodes typically rely on battery-operated and resource-limited microcontrollers. This implies that both area and energy efficiency are critical for smart sensors at the edge.

To bridge this gap, a novel generation of practical and efficient smart and intelligent sensing systems is needed. These systems are expected to have a cost-effective implementation while improving sensing performance. Therefore, in recent years, a variety of advanced techniques focusing on different aspects of smart sensor de-

sign, including ML model optimization, sensor hardware architecture design, and development tools for Electronics Design Automation (EDA), etc., have attracted a great deal of attention. [11] discussed the perspective of the design of ML-enabled intelligent sensors. In this perspective, the design flow can be divided into four steps: data acquisition, sensing model training, model validation, and hardware redesign.

In this thesis, a comprehensive study of smart sensor design has been conducted, encompassing not only the entire research process covering each step in the design flow described above, but also its integration with practical applications. In particular, the overview of the flow of smart sensor design is summarized in Fig. 1.1. In the flow outlined, components related to the technologies or innovative design solutions discussed in this thesis are emphasized. These components can be categorized into three levels:

1) **Algorithm level**: This involves the assessment and analysis of ML models such as Random Forests (RFs) and Deep Neural Networks (DNNs), targeting specific tasks. Various optimization techniques, including quantization and adaptive inference. are considered to develop efficient, yet effective, models suitable for deployment on the target hardware platform.

2) **Hardware level**: At this level, efficient architectures are implemented to support the optimized ML models. Additionally, energy saving techniques for communication interfaces such as approximate computing are taken into account.

3) **EDA Level**: This stage focuses on the development of design automation tools to aid in the placement of smart sensor systems.

Furthermore, it should be noted that the entire design flow is derived from practical applications such as people counting or gesture recognition and is grounded in real-world data-driven inference.

In summary, the purpose of this thesis is to develop novel design solutions to implement efficient architectures for ML/DL, which favor energy savings. To achieve this goal, novel implementation strategies and advanced tools for the synthesis and optimization of heterogeneous designs where digital and analog domains overlap are taken into account. The research spans the whole design hierarchy, covering a multitude of aspects consisting of both front-end design, such as automatic design and

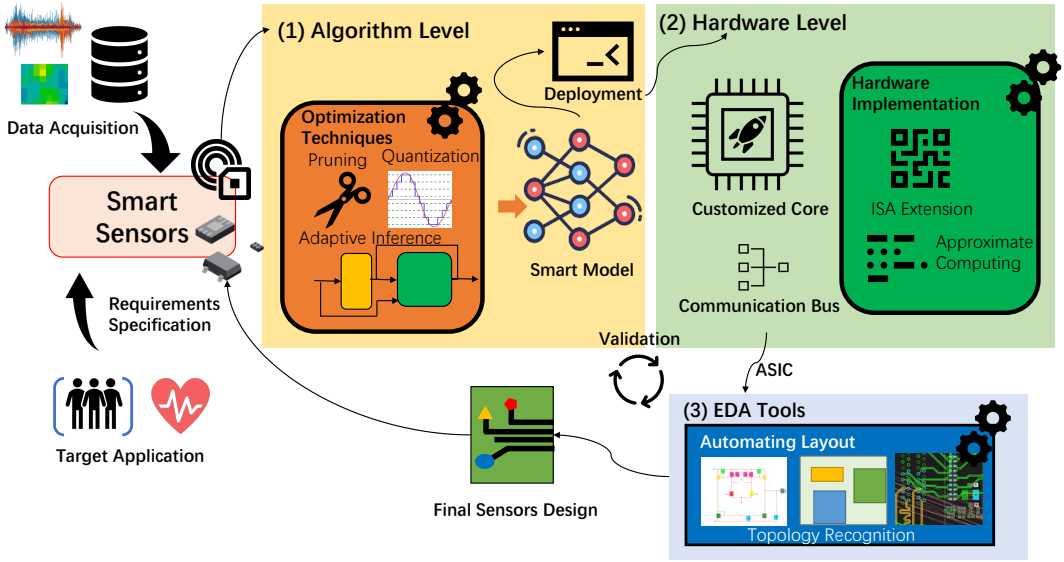


Fig. 1.1 Overview of smart sensor design.

optimization of new architectures, and back-end design such as circuit-level design techniques for area/power optimization of digital-in-analog components. All research activities are conducted through a close collaboration with STMicroelectronics. The subsequent sections will introduce and discuss each aspect in detail.

1.2 Machine Learning-Enabled Smart Sensors

Machine Learning (ML) algorithms have been widely utilized in various IoT applications involving smart sensors in recent years, thanks to their exceptional predictive performance.

In contrast to deterministic approaches that are non-data-driven and necessitate task-specific feature extraction procedures, ML-based solutions take advantage of their capability to autonomously learn crucial features from provided raw data. This implies the generality of ML-based methods to easily fit expanded datasets from new environments without redesign.

ML-enabled smart sensors are equipped with a specific specific ML model and possible data preprocessing [7]. The smart model is trained on acquired data for the target application. The sensing raw data will be processed through the data

preprocessing module and then fed into the trained ML model to perform predictive inference.

The ML-based approaches can be divided into classical ML-based solutions and DL-based methods. Among classic ML works often found in smart sensors, Random Forests (RFs) [3, 23], Support Vector Machines (SVMs) [23, 24], K-Nearest Neighbors (KNNs) [23, 25] and Principal Component Analysis (PCA) [26], are popular across various applications. For instance, based on low-resolution infrared sensors, SVMs and RFs are utilized to achieve occupancy estimation in indoor enviroments [23], as well as human activity recogniton [24, 27, 3]. On the other hand, some works also combine physical and chemical sensors such as Metal-oxide (MOx) gas sensors with classic ML-based solutions. MOx gas sensor operates based on that the properties of a metal-oxide material can be sensitive to specific gases. Among them, [26, 28] take benefit of PCA based on MOx gas sensors to identify patients from exhaled breath, while [29–32] focus on food quality assessment rely on PCA, RFs and SVMs. The favorable outcomes of these approaches, particularly RFs and SVMs have showcased their great predictive performance in solving classification problems based on not only visual sensors but also physical and chemical sensors.

More recently, DL-based approaches have gained increasing attention in smart sensors, thanks to their superior performance. The DL models particularly refer to Neural Networks (NNs) algorithms. With interconnected neuron nodes in a hierarchical architecture, NN models show the strength to efficiently and effectively extract features from high-dimensional data. Therefore, they have become a new paradigm for solving complex tasks.

Hence, a growing number of DL-based approaches have been applied in smart sensors. Among them, Convolutional Neural Networks (CNNs) [13, 14, 12], Long-Short Term Memory (LSTMs) [33, 34], Gated Recurrent Units [35], Transformers [5, 15], etc., are broadly utilized, especially targeting image processing or time-series tasks. For example, addressing the same applications mentioned earlier, [13, 12, 36–39] employ CNNs based on low-resolution infrared sensors for people counting while [40–42] for HAR, capitalizing on the well-established effectiveness of CNNs in analyzing images. Furthermore, some works consider both spatial and temporal information, hence LSTM which has demonstrated its remarkable performance on time series forecasting, is combined with CNNs (CNN-LSTM) for people counting [14] and HAR [33] respectively. Similarly, among DL-based solu-

tions involving MOx gas sensors, [43, 44] select Backpropagation Neural Networks (BPNNs) to address wine identification and gas pollution classification respectively, while [45, 20] use Feedforward Neural Networks (FNNs) to solve air quality monitoring problem. One interesting fact is that, as pointed out by [7], current CNN-based models are strongly relevant across different smart visual sensors while physical and chemical sensors show less correlation.

Although DL models yield superior performance on these tasks, direct execution of ML/DL-based prediction tasks on extreme-edge smart sensors is tricky. ML-based methods are data-driven, and the quality of models is highly relying on the training dataset. Moreover, the inference models that finished training usually cannot be further optimized after deployment.

More crucial is that ML/DL (especially DL) algorithms exhibit high energy consumption and require high computational complexity. These requirements surpass the capabilities of memory- and energy-limited sensor nodes. To effectively implement DL applications at the extreme edge, poses a new set of challenges to find effective yet efficient implementation in both algorithm level and hardware platform. Furthermore, the complex design implies challenges in terms of circuit-level design for sensor systems.

1.3 Energy Efficient Implementation and Optimization

To conceive an efficient and intelligent smart sensor addressing the above challenges, it is imperative to explore appropriate effective ML/DL algorithms. Additionally, innovative hardware implementation strategies supporting these intelligent models are essential. Last but not least, the development of advanced tools tailored for the synthesis and optimization of such heterogeneous designs, is also a crucial aspect of this endeavor.

With this motivation, the design solutions lie in three main pillars: **1)** Algorithm Employment; **2)** Hardware Implementation; **3)** Advanced EDA Tools.

1.3.1 Algorithm Employment

To bridge the gap between heavy ML/DL models and resource-limited and energy-consumption-crucial sensor nodes, employed ML/DL algorithms require a thorough selection and carefully tuned hyperparameters. Initially, targeting different applications and various kinds of sensor nodes, not all ML/DL algorithms can meet the requirements and deliver optimal performance. Next for a selected ML/DL model, the model architecture configuration and its quantization formats require extensive exploration, to achieve the best ML/DL architecture for task performance and cost, in particular, performance and cost including memory occupation, latency and energy consumption are usually conflicting objectives, the trade-off should be considered based on exact application scenarios.

In pieces of literature, efficient and compact algorithm designs from RFs to DNNs [3, 14, 5] are tailored for IoT end-nodes, such as smart sensors. Among them, ML/DL models are first crafted through manual design by a reduction in less crucial layers and thoughtful tuning of hyperparameters configurations, to achieve significantly fewer parameters while maintaining high accuracy.

Moreover, several techniques are broadly utilized on base smart models to further achieve algorithm-level optimization from quantization to knowledge distillation [46]. One category of optimization techniques can be considered as *static* optimization techniques, since they are applied identically to all inputs once the model is optimized. Integer quantization is a well-known optimization that significantly reduces memory footprint and simplifies operations, with a sacrifice of acceptable accuracy drop. It applies a low-bitwidth integer format for inputs and inside parameters of DNNs, instead of the original floating-point format. One popular quantization technique is Quantization-Aware Training (QAT) [47], which is a quantization technique to achieve quantization while avoiding high accuracy loss.

In addition, energy efficiency solutions leveraging run-time trade-offs between accuracy and energy consumption are introduced to further reduce the energy budget for sensor nodes [48]. Different from the static optimization techniques mentioned above, these solutions achieve this goal without the necessity of re-design or re-training, by taking advantage of adaptive inference [14, 15] or run-time device selection [49]. This kind of technique is motivated by the observation that not all inputs are equally hard to process.

1.3.2 Hardware Implementation

Dealing with the challenges of deploying ML/DL models on edge devices like smart sensors, which have limited computational resources and energy budgets, specific hardware features can be incorporated to support efficient computational operations within ML/DLs, thereby further reducing energy consumption, minimizing latency, and optimizing resource utilization.

One of the widely used hardware solutions to support ML models in smart sensors is targeting general-purpose architecture such as CPUs [46], which is customized Microcontroller unit (MCU) cores with extra hardware components to accelerate the execution of DNNs. The inference execution on the customized core invokes custom instructions and takes advantage of newly added hardware computational elements, with the corresponding Instruction Set Architecture (ISA) extension. For instance, [50] propose a mixed-precision ISA extension for RISC-V MCUs to efficiently support quantized DNNs. Thanks to a custom execution paradigm for Single-Instruciton-Multiple-Data (SIMD) arithmetic operations, which consists of fusing a dot product with a load operation, up to 1.64x peak MAC/cycle (where MAC denotes the *number of Multiply-and- Accumulate*) improvement is achieved by this customized RISC-V core.

In this thesis, additional energy-saving techniques targeting the communication interface of smart sensors are also considered. It is motivated by the fact that the large number of data transitions involved in a sensor system dominates all of the energy consumption. Therefore, energy-saving solutions such as Approximative Computing (AxC) [51] can significantly reduce energy with negligible overheads.

1.3.3 Advanced EDA Tools

With the increasing scale and intelligence of smart sensors, new challenges arise in achieving robust synthesis and optimization of heterogeneous designs, particularly in areas where digital and analog domains intersect. Therefore, the autonomous design paradigm has attracted broad attention in order to cope with the growing need for complex Analog mixed signal (AMS) integrated circuits. An important part of AMS design is layout synthesis. The ideal AMS layout automation should translate

a given netlist into a high-quality physical layout without any manual intervention based on modular approaches.

However, while the maturity of digital design automation has been widely acknowledged, the physical design strategies of analog parts in an AMS system still rely on a human-expertise-based process, since the existing design automation of therein analog parts is far less ideal for achieving high-quality handmade design.

In the literature, there exist continuous efforts to increase the opportunity for AMS layout design automation, especially its analog portion [52, 53]. ALIGN and MAGICAL projects [52, 53] aimed to achieve fully automated analog layout generation that benefits from machine intelligence. While these projects focus on the entire workflow of layout automation, including automatic annotation, constraint generation, layout generation, and final block placement, other works may focus on one specific topic in layout design automation. Automatic annotation is one of the essential steps in the whole system and has become a hot topic that attracts many researchers [54–57]. These works can be categorized into two types: circuit analysis and graph matching [54]. While circuit analysis suffers from costly circuit simulations [58], graph matching attracted much attention and generated some interesting results [59–63]. These approaches frame the annotation problem of primitives as a graph pattern matching problem, based on libraries containing several primitive templates, and measure matching by graph similarity metrics [54].

However, considering the complexity of the AMS IC design, there exists a high level of design flexibility, especially on analog parts. Analog primitives, such as the current mirror, may vary depending on the expertise and even preference of designers. In contrast, the performance of the final layout implementation can degrade significantly with tiny changes [64]. To overcome this limitation, recent research trends to utilize the powerful learning capability of Artificial Intelligence (AI) algorithms to improve the robustness and generalizability of current layout automation methodologies.

Chapter 2

Background

This chapter provides a comprehensive overview of the current techniques introduced in the literature, from algorithm-level design, hardware-level design, and the development of advanced EDA Tools.

2.1 Algorithms for Efficient Smart Sensors

ML/DL has garnered significant interest in various IoT applications, from prediction in time series data to computer vision tasks based on embedded devices, due to its impressive predictive capabilities [65, 66]. The direct implementation of DL-driven tasks on smart sensors nodes at edge offers distinct advantages over conventional cloud-based methods. This is achieved by obviating the requirement of transmitting extensive raw data over wireless network links [65, 67, 68]. In particular, executing tasks on the smart sensor itself ensures responsiveness even in poor or absent connectivity scenarios, with a consistent latency. Additionally, the only information communicated with the cloud, if at all, is the final output of the ML/DL-based smart model, such as a classification prediction. This approach is advantageous for maintaining confidentiality, since it minimizes the potential risk of unintended or improper disclosure of sensitive data, including but not limited to videos, images, and other materials [65, 67].

The ML algorithms in smart sensor systems can be split into classic ML algorithms (non-NN algorithms), and NN DL algorithms, as introduced in 1.2. Classic ML algorithms such as RFs, SVMs etc., usually pose greater complexity in configu-

ration than NN-models, demanding expert knowledge to achieve additional manual intervention for fine-tuning parameters to attain desired application outcomes [7]. Meanwhile DL algorithms such as CNNs, Recurrent Neural Networks (RNNs), etc. have showcased a trend to replace classic ML-based approaches for the same application field [7], may due to their superior performance and automatic feature extraction capability.

However, ML and especially DL algorithms typically exhibit a high demand for energy and involve substantial computational complexity. These demands surpass the capabilities of energy-restricted sensor nodes with limited memory resources, typically reliant on MCUs powered by batteries and with limited resources. Effectively deploying ML/DL applications at the extreme edge necessitates a meticulous selection of model architectures and corresponding hyperparameters to bridge this significant gap. This thesis focuses on representative algorithms in both classic ML-based approaches and DL-based solutions, providing a synopsis of the most significant literary works and detailed discussions.

2.1.1 Compact Neural Networks

As DL achieves increasing success and undergoes continuous advancement, NN-based algorithms find widespread application in various domains. In numerous instances, while demonstrating remarkable performance, DNNs are typically crafted for cloud environments, with traits such as considerable computational complexity, extensive memory utilization, and increased power consumption, thus mainly attributed to the vast number of parameters within the network, which amount to billions [46].

Neural Networks

The first concept of Neural Networks (NNs) can be traced back to many decades ago [69]. This set of ML models is inspired by the structure and function of biological brains. By interconnecting artificial neurons with each other in hierarchical structures, input signals can be processed and passed through the layers of the network and produce expected output at the last layer. In particular, neurons inside networks work as basic computational units. Each of them typically operates a weighted sum of the input of the neuron and then applies an activation function to obtain

a nonlinear output. Weights are associated with connections between neurons, which implies the strength of signal transmission between neurons. During the training phase, the output obtained by this progress of the data passing forward (feedforward) is compared to the final expected outcome, and the error is used to perform backpropagation to adjust the weights until the models can produce the desired results [70].

The optimization algorithms used in this backpropagation process play an essential role. Gradient descent has been one of the most widely used optimization algorithms to minimize error during backpropagation since 1980 [71]. This algorithm is based on the current gradient to search the global optimum, and therefore it is often trapped in local optima. Moreover, the performance of the NN algorithms is highly dependent on the size of the dataset. Issues such as overfitting caused by too small training datasets [71] happen, causing significant performance degradation of models in real-world application scenarios. Additionally, as the size of the model increases, the training process becomes progressively more difficult.

Since then, a Deep Belief Network (DBN) based on a novel training method was proposed in 2006 [72]. In 2012, AlexNet [2] was proposed, and it made a breakthrough in the field of DL. At the same time, the rapid development of high-performance devices such as Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) paved the way for the rise of deep learning. Compared to shallow NNs, DNNs are promising for extracting crucial features from big data and achieving much higher accuracy.

Today, the popularity of DL families has been exhibited in extensive applications, and remarkable achievements have been reported in many pieces of literature. Among them, there are several highly popular deep learning models, including CNNs [1, 2, 73–75] and Transformers [76]. These models are renowned for their exceptional performance in a wide range of problem domains.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have become one of the most popular DLs in multidimensional data-based tasks, such as image processing [77]. Instead of the typical matrix multiplication in normal NNs, CNNs utilize the convolution operation. Convolution operations are based on a small two-dimensional matrix,

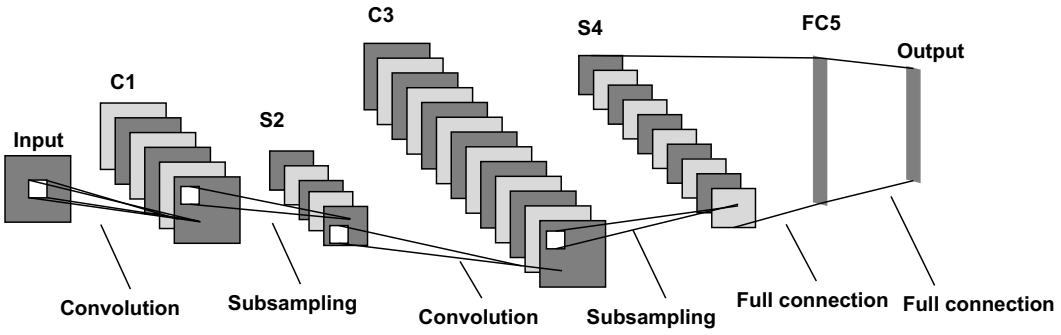


Fig. 2.1 The basic architecture of a CNN model. Cn, Sn, and FCn denotes n-th feature maps from Convolution, subsampling and fully-connected layers respectively.

namely *convolution kernel* (or *filter*), which is moving on the input data as a sliding window. The kernel and input data perform element-wise multiplication, and the results are summed to obtain elements of output. By varying the weights of the kernels, the different features are extracted to contribute to the task-specific prediction.

In theory, the convolution operation in CNN is defined as Equation 2.1 [78], where x_i^j represents the i-th output (namely *feature map*) of j-th layer; x_k^{j-1} donates the k-th input feature map at the (j-1)-th layer; $W_{i,k}^j$ denotes the convolution kernel between the input and the output feature map; n means the number of the input feature maps; b denotes the bias to adjust the output value of the output feature map; finally f($\cdot$) represents an activation function, such as Rectified Linear Unit (ReLU) or Sigmoid, etc.

$$x_i^j = f\left(\sum_{k=1}^{n} W_{i,k}^j \times x_k^{j-1} + b_i^j\right). \tag{2.1}$$

Thanks to convolution operations, the parameters of CNNs and their computational complexity are significantly reduced compared to normal DNNs. The basic architecture of a CNN model is made up of convolution layers and grouping layers[71], as shown in Fig. 2.1. Typically, the architecture consists of an input layer, multiple hidden layers including convolution layers, pooling layers and others (e.g., fully-connected layers), and an output layer. The input tensor is fed into convolution layers and generates feature maps by the trainable kernels, and activation functions are applied to these feature maps to introduce non-linearity. Next, the pooling layer performs sub-sampling operations to reduce the spatial dimensions of feature maps

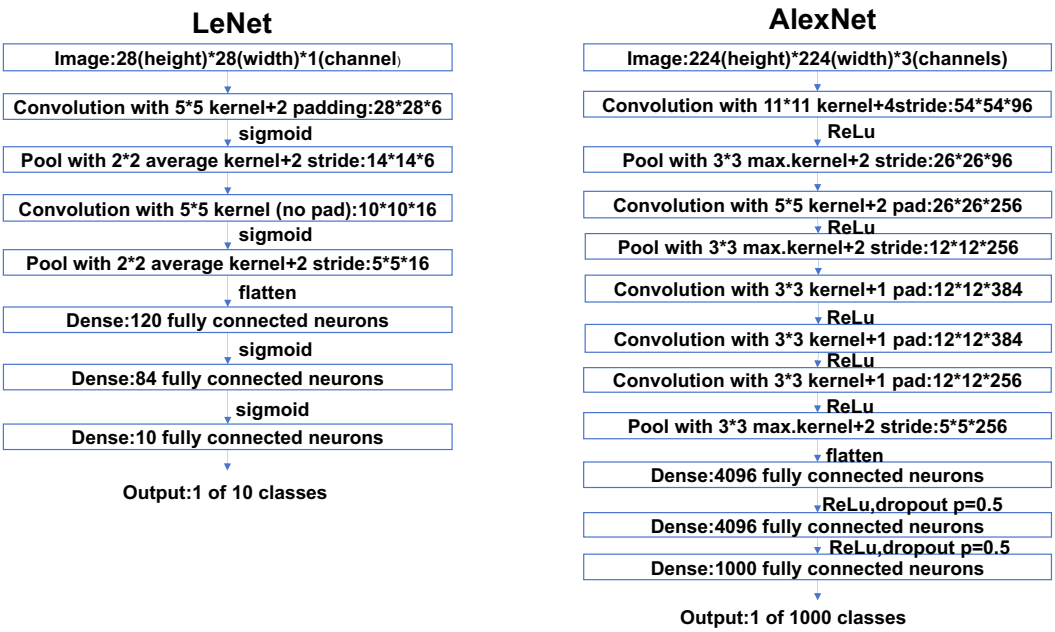


Fig. 2.2 Classic CNN architectures: LetNet-5 [1] and AlexNet [2].

while keeping crucial feature information. The most popular pooling operations are max pooling and average pooling, where the maximum or average value within the pooling window is selected as the output, respectively. The dimension-reduced feature maps pass through the next layers, to further extract features until the final task is achieved.

There are several classic CNN architectures in the literature that are widely used and studied [1]. In Fig. 2.2, two classic CNN architectures are shown as examples. LeNet-5 [1] is one of the earliest CNNs published in 1998, targeting document recognition. This architecture is lightweight with two convolution layers followed by pooling layers and three fully-connected layers at the end. This architectural approach has significantly influenced the design of CNNs, serving as a standardized template. AlexNet [2] deepened CNNs and achieved notable success in the ImageNet image classification challenge. In this architecture, the ReLU activation function is applied. Furthermore, novel architectures such as VGGNet, Inception and ResNet etc. [73–75] were proposed in the next few years. These architectures either grew in size or introduced new functional modules to improve the model performance.

Transformers

The Transformer [76] is a DL model targeting sequential data processing, originally proposed in 2017. It has achieved significant success in various fields, particularly in Natural Language Processing (NLP), and achieved state-of-the-art performance in such tasks as language modeling, machine translation, text classification, etc [79, 80]. Moreover, it has also showcased great results when adopted in computer vision [81, 82], audio processing[83], and other fields.

The key idea of the transformer is to take advantage of the self-attention mechanism to process sequence data. Thanks to multi-head attention, the dependencies between different parts of the input sequence are captured, ensuring superior performance is maintained even when processing long sequences.

The inner logic of the self-attention mechanism is Query-Key-Value (QKV) models [76]. Given an input matrix X, the packed matrix representation of queries Q, keys K, and Value V are then obtained through multiplication with their respective weight matrices. After that, the self-attention can be calculated as Equation 2.2.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.2}$$

where d_k denotes the dimension of keys (as well as queries). Next, multi-head Attention is formed by combining multiple self-attentions. For H attention head, The output from each head is concatenated to form a HxD_v-dimensional tensor. This progress can be formulated as an Equation 2.3, where W^Q, W^K and W^V represents projection matrices for Q, K and V respectively, to linearly project original queries, keys, and values into different subspaces.

$$\begin{aligned} \text{head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), \\ \text{MultiHead}(Q,K,V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right). \end{aligned} \tag{2.3}$$

At last, the concatenated output of multi-head attention is performed linear transformation to map into the desired output dimension.

Fig. 2.3 depicts the architecture of the vanilla Transformer [76]. There are two main components in this architecture: the encoder and the decoder. The encoder is responsible for encoding the input sequence into high-dimensional representations,

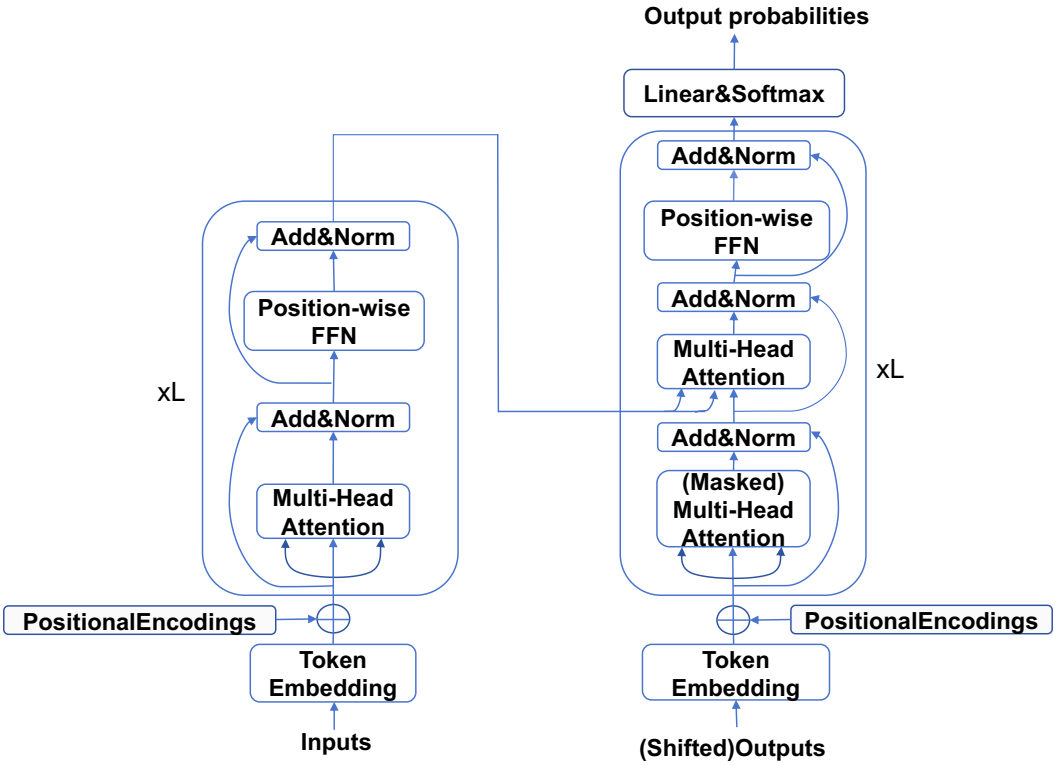


Fig. 2.3 Overview of vanilla Transformer architecture.

while the decoder takes the output from the encoder, along with a partial target sequence that has already been generated, and produces the next portion of the target sequence. Both the encoder and decoder consist of multiple blocks with identical structures stacked together.

Particularly, each encoder block consists of a multi-head self-attention layer and a feed-forward neural layer, both employing a residual connection [75] and followed by a Layer Normalization [84]. The decoder block has a similar structure to the encoder block but consists of two multi-head self-attention layers. Specifically, the first multi-head self-attention layer adopts masked operation, ensuring that each position in the output sequence can only attend to positions before it in the sequence, to generate the output sequence autoregressively. The second attention layer utilizes the output from the encoder and previous layers within the decoder, enabling the decoder to focus on various segments of the input sequence to assist in producing output that aligns with the input. In the end, softmax is applied to produce the final prediction.

In the literature, multiple variants of transformers have been proposed and become state-of-the-art not only in the original NLP target domain but also in computer vision, audio processing, and even multimodal applications [80]. These remarkable successes are attributed to several groundbreaking models such as BERT [85] and the VisionTransformer (ViT) [86], etc. Despite achieving superior accuracy, the size of Transformer models has been exponentially increasing over time, reaching hundreds of millions, or even billions, of parameters.

Compact Models

As described in previous sections, DNN models such as CNNs and Transformers have achieved great success and have greatly contributed to the popularity of DL. However, these DL algorithms were initially developed for cloud computing platforms. Specifically, their requirements of high energy consumption and computational complexity have become a significant obstacle when moving to edge deployment on smart sensors. Indeed, sensor nodes at the edge are typically constrained in resources and critical in energy consumption. Therefore, compact DL models have become a promising solution and have attracted increasing attention.

With CNNs showcasing their superior capabilities across diverse fields, particularly in computer vision, researchers have been increasingly focusing on transferring these models from cloud platforms to edge devices. The state-of-the-art CNNs such as VGGNet [74] are also well-known for their huge number of parameters in the magnitude of hundreds of millions. However, the distribution of parameters is not uniform, but a few layers host the most of parameters in total [87]. Therefore, reducing the size of networks by removing redundant parameters has become an attractive topic. The techniques involving CNN structure simplification can be summarized as network distillation, memory efficient structures, and parameter pruning [87]. On another hand, some studies proposed novel CNN architectures with a compact structure directly targeting the specific tasks [88, 89, 14]. Notably, it has been demonstrated in many studies that by targeting simple tasks, lightweight CNNs with much fewer model sizes can achieve state-of-the-art performances [14].

Similarly, Transformers also suffer from its massive number of parameters, as introduced in the previous sections. To overcome this limitation to achieve larger applicability, several studies proposed compact variants of Transformers [90–93]. These compact models are based on the original BERT and utilize Network

Distillation to extract the most critical architectures. Moreover, some work such as [5] proposed task-specific compact architecture for classic large Transformers, and proven Transformers can achieve success in tinyML-based tasks at edge.

2.1.2 Decision Trees and Tree-based Ensembles

Although DL families have proven highly desirable performance and become go-to models in various applications, classic ML-based approaches still play an important role in edge computing thanks to their lightweight design and superior accuracy. One of the standout representatives of non-deep ML models applied on edge devices are Decision trees and tree-based ensembles.

Decision Trees

Decision Trees (DTs) are a common supervised learning approach in ML and are applied in many applications targeting both classification and regression tasks. The core mechanism of DTs is to generate a tree-like structure based on the extracted features in the dataset, where each leaf node corresponds to a class label or numerical output, depending on the target problem. During training, DTs select the appropriate characteristic and its decision threshold for each internal node, based on its splitting criterion [94], and recursively apply it until a stopping criterion is met, such as maximum depth, etc. The training approach to achieve this DT construction varies in the literature [95–97].

Fig. 2.4 depicts the tree structure of a DT. It consists of nodes and edges, where nodes represent features or categories, and edges represent relationships between feature values or categories. In particular, the nodes in a tree are categorized into the root node (located at the top of the tree), the internal nodes (intermediate), and the leaf nodes (situated at the bottom). The root node is the starting point of the DT and splits the dataset into multiple subsets. The internal nodes continuously split the dataset into smaller subsets by following the left or right branches, determined by comparison results between the values of a feature and a specific threshold inferred from the training data, until reaching a leaf node. Ultimately, the leaf nodes hold the prediction output for the target problem.

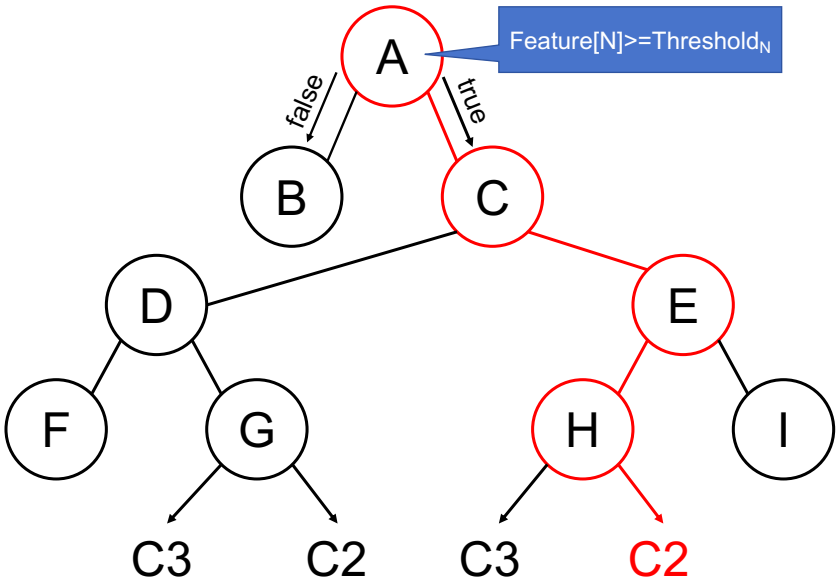


Fig. 2.4 One example of a DT.

As a classic ML-based approach, the advantages and disadvantages of DT are evident [98]. On the one hand, DTs are interpretable and easy to understand, especially compared to DL-based solutions. They can handle both categorical and numerical features, implying their wide applicability to various tasks. More importantly, DTs exhibit fast inference speed and low computational complexity, particularly compared to DL families, making them a promising alternative in edge devices such as sensor nodes. On the other hand, DTs are troubled by overfitting, particularly when handling many features. Furthermore, they are sensitive to minor variations in data and easy to trap in local optima, and both could lead to less-than-ideal performance, especially for complex tasks.

Random Forests

Random Forests (RFs) are a popular ensemble learning approach based on DTs and can alleviate the above limitations. Ensemble learning or multi-classifier systems [99] combine multiple base learners to obtain a more powerful model by leveraging collective solutions. The main principle of this learning approach is that a better solution can be achieved by comprehensively considering decisions from a group of weak classifiers with limited knowledge of the problem domain. The literature has

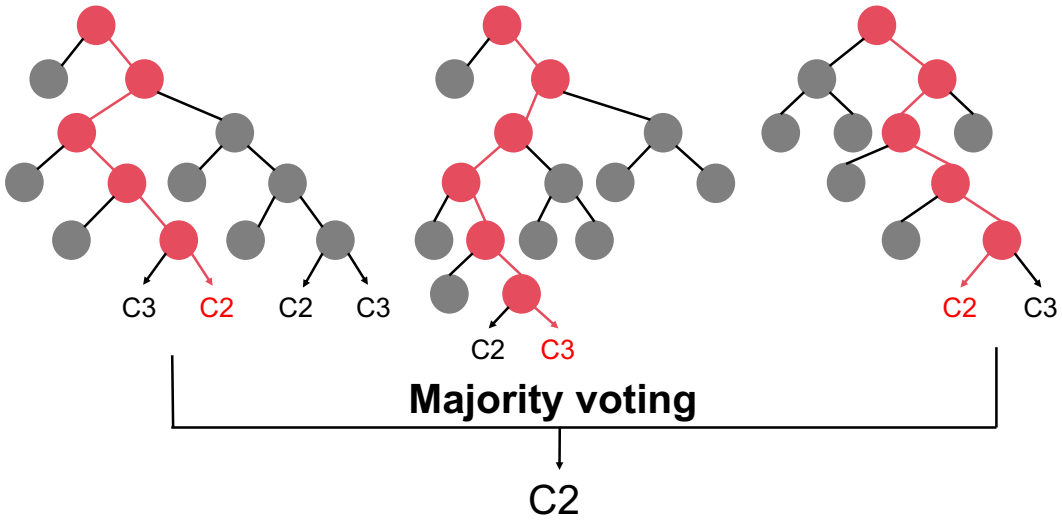


Fig. 2.5 One example of a RF.

several ensemble strategies including majority voting, bagging, boosting, and among others, [100–102].

Taking advantage of these ensemble learning methods, the aggregated result of RFs demonstrates greater robustness to data variance and noise, outperforming that of an individual DT in performance. Fig. 2.5 shows an example of an RF structure. When inference is performed, each DT generates their prediction as class labels or probabilities of each class for the same input samples, and these predictions are used to obtain the final inference result by majority voting.

Although DTs and RFs gain advantages in the deployment involving smart sensor nodes due to their lightweight architecture, fast inference, and high accuracy, their computational complexity increases with the depth of the tree and the number of trees (for RFs). In fact, RFs typically require more computational resources and memory occupation, especially in cases with a large number of trees and features. Assume a single DT with time complexity as $O(D)$, where D denotes the depth of the tree or the level of layers, then for a RF with trees N, the time complexity is $O(N \cdot D)$. Meanwhile, the memory complexity can be formulated as $O(N^2 \cdot D)$ [94]. Note that for the RF version with leaf nodes producing probability of each class instead of class labels, there is additional complexity of the argmax function [94]. In conclusion, similar to large DL models, large RFs also face the conflict between high complexity cost versus relative resource scarcity in edge sensor devices.

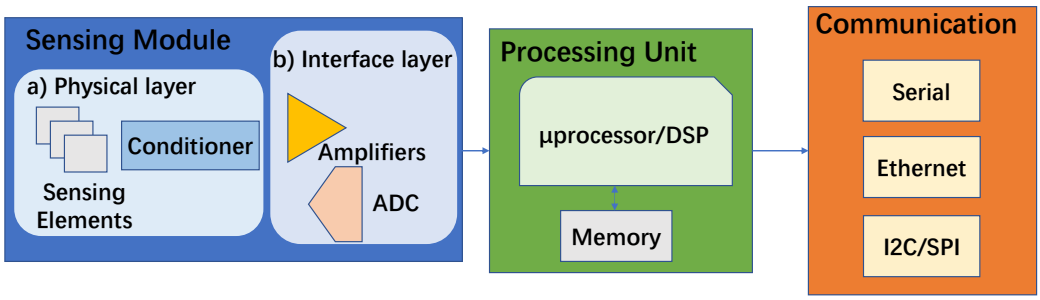


Fig. 2.6 The generic architecture of a smart sensor.

In this case, the efficient implementation of RFs becomes an interesting topic, especially facing edge sensor nodes. Some common libraries and tools offer efficient implementations of RFs such as H2O.ai, XGBoost, etc. [103, 104], utilizing serval optimization techniques such as distributed training, parallelization, etc. to speed training and inference phase.

2.2 Hardware-level Design and Optimization

The hardware design of smart sensors involves integrated computational capabilities and communication interfaces to transmit data. Typically, smart sensors are integrated with MCUs or Digital Signal Processors (DSPs) to achieve data processing tasks such as data preprocessing and feature extraction, etc. in real time. Meanwhile, the processing unit is also responsible for the execution of the ML model developed in the smart sensor. On the other hand, smart sensors are equipped with communication interfaces to transmit data to external devices or networks. In addition, there exist other vital components, such as power management (especially for battery-operated sensor nodes), applying several optimization techniques, security modules such as encryption [105], and so on.

In the literature, the generic architecture of a smart sensor may undergo slight variations [105–107]. As pointed in [106], the integration occurs for discrete components of a smart sensor to reduce cost. In this thesis, a high-level architecture of a smart sensor illustrated in Fig. 2.6, is under consideration. The main components of a smart sensor can be partitioned into three parts: **Sensing Module**; **Processing Unit**; and **Communication Module**.

Sensing Module. The sensing module is responsible for data acquisition. In particular, the module is divided into a) the physical layer, where one or more sensing elements and signal conditioners exist, and b) the interface layer, where amplifiers and Analog-to-Digital Converter (ADC) work. In this module, the sensing elements capture physical quantities such as temperature, image, etc., and the signal conditioner handles the collected signals and makes them suitable for further processing. In the interface layer, the naturally low-level output produced by the physical layer is amplified and converted from the analog output of the sensor to the digital world of microcontrollers and DSPs.

Processing Unit. The processing unit takes charge of the system control and adds intelligence to the smart sensor by supporting relatively complex tasks. Typically, the processor adapts one alternative from MCUs, DSPs, and even other processors (e.g, application-specific ICs) [106]. Memory is usually included in this unit, to store data, configurations, and other relevant information. Various types of memory may selected to be integrated depending on the complexity of the smart sensor, varying from Random Access Memory (RAM) to flash memory. To achieve execution, the corresponding software parts are also needed [105]. For example, a people counting application that uses an MCU-based infrared smart sensor should undergo compilation, with the resulting instructions subsequently fetched, encoded, and executed within the MCU.

Communication Module. The communication modules take care of the transfer of data locally or externally to other devices or systems. The communication interfaces are not only wired such as Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) and Ethernet, but also wireless such as Bluetooth, Wi-Fi, etc.

In this thesis, optimization focusing on Processing Unit and communication module is under consideration. The detailed discussion is in Chapter 5.

2.2.1 Hardware Platforms for Smart Sensors

As described in the previous section, the hardware components of a smart sensor can be divided into three main parts. In the literature, some works also consider the concept of dividing the smart sensor into a sensing element unit and a smart

transmitter part [107]. In fact, the latter, more specifically, the processin unit inside the smart transmitter part is the core to improve accuracy and provide intelligent features to the smart sensor [106].

In the modern world, with the rapid development of semiconductor technologies, dedicated sensor processors vary from MCUs, and Application-Specific ICs (ASICs), to Field-Programmable Gate Arrays (FPGAs). These kinds of highly integrated circuits, namely *System on a Chip* (SoC), typically integrate specialized software that is tightly connected to the system hardware [106].

MCUs commonly used as control centers and intelligent task processing units in smart sensors. The typical MCU consists of microprocessors (namely *core*) and various components including memories (e.g. instruction memory and data memory), clock and I/O capability [106], etc, to form a *core region*. For instance, STM32L4R9ZI, which is a commercial MCU by STMicroelectronics, is an ultra-low power MCU based on the ARM Cortex-M4 core. Particularly, this core features a Floating-Point Unit (FPU) and a set of DSP, plus a memory security block, etc.

MCUs vary by the microarchitecture of their processor cores and can be divided into two main categories: the Complex Instruction Set Computer (CISC) architecture and the Reduced Instruction Set Computer (RISC) architecture. Since the past few years, the RISC-based MCU has become the standard [106].

Today, in the market, several popular architectures serve as the common basis for MCUs equipped with smart sensors. Advanced RISC Machines (ARM) processors are widely adopted in ultra-low-power MCU designs due to their energy-efficient performance. For instance, ARM Cortex-M series processors [108] offer a set of features such as low-power modes, efficient interrupt handling, and support for diverse peripherals and communication interfaces, making them a wise choice for smart sensors. Other selections such as Microprocessor without Interlocked Pipeline Stages (MIPS) processors [109] are also popular and focus on Hardware-Software (HW-SW) collaboration to improve performance. More recently, RISC-V [110] has gained traction in the MCU market, especially for low-power and IoT applications. RISC-V processors are based on open-source architecture and an active ecosystem, offering great flexibility, scalability, and energy efficiency, which makes them suitable for smart sensor integration.

2.2.2 Customized Microcontrollers

RISC-V is an open-source Instruction Set Architecture (ISA) based on RISC architecture, born in 2010 in Berkeley [110]. Thanks to its open collaboration between a wide range of members from different places, RISC-V has become one of the most popular instruction set architectures in a short time. It overcomes the barriers to sharing design in the field of CPU design (central processing units) and enables use across academic and industrial domains and by different designers and developers without costly agreements [110]. Therefore, RISC-V ISA has been growing fast, containing a huge number of extensions, and a variety of open and proprietary cores have been implemented.

RISC-V has been widely adopted in various fields, including embedded systems, Internet of Things (IoT) devices, and more. The RISC-V-based MCU involving edge nodes such as smart sensors has attracted increasing attention, due to its customizability and scalability. Meanwhile, the popularity of ML/DL in such smart nodes poses higher requirements for HW-SW collartation and optimization. For example, while DNNs benefit from quantization (namely, *Quantized Neural Networks*, or *QNN*) in terms of memory footprint savings, the lack of support for such low-bitwidth operations in the arithmetic units in HW leads to waste in terms of energy. A customized MCU in both the microarchitectural and ISA levels to fully integrate a multiprecision arithmetic unit to implement a custom Single-instruction-Multiple-Data (SIMD) operation can significantly improve the efficiency of QNN inference on MCU-based smart sensors.

2.3 EDA Tools for Smart Sensors

2.3.1 Mixed-signal Integrated Circuit and Smart Sensors

As the "intelligence" of smart sensors advances, designing heterogeneous smart sensor systems where analog and digital domains overlap becomes increasingly complex. In fact, a smart sensor system is usually a complex Analog-Mixed-Signal (AMS) IC system combining physical elements (sensing elements), processing (control, data processing), and communication peripherals, in order to process and convert physical quantities from the real world into digital signals for processing

and analysis. Using a MicroElectroMechanical System (MEMS) InfraRed (IR) smart sensor as an example, such a sensor comprises Thermal MOS (TMOS) analog sensing parts and MCUs with other digital functional blocks.

In detail, the vital role of AMS circuits in a smart sensor can showcase in the following aspects:

Sensing Interface and Signal Processing. Smart sensors typically sense physical phenomena under supervision [105] such as temperature, gas, etc., by their sensing elements and produce analog signals. Then these analog signals need to be processed and converted, such as amplification, filtering, ADC, etc. AMS ICs integrate various analog and digital circuits to achieve complex signal processing functions, ensuring that sensor outputs meet the requirements of systems.

Energy Efficiency and Performance Optimization. Smart sensors typically need to operate under a limited power supply based on battery at edge or low power consumption conditions for a long time running to collect data, requiring high energy efficiency and performance optimization. On the other hand, AMS ICs can be optimized in terms of the design of analog and digital circuits to meet such requirements.

Integration and Miniaturization. Smart sensors often need to integrate multiple functions within limited space, requiring high integration and miniaturization capabilities. Meanwhile, AMS ICs integrate analog and digital components on a single chip, achieving the integration of multiple functions while reducing the size and cost of the system.

Economy and Practicality. Last but not least, the cost of an AMS IC depends on the complexity of its implemented functions, which means that various solutions can be selected to fit the needs of smart sensors, ranging from simple to complex. Moreover, the robustness of AMS ICs implies a long production life for a smart sensor.

However, as the complexity of smart sensors has been increasing in recent years, advanced tools and solutions are needed to support this analog and digital co-design, to implement optimized AMS circuits [111].

2.3.2 Analog Mixed-Signal Back-End Design Automation

Electronic Design Automation (EDA) tools, specifically utilized for designing and validating electronic systems, are encountering the imperative need for substantial updates to address the evolving challenges mentioned above. In particular, the autonomous design paradigm has attracted broad attention in response to the growing need for time-to-market complex AMS ICs. An important part of AMS design is layout synthesis. The ideal AMS layout automation should translate a given netlist into a high-quality physical layout without any manual intervention based on modular approaches.

While digital design automation has relatively matured, AMS design still heavily relies on human expertise. Existing automation falls short of achieving the high quality of handmade designs, mainly due to the complexity and variability of analog circuits. Each AMS system includes diverse circuit designs and various parameters, often tailored to individual designers' preferences, posing a significant challenge for generalization and modularization in analog design automation.

To overcome this limitation, recent work that takes advantage of ML and AI to develop new solutions and software for EDA of analog or AMS semiconductor devices [112, 113]. In particular, [113] is a European-Union-funded project AMBEATion (Analog Mixed-signal Back-End Design Automation with Machine Learning and Artificial Intelligence Techniques). Using state-of-the-art tools integrated with AI techniques, along with the advancement of cutting-edge methodologies and workflows, this project aims to overcome the productivity limitation of AMS design.

The workflow of such IC AMS placement automation can be divided into three sections [113], as shown in Fig. 2.7:

IC Schematic Processing. This section is responsible for processing input schemetic/netlist and database files; after data preprocessing, topology recognition, and IP recognition, the final output generates a new version of netlist with annotation.

Layout generation. This section generates placement layout, based on topology recognition and IP recognition in the IC Schematic Processing section. The individual devices are generated by processing PCells, etc., and produce the placement of small topologies (e.g, current mirrors). Similarly, then group

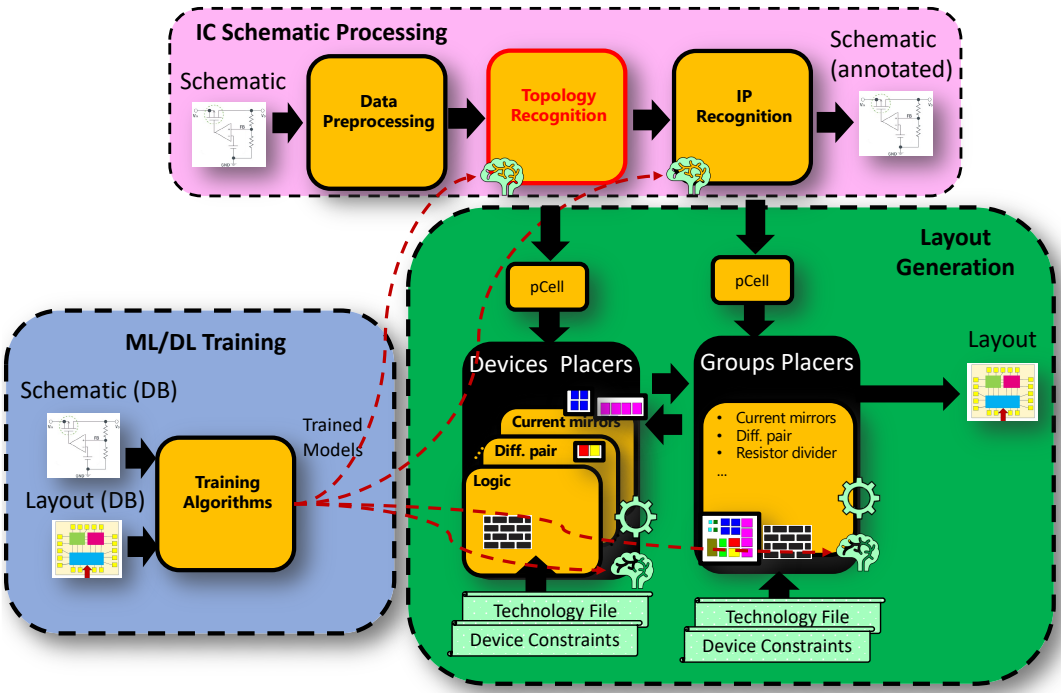


Fig. 2.7 The workflow of such IC AMS placement automation.

placer positions recognized IP containing various topologies in the above section in the layout.

ML/DL Training. This section can be offline and perform training to implement ML/DL techniques. For example, the ML-based approach to automatically recognize topologies in the netlist.

In this flow, one of the critical concerns is robust topology recognition (highlighted in red in Fig. 2.7). Given a netlist, the main elementary functional topologies composing basic elements such as transistors, capacitors, etc. should be recognized in the schematic, in order to properly assign constraints to the layout placer in later modules. In particular, the topologies in AMS circuits can be split into *analog* topologies such as current mirrors, differential pairs, etc., and *digital* functions representing inverters, NORs, and so on.

Nevertheless, the topologies in the AMS circuit tailored by different designers also appear in a great number of variants, especially for analog topologies. For example, a cascode current mirror can vary in transistor sizing and the number of stages to customize performance. Moreover, different designers may employ

unique approaches by varying biasing networks or compensation techniques to determine performance characteristics. Taking into account the multifaceted aspects and complexities of design strategies for large AMS circuits, it is tricky to implement accurate autonomous topology recognition for all potential variants.

Chapter 3

Related Work

This chapter performs a detailed survey of the existing work involving the techniques discussed in the previous chapter, on three pillars comprised of algorithms-level, hardware-level, and advanced EDA tools.

3.1 Algorithms-Level Techniques

To implement compact and efficient smart models into resource-limited and energy-crucial smart sensors, the existing works in the literature can be split into two main parts: compact model design and optimization techniques. The compact models are usually designed targeting specific sensor technologies and application requirements, while optimization techniques such as dynamic inference, etc., are used to further optimize the smart sensor system focusing on energy saving, inference speed up, etc.

3.1.1 ML/DL Algorithms in Smart Sensors

As described in the previous chapter, ML models deployed on smart sensors can be split into NN-based DL models and classic ML models. In particular, in NN-based DL models, CNNs and transformers have obtained great success, thanks to their impressive performance in various applications. On the other hand, RFs as a popular tree-based ensemble demonstrate their natural advantages of lightweight size and high accuracy on many tasks, making it one of the outstanding representations of classic ML-based approaches involving smart sensors. This section introduces and

discusses a review of these models integrated into various types of sensors that target various applications.

NN-based models

CNNs. Since efficient CNNs have been introduced in the last decade including AlexNet, YOLO, VGG, etc. [2, 114, 73], it shows superiority in terms of image processing, compared to classic ML-based solutions such as SVM and PCA [7]. Therefore, it is widely used in visual smart sensors and highly specialized in its solutions targeting different types of images for specific tasks [7], from optical images [115, 116], X-ray images [117, 118] to infrared images [13, 14] etc. Applications span a variety of fields, from object detection and medical diagnosis to biomedical image diagnostics [7].

Various applications utilize CNNs to process optical images exist. [119] proposed a Region-based CNN (R-CNN) and its improved version (Faster R-CNN) aimed at object detection [119, 120]. Furthermore, this model was applied to face recognition and civil infrastructure damage [121, 122], and achieved great results (up to 0.952 true positive rate and 85% precision, respectively).

After that, YOLO frameworks [114, 123] were introduced and quickly outperformed R-CNNs in the same applications. In particular, YOLO solved the large localization and recall errors [7], and achieved optimization when applying the combination of different datasets. [124] utilized a CNN model based on the CaffeNet framework as the smart model to deploy on a sensing system to classify the type of weed among soybean crops. It was trained on 400 images with annotations, and four different types of weeds were recognized. The results showed that the proposed CNN-based smart model can achieve 98% accuracy and outperform a classic ML-based SVM model.

Moreover, many CNNs models, including AlexNetOWTBn, Overfeat, AlexNet and GooLeNet, etc. [125, 126, 2, 74] were proposed to recognize the disease in plants by visual smart sensors. Among them, [127] proposed a CNN-VGG model to diagnose the different types of disease in plants. This model is based on the VGG framework and is trained on a large open data set containing more than 87k images with healthy or diseased labels. The results of this study achieved an accuracy of more than 99% and outperformed other CNNs.

Based on optical images, CNNs have not only proved their great performance in classification problems but also gained success in terms of segmentation and regression problems. [127] used CNNs to segment apples in images and provide the prediction of yield. [128] demonstrated that CNNs can successfully predict agriculture waste.

Although achieving great success in optical images, researchers paid attention to other kinds of images. [129, 130] used CNNs on satellite images to detect and segment water, grassland, etc., from land in the image, and to detect oil spills in oceans, respectively. Moreover, while IR smart sensors have become increasingly popular thanks to their advantages such as low cost and privacy preservation, there exist various studies to apply CNNs on these kinds of sensors. In Section 4.1, more details of CNN-based solutions targeting IR smart sensors are provided.

Transformers. More recently, Transformers have become one of the most popular DL models due to their effective attention mechanism. Therefore, its edge implementation from the initial cloud design attracted people's attention. Transformers were designed to solve sequence-to-sequence problems [76], and then quickly obtained success in various fields, including NLP and computer vision.

However, transformers usually contain tens to hundreds of millions of parameters [131], far beyond the condition provided in an MCU-based smart sensor. To fit the MCU-based edge sensors, several lightweight designs of Transformers were proposed. [131–134, 92, 135].

Among them, [131] proposed a novel Transformer tailored for MCU-based devices such as smart sensors. In this work, a set of new attention kernels was designed to maximize data reuse by optimized operations, to achieve latency and energy while running on the MCUs. [132] proposed a low-precision DL model based on a noel self-attention mechanism named the attention condenser, tailored for speech recognition at the edge. This attention condenser learns a condensed embedding that combines both local and cross-channel activation and performs selective attention. While achieving state-of-the-art test accuracy, the proposed TinySpeech reduced the number of parameters by more than 500 times. [133] designed a Lite Transformer (smaller 18x than [76]) with key primitives named Long-Short-Range Attention (LSRA), which uses both the convolution and the attention mechanism to specialize the local context and the modeling of long-distance relationships, respectively. Inspired by ViTs, [134] introduces a lightweight MobileViT for mobile implementation

and achieves higher accuracy with parameters (7.5 million) than the state-of-the-art CNNs. [92] benefits by knowledge distillation to generate a small transformer from a large one, with over 7x model size savings and 9x speed-up. [135] proposed a highly compact OutlierNets based on autoencoder with only 600 + parameters, achieving an accuracy similar to the existing model with 4 million parameters.

Tree-based models.

As a popular classic ML model, there exist many studies of a variety of ML-enabled smart sensors equipped with RFs to achieve various applications, ranging from accelerometer, gyroscope, and gas sensors to temperature, humidity sensors, etc. [136, 16, 29, 21]. RFs have proven their power to solve both classification and regression problems.

In detail, [136] used RF to detect and monitor Gait Freezing events (FoG) for patients with Parkinson's disease. [16] demonstrated an efficient RF inference by an MCU-based early stopping mechanism that can achieve high accuracy for human activity recognition on a data set collected from a triaxial accelerometer mounted on a smartphone. [29] classify storage quality of Chinese pecans by RFs based on MOx gas sensors. [21] estimate PM2.5 and PM10 thanks to RF based on the temperature, humidity, pressure, and wind speed collected by the sensors.

3.1.2 Optimizations of Algorithms/System Techniques

In addition to a compact and efficient model implementation tailored for MCU-based smart sensors, several algorithm- and system-level optimizations have been reported in the literature. Optimization can be categorized into *static* and *dynamic* techniques. The first is applied to models during the training phase and cannot modify models during inference, while the latter takes advantage of the opportunities to adjust models during real-time inference to adapt to new data and environments. In this section, the related works involving the techniques considered in this thesis are discussed.

Quantization

Quantization is a widely used model compression technique aimed at reducing model size and latency using low bitwidth and redundancy inherent in DNNs [137]. The easiest quantization targeting MCUs is post-training quantization. In this technique, a real number is transformed into a fixed point representation using scaling and offset after the training process [137]. [138–141] are model frameworks that support post-training quantization with 4 to 16 bitwidth. The works involving these frameworks have demonstrated that post-training quantization can reach up to 4x model size reduction and 3x inference time savings.

On the other hand, [142, 139, 140, 143] supports **quantization-aware training**, which injects the simulations of quantization operations during training. This technique enables the containment of relatively high accuracy compared to post-training quantization [137].

Moreover, other quantization techniques such as **mixed-precision quantization**, **binarization** also obtain great success involving implementation on MCUs [144, 145, 3, 146]. Mixed-precision quantization is to carefully craft different bitwidths of weights and activations in each layer in a DL model, to maximize the memory footprint saving while minimizing accuracy drop. Binarization is an extreme case for quantization to set weights and activations to -1 or +1. While significantly reducing memory occupation, relatively expensive operations such as multiplications can be converted to other hardware-friendly operations, further reducing energy consumption involving MCU-based smart sensors.

Dynamic Inference

The dynamic / adaptive inference technique is a kind of flexible optimization performed during runtime inference phases, supporting systems processing different inputs using different strategies [15]. Recent work involving dynamic inference has mainly focused on CNNs [147–156]. Among these, the "big/little" system stands out as one of the earliest and most effective approaches [149]. This system involves the integration of two classifiers with differing levels of complexity and accuracy at runtime to enhance efficiency. The core concept revolves around employing a "little" CNN for straightforward inputs and a "big" CNN for more challenging ones. Therefore, the "little" model classifies an input and tries to output prediction and its

confidence for the results, only when the confidence is lower than a given threshold, and the "big" model is invoked to provide the final prediction to this input. Among them, [150, 151] further enhanced this technique by constructing the little model as a subset of the big one, such as deactivating certain layers during the initial iteration or reducing the bit width of the data.

Staged inference is another kind of dynamic inference, wherein a task is divided into sub-problems that are sequentially tackled by increasingly sophisticated classifiers. The idea behind this approach is the fact that not all classes pose the same level of difficulty for recognition [148]. Consequently, the initial stages focus solely on identifying inputs belonging to easy classes and assigning a "fallback class" to all other samples. Subsequent stages then refine the classification for those inputs previously associated with the fallback class. [153] proposed a diagnostic system that contains two stages, and the first stage is used to classify a patient as healthy or sick, followed by a refinement of the disease classification for sick patients in the second stage. [154] focused on human activity recognition systems, where the first stage focuses on detecting simple activities such as laying or sitting, with the second stage dedicated to more complex actions such as running or cycling.

3.2 Hardware-Level Techniques

3.2.1 Customized CPU

To maximize the efficiency and effectiveness of algorithm-level optimization, such as quantization, the corresponding hardware-level optimization has been studied naturally. As one of the most common hardware platforms for smart sensors, ultra-low-power MCUs lack speed compared to ASICs [50], to efficiently execute quantized NNs.

Therefore, many studies integrate an MCU with enhanced computational units and the corresponding ISA extensions, targeting efficient inferences for ML/DL models. Among them, SIMD instructions tailored for low-precision data were designed [108]. For instance, STMicroelectronics offers a wide range of MCUs based on ARMv7e [108] and ARMv8-M architectures [157], which integrate SIMD instructions for 16-bit and 8-bit data, respectively. On the RISC-V front, XpulpV2 ISA extensions [158] were designed to enable efficient digital signal processing,

using the SIMD paradigm up to 8-bit data. Moreover, [50] proposed a mixed precision arithmetic unit based on a RISC- V processor with ISA extension to enable efficient inference of quantized NNs with bit widths of 2 to 4 bits.

3.2.2 Approximate Bus Encoding

In smart sensor systems, communication is one of the key components and contributes to the significant energy consumption caused by frequent data transfers. In particular, serial interfaces are common in smart sensors [159]. Therefore, numerous low-power data encoding techniques can be applied for serial buses to further improve energy efficiency for a given smart sensor. Among the earliest approaches are those that prioritize lossless (or accurate) transmission, ensuring that data can be precisely decoded at the receiver end without loss of information [160–163]. These encoding methods aim to minimize the number of logic-level changes on the bus, commonly referred to as Transition Count (TC). Many of these techniques employ variations of Differential Encoding (DE) [164], which reduces TC by transmitting the bitwise difference between consecutive words instead of the original data. Well-known encodings such as SILENT [161] and Differential Bar Encoding (DBE) [162] are based on DE principles. However, some approaches seek to address the limitations of DE, particularly the reduced energy savings when consecutive transmitted data are not correlated with each other [160, 163, 165]. However, these advanced lossless encodings often require complex hardware implementations for both the encoder and the decoder, leading to increased silicon area and power overheads. Consequently, while these techniques offer strict output correctness, their potential for TC reduction is constrained, somewhat offset by the anticipated power savings.

In fact, many sensor-based applications have error tolerance properties and allow lossy serial encodings for data buses. These solutions, rooted in AxC principles, prioritize energy savings over output data quality. In a pioneering effort outlined in [166], the authors introduced one of the first lossy encodings tailored to image data. Their method involves saturating the Least Significant Bits (LSBs) of codewords within specified error bounds to diminish the total energy consumption (TC). More recently, a technique named Rake [167] has emerged that strategically flips bits within codewords to lower the TC. Serial T0 [168], on the other hand, employs a constant code with zero TC to transmit data that exhibits minimal variance from the preceding words, thus transmitting only when substantial differences are detected. This strategy

capitalizes on the higher tolerance for errors observed in extended sequences of similar data, referred to as "idle" phases, characterized by low-information-content transmissions. AXSERBUS [159] adopts analogous principles, but offers three operational modes based on the magnitude disparity between consecutive words, achieving a refined balance between energy conservation and data fidelity. Finally, Approximate Differential Encoding (ADE) [164] builds upon DE but incorporates bounded approximations on LSBs to mitigate its limitations with uncorrelated data. ADE and prior lossy encodings focus primarily on TC reduction.

3.3 Electronic Automation Design with Advanced Techniques

Since smart sensors integrate both analog and digital components on the same chip, heterogeneous on-chip integration plays a crucial role in enhancing performance metrics from speed to energy consumption. Therefore, the design of AMS ICs has become a key in this endeavor, as they tightly intertwine both the analog and digital domains to achieve optimal functionality.

As the complexity of AMS ICs increases rapidly, the EDA industry has faced challenges in providing edge-cutting tools to meet urgent needs. Over the past several years, ML and AI techniques have been increasingly incorporated into the EDA industry to enhance efficiency, scalability, and Quality of Results (QoR) for both design and verification tools. In particular, ML has found widespread application across various stages including synthesis, floorplanning, place & route, timing analysis, analog design, and simulation [169, 170]. However, while digital flows have greatly benefited from the integration of ML and AI techniques [169, 170], a significant gap persists in applying these methods within the analog domain [171].

To overcome this limitation, ALIGN and MAGICAL [52, 53] are two of the academic frameworks that made a stride in this progress, using customized device generators. In ALIGN [52], the inspiration was drawn from the learning abilities of ML-based solutions. By mimicking the design process of human experts, NN algorithms were considered to recognize variants of primitives and subblocks. In particular, a trained Graph Convolutional Neural (GCN) network performs classification on bipartite graph representation of target flatten netlists, and output circuit block

recognition, and then a graph matching algorithms were performed to further recognize functional elements inside. MAGICAL [53] focuses on layout and generates layout placements and post-optimization from unannotated netlists. This framework contains components including constraint generation, placement, and routing.

In addition to these state-of-the-art end-to-end layout automation frameworks, there exists some work targeting specific subtopics involving AMS IC design automation. An example is topology recognition, which is an essential stage in the AMS layout automation flow. The topology recognition process aims to identify basic analog functional topologies such as current mirrors, differential pairs, capacitor dividers, etc., within the schematic, in order to apply constraints to the layout placer accurately.

As introduced in Section 1.3.3, graph-based approaches have become a promising paradigm to solve this problem. These approaches convert target schematic netlists into graph-like representations and then frame the topology recognition task as a graph automorphism problem. In Section 6.3, the state-of-the-art topology reconigiton approaches involving AMS IC design automation are introduced.

Chapter 4

Efficient Machine Learning Algorithms for Smart Sensors

This chapter comprehensively studies efficient ML algorithms and their optimizations for smart systems, applied across three sensor-based applications. Extensive experiments were conducted in various scenarios and hardware configurations, demonstrating the effectiveness of the proposed techniques through detailed quantitative analyses on large datasets and real-world devices. In particular, the contributions at the algorithm level targeting smart sensors can be summarized as follows:

DL algorithms for IR smart sensors.

- Targeting privacy-preserving social distancing and people counting tasks based on IR arrays, compact CNNs, and multiple other efficient DL architectures have been proposed and comparatively analyzed. An extensive architecture exploration has been performed, and a rich set of Pareto-optimal solutions in terms of task accuracy and computational complexity has been obtained, providing a wide range of trade-offs for users to deploy DL-based solutions at edge sensor nodes.
- The proposed DL models achieves up to 86.3% best balanced accuracy for social distancing and 82.7% for people counting, significantly outperforming by a state-of-the-art deterministic algorithms [172] (61% and 39.9% respectively). Importantly, per-session cross-validation and cross-dataset validation have been considered in people counting tasks,

ensuring performance when applied in real-world scenarios. In addition, based on architecture explorations, valuable guidelines targeting people counting, which is the one of most common applications on IR sensors, have been extracted to select the optimal model type based on the desired performance range and computational cost.

- Targeting CNN-based social distancing task, an adaptive inference system wherein the complexity of computation can be automatically adjusted based on the difficulty degree of input, has been proposed. The results showed that the total energy consumption can be further reduced by 37-57%, with a limited accuracy drop of <2%.
- Focusing on the DL-solutions deployable on MCU-class platforms for edge sensors, DL-based solutions have been deployed on both commercial MCUs which belong to STM32 series [173, 174] (CNNs and other DL models), and RISC-V platform from academic field [175] (the adaptive system). The detailed deployment results have been obtained including model size, latency and energy consumption for the proposed solutions. The results have demonstrated the efficiency of these solutions on MCU-based IR sensors, and all of them allow real-time inference for years of continuous execution within the power of a single battery.

Compact RFs for acceleration sensors.

- A compact RF implementation of HAR targeting low-power MCUs with a RISC-V instruction set has been proposed. By varying depth and the number of trees in RFs, the Pareto-optimal curves in terms of accuracy and model size/operation cycles have been extracted on two different datasets respectively, to achieve trade-offs for implementation on acceleration sensors. Thanks to C-array representations and quantization techniques, the compact RFs have shown their superiority in hardware-independent metrics, especially in model size (down to $\approx$ 9.5kB memory footprint), while keeping comparable accuracy for activity recognition.

Dynamic inference system for surface Electromyography (sEMG) sensors.

- Based on tiny transformer from [5], an energy-efficient dynamic inference system for gesture recognition based on sEMG sensors has been proposed using staged little/big models. In particular, a lightweight RF is

used as a "rest detector" to trigger more complex transformer models in the following stages, only when the gesture is recognized as "non-rest". Based on the Pareto-optimal tiny transformers by the architecture exploration, the dynamic inference can achieve an accuracy of almost 70%, while offering flexibility to achieve trade-offs in terms of accuracy vs. memory occupation with around 100 points.

4.1 Application 1: Person Counting and Social Distance Monitoring with Multipixel Infrared Sensors

4.1.1 Motivation and Context

People counting is one of the popular tasks involving smart sensors, due to its wide-ranging utility in areas such as safety of the public, urban development, and support for businesses. Its practical applications span from overseeing indoor workspaces, museums, and hospitals to examining pedestrian flow at store entrances, supermarkets, and other public venues. Furthermore, its variant **social distancing**, plays a crucial role in monitoring social distancing compliance and safety regulations, particularly in the context of pandemics such as COVID-19 [12].

There are various technical solutions in IoT for people counting, typically categorized into instrumented and uninstrumented methods [176]. Instrumented approaches take advantage of transceivers on devices that users already possess, such as smartphones, smartwatches, etc. [177]. However, these approaches are constrained by willing involvement and accessibility of equipment, making them challenging to implement in real-world scenarios, especially in public settings. In contrast, uninstrumented solutions do not rely on individual participation and utilize external **smart sensors** like proximity sensors, optical cameras, and infrared arrays [176, 178, 179]. Among these, sensors that detect infrared beams and passive infrared sensors are cost-effective, but may struggle to distinguish multiple nearby individuals and require specific conditions such as object motion, leading to potential inaccuracies [180]. With advances in computer vision and video analysis techniques, vision-based people counting solutions are gaining prominence, often using optical cameras and ML/DL

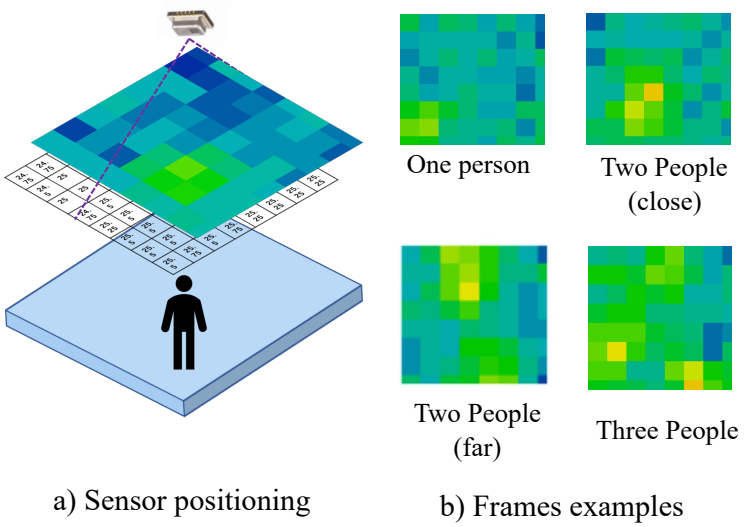


Fig. 4.1 Sensor positioning and IR frames examples.

algorithms to recognize and track individuals in each frame [181–183]. Although effective, these solutions raise privacy concerns as they capture and process sensitive information, such as facial features and body morphology.

In this context, low-resolution **InfraRed (IR) array sensors** emerge as a promising solution, offering benefits such as low energy consumption, affordability, and privacy preservation. These sensors exclusively identify body temperatures, and their limited spatial resolutions (usually 8x8 or 16x16 thermal pixels) merely capture basic body outlines, effectively concealing personal information of individuals. In Fig. 4.1a, an 8x8 IR array [172] is mounted on the roof and located in indoor settings like offices, and so on., using a lens with a field of view. The examples of collected frames and four different people counts cases are shown in Fig. 4.1b. The maximum distance between individuals within the frame varies depending on the sensor height in different environments. Counting individuals in greater regions can be easily accomplished by integrating the results from several sensors strategically positioned for optimal coverage.

Based on such IR frames, the people-counting problem is typically presented as an object recognition challenge [184]. Using such sensors, the task simplifies to classification or regression on the IR frame data, as depicted in Fig. 4.2. Specifically, at time t, denoting x_t as the latest IR frame captured, the input to the recognition model is a single frame $X_t = x_t$ or a sequence of consecutive frames $X = \{x_{t-W+1},...,x_t\}$, where W represents the window size (W = 3 in the illustration). The output is the pre-

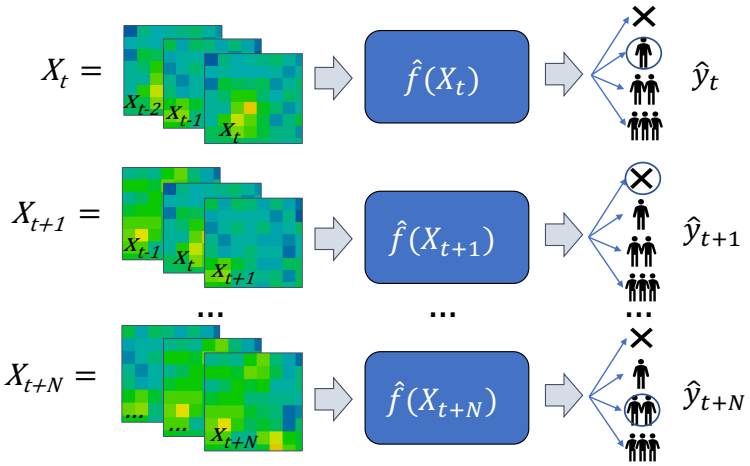
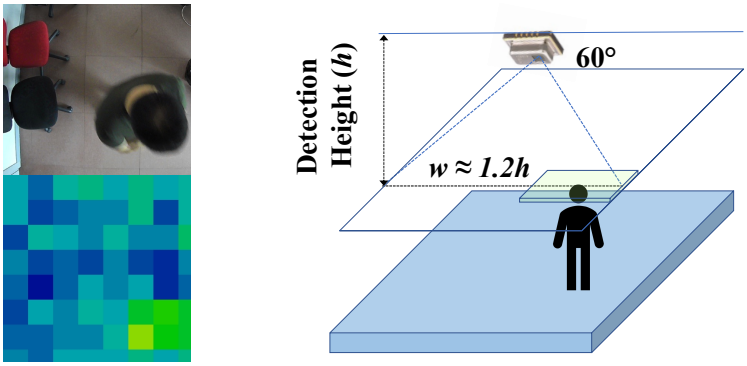


Fig. 4.2 People counting problem formulation based on IR array sensors.

dicted count of people $\hat{y}_t = \hat{f}(X_t)$, which is either a continuous scalar, subsequently rounded to the nearest integer (in the case of the regression problem formulation), for a categorical value that represents one of a set of possible counts (in the classification problem formulation). The mapping between input and output, denoted as $\hat{f}(X)$, can be obtained either by following a deterministic rule-based algorithm or by being inferred from a training dataset using ML/DL methods.

The social distancing problem can be framed as a variant of such people counting formulation. As shown in Fig. 4.3a, IR arrays can sense objects in a specific area. Taking into account the view angle of the IR sensor, the width (w) of the square field of view can be calculated using basic trigonometry. Depending on the design of the IR array, there exists a relationship between w and the distance (h) from the sensor to the identified object. Using a commercial IR sensor Panasonic Grid-EYE (AMG8833) [172] as an example, as illustrated in Fig. 4.3b, The value of w is roughly 1.2x the distance (h) from the sensor to the object being detected. By considering a range of sensor heights and assuming that human heads (the main sources of heat detected by the sensor) are positioned at a minimum of 1.5 meters above the floor, the possible widths (w) can be determined. This leads to a calculable range for the diagonal of the field of view, which indicates the maximum distance between two objects captured within the frame.

Taking into account that the standard suggestion for social distance, which is to keep a minimum distance of 2 meters from the nearest person [185], it can be inferred that any frame containing more than n people indicates a violation. For



(a) (b)

Fig. 4.3 Social distancing problem formulation based on IR array sensors.

example, assume that the maximum distance between two objects within the frame is not greater than 2 meters, then This means that any frame with more than $n = 1$ person represents a violation.

Therefore, the social distancing task is framed as a *binary classification* problem, aiming to predict whether a frame contains $n+1$ ($n+1$=2, in the previous example) or more individuals. Similarly to people counting, as discussed in previous statements, this methodology can be expanded to accommodate taller rooms or wider view angles by adjusting the people threshold to three or more. Additionally, multiple IR arrays can be strategically positioned across the ceiling to cover an indoor space.

To summarize, ultra-low-resolution IR array sensors offer an economical, energy-efficient, and privacy-conscious option for people counting and social distancing. The problem can be formulated as image classification based on a specific ML model. It is important to note that this formulation remains significantly more complex than simple presence detection [186, 187], which involves distinguishing between the absence of individuals and the presence of one or more people. For example, as illustrated in Fig. 4.1, two individuals standing close together may generate a single hot area in the frame, potentially leading to misclassification as a single person.

Therefore, the selection of appropriate smart models to achieve this task becomes an interesting problem. In the literature, there exists a variety of solutions. The next section provided a detailed analysis and discussion with respect to this topic.

4.1.2 Classic People Counting and Social Distance Monitoring Solutions

Classic people counting and social distance monitoring solutions can be divided into two main categories: One category of solutions is based on *deterministic* algorithms, which rely on manually crafted feature extraction techniques customized for particular application contexts. Another category of approaches is *classic* ML-based solutions, such as SVMs and RFs.

Referring to people counting, there exist several works based on deterministic approaches [188–190]. Among them, [188] presented a real-time algorithm for pattern recognition to analyze data from low-resolution IR array sensors installed on doors, effectively identifying the quantity of individuals in a room. Similarly, [189] a sensor mounted on a doorway was employed in conjunction with a body extraction and localization algorithm, along with background determination.[190] proposed a deterministic method that employs a single array sensor placed on a door to monitor the paths of items entering and leaving a space, thereby gauging the number of individuals in the building. Although these methodologies are interesting for making use of only one low-resolution sensor, they focus on a basic and specific cases of people counting, restricted to tallying individuals coming in and going out through a doorway.

[172] described a broad deterministic strategy that uses a ceiling sensor. This technique emphasizes differentiating moving thermal objects from the background using methods such as smoothing, linear interpolation, and categorizing hot regions. Following this, a threshold-based human detection procedure is implemented for each categorized thermal object to determine if it corresponds to a person. Continuous revisions to the base-background image facilitate the automatic removal of immobile warm objects such as heaters.

In addition, research has explored multi-sensor deterministic approaches. For example, a people flow counting algorithm was implemented in [38] to monitor occupancy in intelligent buildings. This algorithm utilizes several low-resolution sensors placed at junctions between different building sections to count the people moving between neighboring areas. On the contrary, the study by [191] offers a framework for indoor people counting utilizing two deterministic algorithms. Their

methodology involves deploying three 16x4 thermal sensors at different locations, each oriented towards x, y, and z directions, respectively.

Classic ML-based solutions in the literature show improvements, especially in terms of accuracy. The investigation in [23] examines the application of three distinct infrared arrays mounted on the ceiling, ranging in resolution from 8x8 to 80x60. Various preprocessing and feature extraction methods are utilized, such as active pixel and active frame detection. Different classification algorithms, including Naive Bayes, KNNs, SVMs and RFs, are then assessed. By conducting experiments on an exclusive dataset, the research reveals that the optimal performance for the lowest resolution array is achieved with an RF. Nevertheless, the study also underscores the substantial impact of the specific set of manually crafted features on the outcomes.

Although people counting with IR sensors has been extensively considered in the literature, the studies targeting social distancing based on low-resolution IR arrays have been much fewer in the past years [192, 193]. During the context of the COVID-19 pandemic, this task has received much attention. Among the early works, [192] used a roof-mounted 24x32 pixel IR sensor and proposed a deterministic solution-based system to achieve indoor social distance monitoring by performing noise removal, threshold processing, and zone detection processing. [193] utilized a wall-mounted IR array and considered a Bayesian framework to recognize based both on body temperature and location of individuals indoors.

It is noteworthy that as explained in the previous sections, under suitable conditions, the above solutions for people counting can be adapted for social distance monitoring. By positioning the IR sensor to detect when a certain number of individuals are within its field of view, the system can identify violations of social distancing rules. The monitoring algorithm can then calculate the count of people and compare it against a predetermined threshold, triggering an alarm when necessary.

4.1.3 Limitations in the Existing Solutions

Although traditional people counting and social distancing based on low-resolution IR smart sensors achieved some success thanks to deterministic approaches and classic ML-based solutions, they also suffer from their inherent limitations. In fact, deterministic algorithms rely on manually designed feature-extraction processes customized for particular application contexts, which implies that these solutions

often lack adaptability when applied to new environments. For example, [188–190] focuses solely on counting individuals entering or exiting a room using IR arrays mounted on doorways. While allowing monitoring of more general cases with IR arrays mounted on ceilings or side walls, [194, 191, 172] either employ a network of multiple IR sensors or relies on a single low-resolution IR array but with unexpected accuracy. Although classic ML-based solutions can alleviate its time-consuming hard-craft feature extraction, they cannot completely eliminate it. In fact, their accuracy is highly dependent on the selected feature set.

In this scenario, DLs offer a promising avenue to eliminate manual feature extraction while achieving higher accuracy. In terms of people counting tasks based on low-resolution IR smart sensors, the previous research has already begun attemps [36, 195–197]. In [36], the researchers employ a CNN with nine convolutional layers and two dense layers to analyze information from an IR sensor with a resolution of 32x24 pixels installed on the ceiling. This setup allows for the identification and enumeration of individuals indoors. Moreover, their approach provides the option to gather lower-resolution data (as small as 8x6 pixels) to cut costs, achieved by using a distinct 8-layer CNN to upscale the frame resolution. In [195], an algorithm has been created to detect heads and count people for wall sensors, using a compact CNN model that is suitable for use on low-power devices with limited memory. This approach focuses on a relatively high-resolution 80x60-pixel IR array. [196] explores FNNs, CNNs, and GRUs for indoor occupancy estimation using ceiling-mounted 24x32 resolution IR arrays. Lastly, [197] uses a ceiling-mounted 24x32 resolution IR array and utilizes an encoder-decoder CNN architecture, which is a simplified variant of U-Net, to determine the location of individuals within the image.

On the other hand, focusing on social distancing, studies based on 8x8 IR sensors utilizing DL models such as YOLOv4 were proposed [198]. Similarly, previously discussed DL-based solutions that focus on people counting can be used as variants to solve social distancing.

To summarize, DL-based solutions achieve end-to-end inference and state-of-the-art accuracy for people counting and social distancing tasks. However, this does not mean that these approaches can be easily applied in a smart sensor. The crucial limitations are that these DL-based solutions are currently deployed solely on high-performance, energy-intensive PC-class processors [23, 36], or are responding on relative high-resolution IR arrays [38, 195]. Moreover, large DNNs cannot be

applied directly on a smart sensor, which is a resource-limited and energy-crucial platform at edge.

In detail, such limitations existing in the deployment of DL solutions can be considered in the following aspects:

Model size. As first designed for cloud-based approaches, DL models, especially those with numberous layers and parameters, tend to have a large model size. While Cloud computing platforms and even PC-class processors can store these DL models easily, deploying such huge models on resource-limited sensor devices can be impractical due to its limited storage capacity. For instance, targeting people counting task, [23] proposed a CNN model with 520KB model size, while STM32L412KB [199], one of STM32 MCU series, with only 128KB of Flash memory, makes it impossible to deploy such DL-based solution on sensors based this MCU.

Processing speed. The applications based on sensors often requires real-time processing of sensing data to make quickly responses. However, DL-based solutions often require high computational complexity, make it difficult to achieve real-time performance on edge sensors. In particular, delays in processing can impact the effectiveness of the system and may even lead to incorrect results.

Energy efficiency. High computational complexity nature in DL-based solutions also implies high energy consumption. Nevertheless, applications at edge often operate under strict power constraints, such as battery-powered sensors. Therefore, deployment of such DL models may require frequent recharging or battery replacement, significantly limiting the operational lifespan of devices.

Due to these limitations, how to deploy DL-based solutions on resource-limited sensor nodes while maintaining their state-of-the-art task performance for applications at edge becomes an attractive topic in the real-world development of sensors. It is a significant concern for developers to consider the trade-offs between acceptable model complexity fitting the resource-limited sensors and high task performance achieved by DL-based solutions. To achieve this goal, lightweight DL models fitting in edge devices with fewer layers and parameters have been studied, and techniques such as quantization have been widely used to reduce model size. At the same time, the performance accuracy of DL models may drop slightly due to these techniques.

On the other hand, it makes the consideration of model type in DL family becoming one of important factors in practical application development in smart sensors. Depending on the various architectures from different DL families, the accuracy performance, operations and number of parameters vary. For example, targeting simply object detect tasks based on IR images, some lightweight CNNs can achieve acceptable accuracy, while more complex models adding LSTM layers achieves higher accuracy. This fact reveals the different DL alternatives may impact development of smart models in sensors for specific tasks. In particular, Section 4.1.5 provide a complete and in-depth discussion focusing this point.

4.1.4 Efficient Deep Learning Models for Social Distancing

In this section, the contributions of this thesis concerning the deployment of efficient DL models for social distancing are presented in detail. The work described in this section has been previously published in [12, 13].

CNNs and IR-based applications

To take advantages of DL to overcome classic solutions for the target IR-based applications as discussed in the previous sections, one of important questions is that teh selection of DL models for the specific task. As discussed in 4.1.2, IR-based social distancing task has not attracted the same level attention as other applications until COVID-19 pandemic, since the needs for this application was increasing sharply. In fact, most applications based on DL-solutions with low-resolution IR arrays are targeting activity recognition [33, 40, 200, 41], person detection [187], and people counting [36, 195–197]. Among these works, CNNs become the first choice as either the standalone model [40, 200, 187, 36, 195–197] or combining with other DL layers (LSTM, GRU) [33, 41]. This is due to the effectiveness of CNNs on visual image processing, which has been widely validated in numerous studies. In turn, these works further demonstrated the superior performance of CNNs on IR frames. Nevertheless, compared to the target social distancing, activity recognition and people counting are more complex tasks, while presence detection is easier. Considering this fact, the existing works either using additional pre-processing to thermal images [33] and multiple IR sensors positioning different directions [40], or complex model architectures with deep layers [40, 36, 195–197] or combining with

LSTM/GRU layers [33, 41]. Moreover, these works use relatively high-resolution sensors including 24x32 to 80x60 (e.g, [196, 195] etc.), making the privacy cannot be guaranteed as the same level with ultra-low-resolution ones (e.g, only 8x8 pixels).

Therefore, there are two questions can be proposed for the DL models targeting social distancing:

Question 1: Since CNNs have shown their superiority in several IR-based tasks, can they handle social distancing tasks based on a *single* IR array with *ultra-low-resolution* outputs as small as 8x8 pixels?

Question 2: What is the trade-off between the complexity of CNN architectures and task performance targeting edge sensors on social distancing applications?

To answer these two questions, this work focuses on the dedicated implementation of a CNN-enabled social distancing system on microcontrollers with an 8x8 IR array. In particular, CNNs with effective layers in existing pieces of literature but with smaller sizes are introduced, more details are illustrated in the following section.

Efficient CNNs

CNNs [201] are widely recognized as one of the most effective DL models for analyzing visual imagery. To prove that simple CNNs are applicable for smaller-scale tasks and can attain state-of-the-art performance in a TinyML scenario with reduced computational complexity compared to typical CNNs with deep architectures, a series of straightforward CNN classifiers are developed, to implement the proposed social distance monitoring system based on IR arrays.

These models are designed following a common simple template inspired by classic CNN architectures such as LeNet-5 [201], as illustrated in Fig. 4.4. The template comprises two convolutional layers with Rectified Linear Unit (ReLU) activation, one Max Pooling layer, and two Fully Connected (FC) layers. The first FC layer utilizes ReLU activation with a hidden size of 64, while the second FC layer employs sigmoid activation with a single output neuron, generating the probability of a violation of the social distancing rule.

Beginning with this template, a comprehensive architectural exploration is conducted. Various network configurations are derived by removing certain layers

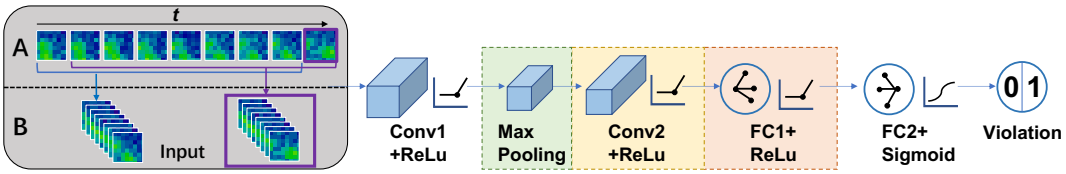


Fig. 4.4 CNN architecture and input generation.

enclosed in dashed boxes in Fig. 4.4. Specifically, architectures are explored with: i) one or two convolutional layers, ii) one or two FC layers, and iii) with or without Max Pooling. Additionally, the number of channels in each of the two convolutional layers is varied from {8, 16, 32, 64}.

Two variations of each CNN architecture are developed, differing in the processed input. The first network is fed with a single thermal frame X_t and is trained to predict whether the corresponding number of people y_t exceeds 1, indicating a violation of social distance. The second variant receives a sliding window of W consecutive frames $(X_t, ..., X_{t+W-1})$ and is trained to identify social distance violations in the last frame of the window (i.e., $y_{t+W-1} > 1$). The input tensor shapes for these variants are (8,8,1) and (8,8, W), respectively, meaning that the W frames are treated as distinct input channels for CNN. These two input types are depicted in the left portion of Fig. 4.4 as A and B, respectively. To ensure clarity, a single input sample of each type is enclosed in a purple box.

The second variation of the CNN model benefits from accessing previous frames, enabling it to identify situations where a single heat source is associated with two nearby individuals (as depicted in Fig. 4.1) based on their movements. However, this version of the model involves increased computational complexity, which is especially critical for on-device inference. In the experiments carried out, a window size of $W = 8$ is chosen to ensure a fair comparison with a cutting-edge deterministic algorithm detailed in [172], which utilizes an 8-frame window for background subtraction. Furthermore, as a strategy to decrease memory usage and enhance the speed and energy efficiency of the considered CNN models, all parameters and activations are quantized to 8-bit integers [202].

Adaptive Inference

Instead of *static* optimization such as quantization in the previous section, there exist opportunities to further reduce energy using the dynamic approach. Given that not all inputs pose the same difficulty in classification, a CNN can be particularly valuable in discerning whether the thermal pattern detected by the IR sensor corresponds to a single person (acceptable) or two individuals in close proximity (indicating a violation of social distancing). On the contrary, a much simpler classifier suffices to identify the presence of at least one person in the frame. Building upon this insight, an adaptive inference system is proposed for low-resolution IR arrays that target social distancing. The term *adaptive inference* denotes a runtime optimization mechanism where the computational complexity is automatically adjusted based on the difficulty level of the inputs. Examples of adaptive systems include big/little models [149, 203], early-exit strategies [204, 16], and hierarchical/staged classifiers [205]. Following the latter approaches, the proposed adaptive system is a straightforward, yet efficient hierarchical classifier centered on a lightweight wake-up trigger.

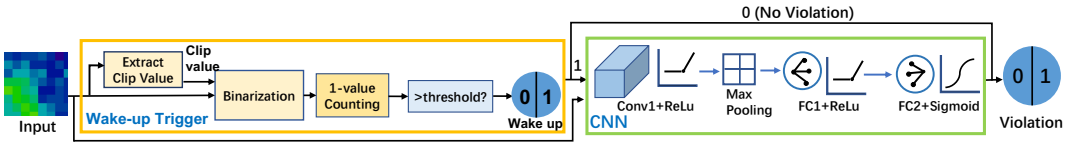


Fig. 4.5 Overview of the adaptive inference system.

To implement such an adaptive inference system, we integrate a straightforward and deterministic classifier termed the Wake-up Trigger with a compact, 8-bit quantized CNN. The entire adaptive framework is depicted in Fig. 4.5. The social distance monitoring task is divided into two stages, each stage being framed as a binary classification problem. Initially, the wake-up trigger identifies whether there is at least one person present in the frame. If this initial stage yields a "no-person" prediction, the inference process halts immediately, resulting in a 0 output, signifying no violation of social distancing. However, if the wake-up trigger detects the presence of a person, the second stage is activated, initiating another binary classification using the CNN. This stage produces the final output, determining whether a violation of social distancing has occurred or not.

As depicted in Fig. 4.5, the Wake-up Trigger primarily functions as a threshold-based classifier. It operates by converting the pixels in the IR frame into binary values (0/1) based on the measured temperature. Subsequently, the count of pixels

with a value of 1 is tallied and compared against a predetermined threshold. If this threshold is surpassed, it is inferred that at least one person is present in the frame, prompting the CNN to invoke it. The clipping value utilized for binarization is recalculated periodically by analyzing a window of N consecutive frames where no individuals are detected and determining the maximum temperature among all pixels (illustrated by the "Extract Clip Value" block in the figure). This approach is based on the observation that when a person passes under the IR array, their head will cause a temperature elevation compared to the background, especially because of the shorter distance from the sensor compared to other heat sources in the background (e.g., heaters). Essentially, the input binarization performed by the Wake-up Trigger serves as a form of background removal. In experiments, $N = 8$ was selected for this window size.

Once the human presence is detected by the Wake-up Trigger, the CNN classifier is triggered. In particular, in Section 4.1.4, 8-bit quantized CNN architectures are considered to predict social distancing violations. Specifically, among the various CNN variants presented in that work, the one that demonstrated the highest balanced accuracy is chosen, as depicted in Fig. 4.5. Furthermore, to enhance the final classification accuracy, Batch Normalization (BN) is added. The CNN architecture comprises a single Convolutional (Conv) layer with BN and ReLU activation, followed by a Max Pooling layer and two FC layers with ReLU and sigmoid activations, respectively. The Conv layer is configured with 64 channels, and the hidden size of the first FC layer is also set to 64.

4.1.5 Efficient Deep Learning Models for People Counting

As discussed in section 4.1.3, existing studies have demonstrated the potential of DL to achieve remarkable performance in people counting tasks using IR arrays. Nevertheless, the existing literature lacked a comprehensive comparative analysis of different efficient DL architectures tailored for IR-based people counting. Such an analysis should not only evaluate their accuracy but also consider the deployment costs associated with memory and energy constraints, especially in such edge nodes as smart sensors.

Notably, such extensive comparative exploration for DLs is vital to the design of the smart model in IR smart sensors. First, previous literature that used DL-based

solutions for people counting only considered a single DL model, which limits the exploitation of the maximum potential of DLs since the optimal solutions in the design space are never searched. Second, the promising results presented in several of these studies [195, 38, 23] are overshadowed by biased data partitioning methods, which often involve random sampling at the frame or sliding window level. This data-splitting approach oversimplifies the task, leading to unrealistic evaluations.

Therefore, in this section, Six well-known families of DL models are being examined for predicting the number of people in IR frames, investigating certain important hyperparameters of each model. A visual depiction of all the models under consideration is presented in Fig. 4.6. The work described in this section has been previously published in [14].

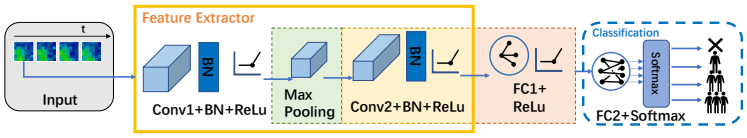
Single-frame CNN. The initial architecture considered is a basic CNN, which has demonstrated effectiveness in numerous image-based pattern recognition tasks and, importantly, in the previous social distancing task. The general architecture of basic CNNs considered is similar to the ones in the previous social distancing task, distinguished by the inclusion of additional BN layers, as illustrated in Fig. 4.6a. It comprises of up to 2 Conv layers followed by a BN layer and employed ReLU activation, along with an optional Max Pooling layer, and up to 2 FC layers. The first FC layer features 64 hidden units and a ReLU activation function, while the output layer contains several neurons corresponding to the possible count "classes" (ranging from 0 to 3 people, translating to 4 output neurons in the experiments).

Using this template as a foundation, we conducted an extensive exploration of architectures by retaining or eliminating the layers depicted within the dashed boxes in Fig. 4.6a. Specifically, the architectures were explored following strategies:

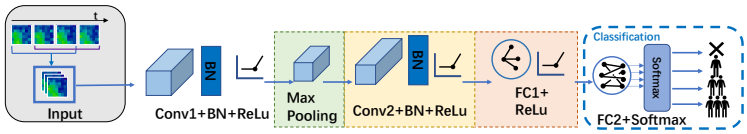
1. contains 1 or 2 Conv layers, each followed by BN;
2. contains 1 or 2 FC layers;
3. contains 0 or 1 Max Pooling layers.

In addition to modifying the layer number, various numbers of feature maps (referred to as *channels*) in each Conv layer were explored, with choices varying from 8 to 64. The sizes of the convolutional and pooling kernels

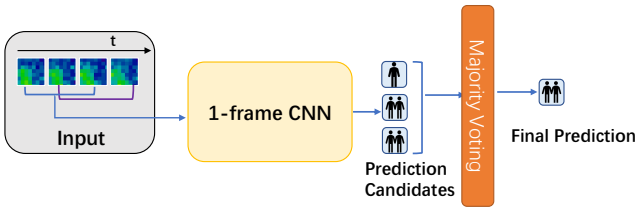
remained consistent at 3x3 and 2x2, respectively. The input fed to this CNN architecture is a single IR array frame, labeled $X_t = x_t$, having a tensor shape of (8, 8, 1). Altogether, 48 unique single-frame configurations were utilized in this model.



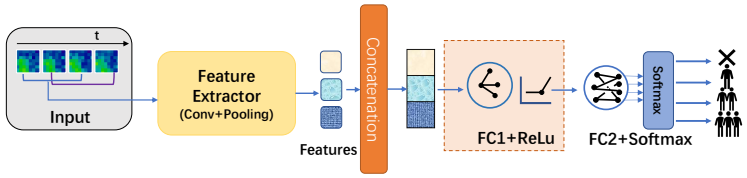
(a) Single-frame CNN



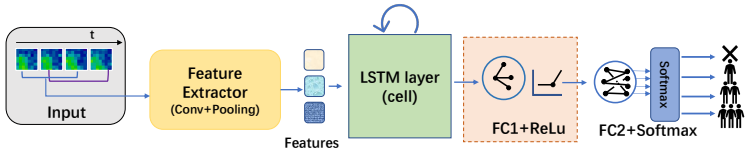
(b) Multi-channel CNN



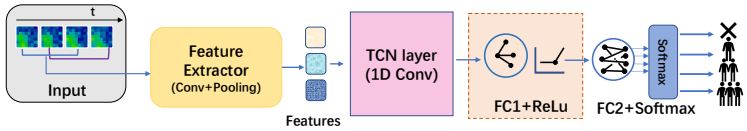
(c) Majority Voting CNN



(d) Concatenated CNN



(e) CNN-LSTM



(f) CNN-TCN

Fig. 4.6 DL model architectures.

Multi-channel CNN. Unlike the CNN *Single frame CNN* model that operates on a single IR array frame, all other models aim to leverage the temporal

information embedded within a sequence of consecutive frames to improve the accuracy of people counting. The principle behind this approach is that analyzing a sliding window of IR frames as input can obtain information on the movement patterns of objects, thereby enhancing prediction accuracy, especially in complex scenarios. For example, as depicted in Fig. 4.7, in the final frame, a specific hot region (indicated by a purple box) can be accurately linked to two nearby individuals instead of being assigned to only one person. This linkage is established by tracking the movements of the two individuals (depicted by red and orange circles) in the preceding frames.

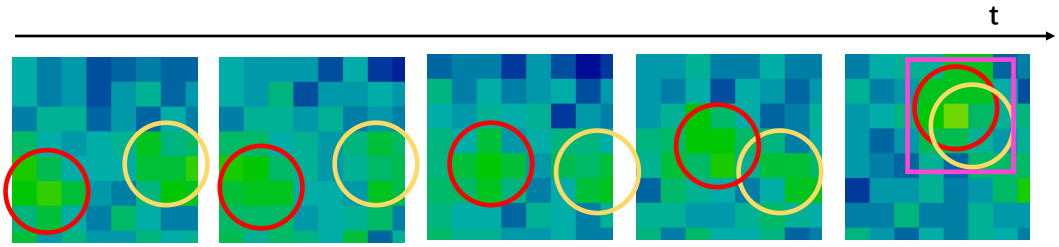


Fig. 4.7 An instance of an infrared frame sequence involves two individuals in motion as they approach each other closely.

The initial and most straightforward approach to processing multiple IR frames involved feeding these multiple frames into a CNN as distinct input channels. Specifically, given the length of the sliding window as W, a tensor with a shape of (8, 8, W) is constructed by stacking IR frames $X_t = \{x_{t-W+1},...,x_t\}$ along the channel axis. This tensor is then paired with the people count label of the last frame y_t for both training and testing purposes. These inputs and outputs serve as the foundation for all subsequent multi-frame architectures discussed below. The architecture template of the Multi-channel CNN model is illustrated in Fig. 4.6b. The same hyperparameter configurations used for Single-frame CNNs were evaluated, including the number of layers and the number of Convolutional channels. Furthermore, we varied the window size W within $\{3, 5, 7, 9\}$. This exploration is significant because, intuitively, a window that is too short limits the benefits of accessing past frames, whereas an excessively long window provides irrelevant information from too-distant past frames, thereby increasing the time and memory complexity of the initial Conv layer.

Majority Voting CNN. Majority voting serves as a straightforward yet effective ensemble learning strategy that takes advantage of multiple classification outcomes to produce final predictions with reduced variance [206]. Over recent years, numerous studies have employed this technique, using different classifiers, multiple instances of the same model trained with variations, or a single trained model fed with diverse inputs. In this work, the latter approach was adopted, applying majority voting (i.e., *mode inference*) to the W predictions generated by running a Single-frame CNN on each frame within the sliding window. A schematic overview of this approach is depicted in Fig. 4.6c. One clear advantage of this technique, particularly in edge-inference scenarios, is that it demands approximately the same memory as a single-frame CNN, potentially enhancing prediction accuracy by mitigating occasional misclassifications. However, the latency and energy expenditure for inference is approximately W times higher compared to a Single-frame model. Once again, W varied by values within $\{3, 5, 7, 9\}$. It is worth noting that majority voting necessitates an odd window size. Hence, for a fair comparison, all other multi-frame architectures have been evaluated solely with odd W values.

To simplify the complex architectural exploration, in this work, only the models derived from Pareto-optimal Single-frame CNNs were considered to be used as independent predictors to generate the majority-voting models. In essence, majority voting was applied on top of all single-frame CNNs identified within the Pareto front in terms of people counting accuracy versus model size or computational complexity (represented by *the number of operations*).

Concatenated CNNs. While majority voting offers the advantage of not requiring additional trainable parameters, its primary drawback lies in its inability to assign varying importance to different IR frames within the sliding window. Intuitively, more *recent* frames should be more important in determining the count of people, especially with larger window sizes (W). Although this could potentially be addressed through a weighted voting mechanism, such a solution requires intricate manual tuning of the weights assigned to each frame. Therefore, the subsequent DL models in this work tackle this limitation by adopting a concatenation approach of features, as proposed in previous research [207, 208]. In this approach, illustrated in Fig. 4.6d, each frame within the sliding window undergoes individual processing through a feature extractor module to extract time-independent features. Subsequently,

all outputs are flattened and concatenated into a unified feature vector, which is further processed by two FC layers to generate the final prediction. This methodology enables the training process to automatically assign appropriate weights to the features extracted from different frames.

To construct concatenated CNNs, the configurations of Conv and Pooling layers (i.e., the components highlighted by an orange box in Fig. 4.6a) were identified within each Pareto-optimal Single-frame model as potential feature extractors. Additionally, beyond exploring the four values of W as in previous, the number of neurons also varied in the first FC layer within {8, 16, 32, 64}. Overall, with N Pareto-optimal feature extractors available, we evaluate a total of 4 * 4 * N concatenated CNNs.

CNN-LSTM. The subsequent multi-frame model explicitly addresses the temporal dependency between frames by replacing simple feature concatenation with a Long Short-Term Memory (LSTM) cell. Numerous studies have explored CNN-LSTM models to integrate spatial and temporal information effectively [209–211]. A relevant study for this work is [33], which used a CNN-LSTM architecture for human activity recognition based on a 16x16 IR array. These investigations have showcased the remarkable performance achieved by CNN-LSTMs. However, it is noteworthy that LSTM cells are less hardware-friendly compared to CNNs. Therefore, the comparison between this model and other architectures is interesting, considering the trade-off between complexity and performance.

The CNN-LSTM architecture considered, depicted in Fig. 4.6e, follows a template where the outputs of the W feature extractors are flattened and sequentially fed into the LSTM cell. Subsequently, one or two FC layers are connected to the last hidden state generated by the LSTM to produce the output prediction. The same feature extractor selection strategy was applied as previously described for concatenated CNNs. Additionally, W was varied as before, as well as different numbers of hidden units were explored in the LSTM cell, selecting from values in {8, 16, 32, 64}. Once again, considering N Pareto-optimal feature extractors, a total of 4*4*N CNN-LSTM architectures were evaluated.

CNN-TCN. Last but not least, the model type considered is based on Temporal Convolutional Networks (TCNs), which have recently emerged as a more

hardware-friendly alternative to LSTMs and have found application in various edge-relevant tasks. TCNs are essentially 1D CNNs but with the distinct characteristic of employing *causal* convolution, making them suitable for time-series processing.

Compared to LSTMs, TCNs exhibit greater data reuse and are more resilient to integer quantization, making them advantageous for edge deployment. Therefore, the last architectural template in this work combines the outputs of typical 2D CNN feature extractors applied to single IR frames with a single TCN layer, as depicted in Fig. 4.6f. The TCN output is then flattened and fed into 1 or 2 FC layers to generate a prediction. The 1D Conv kernel size is contained at 3x1 and the dilation at 1. Along with the variation W as in previous models, the number of output channels of the TCN layer was explored, considering values in {8, 16, 32, 64}. Therefore, with N feature extractors, a total of 4 * 4 * N architectures were also explored in this case.

4.1.6 Experimental Results

Datasets

While there are various publicly accessible datasets comprising infrared (IR) array sensor images, none of them meet the specific requirements of the intended application. The existing datasets typically feature high-resolution sensor data (ranging from 160x120 pixels to 640x480 pixels) tailored mainly for applications in fields such as robotics, autonomous driving, and related areas [212–215]. The sole known publicly available dataset containing low-resolution IR images is referenced as [216]. Nonetheless, this dataset is designed for more complex tasks related to human activity recognition and involves combining data from three distinct IR sensors positioned on walls in varied orientations.

To address this gap, a brand new labeled dataset called **LINAIGE** (Low-resolution INfrared-array data for AI on the edGE) [13], specifically tailored for person counting and social distance monitoring has been collected, which is now available open source. A ceiling-mounted sensor was selected because it offers greater consistency in indoor environments, where variations in ceiling height are typically less significant than differences in room size. Consequently, the shape of a person captured from above in the thermal image remains relatively consistent across different en-

vironments. Similarly to [216], the IR frames in the data set were captured using a Panasonic Grid-EYE (AMG8833) [172], which generates an 8x8 array and features a viewing angle of 60 degrees.

In order to gather the data, a setup was created using a Raspberry Pi 3 that was outfitted with the Grid-Eye sensor and a regular USB camera, both oriented in the same way. The camera and sensor captured synchronized frames at a frequency of 10 Frames Per Second (FPS) in different indoor settings like offices, research labs, and hallways, across six distinct **sessions**. In each session, a maximum of five volunteers were invited to participate in activities including standing, walking, and so on, within the range of sensor.

At the end of the data collection, a semi-automated method was employed to expedite the labeling of the IR data. More precisely, a pre-trained Mask R-CNN [107] model was applied to optical images to identify and label individuals in each frame. Subsequently, the identified count of individuals was matched with the respective IR frame.

Human annotators were then presented with both optical and thermal images and tasked with confirming or adjusting the people count proposed by Mask R-CNN. Human verification was essential not only to rectify misclassifications made by Mask R-CNN but also due to difficulties in precisely aligning and matching the viewing angles of the camera and IR array. In instances where imperfect alignment occurred, certain IR frames depicted the heat signature of a person (often near a corner) that was not visible in the corresponding optical frame. Annotators were given the choice to mark such "hard-to-label" frames with a binary *confidence* level, which could be excluded from the training and testing sets.

Detailed information for each collected session is provided in Table 4.1. An example of optical and IR frames captured during the data collection process is shown in Fig. 4.3a.

Taking into account the problem of **social distancing**, based on the statistical information in the dataset as shown in Table 4.1, the sensor height range spans [2.4: 2.7] meters, therefore, refer to the problem formula in Section 4.1.1, the length of the detection area (w) spans a range of [1.08: 1.44] meters. As a result, the detection area is up to $\approx 2\ m^2$, and the diagonal of the field of view, representing the maximum distance between two objects within the frame, falls within the range of [1.53 : 2.04] meters. Therefore, it can be concluded that any frame containing more than

Table 4.1 LINAIGE Dataset information in social distancing. Viol = violations.

Session	N. Frames	Viol. [%]	N. People	Sensor Height [m]	Room
1	18124	34.3	0-5	2.4	A
2	1581	54.5	0-2	2.7	B
3	1519	6.4	0-3	2.4	A
4	196	5.6	0-2	2.4	A
5	2202	22.7	0-3	2.4	A
6	1850	32.3	0-3	2.7	C
Tot.	25472	32.5	-	-	-

1 person corresponds to a violation. The statistical information on the violations is also reported in Table 4.1.

According to the information in Table 4.1, all frames that belong to the "hard-to-label" have been excluded from both the training and the testing datasets. The training set consists of Session 1, while all other sessions make up the test set. This division by session guarantees that test instances are taken from either a different room than the training data or from the same room, but on different dates or at different times. In contrast, a completely random division where consecutive frames from the same session could be placed in different data groups would lead to a too simplistic and unrealistic portrayal of the issue.

Targeting the **people counting problem**, the same dataset was utilized but in a new version with improved data quality. In detail, there are two changes: i) The very rare frames with > 3 people (0.66% of the total) are removed. It is based on the fact that in such a small detection area (up to 2 m^2) in the dataset, it is very rare to contain four or more people. The statistical data also supported this finding, indicating that such uncommon occurrences occurred only once during a session, making it challenging to train and perform cross-validation for ML/DL models. ii) The shortest session (session 4 in Table 4.1) is removed, which contained only 196 frames, equivalent to approximately 20 seconds of data. This session has the potential to inaccurately influence the metrics for recognition performance. After making these modifications, the revised dataset now comprises 25110 samples, which are organized into 5 sessions. Each session is provided with a timestamp, environment label, and room temperature.

Table 4.2 LINAIGE Dataset statistics and CV strategy in people counting.

Train Fold						**Test Fold**					
Session	Sample N.	People Counts Statistics [%]				Session	Sample N.	People Counts Statistics [%]			
		0	1	2	3			0	1	2	3
1, 3, 4, 5	23529	26.07	43.49	23.61	6.83	2	1581	14.86	30.68	54.46	0
1, 2, 4, 5	23591	22.37	44.03	26.84	6.77	3	1519	71.89	21.72	5.66	0.72
1, 2, 3, 5	22908	25.3	41.85	26.17	6.67	4	2202	26.02	51.27	19.16	3.54
1, 2, 3, 4	23260	24.69	43.02	26.08	6.20	5	1850	33.78	38.38	18.92	8.92

Similarly, in all experiments conducted in this task, frames labeled "hard-to-label" were excluded from both the training and testing datasets for both the proposed method and state-of-the-art comparisons. Additionally, diverging from the approach in the social distancing task, where a simple per-session train/test split was utilized, a per-session Cross-Validation (CV) approach was adopted. This was done to ensure that the model evaluation remains independent of the characteristics of any specific test session. The cross-validation strategy is illustrated in Table 4.2.

Given that session 1 in the dataset contains a notably higher number of frames compared to the rest (17958 frames versus a maximum of 2202 for other sessions, which comprise 71% of the total data), it was consistently included in the training set. Sessions 2, 3, 4, and 5 were alternatively assigned as the test set in different iterations, while all other data remained in the training set. This rotation produced 4 CV folds. This strategy of *leave-one-session-out* CV ensures unbiased model assessment by guaranteeing that test frames are from a different setting in terms of environment, date time, and room temperature compared to training frames. This mirrors a real-world scenario where the system is tested in a setting different from where it was trained. In contrast, a completely random per-frame division would lead to leakage of information between training and testing, oversimplifying the task.

Training Protocol

Developed for the target social distancing system, all variations of CNN structures and input dimensions can be found in section 4.1.4, where a total of 96 architectures were trained. Each model was trained five times using different random seeds, with a maximum of 500 epochs and early stopping implemented after 10 consecutive epochs without improvement. The training utilized a binary cross-entropy loss function in conjunction with the Adam optimizer, which was initialized with a learning rate

(LR) of $1 \cdot 10^{-3}$. The learning rate was decreased by a factor of 0.3 when progress plateaued, with patience of 5 epochs.

After the initial training with floating-point numbers, the parameters, inputs, outputs, and intermediate activations of the models are converted to 8-bit integers using the TensorFlow Model Optimization (TFMOT) API [202]. This step is crucial to further reduce memory usage, latency, and energy consumption when deploying the models on MCU-based IoT nodes constrained [147]. Subsequently, Quantization-Aware Training (QAT) [217] is applied to mitigate the accuracy degradation resulting from quantization. The same training protocol as described above is used, with a difference: the initial learning rate is configured as 5×10^{-4}.

To address the class imbalance observed in the target dataset (refer to Table 4.1), sample weights were used to the loss function. These weights were set equal to the inverse of the class probabilities. Following the training of all CNNs, those that were in the Pareto front were analyzed and discussed, as detailed in Section 4.1.6. Note that the quantized CNNs in the adaptive system (see section 4.1.4) were obtained in the same training protocol and then selected from Pareto-optimal solutions in terms of accuracy vs. the number of parameters or the number of operations.

Designed for counting people, where 6 families of DL are being considered, all models were trained by utilizing the CV strategy of leave-one-session-out outlined in Section 4.1.6. Initially, standard training for the floating-point model was performed using Keras / TensorFlow 2.0 [202] for each fold, with a maximum of 500 epochs. The training protocol is the same as social distancing, with three differences: i) a categorical cross-entropy loss function was optimized using the ADAM optimizer, for both floating point training and QAT training; ii) during QAT training, The values for adjusting the learning rate scheduling and early stopping patience are set to 10 epochs and 20 epochs, respectively.

Moreover, it is important to mention that the TFMOT API does not currently support QAT for LSTM cells. As a result, CNN-LSTM models are deployed to the MCU in floating point format, which poses a notable practical constraint on CNN-LSTMs.

Experimental Results of Social Distancing

Result of efficient CNNs.

Initially, effective CNNs were trained and quantized utilizing Keras/TensorFlow 2.0 [202], and the X-CUBE-AI v6.0 toolchain [218] was employed to transform the trained models into optimized C code for inference. In light of the imbalanced classes in the target dataset, the evaluation of performance primarily centers on balanced accuracy (Bal. Acc.), calculated as the average of sensitivity and specificity. Additionally, measurements such as Area under the ROC curve (ROC-AUC), F1-Score (F1), and standard micro-average accuracy (Acc.) [219] are also included. The results for all metrics are presented as the mean ± standard deviation across the 5 training iterations.

The hardware-independent expense of CNNs, which are utilized for generating Pareto curves, is evaluated in terms of the model size (number of parameters) and the quantity of Multiply-and-Accumulate (MAC) operations. Following this, the efficiency of the models running on a commercial MCU known as STM32L476RG [173] is further evaluated with respect to the inference speed, energy consumption, and memory utilization, as in Table 4.3. In particular, the model size (Size) is reported by X-CUBE-AI toolchain, and total Flash occupation (Mem.) is obtained by STM32CubeIDE [220]. The average inference cycles are measured by STM32CubeIDE by running 1000 times, and then based on the clock frequency and supply voltage as STM32L476RG datasheet [173], the latency and energy are estimated as:

$$Latency = Cycles/Frequency, \tag{4.1}$$

$$Energy = Power \times Latency. \tag{4.2}$$

The comparison is made primarily against the approach introduced in [172]. The method from [172] is run as is and the count of individuals is directly transformed into a violation of social distancing whenever two or more people are detected within a frame.

Fig. 4.8 demonstrates the Pareto curves produced by different CNN architectures concerning balanced accuracy vs. the number of parameters and MAC operations. The two curves depict the single-frame and windowed input variations (labeled as A and B in Fig. 4.4), where each data point represents a unique architectural configuration, such as the type and number of layers, channel numbers, etc. The

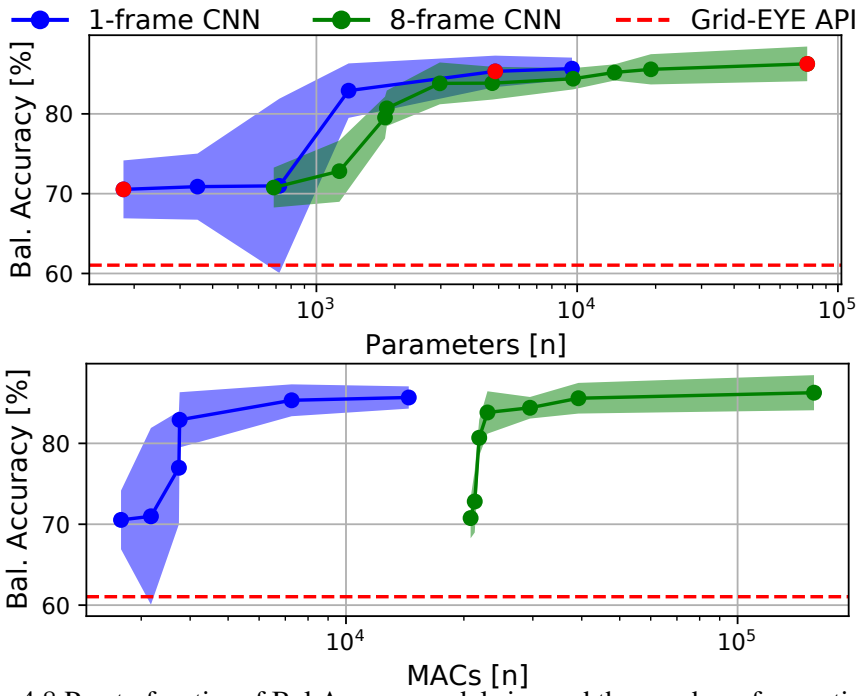


Fig. 4.8 Pareto frontier of Bal.Acc vs. model size and the number of operations.

points indicate the average accuracy across 5 training iterations, while the colored bands show the range of ± standard deviation. The horizontal dashed line represents the accuracy obtained by the baseline method introduced in [172].

The results indicate that all suggested CNN architectures show a notable improvement in accuracy compared to the baseline. Specifically, single-frame networks achieve impressive accuracy levels (> 80%) with approximately 1k parameters and less than 10k MACs. On the contrary, windowed networks have higher computational complexity, particularly in terms of operations, but achieve the highest overall accuracy (86.3%) with 75k parameters and 157k MACs.

Overall, it was noted that both types of models demonstrate improved efficiency when employing a limited number of channels in convolutional layers (architectures with 64 channels do not fall within the Pareto front). This is probably due to the dataset being relatively small and uncomplicated, where an excessive number of features could result in overfitting.

Subsequently, three designs were chosen from the parameters panel in the Pareto curve (highlighted by red dots in Fig. 4.8) and implemented on the target MCU. Specific outcomes are detailed in Table 4.3. The selected CNNs consist of: i) the most precise model, ii) the most compact model that achieves a < 1% balanced

accuracy reduction compared to the most accurate model, and iii) the smallest model overall. In addition, the code from [172] was compiled and evaluated on the same MCU for comparative purposes.

For the CNNs under consideration, the table presents both the quantized model size in bytes (Size column) and the total occupied Flash memory, which includes the code size (Mem). It is evident that the proposed CNNs outperform [172] in all the accuracy metrics evaluated. The absence of ROC-AUC data for [172] is due to this method that produces a deterministic 0/1 value instead of a probability score. In terms of balanced accuracy, the proposed models exceed [172] by 9.5-25.2%, depending on the specific selected CNN. Additionally, MinSize and MaxAcc-1% are respectively 5.6x and 4.1x faster and more energy-efficient than [172].

Even though the total memory footprint of the considered models is larger, this is mainly due to the extensive size of the X-CUBE-AI code ($\approx$ 60kB). Furthermore, all three CNNs fit easily within the 1MB Flash memory of the target MCU. In particular, due to the use of an 8x8 sensor with very low resolution, MinSize/MaxAcc CNNs are 60.6x/8.3x faster compared to the networks proposed in [196], and impressively 235000x/32000x more energy efficient than the approach presented in [195], which are the only two studies suggesting MCU implementations of ML tasks based on IR arrays. However, it is important to note that, in addition to using different inputs, [196, 195] also tackles a slightly different task (person counting), making a direct comparison with this study challenging.

Table 4.3 Evaluation and deployment Results of three selected architectures on STM32L4 @ 80MHz (12.8 mW average power). 1/8F = 1-frame/8-frame input, Cn = Convolution with N channels, P = Max Pooling, FC = Fully-Connected.

Model	Bal. Acc. [%]	ROC-AUC	Acc. [%]	F1	Size [B]	Mem. [B]	MACs	Energy [µJ]	Latency [ms]	Architecture
MinSize	70.5±3.6	0.83±0.04	75.6±0.9	0.56±0.05	0.18k	63.2k	2.6k	9.38	0.728	1F-C8-P-FC
MaxAcc-1%	85.3±2.0	0.93±0.04	86.4±1.9	0.77±0.03	4.9k	68.2k	7.3k	12.93	1.005	1F-C8-P-FC-FC
MaxAcc	86.3±2.2	0.99±0.01	87.6±1.8	0.78±0.03	74.7k	138.8k	157k	68.57	5.328	8F-C32-FC-FC
[172]	61	-	73	0.41	-	33.5k	-	53.09	4.125	-

Result of adaptive inference.

The CNN architecture in the adaptive system (refer to Section 4.1.4) was selected from the Pareto-optimal solution in terms of balanced accuracy, specifically, **MaxAcc** models in the previous section, with improvements by adding BN layers. Then this CNN model was trained from scratch under the same training protocol.

Classification quality was evaluated using standard accuracy and balanced accuracy (Bal. Acc.), and F1-Score, which are more meaningful for highly class-imbalanced datasets. The results involving CNN were reported as mean ± standard deviation over the 5 training runs. All models were deployed on Quentin [221], a 22nm, 32-bit single-core RISC-V platform with 520kB of memory. The latency and energy consumption for the inference was estimated using the GVSoC virtual platform [222], with a clock frequency of 205.1 MHz and a supply voltage of 0.54V, following the same principle in Equation 4.2 and 4.1. All deployed models were written in C; for CNNs, the optimized kernel library presented in [223] was used, in its single-core version.

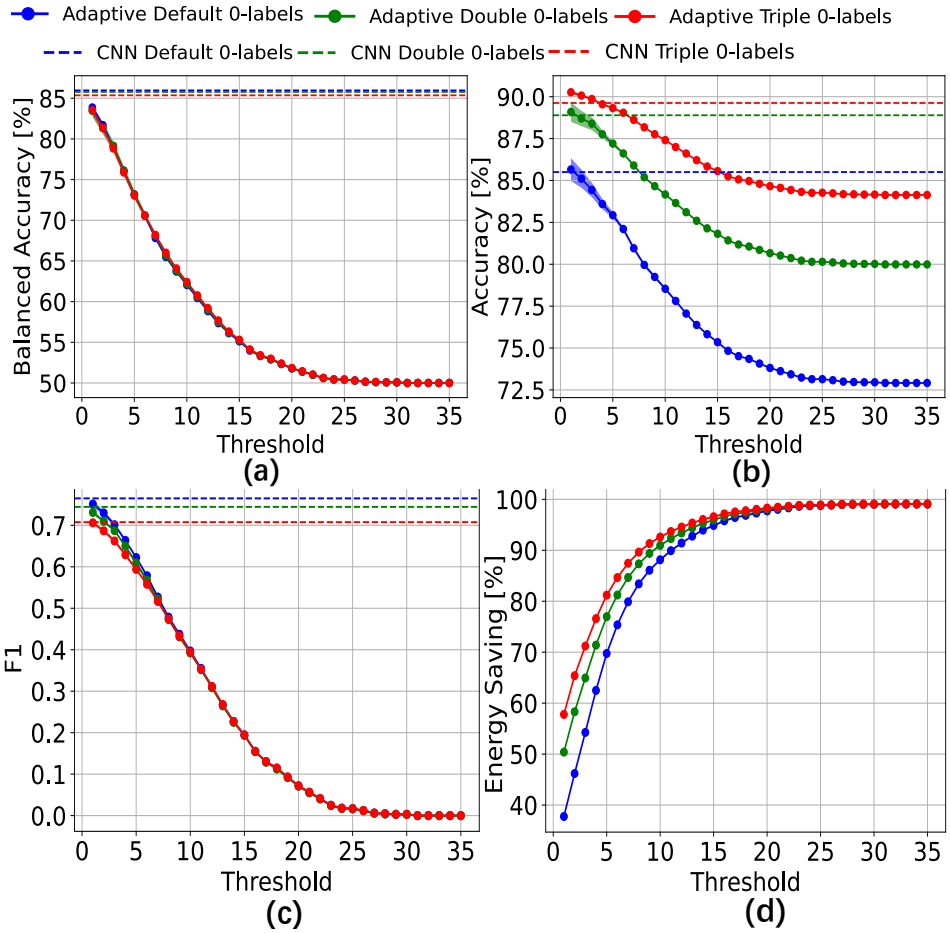


Fig. 4.9 Evaluation results of the adaptive system in terms of threshold vs Bal. Acc. (a), vs Acc. (b), vs F1 (c) and vs energy saving (d).

Fig. 4.9 shows the classification performance achieved by the proposed adaptive system. In particular, the solid lines and dots in 4.9a-c depict the results in terms of balanced accuracy, accuracy, and F1-score, respectively. Different points are obtained

by varying the configurable threshold of the wake-up trigger, representing the mean score obtained over the 5 training runs. The standard deviation is represented as a colored band (although it may be hardly visible in most curves). Dashed lines indicate the results obtained by a static CNN, which is identical to the second "stage" of the adaptive method. Additionally, Fig. 4.9d illustrates the average energy savings in percentage achieved by the adaptive system compared to the static CNN. The three colors correspond to three different versions of the LINAIGE [13] test set. Specifically, based on the observation in practical application scenarios, long sequences of frames with no people are likely to be observed, three variants of the test data are considered: i) Default, representing the original LINAIGE test set, ii) Double, a version of the test set where all frames corresponding to a ground truth label of 0 people are duplicated, and iii) Triple, similar to the previous one but with triplicated 0-labels.

A key observation from the figure is that the overall performance of the adaptive system decreases as the threshold values increase, with the best results achieved when the threshold equals 1. This outcome is anticipated, as a higher threshold leads to fewer invocations of the second-stage CNN. Beyond threshold values exceeding 25, the adaptive system ceases to function effectively and behaves as a constant classifier, consistently predicting 0. Conversely, lower thresholds are more cautious, reducing the number of false negatives but potentially increasing the number of unnecessary CNN invocations, even for frames devoid of people (i.e., false positives). However, it is noteworthy that the CNN can still rectify these false positives by predicting the "No Violation" class, thus enhancing the overall accuracy, balanced accuracy, and F1-score. Correspondingly, the trend in energy savings displays an inverse pattern, increasing as the wake-up threshold increases (refer to Fig. 4.9d).

The most interesting results are obtained with a threshold of 1: the adaptive system achieves a balanced accuracy of around 83%, which is only 2% lower than that of a static CNN solution. Moreover, the F1-score and standard accuracy remain substantially identical to the static case. Simultaneously, by invoking the CNN only when the presence is detected in the frame, the average energy consumption is reduced from 37% to 57%, depending on the number of frames with 0-labels. As expected, savings increase when there are more frames without people, demonstrating the potential effectiveness of the proposed lightweight wake-up trigger in a real-world scenario.

Table 4.4 Detailed evaluation and deployment results of models targeting social distancing on Quentin @ 205.1 MHz

Model	Bal. Acc. [%]	Acc. [%]	F1	Model Mem. [B]	Tot. Mem. [B]	Energy [µJ]	Latency [µs]
8-bit CNN (MaxAcc)	85.96±0.47	85.50±1.05	0.76±0.01	37.97k	56.07k	1.20	316
Wake-up Trigger	71.30	65.28	0.57	66	12.25k	0.01	2.96
[172]	61	73	0.41	1.71k	23.31k	2.53	665
Adaptive	83.86±0.29	85.65±0.68	0.75±0.01	38.04k	56.24k	0.75	198

Table 4.4 compares the proposed adaptive system (with a threshold of 1) with the two individual classifiers (wake-up trigger and CNN), as well as to the state-of-the-art deterministic algorithm proposed in [172]. In addition to the classification scores, the table also presents the data memory occupied by each model once deployed on the target MCU, the total memory including the code size, and the average energy consumption and latency for a classification. The "Wake-up Trigger" row pertains to the threshold-based classifier used "stand-alone", directly for social distance monitoring. In this case, the best performance is achieved with a threshold of 2. All results refer to the default *Default* LINAIGE test set.

As demonstrated, the method exhibits significant performance superiority over both the standalone wake-up trigger and the approach proposed in [172] in all classification metrics (up to a + 20% improvement in balanced accuracy), while achieving results comparable to static CNN, similar to those outlined in [13]. Currently, there is a notable reduction in average energy consumption and latency compared to CNN (37. 5%), with negligible total memory overhead (<0.3%). This underscores the effectiveness of deploying the proposed adaptive system on MCU-class platforms.

Experimental Results of People Counting

In experiments of people counting task, 6 different DLs were trained and quantized. Training and quantized models are subsequently converted to TensorFlow Lite (TFLite) format [202]. Ultimately, the TFLite files are handled through the X-CUBE-AI toolchain 7.2.0 [218] to produce customized C language implementations optimized for the specific deployment target, specifically the STM32L4A6ZG MCU from STMicroelectronics, known for its 32-bit Cortex-M4 core [174]. The latency and energy usage outcomes are derived from the MCU functioning at 80MHz, powered by a 1.8V supply voltage.

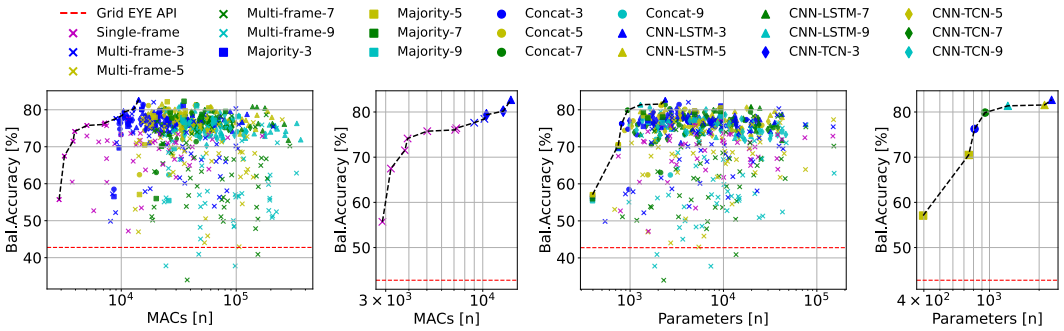
To assess the performance of the models, the primary metric under consideration is the Balanced Accuracy (Bal. Acc.), which denotes the average recall across all classes. This metric is favored over standard accuracy (Acc.), which measures the proportion of correct predictions, particularly for datasets with class imbalances. Furthermore, the F1-Score (F1) is computed, representing the harmonic mean of precision and recall, with the weighted average calculated for scenarios involving multiple classes. Additionally, the Mean Absolute Error (MAE) and Mean Squared Error (MSE) between the actual and predicted counts of individuals are provided. Although the task is classification-oriented, MAE and MSE are included, as they offer insights into the magnitude of errors. For example, in a scenario where the true count of individuals is 3, a model predicting 2 incurs a "less" error compared to one predicting 1. All metrics are displayed as the mean ± standard deviation across the 4 cross-validation folds, with each fold's impact weighted by the number of its test samples relative to the total test samples.

In order to assess the hardware-agnostic computational complexity of each model, the number of parameters is employed as an indicator of model magnitude, while the count of MAC operations, which are the primary operations in DL inference, is utilized as an assessment of energy consumption and processing time. A set of Pareto-optimal models, considering Bal. Acc. versus parameters and MACs, is implemented on the target MCU.

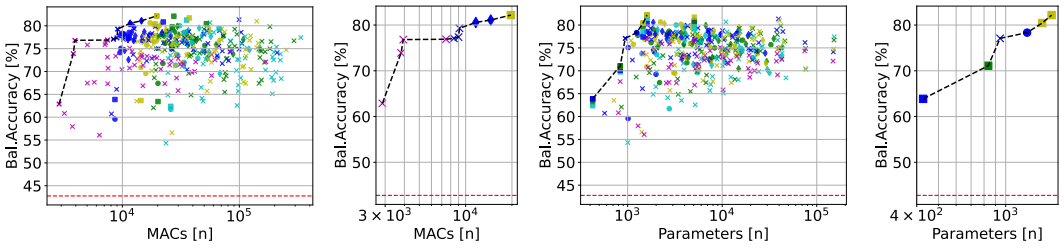
The deployed models are assessed in terms of their overall memory usage, inference cycles, energy consumption, and latency on real hardware, specifically, the STM32L4A6ZG MCU. In terms of memory usage, both model size and total Flash usage are calculated. The model size is derived from the X-CUBE-AI toolchain when generating C code for DL models, while the use of Flash is determined using STM32CubeIDE [220] during deployment.

The CPU cycles per inference, which determine the total latency, are also measured using STM32CubeIDE, and used to calculate the energy consumption based on the average active power of the MCU from the datasheet. The STM32L4A6ZG MCU is considered to operate at an 80MHz clock frequency, with a 1.8V supply voltage.

The latency and energy consumption predictions for each architecture are determined by executing the models and baselines 1000 times, as Equation 4.1 and 4.2,



(a) Floating point architecture exploration results



(b) 8-bit quantized architecture exploration results

Fig. 4.10 Evaluation Results in terms of Bal.Acc. vs. parameters number and operations number for considered DL models in people counting. The left figure shows all models, while the right shows the isolated Pareto curves.

and particularly, are presented as the mean ± standard deviation, as summarized in Table 4.6.

The main reference point for comparison is [172], which is the only publicly accessible solutions for people counting that utilizes a ceiling-mounted IR array with a sensor of identical resolution in this work. The C code of [172] was compiled and run on the target MCU with the same compilation settings as the proposed models. Following that, [172] was evaluated on the LINAIGE dataset. Additionally, the proposed models were qualitatively compared with [36, 195, 23], although these works target different datasets and hardware platforms.

Fig. 4.10 demonstrates the outcomes of the architecture exploration. The upper and lower graphs show the results before and after 8-bit quantization, respectively. Each data precision and target metric (MACs or parameters) is presented for the complete set of models on the left and a zoomed-in view of the Pareto frontier on the right, highlighted by a black dashed line. The performance in people counting is indicated by the average Bal. Acc. over the CV test folds. Different marker shapes represent various model types, while colors indicate different sliding window sizes

W. It is worth noting that LSTM-based models are not shown in Fig. 4.10b due to the lack of quantization support by TFMOT. The performance of the comparison baseline [172] is illustrated by a horizontal red line. The detailed graphs demonstrate the extensive architectural exploration, covering models across a wide range of MACs (2.9k-364k) and parameters (0.4k-153k). For Pareto-optimal models, the MACs range from 2.9k to 14k for float models and from 2.9k to 20k for quantized models, while parameters vary from 0.4k to 2.4k and 0.4k to 1.6k, respectively. The Bal. Acc. ranges from 55.70% to 82.70% for float architectures and from 62.88% to 82.17% for quantized models.

All Pareto-optimal models demonstrate superior performance compared to the deterministic approach of [172], indicating the advantage of data-driven methods for this task. Additionally, all six considered model families are represented in at least one Pareto front, indicating that focusing solely on a single architectural template would be suboptimal. Single-frame CNNs achieve optimal trade-offs primarily in the lower end of the accuracy range. Meanwhile, models with W = 9 are less frequently found on the frontier, suggesting both the importance of processing a sequence of IR frames for achieving high accuracy and the diminishing returns of excessively long sequences leading to overfitting. Lastly, comparison between Fig. 4.10a and Fig. 4.10b reveals that quantization does not result in significant accuracy drops; instead, it often leads to a Bal. Acc. increase in most models, particularly the ones with lower complexity. This phenomenon is attributed to its well-known regularizing effect [47], which aids in reducing overfitting.

Examining each chart in more detail reveals that every Pareto front is characterized by a unique blend of model types, suggesting that different designs are more effective for optimizing either the model size or the number of operations. Specifically, in the plots showing Bal. Acc. versus MAC, Single-frame and Multi-channel CNNs (denoted by crosses) stand out on the Pareto front for both floating-point and 8-bit models. In contrast, when considering parameters as the cost metric, the front is predominantly occupied by Majority voting (squares) and Concatenated CNNs (circles). This difference arises because, in the case of $W > 1$, Multi-channel CNNs necessitate only additional MACs in the initial Conv layer, while subsequent layers remain unchanged compared to $W = 1$. Conversely, Majority-voting and Concatenated CNNs entail the repeated execution of the entire network or feature extractor on each frame, leading to an almost linear increase in total MACs with W. As a result, despite the advantages of higher W in terms of Bal. Acc., these models

Table 4.5 Analysis of the characteristics of DL families for people counting based on IR sensors.

Model	Best For	Bal. Acc. Target	Max Input Win.
Single-frame	Latency/Energy	Low	1
Multi-frame	Latency/Energy	Mid	3
Majority	Memory	Whole Range	7
Concat	Memory	Mid	7
CNN-LSTM	Memory	High	9
CNN-TCN	Latency/Energy	High	3

incur notably higher computational expenses. Nonetheless, the growth in model size is relatively minimal, nearly negligible for Majority-voting CNNs and limited to the final FC layers for Concatenated CNNs. Consequently, when parameters are taken as the cost metric, these models have the potential to outperform Multi-channel CNNs and appear on the Pareto frontier.

As expected, the most complex models, namely CNN-LSTM and CNN-TCN, are positioned in the high-accuracy section of the Pareto curves. Specifically, the floating-point model that achieves the highest accuracy is a CNN-LSTM, as a Bal. Acc. of 82.7% with around 14k MACs and 2.38k parameters. Meanwhile, two CNN-TCN models emerge close to the top of the Pareto front in terms of MACs for quantized models. Nevertheless, generally speaking, many instances of these model types tend to struggle with overfitting, resulting in less than optimal performance despite their considerable computational expenses in terms of MACs and parameters. This tendency is observable as they predominantly cluster on the right side of the comprehensive charts. In conclusion, it is evident that simpler and more effective approaches for integrating multiple IR frames (such as multi-channel processing, feature concatenation etc.) are more favorable for this relatively uncomplicated task and modest dataset.

Table 4.5 provides an overview of all DL models assessed, emphasizing their characteristics and specifications as indicated by the findings in this work. The table outlines general trends as specific numerical outcomes may differ depending on the sensors or datasets used. The Pareto analysis for each type of model is condensed for both floating-point and quantized versions, encompassing:

i) The effectiveness of models for reducing memory usage or reducing latency/energy consumption depends on whether they are frequently located on the

parameters or MACs Pareto frontier respectively, as indicated in the "Best For" column.

ii) The preferable accuracy range for the model (Bal. Acc. Target column) implicitly determines the associated resource range (memory, latency, or energy).

iii) The maximum length of the IR frames window that produces Pareto-optimal outcomes for the particular model family (Max Input Win.).

Approximately, the Low, Mid, and High Bal. Acc. ranges align with < 75%, 75% - 80%, and > 80%, respectively. This table provides a convenient guide for system designers. For instance, it suggests that single-frame CNNs are appropriate for achieving rapid and energy-efficient inference when extremely high accuracy is not essential. Likewise, it proposes that Majority voting is efficient in minimizing memory usage across all accuracy levels, and CNN-LSTMs are the only models for which an increased window length > 7 proves advantageous for enhancing accuracy, among other findings.

To evaluate the deployment performance, five floating-point and five quantized architectures were selected from the Pareto frontier derived in the previous evaluation for deployment on the target MCU. Specifically, the following models were deployed:

i) The model that achieves the highest balanced accuracy (Top);

ii) The most compact model (Size-L) and the one with the fewest MACs required (MAC-L);

iii) The most compact models with the fewest MACs that maintain a balanced accuracy decrease of less than 5% compared to the Top (Size-H/MAC-H).

Table 4.6 shows the detailed deployment outcomes for these configurations on the STM32L4A6ZG MCU. Quantized models are identified with a "-Q" suffix. In addition to the accuracy metrics for people counting, the table also includes information on memory usage, energy consumption, and inference speed for each model. Memory usage encompasses both the model size and the total occupied Flash, which also includes the code size. The same metrics are provided for [172] for comparison purposes. The architecture of each deployed neural network is summarized in the rightmost column, with symbols in square brackets denoting the model type, using the same marker shape as in Fig. 4.10 (for example, ▲ corresponds to a CNN-LSTM). The numbers in brackets indicate the value of W. The layer sequence in the model is represented as follows: "Cn" denotes a convolutional layer

with n output channels, with implicit Batch Normalization (BN) and Rectified Linear Unit (ReLU), "FC" denotes a fully-connected layer, "P" denotes a max-pooling layer, "Lm" denotes an LSTM cell with hidden size m, and "Cat" denotes a concatenation operation.

Table 4.6 Deployment and evaluation results of considered architectures for people counting on STM32L4A6ZG @ 80MHz.

Model	Bal. Acc. [%]	Acc. [%]	F1	MSE	MAE	Model Size [kB]	Tot. Mem. [kB]	Energy [µJ]	Latency [ms]	Architecture
Top	82.70 ±6.15	84.34 ±7.84	0.85 ±0.07	0.18 ±0.09	0.16 ±0.08	9.28	82.38	80.26 ±0.10	5.16 ±0.0064	[▲3] C8-P-C8-L16-FC
Size-H	76.25 ±5.54	78.13 ±9.08	0.79 ±0.08	0.24 ±0.09	0.23 ±0.09	2.97	68.73	54.96 ±0.01	3.53 ±0.0006	[•3] C8-P-C8-Cat-FC
MAC-H	77.62 ±5.98	78.04 ±8.18	0.80 ±0.07	0.27 ±0.11	0.24 ±0.09	5.7	42.95	29.25 ±0.01	1.88 ±0.0003	[×3] C8-P-C16-FC
Size-L	57.08 ±11.37	51.10 ±22.49	0.52 ±0.23	0.78 ±0.70	0.58 ±0.36	1.45	37.88	85.75 ±0.06	5.51 ±0.0036	[■5] C8-P-FC
MAC-L	55.70 ±11.86	50.35 ±21.31	0.51 ±0.21	0.81 ±0.68	0.59 ±0.35	1.45	37.59	17.18 ±0.01	1.10 ±0.0007	[×1] C8-P-FC
Top-Q	82.17 ±6.42	86.06 ±5.59	0.86 ±0.05	0.15 ±0.05	0.14 ±0.05	1.71	78.01	120.43 ±0.02	7.74 ±0.0010	[■5] C8-P-C8-FC-FC
Size-H-Q /MAC-H-Q	77.08 ±6.05	79.48 ±6.53	0.81 ±0.06	0.24 ±0.07	0.22 ±0.07	0.9	76.32	27.70 ±0.02	1.78 ±0.0010	[×3] C8-P-C8-FC
Size-L-Q	63.87 ±10.76	70.83 ±13.79	0.70 ±0.14	0.33 ±0.12	0.30 ±0.13	0.41	71.56	61.90 ±0.02	3.98 ±0.0010	[■3] C8-P-FC
MAC-L-Q	62.88 ±7.52	68.97 ±14.03	0.69 ±0.14	0.36 ±0.13	0.33 ±0.14	0.41	71.39	20.45 ±0.01	1.32 ±0.0007	[×1] C8-P-FC
[172]	42.77 ±14.50	57.54 ±11.50	0.56 ±0.12	0.61 ±0.21	0.49 ±0.14	-	20.07	60.34 ±0.005	3.88 ±0.0003	-

As shown, all quantized models, in addition to the majority of floating-point models (except Size-L and MAC-L), exhibit a notable superiority over the approach presented in [172] in all accuracy measures evaluated. In terms of balanced accuracy, the models considered outperform [172] by 20.1% to 39.4% and 12.9% to 39.9% for integer and floating-point data representations, respectively. Furthermore, both MAC-H and MAC-L implementations demonstrate faster inference and energy efficiency (ranging from 2.06x to 3.51x) compared to [172], while still maintaining significant performance benefits. For example, MAC-H-Q is 2.18x faster and more energy efficient than [172], while also achieving a +34.3% balanced accuracy, +21.9% accuracy, a 1.44x higher F1 Score, and 2.54x/2.22x lower MSE/MAE.

The selected architectures are characterized by remarkably small model sizes, with the tiniest one taking up only 0.408 KB. Despite this, the overall memory usage exceeds that of [172], mainly because of the significant code size of X-CUBE-AI libraries, which accounts for up to 97% of the Flash usage. As shown in the table, the memory consumption is influenced by the types of layers in the model (for instance, the "Top" floating-point model requires more memory partly due to the additional

incorporation of LSTM-related code). Moreover, quantized models generally exhibit a larger code size compared to floating-point models on average, possibly because of the more intricate logic involved in managing scaling factors and re-quantization operations. Nonetheless, all models under consideration necessitate 37.6-82.4 kB of Flash, representing 3.7%-8% of the 1 MB available on the MCU used in the experiments, demonstrating their suitability for memory-constrained IoT nodes.

All models have a latency of less than 10ms, meeting the real-time requirement due to the 10FPS acquisition rate of the target dataset. Additionally, with a compact 1400mAh@3.7V battery, and simplifying by disregarding non-idealities and conversion losses, a model like MAC-H-Q could perform continuous inferences at that frame rate for over 2 years without requiring a recharge.

Table 4.7 compares the presented work with the most relevant ML/DL approaches for people counting with IR array sensors. The comparison is qualitative due to differences in datasets and hardware deployed in previous work. Alongside the input frame size (Input), the table includes accuracy (Acc.), F1, and MSE, as well as model size (Size) and the number of operations (OPs) per inference, serving as hardware-independent complexity metrics. For DL solutions, OPs are approximated with the number of MACs. When Size or OPs are not provided, they are calculated based on the layers' geometries. For [23], Size and OPs could not be estimated, as the authors did not report the number and maximum depth of the decision trees comprising their best-performing RF.

Two outcomes in this work are presented: the first respects to the "Top-Q" model detailed in Table 4.6, which was identified through the per-session CV approach outlined. Furthermore, in contrast to [195, 196, 23] who employ a random sampling technique to divide training and test sets, the optimal quantized results achieved with this splitting method are also reported. Specifically, an architecture exploration is carried out using an 80%/20% random split for training and testing, and the average results on the test set across 4 iterations are reported. It is crucial to highlight that the Top-Q* model features a distinct architecture from Top-Q, specifically a quantized CNN-TCN model structured as follows: [♦9] C8-P-C32-TCN32-FC-FC, where TCNo denotes a TCN (1D Conv) layer with o output channels.

The primary point of comparison for cutting-edge DL methods is [36], which employs a per-session split and a similar input resolution. In their study, a comparable F1 Score is attained with the Top-Q. However, the proposed model in this study

Table 4.7 Comparison with the state-of-the-art ML/DL solutions for people counting.

Result	Input	Acc. [%]	F1	MSE	Size [kB]	OPs
[36]	8x6	n.a.	0.88	n.a.	450	$34{\cdot}10^6$
[195]*,†	10x10	95.9	n.a.	n.a.	13.6	$117{\cdot}10^3$
[196]*	32x24	**98.9**	n.a.	**0.01**	400	$400{\cdot}10^3$
[23]*	8x8	94.6	0.95	n.a.	n.a.	n.a.
[197]§	32x24	94.1	n.a.	0.057	520.8	$25{\cdot}10^6$
Top-Q	8x8	86.1	0.86	0.15	**1.71**	$\mathbf{20{\cdot}10^3}$
Top-Q*	8x8	95.3	**0.95**	0.05	20.1	$80{\cdot}10^3$

(*) Train/test split based on random sampling, instead of per-session.
(§) Train/validation/test split based on sequences splitting in different locations, instead of per-session.
(†) This approach processes a single 10x10 sliding window; also perform people *localization*.

achieves a reduction of 263 times in size and 1700 times fewer OPs, thanks to the considerably smaller model size, and the elimination of the necessity for an additional super-resolution network. [197] also utilizes a time-based data split, although still simpler than the considered strategy in this work: frames gathered in the same location at different times are allocated to different data buckets. Their study achieves a higher Acc. than the Top-Q (94.1% vs. 86.1%), mainly due to the 12x higher resolution input. Additionally, their model necessitates approximately 130k floating-point parameters, resulting in a model size of 521kB, which is 304x larger than that of Top-Q. Similarly, the number of OPs is in the order of millions, over 1000x larger than Top-Q.

Given the availability of the dataset from [197], two additional experiments were carried out using it. Initially, the images were down-sampled to 8x8 resolution, and any samples containing more than 3 people were excluded to ensure a fair comparison with LINAIGE. Subsequently, the "Top" model from Table 4.6 was trained exclusively using the data from [197], following the same training protocol as in the original work. An accuracy of 72.6% was achieved, which is lower than the performance of their model, yet acceptable considering the lower resolution of the inputs in this study and the significant over 1000x reduction in complexity.

Furthermore, the dataset referenced in [197] consists of only around 9k samples with fewer than 3 individuals, in contrast to the > 20k samples in LINAIGE. Consequently, an effort was made to incorporate the down-sampled data from [197] to

augment the LINAIGE training dataset in each CV fold. In this context, the "Top" model demonstrated improvements in all the average classification metrics (Acc. + 4. 8%, F1 + 0.04, MSE -0.06, MAE -0.05), except for Bal. Acc. (-1.2%), compared to training exclusively in LINAIGE. This suggests that leveraging a larger dataset could potentially further improve the outcomes of the proposed efficient DL models, particularly the more complex architectures.

When examining a random data split, Top-Q* achieves slightly lower accuracy and higher MSE in comparison to the work of [196], but operates with a smaller resolution input and requires a model that is 233x smaller with 20x fewer inference operations. It also obtained similar accuracy and F1 score when contrasted with the RF-based approach presented by [23]. In addition, the model introduced by [195] utilizes a smaller model than Top-Q* to achieve slightly higher accuracy on a 10x10 input. Nevertheless, the inputs processed by [195] consist of patches extracted from a significantly higher resolution input (80x60), which is then upscaled to 120x90 and 160x120. These three variations of the image are subsequently processed by the CNN on 10x10 sliding-window patches. Consequently, the total number of inference operations for this approach is huge (approximately 450×10^6 based on calculations), resulting in very high latencies and energy consumption. The authors report a total latency of 63s and an energy consumption of 2.2J, which are orders of magnitude higher than those achieved by the proposed models. It is important to note that [195] not only focuses on counting people in the frame but also on localizing their heads, a task made feasible by their higher-resolution input. The reported accuracy of 95.9% in the table with respect to head detection on a single 10x10 patch, while the final counting accuracy is only 53.7%.

In conclusion, the comparisons show that the suggested models achieve similar counting accuracy with significantly reduced complexity on average when compared to the latest solutions. This is especially crucial for implementation in IoT smart sensors, which operate under strict memory constraints and energy limitations, often relying on batteries, and are required to function for extended periods without the need for recharging. The small and effective Dl models investigated in this study could facilitate innovative people counting solutions that prioritize privacy and are widespread in settings where access to the power grid is limited, a scenario that would render most energy-intensive state-of-the-art solutions impractical.

4.1.7 Conclusion and Discussion

Focusing on people counting and social distancing, the two most popular applications involving IR smart sensors, efficient smart models based on DLs were accessed and deployed. In particular, compact and energy-efficient CNNs were evaluated both in terms of hardware-independent performance including accuracy and computational complexity, the results indicate that such DL-based solutions can achieve state-of-the-art accurate social distance monitoring directly on MCU-based IR sensor nodes.

Furthermore, dynamic optimization can be applied to further improve energy efficiency with negligible overheads, by an energy-efficient adaptive inference comprising a cascade of a lightweight wake-up trigger and an 8-bit quantized CNN activated solely for challenging frames. Implementing this adaptive system on an MCU, which is a common core in smart sensors, has shown that when handling the output of an 8x8 low-resolution IR sensing frame, energy consumption is reduced by 37-57% compared to a static CNN-based approach, with a minimal accuracy decrease of less than 2% (83% balanced accuracy).

Finally, a systematic study on efficient DL architectures for person counting based on ultra-low-resolution IR smart sensors has been conducted, yielding a wide range of trade-offs between classification scores, memory occupation, latency, and energy consumption. Different types of DL models have been shown to be preferable for different objectives. The resulting Pareto-optimal models achieve counting accuracy scores significantly higher than those of a publicly available deterministic solution [172] (up to 82.70% balanced accuracy vs 42.77%), and comparable with a state-of-the-art DL approach [36] (0.86 vs 0.88 F1-score), while reducing the latency and energy requirements by up to more than 2x compared to the former, e.g., 1.78ms/27.70µJ vs 3.88ms/60.34µJ per inference at approximately +34.3% balanced accuracy for the proposed method.

In conclusion, the proposed efficient DL architectures and adaptive inference strategies achieve state-of-the-art performance on multiple metrics including accuracy, balanced accuracy, F1-score, etc., targeting people counting and social distancing applications, and can be directly implemented as smart models in MCU-based resource-limited and energy-critical IR smart sensor nodes, enabling continuous

real-time inference (< 10ms latency), with years of autonomous operation, while requiring less than 100kB of memory.

Moreover, there exist several interesting points that can be further discussed referring to the proposed DL-based solutions and their performance in practical scenarios.

Approach Generalization. The first interesting point is the model generalization in the practical applications in the real world. In general, model generalization refers to the ability of models to achieve expected performance on unseen data which never appears in training sets or validation sets during development. The more generalized models mean the more robust accurate predictions in a wide range of real-world scenarios. In this application, there are several factors are under consideration with respect to ensuring model generalization, and more details are discussed as follows:

- **Dataset splitting.** Datasets used in the training of models have a direct and significant impact on model generalization. As discussed in the experiments, in real-world scenarios, the environmental factors including places and time are impossible as same as the data used in model developments. Therefore, targeting people counting and social distancing tasks, the collection of LINAIGE datasets used in the experiments ensuring the variability of data in terms of locations and time are important to make as close as possible to real application scenarios.

  On the other hand, the data splitting is another significant aspect to be considered to reflect the model generalization in real cases. As explained in Section 4.1.6, the data splitting in the social distancing and people counting experiments used *per-session* splitting, which ensures the data sets used in validation are from either different places or times as training data, simulating conditions in the final applications. In contrast, a random per-frame splitting cannot ensure the testing data excludes frames from the same environment and time windows. This may lead to information leakage between the training and validation sets, resulting in deceptively high evaluation results.

- **Cross Validation.** Cross-validation strategy is a common approach to evaluate the performance of models on different data. By splitting

datasets into *N* folds, each fold is used as a test set in turn, while the remaining folds are used as train sets. A model with great generalization ability should achieve stable performance scores in all test folds. As explained in Section 4.1.6, targeting the people counting task, *leave-one-session-out* cross-validation approach was applied based on per-session data splitting. This approach further guaranteed the generalization performance of proposed models, since the evaluation results do not depend on a specific test session.

Moreover, the results of performance metrics were obtained by weighted average over cross-validation folds, where the weight is calculated by the number of test samples over the total samples. All the results were shown in mean $\pm$ standard deviation format, reporting the variability of performance of proposed models in different environments. In this way, the smaller of $\pm$ standard deviation range, the robust of performance of models facing different environmental factors such as sites and time.

In Table 4.6, where detailed evaluations were performed on Pareto-optimal models for people counting, models with an average balanced accuracy greater than 76% exhibited a standard deviation of no more than ±6.5%, indicating relatively better generalization abilities. It is worth noting that these models usually have larger model sizes. In contrast, models with a lower balanced accuracy (less than 63%) showed a larger standard deviation (approximately 11%), meaning significant performance variability across different cross-validation folds. In fact, it is expected since the worse accuracy performance of these models with smaller architecture is exactly caused by their relatively poor model generalization abilities. These findings demonstrate the effectiveness of cross-validation and metric reporting format to evaluate model generalization ability. In addition, targeting this specific dataset and task, it may suggest that compared to simple architectures, models with more complex architecture exhibit more robust generalization ability to deal with different unseen data.

- **Cross-dataset Validation.** In this application, the LINAIGE dataset was collected since there were not any existing public datasets containing IR frames that fit the needs of target applications, both in resolution level (low to 8x8 pixels) and the sensor placement(a *single* ceiling-mounted

sensor). However, to further validate the generalization of proposed models on different datasets, a public dataset from [197] containing relatively high-resolution IR frames was down-sampled to the same level resolution as LINAIGE, and used as another validation dataset. As shown in 4.1.6, the most accurate DL architecture achieves 72.6% balanced accuracy on 1000x shrunken frames from this dataset, by only 9k samples available to fit the goal of the people counting work. On the other hand, a data augment was performed by using the dataset from [197] during the training of proposed architectures, and the cross-validation results show that all performance metrics report a trend of improvement. It allows the users to use the proper amount of additional data to make the proposed architectures reach better performance in practical scenarios.

- **Sensor types.** In the practical world, there exist different types of commercial sensors on the market, and their resolutions differ from a wide range from 80x60 [195] to 8x6 [36]. As discussed in Section 4.1.3, there exist several solutions based on relative high-resolution IR arrays [36, 195–197]. It is interesting to consider the impact of sensor types on the generalization of proposed solutions. In [36], the author proposed a CNN model based on high-resolution IR (32x24) for counting people and provided additional solutions based on low-resolution sensors plus an upscaling technique. The experimental results show that the same CNN achieves all performance metrics (true Positive Rate, precision, recall, and F1-score) over 97% on 32x24 frames but only around 72% directly on 8x6 IR images. This fact demonstrates the impact of the resolution level of IR data on model generalization, and compared to high-resolution frames, IR data with lower resolution make the target tasks more challenging. Therefore, the proposed models may achieve better accuracy performance on the IR images with higher resolution, since the hot profile of objects is easier to be detected. However, it is worth noting that higher-resolution IR frames cannot guarantee the same level of privacy-preserving level, which is exactly the main concern in this work.

Challenge and limitations. Although there exist several considerations concerning the generalizability of the solutions, there still exist two limitations beyond

the specific experimental setups. In the considered social distancing and people counting experiments, the application scenarios are assumed in indoor environments represented by workplaces, where the IR arrays can mounted on the ceiling, and the room temperature is stable. Therefore, indoor scenarios with unusual room temperatures (e.g., boiler rooms) may lead to different performances of the proposed models. In this situation, environments with higher temperatures may complicate object recognition. Another kind of case is outdoor environments, where the IR sensors may affected by various heat sources including vehicles, sun etc. [172], making it challenging to achieve the same accuracy to detect the targets for the proposed models. However, the conclusions drawn from experimental outcomes, such as the superior performance of advanced DL architectures (e.g., LSTMs) and the utilization of sliding window sampling in such contexts, remain practically relevant.

On the other hand, the placement of sensors may have impacts on the generalizability of the proposed solutions as well. In particular, the proposed solutions are based on ceiling-mounted IR arrays. It is from the two considerations. First, the height of the ceiling in indoor environments is relatively stable, making the distance between the sensors and the head top of objects less variable compared to the distance to their bodies. Second, compared to ceiling-mounted sensors, wall-mounted sensors face challenging situations such as overlapping heat profiles of people who stand in front and behind, complicating the target tasks. Therefore, in this application, the sensors are considered as ceiling-mounted with heights from 2.4m to 2.7m, which fits the typical ceiling height in homes and offices [224]. It is important to note that, for social distancing, the problem formulation is based on the social distance recommendation and the maximum distance between objects in the detection area, and in the specific effective detection area, the task is modeled as a binary classification to predict whether more than 2 objects in one frame, as detailed in Section 4.1.1. In practical cases, a larger space can be covered by combining multiple sensors mounted on the ceiling.

Overheads. In particular, while the dynamic inference system can further reduce energy consumption thanks to the adaptive strategy based on inputs, the staged architecture containing more than one model also brings overheads in terms of memory occupation. These memory overheads mainly depend on the number of models equipped in the system. In this application, the choice of a

lightweight trigger stage is precisely out of consideration for negligible overheads. As shown in Table 4.4, the memory overheads of the adaptive system are less than 0.3% compared to the standalone base model. Interestingly, latency may be considered to bring overheads intuitively, since the "difficult" input will pass through more than one stage. However, as reported in Table 4.4, the average latency of the adaptive system for predicting thousands of inputs is less than one of the standalone base models. This is because, in the practical scenarios, the easy inputs constitute the majority of collected sensing data. This demonstrated the fact that the most of time, only the wake-up trigger is invoked, which is exactly the principle of this system targeting social distancing.

4.2 Application 2: Human Activity Recognition with Acceleration Sensors

4.2.1 Motivation and Context

Human Activity Recognition (HAR) using Inertial Measurement Units (IMUs) is a prevalent application that benefits from edge computing, often based on devices such as acceleration smart sensors. In the literature, current HAR solutions with smart sensors rely mainly on classic ML-based approaches, particularly simple and hardware-friendly tree-based algorithms such as DTs and RFs [225–227].

Thanks to their remarkable classification performance, lightweight architecture, and great interpretability, tree-based models represented by RFs have achieved success and increasing edge-device-based solutions considered a small number of operations and memory occupation. In particular, compared to DTs, RFs alleviate overfitting issues for simple smart tasks in smart sensors represented by HAR, which often results in comparative accuracy close to DL-based solutions.

Although less costly than DL-based approaches, the implementation of RFs also has a significant impact in terms of inference time and energy consumption of such smart models running on a smart sensor, further determining the battery life of MCU-based sensors. Therefore, compact implementations tailored for MCU-based

HAR inference are crucial for edge devices such as smart acceleration sensors. The work described in this section has been previously published in [3].

4.2.2 The State-of-the-art Human Activity Recognition Solutions

In the past few years, there has been a trend in research focusing on ML-based HAR solutions targeting edge devices such as acceleration smart sensors, obtaining impressive success across various domains including health monitoring and fall detection.

One of the early proposed approaches to Human Activity Recognition (HAR) tailored for edge devices was introduced in [228]. The authors implemented an SVM classifier using fixed-point arithmetic to reduce computational costs while minimizing accuracy loss compared to a floating-point version. [229] evaluated four classic ML algorithms, including KNN, SVM, ANN, and RF, for HAR tasks based on acceleration data gathered from smartphones, using raw data and magnitude features, respectively. The experimental results show that RFs outperformed other classifiers in this task during a 5-fold cross-validation and a leave-one-subject-out evaluation, particularly in tasks that distinguish among various types of Activities of Daily Living (ADL).

Similarly, [230] compared four supervised classifiers (KNN, SVM, Gaussian Mixture Models (GMM) and RF) and several unsupervised models. They found that KNN yielded the best results among supervised solutions. Furthermore, [231] applied a low pass filter preprocessing method followed by the evaluation of five classifiers and studied the combination of multiple classifiers. Finally, [232] proposed an innovative HAR system leveraging DL, utilizing a CNN to recognize different activities.

4.2.3 Limitations in the Existing Solutions

However, the existing solutions discussed in the previous section mainly focus on high classification performance, proposing models with larger memory footprints and computational complexities. More recently, DL approaches have emerged, achieving state-of-the-art accuracy in various HAR datasets [233, 234]. Nevertheless, these DL models often contain millions or even billions of parameters, which require

significantly more computational power and memory than what is typically available on low-power acceleration smart sensor nodes.

On the other hand, classic ML-based solutions require relatively less computational complexity, making them still have big advantages in application scenarios involving smart sensors. In fact, lightweight non-NN-based solutions are often adequate for simple tasks such as HAR, producing similar results with NN-based approaches [16]. However, it also implies the fact that the design of classic ML-based solutions, especially feature extraction, will have a significant impact on the final smart models deploying on the smart sensors. For example, the RFs in [229] contain s300 bagged classification trees, while achieving remarkable performance. Considering the inference time and energy consumption of RFs increase linearly with the number of trees [16], *deep* (respect to the number of levels of trees) and *large* (respect to the number of trees), RFs also suffer from the conflict between their large memory occupation and limited resource in edge sensors.

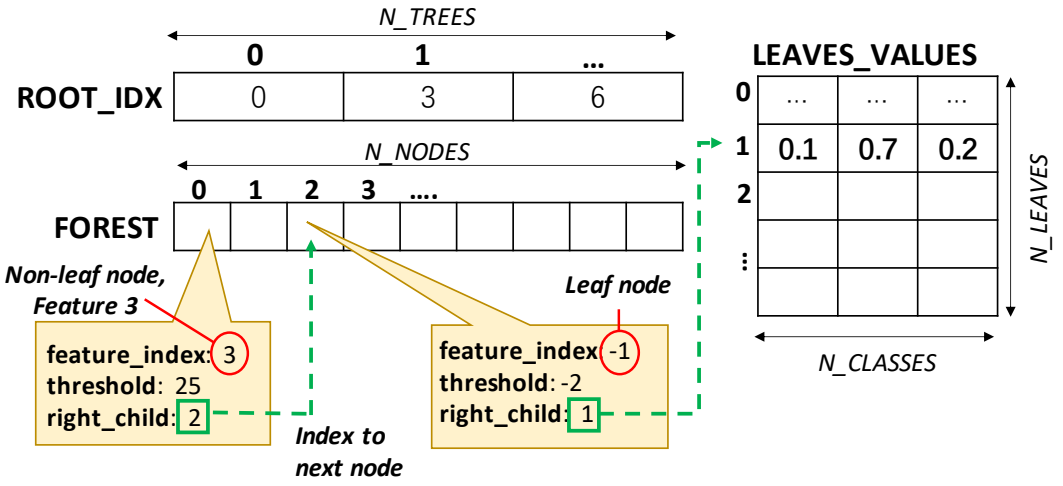
4.2.4 Proposed RF Implementation

Since RFs have become one of the well-known state-of-the-art HW-friendly smart models targeting HAR tasks based on acceleration smart sensors, in this work a compact implementation tailored for this specific task based on a general-purpose hardware RISC-V processor is proposed.

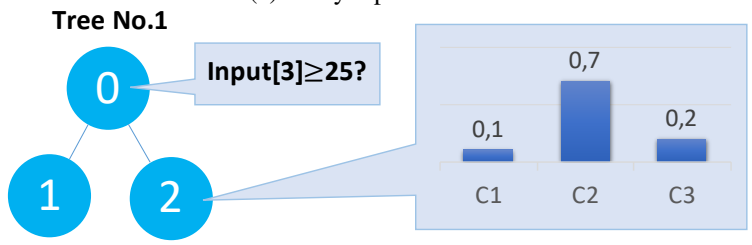
Fig. 4.11a depicted the proposed optimized implementation of RFs. In particular, it is inspired by OpenCV [235], designed for smart sensors with limited memory. Instead of using lists as in OpenCV, the RF is represented by multiple C-arrays to improve memory locality and streamline the control flow. The RF implementation operates on an *INPUT* matrix that contains raw data or extracted features. At the heart of the implementation lies the *FOREST* array, which encapsulates the descriptions of all nodes in the RF. The node elements are implemented in C struct format with three essential fields, in detail:

feasture_index. Indicates the column index in *INPUT* corresponding to the feature for comparison against the node's threshold. Leaf nodes are marked by a value of -1 in this field;

threshold. Specifies the threshold used for comparison in the node;



(a) Array representation



(b) Corresponding (partial) RF

Fig. 4.11 A compact RF implementation.

right_child. Denotes the index of the right child of the node within the *FOREST* array. To conserve memory, only the right child is explicitly stored, while the left child is implicitly assumed to be the subsequent node in the array. For leaf nodes, the right_child index points to an entry in the *LEAVES* array.

The *LEAVES* array is responsible for storing the output predictions for all leaf nodes. Unlike storing a single class label, this implementation stores class probabilities for all classes in the leaves. This approach not only improves accuracy, but also aligns with the scikit-learn implementation [236], which is a widely used ML library in Python. In addition, the *ROOT* array stores the indexes of the roots of all trees in the *FOREST*.

An illustrative example of this implementation, depicted in Fig. 4.11b, highlights the content of selected *FOREST* elements relative to the first tree of the RF. Inferences are initiated by iterating over all elements in *ROOT* and traversing each tree. The predicted probabilities of different trees are accumulated, and a final *argmax* operation determines the class label.

Since many MCUs targeting embedded control applications lack a FPU to support floating-point operations, in this work, the compact implementation of RFs is tailored to fit the meets. Specifically, thresholds and probabilities in RFs are quantized to 8-bit integers. Moreover, the indexes in the various arrays are also quantized to 8 or 16 bits, depending on the maximum RF size fitting in the target hardware platform, to minimize the memory footprint.

4.2.5 Experimental results

The experiments were performed on two distinct HAR datasets, to evaluate the performance of proposed RFs. In particular, one dataset is publicly available, while the other is proprietary. The former dataset, UniMiB-SHAR [229], contains 11711 records collected using sensors equipped inside an Android smartphone, categorized into 17 activity classes. Each record consists of a vector of 151x3 tri-axial accelerometer values. The original study [229] benchmarked this dataset with various classifiers, utilizing raw data or acceleration magnitude features, and in particular, the considered RF was demonstrated to achieve the best performance.

In addition, experiments were performed on an internal proprietary dataset of STMicroelectronics, denoted as "Walk", containing 667 records. Each record is composed of multiple non-overlapping windows, each containing 32 tri-axial accelerometer samples. In this dataset, the HAR focuses only on distinguishing "walking" from other activities.

A state-of-the-art DL-based solution using Binary Neural Networks (BNNs), from [3] is used as a comparison. This BNN model is designed for low-power general-purpose processors with a RISC-V instruction set, offering advantages such as minimal memory footprints and low inference complexity by replacing arithmetic operations with bitwise operations. The detailed implementation is out of the scope of this work, the interested reader can refer to [3]. In the experimental results, both the RFs and the comparison baselines (denoted as BNN) were evaluated with hardware-independent performance metrics in terms of accuracy vs. memory and inference cycles. In particular, the inference clock cycles were estimated using the virtual platform GVSOC [222].

Fig. 4.12 presents a comparison between RF and comparison baselines in the two targeted HAR datasets: (a) UniMiB-SHAR, and (b) Walk. The results are depicted

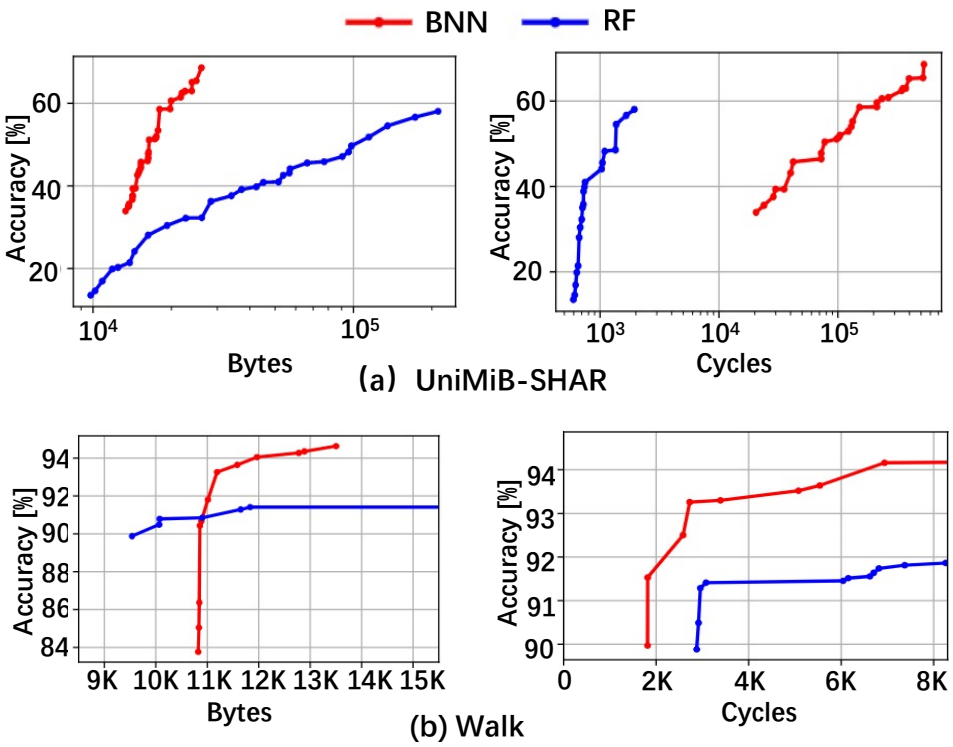


Fig. 4.12 Accuracy versus inference cycles and memory occupation of RFs and [3].

as Pareto frontiers in the accuracy versus total memory (figures on the left side) and accuracy versus cycles planes (right side), encompassing end-to-end inferences, which include both feature extraction and classification processes. In particular, Fig. 4.12 shows RFs with inputs as raw data or extracted features depending on the target datasets. The rules followed from the original datasets, specifically, target UniMiB-SHAR, raw data is used as inputs for RFs, as in [229]; while targeting Walk private datasets, seven features computed on each of the three accelerometer axes within each 32-sample window are utilized, including average, variance, energy, max, min, peak-to-peak and number of zero-crossings, more details can be found corresponding to [3].

Each classifier is represented by different points corresponding to Pareto-optimal sets of hyperparameters, determined via an extensive grid search. For RFs, variations in the number of trees and their maximum depth were explored, while for BNNs, the exploration included the depth of networks(number of layers) and the type and parameters of each layer (e.g., convolutional, fully connected). Note that the two graphs that refer to the UniMiB-SHAR dataset are semi-logarithmic in scale.

On the UniMiB-SHAR dataset, RF models achieve accuracy levels of up to 60% with a minimal number of cycles by employing a small number of trees with high depth, which implies much less energy consumption. More importantly, RFs dominate on the Pareto-optimal frontier in terms of inference cycles, thanks to their wide range of accuracy achieved that only requires minor variations in the number of cycles. On the contrary, the baseline comparison BNNs can achieve approximately 10% higher accuracy than RFs, however, with the sacrifice of a large number of operation cycles. In fact, achieving a similar accuracy of around 60%, BNNs require approximately 75x more cycles compared to RFs.

However, while improving accuracy, the depth increase of the RFs incurs an exponential increase in memory usage, exceeding 200 kB in some cases. On the contrary, BNNs exhibit a higher initial baseline memory requirement (approximately 12 kB) due to their larger code size. However, they significantly improve accuracy with only a slight increase in memory usage.

On the Walk dataset, which is simpler in terms of classification categories, all models exhibit reduced cycle and memory requirements. The primary distinction between RFs and BNNs lies in the level of information extracted from the features of this dataset. Enhanced feature extraction in RF models leads to high accuracy. However, this feature extraction process significantly increases the cycles required, Preventing RF-based solutions from appearing on the Pareto curves in terms of inference cycles.

However, concerning memory occupation, RFs show a high classification performance even with a very small memory footprint (down to approximately 9.5kB), while BNNs show a higher initial baseline memory as in UniMiB-SHAR.

4.2.6 Conclusion

HAR is one of the most common tasks involving ML algorithms in smart sensors. HAR is typically framed as a classification task aimed at identifying human activities based on inertial sensor measurements. Previous studies have shown that state-of-the-art HAR at the edge is often achieved using RFs. However, there was a lack of an efficient RF implementation available for general-purpose hardware, such as RISC-V processors.

As a response, a compact RF implementation of HAR was proposed, targeting low-power general-purpose processors. This implementation represented the RF using multiple C arrays instead of lists to improve memory locality and simplify the control flow. The main data structure representing the model was an array containing the description of all nodes in the RF, with its elements structured as C structs and optimizations such as implicit left child applied. Moreover, quantization was considered in the implementation to optimize memory usage, with different bit widths applied in arrays.

Experiments were conducted using two RF variants with different inputs (raw data and features) on two datasets, one public and one proprietary while varying the number of trees and maximum depth. The results were analyzed and reported as Pareto curves in accuracy versus cycles/memory. The experimental results show that the proposed compact RFs provide various promising low-power solutions targeting extremely memory-constrained application scenarios.

4.3 Application 3: Gesture Recognition with Surface Electromyography Sensors

4.3.1 Motivation and Context

Gesture recognition has emerged as a widely utilized application leveraging smart sensors, offering significant potential in human-machine interfaces (HMIs), facilitating a range of innovative interactions with robots and computers, particularly beneficial for individuals with physical or hearing impairments [237, 238]. Vision-based gesture recognition stands out as a prevalent approach, analyzing videos of hand gestures captured by optical cameras [239, 240]. However, this method is heavily dependent on visual frame quality, making classification susceptible to environmental variables such as camera placement, obstructions, lighting conditions, and more.

In contrast, gesture recognition based on sEMG smart sensors, which directly capture behavioral intentions through muscle contractions at various arm positions, is gaining traction within the scientific community. Although more invasive compared to vision-based alternatives, requiring users to wear potentially uncomfortable

electrode arrays on their arms [4, 241], sEMG-based solutions offer advantages in wearability, reduced raw input data bandwidth, and resilience to environmental factors. In addition, these solutions are not confined to camera-equipped environments and can facilitate gesture recognition for individuals with amputations.

Classic ML algorithms were widely used solutions for gesture recognition tasks based on sEMG, based on carefully crafted feature extraction [241–244]. However, it is tricky to select the appropriate features for the optimal classification performance considering all factors. Therefore, more recently, DL algorithms have become novel state-of-the-art solutions, thanks to their ability to automatically learn crucial features in training phases [238, 245–247].

Nevertheless, implementing a DL system in sEMG smart sensors presents significant hurdles. Many DL algorithms were initially developed for cloud computing, far beyond the capabilities of memory- and energy-constrained mobile sEMG smart sensors. Typically, these devices are battery-powered with power constraints in the range of tens of megawatts, and have less than 1MB of working memory [248]. Consequently, they cannot accommodate the substantial memory requirements and extensive computational operations required by cloud-based models [245, 247, 249]. To address this issue, researchers have proposed lightweight yet accurate DL models specifically optimized for edge deployment, tailored for such wearable sEMG smart sensors.

Various optimization techniques can be employed to craft such models, including architectural modifications achieved through architectural search and pruning [250], as well as reductions in the bitwidths used to represent weights and intermediate activations through quantization [47]. Through these techniques, massive DL models can be streamlined to suit edge devices while still maintaining superior performance compared to classic ML, which highly relies on manually crafted features.

However, with the recent boom in innovation in deep learning architectures, there tends to be a time delay between the initial design of a new cloud model and its adaptation for edge computing. Therefore, while Transformers, a more recently proposed DL architecture originally developed to address sequence-to-sequence problems [76], achieving state-of-the-art performance in various applications such as NLP and computer vision, the popular implementation, including BERT, GPT-3, ViT [85, 251, 86] suffer from their huge model size that contains millions or billions of parameters. This makes them inapplicable for deployment in smart sensors. In

contrast, while there have been many studies on edge deployment and optimization focusing on CNNs, etc., relatively few edge-oriented transformers have been proposed. Therefore, in this thesis, gesture recognition based on efficient transformers tailored specifically for edge sEMG-based smart sensors was demonstrated that achieve exceptionally high energy efficiency while maintaining state-of-the-art performance. Additionally, dynamic inference techniques [147] further enhance the flexibility and efficiency of the proposed gesture recognition system. The work described in this section has been previously published in [15].

4.3.2 The State-of-art Gesture Recognition Solutions

Recently, many methodologies for hand gesture recognition utilizing sEMG signals have been proposed, serving both academic and commercial objectives. Typically, the analog front end of a sEMG smart sensor acquires the biopotential of objects and then the data preprocessing and feature extraction steps are adopted to process the collected signals. Finally, a smart model is utilized to perform the classification of hand gestures and produce the final prediction. In the literature, classical ML algorithms such as SVMs, RFs, linear discriminant analysis (LDA), or artificial neural networks (ANNs) [241–244, 246], or more contemporary DL models [238, 5, 252–255], are widely used. For example, researchers in [256, 257] achieved over 90% accuracy in hand gesture classification using ANN with five time-domain features. Another study [258] achieved 97.1% classification accuracy to recognize three grasps that use root mean square values as features for an SVM. In particular, [241] achieved remarkable results in three datasets, NinaPro DB1, DB2, and DB3, comprising various gestures, utilizing features extracted in the time and frequency domains.

However, these approaches are typically limited to single-session setups, neglecting the challenge of inter-session accuracy degradation when classifying gestures from sessions that have never been seen before.

Achieving high classification accuracy in sEMG signals is challenging due to various influencing factors such as anatomical differences, posture variations, fatigue, perspiration, changes in the skin-electrode interface, user adaptation, and electrode repositioning over consecutive days of use [4, 242, 5, 252, 259]. These factors introduce significant variability, impeding generalization and hampering the development of robust real-time recognition systems. For example, studies have

shown that classic ML algorithms experience up to a 30% decrease in accuracy when trained on data from a single session and tested on other sessions [4, 260].

To address these challenges, a cutting-edge approach involves employing multi-session training, which aims to enhance recognition robustness against temporal variability. This strategy has become feasible with the availability of multi-session sEMG datasets such as NinaPro DB6 (consisting of 10 sessions and 8 classes). There exist several studies based on these datasets [4, 261, 253, 252, 254, 255]. However, most of these studies did not focus on real-time classification or integration of the proposed algorithms into low-power edge devices, such as smart sensors. The closest work is [252], which demonstrated that their algorithm meets the real-time classification requirement of 15 ms and 9.8 mJ energy consumption, essential for real-world applications involving sEMG smart sensors.

Meanwhile, [5] proposed *Tiny* Transformers named Bioformers tailored for edge devices like sEMG smart sensors. The results showed that Bioformers reached the state-of-the-art inter-session accuracy as in the previous works while significantly reducing energy by 98%. Nevertheless, all of the approaches only consider static optimization such as quantization, missing opportunities to further reduce energy by dynamic inference. Moreover, in real-world smart sensors, smart models with high flexibility benefit from the ability to fit wider application scenarios, providing more practical solutions targeting such gesture recognition applications.

4.3.3 Proposed Method

In this work, an energy-efficient sEMG gesture recognition system designed to operate on a wearable sEMG smart sensor was proposed, utilizing tiny transformer neural networks. The overall approach proposed to develop such systems is illustrated in Fig. 4.13, comprising two primary components.

The initial phase (depicted on the left side of Fig. 4.13) is conducted during the design stage. It involves exploring the parameter space of transformer architectures to identify a Pareto-optimal set of solutions concerning accuracy versus model size and accuracy versus the number of MAC operations. These non-functional metrics serve as proxies for memory occupancy and latency/energy consumption, respectively. The resulting tiny attention-based models from this search phase are termed "bioformers" [5].

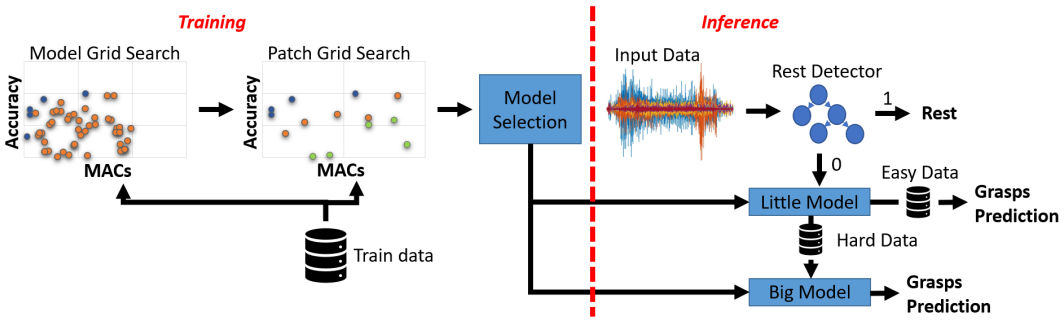


Fig. 4.13 Overview of the efficient gesture recognition solution.

The second part of the methodology (illustrated on the right side of Fig.4.13) addresses runtime operations. A multi-stage dynamic inference mechanism wherein two different bioformers, along with an optional non-deep classifier, are combined to reduce the average energy consumption of the system while minimizing accuracy drop.

Datasets

To evaluate the performance of the proposed method, the Ninapro DB6 dataset [4] is employed. This dataset is widely recognized as one of the most popular public datasets for sEMG-based hand gesture recognition.

In detail, Ninapro DB6 comprises data obtained from 10 non-amputee participants (7 males and 3 females, with an average age of 27 ± 6 years). Each participant was instructed to replicate identical experiments in 10 distinct sessions. In each session, the participant repeated the same hand gesture 12 times. These sessions were conducted twice daily for five consecutive days, once in the morning and once in the afternoon.

The recorded gestures consist of seven distinct grasps closely resembling everyday activities, in addition to a rest position. Each repetition of a gesture lasts approximately 6 seconds, followed by a 2-second rest period. Regarding the hardware utilized for data collection, the sensor array is composed of 14 Delsys Trigno sEMG Wireless electrodes. These electrodes are placed on the upper half of the forearm to simulate amputation of the lower half. Each sensor captures data at a sampling rate of 2 kHz.

In this work, the 10 recording sessions were divided into two groups, each comprising 5 sessions: the training set and the test set. All training sessions precede the testing sessions to ensure temporal coherence between sessions (specifically, sessions 1 to 5 were used for training, while sessions 6 to 10 were designated for testing). Data preprocessing involved partitioning the data into 150-ms windows (equivalent to 300 samples), with a time shift of 15 ms between consecutive windows. For this dataset, network training focused on stable gestures, where contraction dimensions of 300 (time samples) × 14 (sensors) were retained as the final input windows. In other words, the first and last 1.5 seconds of each gesture were removed, since they are considered as contraction transients. Examples of input windows fed into neural networks are illustrated in Fig. 4.14 for three distinct gestures.

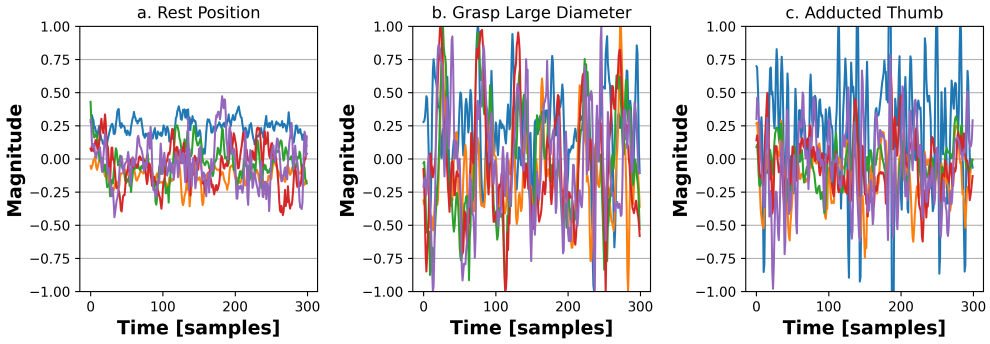


Fig. 4.14 Examples of sEMG signals from Ninapro DB6 [4] respect to three gesture classes.

Bioformer Architectures

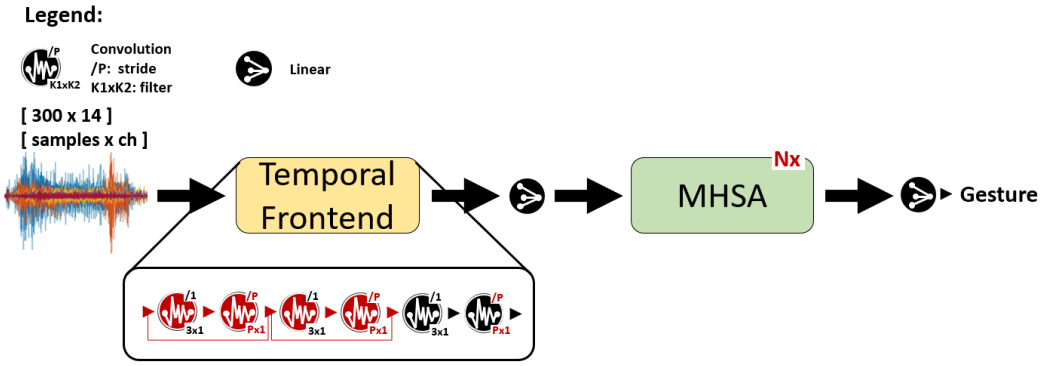


Fig. 4.15 The considered architecture template of bioformers [5].

The tiny transformers utilized in this work, namely *bioformers*, are from [5]. Drawing inspiration from ViTs, which have demonstrated superior performance over CNNs in high-end computer vision tasks [86], the architectural template of Bioformers is structured similarly to ViTs. It consists of a sequential connection of three main components, as depicted in Fig. 4.15:

- An initial set of 1-dimensional convolutional layers (referred to as the temporal frontend), responsible for extracting patches and local features from the input sEMG time series. This component is a key difference compared to ViTs.
- One or more Multi-Head Self-Attention (MHSA) layers, which combine these patches to extract higher-level features.
- A single fully-connected output layer, applied exclusively to the first patch (a special class token added at the beginning of the input sequence), to assign a gesture label to the input.

Based on these bioformer architecture templates, extensive architecture exploration was performed through a grid search, to explore the trade-off between task accuracy and complexity. This search process is divided into two phases, as depicted on the left side of Fig. 4.13. Initially, the model architecture was explored, with flexibility in the depth of the convolutional frontend and the attention component, while also tuning some key hyperparameters. Throughout this phase, variants of bioformers were considered that process "default" input patches of intermediate size. Subsequently, in the second phase, the patch size for the Pareto-optimal models obtained in phase 1 was varied, as this hyperparameter significantly influences both accuracy and complexity. The separation of these two phases was intended to maintain the manageability of the search process.

In particular, front-ends consisting of a minimum of 1 and a maximum of 6 1-dimensional Convolutions (Conv) are explored. Each Conv is followed by Batch Normalization (BN) and Rectified Linear Unit (ReLU) activation functions, although not depicted in Fig. 4.15 for simplicity. Specifically, the investigation considers 1, 2, or 3 frontend blocks, with 1 or 2 layers each. The first (optional) Conv layer of each block employs a 3×1 filter size and a stride of 1. The second Conv uses a larger stride, equivalent to the filter size, to process input signal patches [86, 260]. The patch size matches the convolution stride, denoted as P in Fig. 4.15. The three

frontend blocks have 32, 16, and 14 output channels, respectively. To maintain the compatibility of the tensor shape when using a single patching layer, the number of output channels of the last block remains constant at 14, equal to the number of channels in the input sEMG signal. When utilizing less than 3 blocks, they are removed sequentially from first to last (e.g., the initial block with 32 output channels is the first to be removed). In blocks containing 2 layers, the first convolution adjusts the number of channels, while the second maintains them constant while reducing the temporal dimension due to the stride. During the initial exploration phase of model architecture, patch dimensions (P_0, P_1, P_2) are considered as (, 10), (, 5, 3), and (1, 5, 3) for bioformers with 1, 2, and 3 frontend blocks, respectively. The selection of larger patch sizes for 1-block frontends is motivated by the quadratic complexity of the attention block concerning the input size sequence. Thus, patch sizes are chosen to ensure that the input sequence length passed to the MHSA part is always less than 30, maintaining a low complexity of the bioformer architecture.

Moreover, based on [5], the attention components of the template are also varied in architecture exploration. In particular, bioformers are configured with 1 or 2 MHSA blocks. The number of attention heads H varies in the set $\{2, 4, 8\}$, and the size of each head's projection (DH) in $\{8, 16, 32\}$. Assume C refers to the number of channels, the hidden dimensions of the two linear layers are consistently set at C = 64 and C' = 128. An additional linear layer before the first MHSA block projects the output of the convolutional frontend onto C channels.

When MHSA and front-end variants are combined, a total of 108 different bioformer models are considered in the initial exploration phase. It is important to note that although hyperparameter ranges are manually determined, empirical verification confirms that values outside these ranges result in poorly performing or non-converging models. In the results presented in Section 4.3.4, only a subset of the 108 models is reported, specifically those demonstrating interesting accuracy versus complexity trade-offs.

Once a set of Pareto-optimal bioformers with default patch sizes (P=10) is identified, further exploration is carried out to assess the complexity versus accuracy trade-off induced by varying P to obtain additional network configurations. Specifically, patch size values in $\{5, 10, 30, 60\}$ are considered, as values outside this range tend to yield inferior results for the models placed on the Pareto frontier after the initial exploration phase.

Dynamic Inference

The outcome of the architecture optimization and subsequent Pareto analysis is a collection of bioformers, each achieving varying test accuracy and involving different numbers of parameters and MAC operations. In a standard machine learning operations (MLOps) workflow, the model to be deployed in the field would be selected from this set based on the specific hardware requirements. In other words, selecting the most accurate model that also meets energy and latency limitations in a particular sEMG smart sensor design.

Although this approach provides some level of flexibility, designers are still limited to choosing from a small number of options that only offer a coarse-grained sampling of the design space. Moreover, the selection is made once-for-all, which may not be optimal intuitively.

In fact, it is not always the case to prefer a complex yet accurate model targeting this gesture recognition task based on sEMG smart sensors. When the battery of smart sensors is not well charged and/or when the input is considered as "easy" (e.g., a sample corresponding to the rest class), there might be a preference for a simpler and less energy-intensive model. This added flexibility can be provided by dynamic (or adaptive) inference techniques [147], which optimize the operation of machine learning applications at runtime. These techniques remove the conventional assumption that the same model must be executed identically for all processed inputs.

Therefore, a staged and big/little inference to gesture recognition based on sEMG smart sensors is proposed in this work. The resultant system comprises three stages that incorporate two bioformers alongside a shallow ML model, as depicted on the right side of Fig. 4.13.

In detail, the first stage of the dynamic system is realized with a small RF, aiming to classify input into either rest or non-rest categories. To train the RF, all inputs from the target dataset excluding those belonging to the rest position class were considered as non-rest samples. The basic principle behind this stage is twofold: (i) the rest samples are expected to be the most common in a real-world deployment scenario and (ii) they represent sEMG signals with low variability, making them easily distinguishable from signals associated with specific gestures. This distinction can be observed by comparing the three graphs in Fig. 4.14, which illustrate examples

of sEMG signals corresponding to one rest input and two other gestures, extracted from the target dataset.

A different rest detector was trained for each subject, employing the same data split outlined in Section 4.3.3. RFs consisting of a maximum of 12 decision trees, each with a maximum depth of 5, were considered. The model was fed with a single feature *waveform length (WL)* per sEMG channel [4], which is defined as Equation 4.3,

$$WL_i = \sum_{k=1}^{300} \left\| X_{i,k} - X_{i,k-1} \right\| \tag{4.3}$$

where 300 is the number of samples in a window, chosen based on previous studies using classical ML models for this task [4, 256, 257]. With 14 sEMG electrodes (that is, channels), a total of 14 features are provided to this RF-based rest detector. Note that the memory and computational requirements to execute this model, including feature extraction, are negligible even in comparison to the simplest bioformer.

The next step involved feeding the non-rested samples into subsequent stages, featuring a **big/little** classifier created by combining two bioformers with varying levels of accuracy and complexity. To determine whether to stop after the "little" bioformer (stage 2) or proceed to the "big" one (stage 3), the score margin (SM) [149] is used as the classification confidence metric. The score margin can be calculated using the following formula:

$$SM = \hat{y}^{(little)}_{i_{max}} - \hat{y}^{(little)}_{i_{2nd-max}} \tag{4.4}$$

where $\hat{y}^{(little)}_{i_{max}}$ and $\hat{y}^{(little)}_{i_{2nd-max}}$ refer to the highest and second highest class probability scores produced by the little model, respectively. The confidence metric SM is used for comparison against a configurable threshold T_h to determine the final prediction of the outcome of either stage:

$$\hat{y} = \begin{cases} \hat{y}^{(little)} \text{ if } SM > T_h \\ \hat{y}^{(big)} \text{ otherwise} \end{cases} \tag{4.5}$$

The principle behind this stopping criterion, which is commonly employed in such systems, is that a high value of *SM* indicates a confident classification (where only one class is highly probable), and, conversely, a low value indicates uncertainty. Various criteria can be utilized to choose the two bioformer models that make up the big/little system. Typically, this choice is influenced by the required range of accuracy, energy, and latency values that the system has to support during runtime. When specific constraints are lacking, a natural approach is to select the two extremes of the Pareto frontier concerning accuracy versus the number of operations, identified after the architectural exploration phase.

The proposed dynamic system offers a range of configuration options that can be adjusted in real time. One such parameter is the threshold (Th) of the big-little bioformers, which can be modified to achieve various trade-offs between complexity and accuracy. For example, setting a high Th value means that the classification stops at the second stage ("little" bioformer) only when predictions strongly favor a single class, leading to a conservative system that prefers invoking the "big" model more frequently to maximize accuracy. On the contrary, a low value of Th prioritizes energy savings by stopping more often in the "little" stage. The experimental results (refer to 4.3.4) demonstrate that adjusting Th allows various running conditions to be achieved with just two models.

Another aspect to consider is the option to deactivate the RF-based rest detector. In this scenario, the two bioformers retain the ability to classify inputs as belonging to the "rest" class independently, offering yet another trade-off between complexity and accuracy. While the rest detector effectively filters out "easy" inputs, reducing the average computation per input, its absence may lead to higher accuracy because it avoids false positives produced by the rest detector, i.e., inputs wrongly classified as "rest", cannot be corrected by subsequent stages. Consequently, disabling the rest detector shifts the operational point towards regions of higher latency/energy but potentially higher accuracy. In addition, the retention of the "rest" class by the bioformers could also help in addressing some false negatives from the rest detector, where inputs wrongly classified as "no rest" could be corrected.

In summary, by adjusting both the threshold (Th) and deactivating the RF-based rest detector, the system offers a spectrum of 10 different operating points that can be switched in real time, covering a wide range in terms of accuracy and energy/latency trade-offs. Evaluating the early-stopping policy for the big/little bioformer has

minimal impact on total prediction time and energy consumption, and the memory footprint for storing the rest detector is negligible. The main overhead of the dynamic system lies in the additional memory required to store two bioformers instead of one. However, as elaborated in Section 4.3.4, this added flexibility justifies the overhead. Importantly, achieving a comparable range of operating conditions with multiple independent static models (without employing a multi-stage dynamic approach) would incur substantially higher overheads.

4.3.4 Experimental Results

Setup

The bioformer models were trained and validated on the NinaPro DB6 dataset, as detailed in Section 4.3.3, using Python 3.8.13 and PyTorch 1.10.2. Initial training and architecture exploration were conducted with standard single-precision floating-point (FP32) representation for weights and intermediate activation tensors. Subsequently, QAT was performed to convert FP32 tensors into 8-bit integers (INT8) before deployment. This step, aimed at reducing the memory requirements of bioformers, was essential because not all edge sensor hardware platforms were equipped with floating-point units. The built-in QAT functionality of PyTorch was utilized for this purpose.

To evaluate the proposed dynamic system, the quantized models were deployed on the GAP8 SoC utilizing the optimized library of kernels for MHSA and convolutions as described in [131] and [262] respectively. The GAP8 SoC, developed by GreenWaves Technologies, features a fabric controller composed of a single RISC-V core responsible for managing peripherals and orchestrating program execution. In addition, it includes a cluster of 8 identical RISC-V cores with shared 64 kB scratchpad memory, designed to accelerate intensive workloads. Both the fabric controller and the cluster share a 512 kB L2 memory.

In this work, the state-of-the-art CNN-based gesture recognition solution proposed in [252], known as TEMPONet, was considered as a comparison. TEMPONet had previously achieved state-of-the-art accuracy on the same target datasets in this work. To ensure fairness in the comparison, an identical training protocol was used for both the models considered and the baseline comparison.

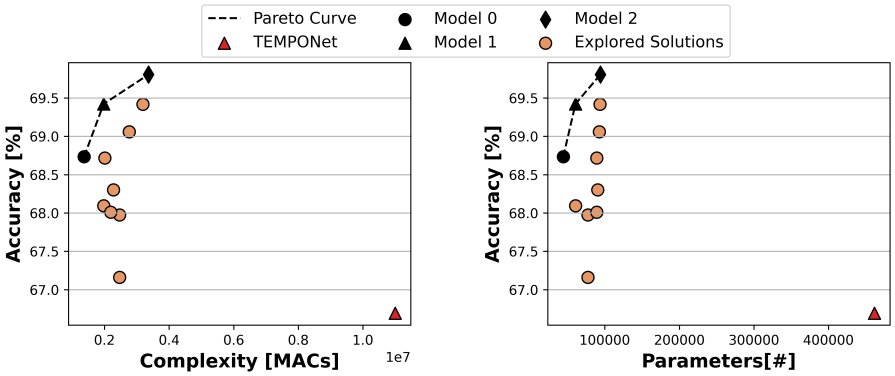


Fig. 4.16 The results of architecture exploration of bioformers.

Architecture Exploration

Fig. 4.16 shows the results of the initial phase of the exploration of the bioformer architecture. As detailed in Section 4.3.3, various bioformer variants were examined using default patch sizes, while adjusting hyperparameters such as the number of layers in the convolutional frontend, the number of MHSA blocks, the number of attention heads and the size of each head's projection. Two graphs are presented in Fig. 4.16, which illustrate the Pareto curves generated by some of these architectures in terms of accuracy versus the number of MAC operations and accuracy versus the number of parameters, respectively. Accuracy results are applicable to floating point models at this stage. Each marker denotes a specific architecture configuration. Note that only configurations near the Pareto frontier are displayed in the figure. Models with low accuracy, failure to converge, excessive complexity, or excessively large size are omitted to facilitate visualization.

In both graphs, Pareto-optimal solutions are denoted by black dashed lines. Remarkably, the same set of three models emerges as optimal when considering both MACs and parameters as complexity metrics. These three bioformers are labeled Model 0 (marked with • in the graphs), Model 1 (▲), and Model 2 (⧫), arranged in order to increase accuracy. With default patch sizes P=10, these models exhibit a relatively narrow accuracy range of 68.73–69.81%, while MAC operations vary more widely between 1.37–3.36 M, and parameters range from 44.35–94.09 k. The detailed architecture configurations of the three Pareto-optimal bioformers are as follows:

Model 0. H = 8, DH = 8, 1 block with 1 Conv layer in the frontend;

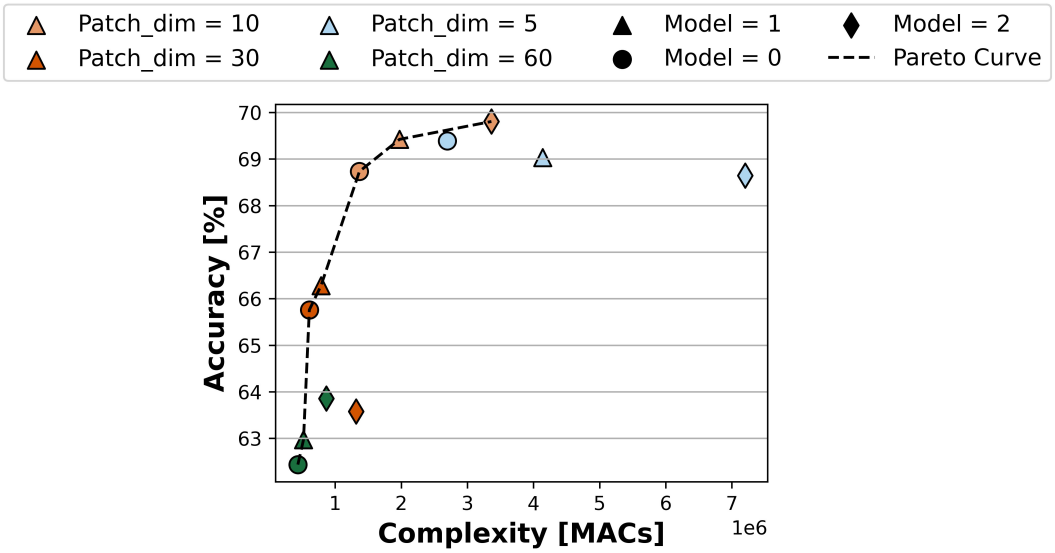


Fig. 4.17 The results of patch size exploration in terms of complexity.

Model 1. H = 8, DH = 16, 1 block with 1 Conv layer in the frontend;

Model 2. H = 8, DH = 32, 1 block with 2 Conv layers in the frontend.

These results offer several intriguing insights. Firstly, having a substantial number of heads (H) in MHSA blocks is crucial to achieving high accuracy in this task. Secondly, increasing the size of each projection (DH) represents a straightforward approach to balance additional operations and parameters for improved accuracy. Lastly, a deep temporal frontend is not essential, as all Pareto-optimal solutions feature a single frontend block at most, and no more than two one-dimensional convolutional layers.

All three bioformers exhibit substantial performance improvements over the baseline TEMPONet comparison (red triangle). It should be noted that the highest accuracy is 69. 81% achieved in the DB6 dataset represents the most recent state of the art in training utilizing a time-based data split (i.e., using only past data to predict gestures at future time points), as described in Section 4.3.3. While there is room for further improvement, this accuracy suffices for various practical applications.

Fig. 4.17 illustrates the results in terms of the complexity of exploring the patch size hyperparameter. This exploration began with the three Pareto-optimal bioformers with default patch sizes as shown in Fig. 4.16, and then varying the patch size by values in the set {5, 10, 30, 60}. Each Pareto model is distinguished by a

unique marker shape, while the four different colors represent different patch sizes. Once more, the Pareto frontiers are highlighted by black dashed lines.

Adjusting the patch size substantially enriches and expands the range of optimal solutions in the accuracy versus MACs space. This expansion now encompasses models with accuracy ranging from 62.43% to 69.81%, requiring 0.44 to 3.36 million MAC operations. This outcome is expected because the number of operations in the forward pass of an MHSA block increases quadratically with the length of the sequence S, which is directly influenced by the size of the patch P (larger patches result in shorter sequences and vice versa). However, the operations in the temporal frontend are independent of P. This is because the first convolution, if used, has a fixed 3×1 filter, and the second convolution performs a single MAC operation per input value, regardless of P (because of the use of a stride equal to the size of the kernel). Consequently, an inverse relationship can be observed where models with larger patch sizes require fewer operations and generally achieve lower accuracy.

In summary, the exploration of patch sizes demonstrates that the three bioformers (Model 0, 1, and 2) maintain their positions on the Pareto front when varying the patch size. Moreover, increasing the patch size proves to be highly effective in exchange for a reduction in the number of inference operations for a decrease in accuracy.

Dynamic Inference

Based on the previous exploration, the models equipped in stage 2 and stage 3 of dynamic inference systems were selected. Fig. 4.18 shows the results achieved by the dynamic inference system outlined in Section 4.3.3. Specifically, the system is constructed by choosing the two bioformers with the fewest/most MAC operations as the foundation for the big/little scheme, refer to Fig. 4.17. These correspond to Model 0 with $P = 60$ and Model 2 with $P = 10$.

The green and blue points in Fig. 4.18 represent the potential operating conditions of the dynamic inference systems with/without the RF rest detector enabled, respectively. These points are obtained by varying the confidence threshold (Th) of the big/little scheme from 0 to 1, determining when to stop after stage 2 or proceed to stage 3. For comparison, the figure also shows the results of independent Pareto-optimal static models spanning a comparable range of MACs, depicted as orange

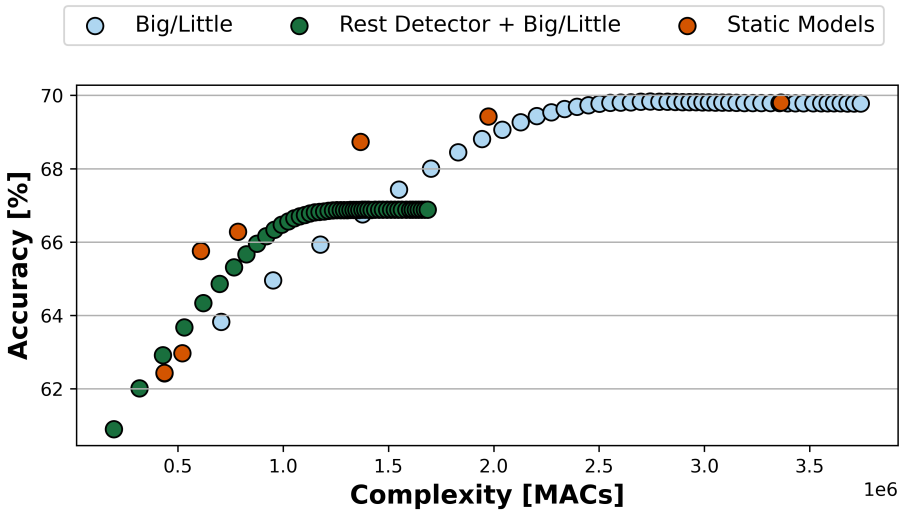


Fig. 4.18 The comparison in dynamic inference results.

dots and directly derived from Fig. 4.17. The reported number of MAC operations for the dynamic approach is the average over the test set for all subjects.

As illustrated in the figure, adjustment Th enables the achievement of various operating points in the accuracy versus the number of operations plane. In particular, higher values of Th yield greater accuracy at the expense of increased operations and vice versa. In addition, enabling or disabling the rest detector generates two distinct sets of operating points: one more efficient yet less accurate (with rest detector enabled, shown in green), and the other more accurate, but requires a higher number of operations per input on average (rest detector disabled, in blue).

In summary, by combining these two parameters, the three-stage dynamic system can cover a wide spectrum in accuracy (ranging from 60.90% to 69.78%) and MACs (from 196.4 thousand to 3.74 million), which subsequently result in varied average energy consumption and latency values. Overall, the combined adjustment of these parameters offers slightly inferior but comparable accuracy versus MAC trade-offs compared to static solutions. However, the dynamic approach yields approximately 100 distinct operating points, which provide significant advantages. Unlike static solutions, the benefits of a dynamic approach become evident, especially with regard to memory overhead, as discussed further in Section 4.3.4. In essence, the dynamic system achieves this wide trade-off range with only two bioformers (plus the negligible rest detector). On the contrary, reaching a similar range of accuracy and complexity using standalone static models (depicted as orange dots)

would necessitate the deployment of seven different bioformers, resulting in obvious overheads in terms of memory and storage. However, the flexibility offered by the latter approach would still be more limited, including only 7 points compared to 100.

Deployment

Table 4.8 presents the results of the implementation of various dynamic inference solutions on the GAP-8 SoC platform. Furthermore, for comparison, the CNN TEMPONet from [252] was also quantized and deployed.

The three leftmost columns of the table show the configurations of dynamic systems based on utilized models for each stage. For bioformers, the notation follows the format -*N*-(*P*), where *N* denotes the model number (0, 1, or 2) and *P* represents the patch size. Configurations that do not employ all stages are indicated with "n.a." values in some columns. The "Technique" column specifies the type of dynamic system employed, where "Rest" indicates the utilization of the RF-based rest detector, and "Big/Little" signifies the combination of two distinct bioformers. Accuracy, MMAC, latency, and energy consumption are reported for each configuration, with the last two metrics averaged over the test sets.

Table 4.8 Detailed deployment results of dynamic inference on GAP8.

Stage-1	Stage-2	Stage-3	Technique	MMAC [#]	Latency [ms]	Energy [mJ]	Accuracy
n.a.	n.a.	- 0 - (10)	None	1.37	1.130	0.058	67.04%
n.a.	n.a.	- 2 - (10)	None	3.36	2.797	0.143	65.28%
n.a.	n.a.	- 0 - (60)	None	0.44	0.364	0.019	60.82%
RF	- 2 - (10)	n.a.	Rest.	1.686	1.402	0.071	62.55%
n.a.	- 0 - (60)	- 2 - (10)	Big/Little	2.483	2.067	0.105	65.20%
RF	- 0 - (60)	- 2 - (10)	Rest.+Big/Litte	1.274	1.061	0.054	62.54%
RF	- 0 - (10)	n.a.	Rest.	0.685	0.570	0.029	64.69%
n.a.	- 0 - (60)	- 0 - (10)	Big/Little	1.319	1.098	0.056	67.02%
RF	- 0 - (60)	- 0 - (10)	Rest.+Big/Litte	0.631	0.525	0.027	64.72%
[252]	-	-	-	16.00	21.828	1.113	61.95%

The first three rows present the metrics of the three static models utilized as the foundation for the big/little bioformers, provided for comparison purposes. Subsequently, the following six rows detail the metrics obtained from two dynamic systems constructed using these bioformers as components. Specifically, the least energy-consuming bioformer (Model-0 with P = 60) was used as the "little" model in all cases. Regarding the "big" model, the bioformer achieving the highest floating point accuracy (Model-2 with $P = 10$) or the one attaining the highest int8 accuracy

(Model-0 with $P = 10$) is considered. The former corresponds to the same combination of big/little models depicted in Fig. 4.18. Each of these two dynamic systems is presented in multiple configurations, namely: (i) a solution that directly combines the rest detector with the big model without leveraging the big/little technique; (ii) the big/little system with the rest detector disabled; (iii) the big/little system with the rest detector enabled. As each of these combinations can operate under tens of different operating conditions (achieved by changing Th), the latency and energy results corresponding to the value Th are reported that yields the maximum accuracy with the fewest MACs. This is equivalent to the leftmost extreme of the horizontal green and blue segments in Fig. 4.18.

The results indicate that big/little bioformers, despite having nearly the same accuracy as the big models from which they are derived (67.02% vs. 67.04% and 65.20% vs. 65.28%, respectively, for Model-0 and Model-2 as "big" NNs), can reduce the average inference energy consumption by 1.03×–1.35× (56 µJ vs. 58 µJ and 105 µJ vs. 143 µJ, respectively). Furthermore, the efficacy of the proposed three-stage solution is demonstrated by achieving the same accuracy as a system using only the RF rest detector combined with the "big" model (64.72% vs. 64.69% and 62.54% vs. 62.55% for the two cases), while reducing the average consumption per input by 1.07×–1.31×. Thus, it can be concluded that proposed multi-stage dynamic approach, combining staged and big/little inference, can further reduce energy consumption and enhance flexibility compared to any scheme involving only two levels. Furthermore, these results are derived from the NinaPro DB6 test set, which comprises a comparable number of "rest" and "grasp" samples. In a real deployment scenario, where the proportion of "rest" samples is likely higher, the dynamic system would likely yield even larger energy savings.

In terms of memory occupation, the entire three-stage system requires a total of 132 kB/184 kB when Model-0 and Model-2 are utilized as "big" models, respectively. While there is a noticeable memory overhead compared to a single static model, the system still fits largely within the L2 memory of GAP8. On the contrary, the total memory occupied by all seven dynamic systems exceeds the L2 memory of GAP8.

In particular, the last row reports the result of the deployment of the TEMPONet comparison baseline TEMPONet [252]. Importantly, while achieving a similar accuracy (62.54% vs. 61.95%), the dynamic inference solution equipped the rest detector, Model-0 with P = 60 as the "little", Model-2 with P = 10 as the "big" model,

saved over 95% energy consumption and achieved up to 21x speed up, compared to TEMPONet. On the other hand, the solution of a static model that achieves the greatest energy savings (Model-0 with P = 60) reduced the inference time and energy consumption of 98%, with only a decrease in precision of 1. 13%.

4.3.5 Conclusion

Hand gesture recognition based on sEMG smart sensors can derive benefits from on-device execution, offering faster and more predictable response times and improved energy efficiency. However, deploying state-of-the-art DL-based smart models for this task on memory-constrained and battery-operated wearable sensor nodes, necessitates a meticulous optimization process, both during the design period and running time. In particular, the former involves appropriate tuning of the DL model architectures, and the latter provides flexibility to select appropriate solutions in real time.

With the aim of targeting this application, both optimization objectives are pursued, presenting a novel gesture recognition system that enhances state-of-the-art models in terms of accuracy and efficiency. At the DL model architecture level, tiny transformer models (referred to as bioformers) are applied for the first time to sEMG-based gesture recognition. Through extensive architecture exploration, it is demonstrated that the most accurate bioformer achieves a higher classification accuracy on the popular Non-Invasive Adaptive hand Prosthetics Database 6 (Ninapro DB6) dataset compared to the state-of-the-art CNN TEMPONet (+3.1%).

When deployed in the low-power SoC GAP8, bioformers based on RISC-V that outperform TEMPONet in accuracy consume 7.8×–44.5× less energy per inference. At run-time, a three-level dynamic inference approach is proposed, combining a shallow classifier, i.e., an RF implementing a simple "rest detector," with two bioformers of different accuracy and complexity, which are sequentially applied to each new input, stopping the classification early for "easy" data. With this mechanism, a flexible inference system is obtained, capable of operating at many different operating points in terms of accuracy and average energy consumption. In GAP8, a further 1.03×–1.35× reduction in energy consumption is achieved compared to static bioformers at iso-accuracy.

To summarize, this work demonstrated the success of a tiny Transformer targeting gesture recognition based on sEMG smart sensor to achieve state-of-the-art performance with lightweight architectures. Moreover, the proposed dynamic inference can further provide extremely flexible solutions for the implementation of smart models in such smart sensors, achieving a wide range of trade-offs in terms of accuracy and computational complexity, making the proposed solutions applicable to a wide range of practical scenarios.

Chapter 5

Hardware Acceleration for Smart Sensors

This chapter introduced hardware-level optimization in smart sensor hardware systems, to further achieve energy efficiency by collaborating on algorithm-level design. In particular, the contributions at the hardware level targeting smart sensors in this chapter can be summarized as follows:

Hardware Optimization of DL for a smart sensors prototype.

- An optimized RISC-V ISA core is proposed to provide efficient computation for low-precision DL models. Based on a lightweight RISC architecture, vectorized operations at multiple-bit widths have been implemented by a newly introduced HW unit and the corresponding ISA extension. The experimental results have demonstrated its energy efficiency in executing quantized DL models with 4-bit and 8-bit precision, compared to the other hardware platforms from both commercial and academic fields.

Approximate I2C bus encoding.

- An energy-efficient approximate serial bus encoding namely ACME targeting the I2C bus is proposed. By maximizing the number of logic-1 states and reducing transitions, ACME outperformed the state-of-the-art approximate serial bus encoding on two datasets, and an average energy

saving of over 54% can be achieved with an error margin below 0.1%. The synthesis results indicate that the hardware implementation overhead is negligible.

5.1 Motivation

In the previous chapter, compact and efficient ML/DL architectures and energy-efficient inference strategies are discussed. However, it is insufficient to focus solely on algorithm-level optimization. The rapidly increasing computational complexity in the modern world has been proposing an urgent need for HW-SW co-design, to implement optimized IoT smart sensors.

One effective approach involves integrating specific hardware features in the MCU, which is a common platform basis of smart sensors at edge, tailored to support efficient computational operations within ML/DL frameworks. Thanks to such hardware acceleration, it benefits intelligent task execution in smart sensors in the following aspects:

Energy Efficiency. Using hardware acceleration can lower the energy consumption of sensor systems. A number of repeated operations may be skipped or merged by simpler operations by dedicated hardware modules, achieving higher energy efficiency compared to using general-purpose processors for the same tasks.

Real-time Requirements. Smart sensors usually involve applications requiring real-time response. By several techniques such as SIMD, hardware acceleration can perform multiple computations at the same time, thereby providing lower latency. This is crucial for certain applications such as smart security systems or medical monitoring devices.

5.2 Customized Core for Low-precision DNN Inference

5.2.1 Context and Motivation

Currently, most smart sensors rely on MCUs to perform data processing and analysis tailored to their particular tasks. As general-purpose processors, MCUs benefit from their flexibility and quick market entry. Their programmability and wide selection of peripheral options provide numerous design opportunities [106]. Therefore, MCUs become one promising platform to integrate specific hardware features to efficiently support smart models in smart sensors.

RISC-V is an open-source ISA based on RISC principles. Thanks to its simplicity, extensibility, and active community, RISC-V architecture has become a new paradigm for designing processor cors, used in the MCU embedded in smart sensors. There exists a variety of MCUs tailored for embedded systems equipped with RISC-V-based processors, both in the academia and industrial fields, including RI5CY [158], IBEX [263], SiFive E31 [264] etc.

Among them, IBEX is a lightweight 32-bit RISC-V CPU core developed in SystemVerilog. It is highly configurable and particularly well-suited for embedded control applications. In particular, this core is optimized for area and implemented with a 2-stage pipeline (with an optional third pipeline *Writeback* stage): *Instruction Fetch (IF)* and *Instruction Decode and Execute (ID/EX)*.

However, while MCUs offer the capability to handle various tasks due to their generality, this versatility also imposes limitations on their maximum performance for any one specific task. On the other hand, as ML/DL algorithms have been playing an increasingly important role in smart model design involving smart sensors, the requirements to efficiently execute the operations of these models become urgent, especially for ultra-low-power MCUs.

Moreover, the NN-based models have demonstrated superior performance involving smart sensor applications such as human activity recognition, people counting, etc [14, 3]. To fit the resource-limited condition of MCUs, which are usually only equipped with integer units, and reduce memory occupation, these ML/DL models are typically quantized into low-precision. In Chapter 4, these optimizations of

compact smart models are introduced in detail. Therefore, there exists an energy waste by using original computational operations provided by MCUs. For instance, to perform the inference of quantized NNs at 8 bits, the register size in a RISC-V MCU (e.g., 32 bits), exceeds the amount of data being processed, resulting in unused or "wasted" bits within the register. This inefficiency can lead to suboptimal memory usage and potentially impact performance and resource utilization in smart sensor systems. Therefore, customized ISA extensions such as integer SIMD can significantly reduce such energy waste.

5.2.2 Limitations in the Existing Solutions

In the literature, several works made efforts to provide quantized NN libraries to be deployed on the MCUs. [265] proposed CMSIS-NN library developed by ARM for implementing 16-bit and 8-bit QNNs on Cortex-M-based systems. [266] focused on RISC-V, and provided support targeting extremely quantized NNs with bitwidths low to 1-bit, as well as mixed-precision formats. However, these libraries still suffer from the gap between lack of hardware support at the ISA level, as described in the previous section. MCUs such as RV32IMCXpulpV2 [158] and ARMv8.1-M [157] provide support to 16-bit or 8-bit data operations, however, they suffer from the inference of sub-byte QNNs (e.g., 2-bit or 4-bit) since low-precision data have to be unpacked first, and it leads area and energy overheads.

[50] solved this problem by proposing an energy-efficient mixed-precision arithmetic unit tailored for the execution of low-bit-width QNNs. In particular, this unit supports sub-byte SIMD operations at 8-, 4-, and 2-bit precision levels. However, this work focused on a "larger" MCU that has a 4-stage pipeline and optional 32-bit FPU, with multiple ISA extensions such as hardware loops, etc. While providing capabilities to support various quantized DL algorithms, such MCUs may have relatively high costs and be overqualified for running simple tasks such as people counting in low-cost smart sensors.

5.2.3 MAUPITI System and Smart Sensor Prototype

In this thesis, a hardware system targeting IR smart senor prototype (namely **MAUPITI**) containing a customized core supporting low-precision DNNs is introduced. This

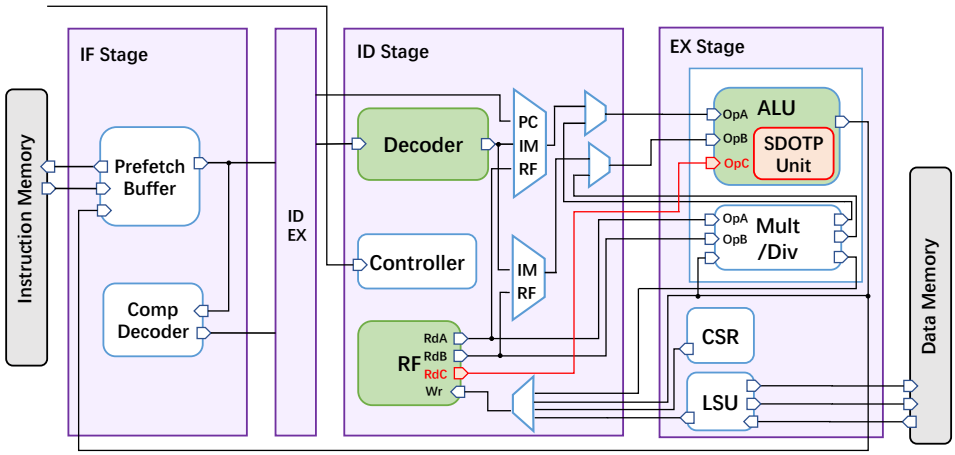


Fig. 5.1 The architecture of customized IBEX RISC-V core.

customized core is based on the IBEX [263] RISC-V core with a customized ISA extension. Specifically, such MAUPITI system is tailored for efficient people counting tasks, which is the one of most common tasks involving IR smart sensors. The smart models are optimized to use precision as INT4 and INT8, therefore, to exploit the maximum optimization in this novel smart sensor prototype by collaboration of SW-HW, HW acceleration features are introduced into the MCU core. The work described in this section has been previously accepted in [6].

Core Customization

The IBEX core has been customized to enable support for quantized DNNs utilizing INT4 and INT8 data formats, with the incorporation of an efficient arithmetic unit facilitating low bit-width integer SIMD Sum of Dot Product (SDOTP) operations. In Fig. 5.1, hardware blocks highlighted in green denote modifications, while the orange block represents a completely new addition.

In detail, the SDOTP unit executes a MAC operation between two 32-bit registers (RS1 and RS2), interpreting their content either as four 8-bit or eight 4-bit signed values. Additionally, an extra 32-bit register (RD) serves as input and output for the accumulation process. To ensure single-cycle latency, the dot product computation employs four independent 8-bit multipliers and eight 4-bit multipliers, along with an adder tree to sum up the partial products and the additional 32-bit operand.

Replicating the multipliers for the two bitwidths, instead of sharing them, effectively moves the new block out of the core's critical path, albeit with an acceptable

increase in area. Key hardware modifications also extend to the Decoder, which has been expanded to support the new opcodes, and the Register File (RF).

The SDOTP unit utilizes RD as both source and destination, whereas the vanilla IBEX ALU only supports two input operands. Thus, a third input (OpC) has been added to the ALU, which always takes data directly from the register file, with the MAC result subsequently written onto the same register. Consequently, a read port (RdC) must be added to the RF to accommodate this requirement, as illustrated in Fig. 5.1.

With respect to [50], a State-of-the-Art (SotA) mixed-precision ISA extension for RISC-V MCUs, the proposed solution places much more emphasis on containing area (i.e., cost) overheads, sacrificing some performance and flexibility, as we target a low-cost smart sensor application. Consequently, unsigned variants of the SDOTP are not implemented, nor are 8x4-bit or 4x8-bit versions to support different precisions for weights and activations. Similarly, the implementation of 2-bit SDOTP is omitted since QAT experiments revealed significant accuracy degradation associated with 2-bit precision.

Furthermore, only the register addressing mode for the source operands is supported, necessitating separate loads before a dot product. Additionally, combined MAC and Load functionality is not implemented. Lastly, separate instructions for simple dot product (DOTP) are avoided, as these can be reduced to SDOTP by setting the third input register RD to 0. These simplifications collectively contribute to a significant reduction in area overhead to less than 7% compared to the vanilla IBEX core. In contrast, [50] reported a 17.5% increase in the core area despite starting from a more complex baseline core.

MAUPITI Platform for Smart Sensor Prototype

The illustration in Fig. 5.2 shows the complete MAUPITI Platform in IR smart sensor prototype, fabricated using 130nm CMOS technology, operating at a clock frequency of 20MHz, and capable of capturing thermal images at a frame rate of 10 Frames Per Second (FPS). This system comprises three primary components:

16x16 array of Thermal MOSFET (TMOS) sensors. A 16x16 array of TMOS sensors (1) designed to detect infrared radiation-induced heating effects. The

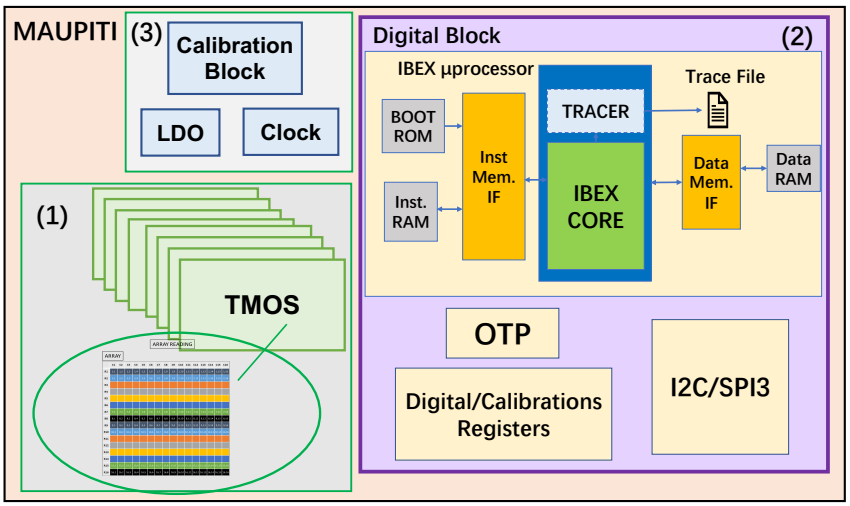


Fig. 5.2 The complete MAUPITI System.

array features 8 parallel analog front-end processing chains, each responsible for acquiring one row of pixels. This setup enables a two-step acquisition process for a complete frame. Each TMOS sensor draws approximately 1µA at 2.4V, resulting in a total power consumption of 0.62mW for the entire array.

Digital processing block. A digital processing block (2) houses a specialized IBEX core (highlighted in green), along with its instruction and data memory modules (each 16KB in size), and associated interfaces. Additionally, this block incorporates an 80B One-Time Programmable Memory (OTP), an instruction tracer, calibration registers for TMOS sensors, as well as I2C and SPI3 communication interfaces. In FF conditions, the digital block consumes approximately 0.9mW of power.

Standard circuitry block. The system includes standard circuitry blocks (3) for functions such as clock and reset management, voltage regulation, and other essential tasks.

In summary, the MAUPITI system is a sophisticated arrangement tailored for thermal imaging applications, featuring efficient sensor arrays, robust digital processing capabilities, and essential supporting circuitry, all optimized for low power consumption and high performance.

Table 5.1 Deployment results of three DL models from [6] on different HW platforms.

Model	Platform	Code [B]	Data [B]	Energy [μJ]
Top	STM32	22840	10420	9.381
	IBEX	3476	1104	6.003
	MAUPITI	4152	1104	4.927
- 5%	STM32	22970	9060	6.854
	IBEX	3384	648	5.005
	MAUPITI	4052	648	4.525
Mini	STM32	22950	8410	5.640
	IBEX	2700	416	4.342
	MAUPITI	3208	416	4.067

5.2.4 Experimental Results

Table 5.1 presents the deployment results of three different DL models targeting people counting tasks involving smart sensors, on three different HW platforms. In particular, these DL models are selected from [6], and specifically, they are i) the top-scoring model (Top), ii) the smallest model in terms of memory with a maximum accuracy drop smaller than 5% (-5%), and iii) the smallest overall (Mini). It is notable that, in this case, the latter corresponds to the models with the lowest MACs. The code size (Code), memory occupation (Data), and energy consumption per inference on three different platforms are reported for the aforementioned architectures.

Specifically, a comparison is made between the proposed MAUPITI platform and an unmodified IBEX core, utilizing no custom instructions. For these two targets, the deployment toolchain is applied with identical compilation flags. Additionally, a comparison is drawn between MAUPITI and an off-the-shelf MCU solution, namely a STM32L4R5 [267] core with models deployed on 8 bits only using the proprietary X-CUBE-AI toolchain, which does not support mixed-precision.

When comparing with IBEX, it is observed that the Top model achieves the largest gains in terms of energy efficiency. Despite a 7% area overhead and a 2.2% power overhead, MAUPITI achieves up to a 17.9% reduction in energy consumption. It is worth noting that the efficiency benefits of the low-precision SIMD in MAUPITI are limited by the small geometry of the considered layers, namely the small number of output channels, and would be higher for larger DNNs.

5.3 Energy-Efficient Approximate Bus Encoding for I2C

5.3.1 Motivation

Communication is one of the important components in smart sensor systems; refer to section 2.2. In fact, ultra-low-power smart sensor systems featuring numerous peripherals and off-chip serial interconnects significantly contribute to the total energy budget, making energy savings more urgent.

Most smart sensor networks are equipped with numerous sensors, typically connected to processing elements (e.g., MCUs) via serial buses utilizing standard protocols such as SPI, Controller Area Network (CAN), I2C, and others, data transfers play a crucial role in the total energy consumption. Previous studies have demonstrated that in ultra-low-power IoT nodes such as smart sensors, the energy consumed by off-chip bus transmissions can be comparable to or even exceed the energy consumed by processing operations [159].

Over the past decade, Approximate Computing (AxC) has emerged as a promising paradigm for designing energy-efficient digital systems [51]. This approach leverages the error tolerance of many emerging applications such as multimedia processing and machine learning, allowing them to withstand errors at various levels during execution with minimal impact on final results [51]. In numerous previous studies, AxC has been used to optimize the serial bus interfaces of sensor-rich devices [167, 168]. By encoding the data in a lossy manner, these studies achieved savings by minimizing the number of electrical level transitions on the bus [167, 168].

I2C is one of the most common communication protocols applied in the communication components of smart sensors. In I2C peripherals, both the Serial Clock Line (SCL) and the Serial Data Line (SDA) employ an open-drain physical connection, capable of driving the bus line only to a low level. To bring the line to high voltage when the bus is idle, a pull-up resistor is employed. Consequently, in addition to the dynamic consumption resulting from level changes, a substantial amount of static energy is dissipated by the pull-up resistor continuously when the line voltage is low, corresponding to logic-0 values.

In this work, it is demonstrated that this static component is actually the dominant factor, underscoring the sub-optimal nature of existing approximate bus encodings when applied to I2C. Therefore, a novel energy-efficient approximate bus encoding named AxC Maximum-one Encoding(ACME), targeting specifically I2C connections, was proposed. The work described in this section has been previously published in [268].

5.3.2 Proposed Method

I2C Power Model

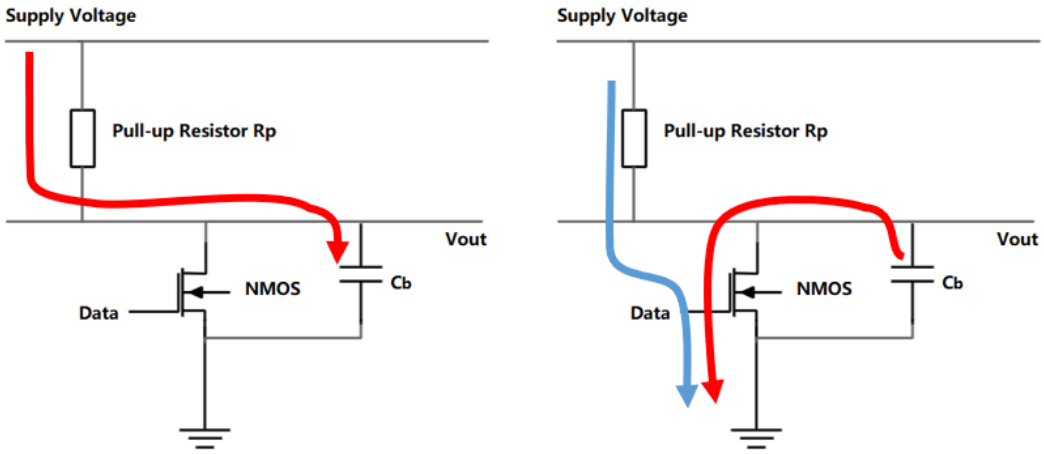


Fig. 5.3 I2C open-drain architecture and current flow at High/Low logic levels.

In the I2C protocol, an open-drain architecture is enforced for both the SDA and the SCL. This configuration involves an NMOS transistor and a pull-up resistor Rp, as illustrated in Fig. 5.3. Additionally, C_b represents the cumulative bus capacitance, encompassing the contributions from the printed circuit board (PCB) and the drivers of all devices connected to the bus [269].

The NMOS transistor functions as a switch driven by the data to be transmitted in the I2C protocol. To transmit a logic-1, the NMOS is turned off, and Vout is pulled up to the supply voltage by Rp. Conversely, when transmitting a logic-0, the switch is turned on, connecting Vout to the ground. Therefore, whenever Vout is at a logic low level, a significant static energy consumption occurs due to the current

flowing through the pull-up resistor Rp [270]. In addition to this static consumption, during a 0-to-1 transition, dynamic dissipation occurs due to the charging of the bus capacitance Cb. Similarly, the discharge of Cb consumes dynamic energy during 1-to-0 transitions. The static and dynamic current flows are depicted in blue and red, respectively, in Fig. 5.3a (b) for a high (low) logic level.

Previous research [270] has demonstrated that static energy is the dominant factor in the total energy consumption of an I2C channel. However, in this study, a comprehensive model that incorporates both static and dynamic energy is considered, to evaluate the proposed approximate encoding, applied to the SDA line. Specifically, Equation 5.1 and Equantion 5.2 to compute the static and dynamic energy, respectively are used:

$$E_{st} = \frac{V_{dd}^2}{R_p f} \tag{5.1}$$

$$E_{dy} = V_{dd}^2 \cdot C_b \tag{5.2}$$

Where V_{dd} represents the supply voltage, R_p denotes the pull-up resistance, C_b signifies the bus capacitance, and f indicates the transmission frequency. Equation 5.1 is applicable solely when the line is in a logic-low state, whereas Equation 5.2 holds true for each level transition. The total energy is then straightforwardly calculated by summing the static and dynamic components.

Approximate Computing Maximum-one Encoding

A new lossy serial encoding is proposed in this work, specifically targeting error-tolerant peripherals like sensors utilizing I2C connections. The approach, termed ACME, draws inspiration from both the standard DE and its approximate counterpart ADE [164]. ACME aims to diminish total energy consumption on the bus, thereby reducing dynamic energy, while concurrently augmenting the transmission of logic-1s to curtail static energy, aligning with the I2C power model described by Equation 5.1 and Equation 5.2. The encoding algorithm of ACME exclusively operates on data words, omitting segments of the I2C protocol that cannot be approximated (such as address bits). It can be succinctly summarized by the following two

equations:

$$b_{ax,i}[t] = \begin{cases} 0 \text{ if } i < l \\ b_i[t] \text{ otherwise} \end{cases} \quad \forall i \in [0, n-1] \tag{5.3}$$

$$B_i[t] = b_{ax,i}[t] \odot b_{ax,i}[t-1], \ \forall i \in [0, n-1] \tag{5.4}$$

In the equations provided, b represents the input word, b_{ax} stands for an intermediate, approximate version of b, B denotes the final codeword, and n represents the word length. Additionally, $\odot$ denotes the binary XNOR operator, i indicates the i-th bit of a word, and t signifies the t-th word in the temporal sequence transmitted on the bus. Lastly, l represents the number of approximated Least Significant Bits (LSBs), serving as the primary parameter in ACME to balance the approximation error and the energy reduction on the bus.

In [161], codewords are derived as the bitwise difference between consecutive words. This strategy is driven by the observation that when two successive inputs exhibit similar binary values, their bitwise difference results in a sequence of 0s in the Most Significant Bits (MSBs), thereby avoiding intra-word transitions once serialized. ACME draws inspiration from this straightforward yet effective concept but substitutes the bitwise difference (XOR) with an equality operation (XNOR) (see Equation 5.4). Consequently, identical MSBs in consecutive words yield codewords featuring extended sequences of 1s (as opposed to 0s), thereby reducing static energy on an I2C connection. Furthermore, as depicted in 5.3, before codeword generation, ACME approximates each word by truncating its l-Least Significant Bits (LSBs) to 0 (this constitutes the lossy aspect of the encoding). Such truncation introduces a bias in the approximation but yields the beneficial outcome of ensuring identical LSBs across all words. Subsequent XNOR operations and serialization result in another sequence of constant 1s, further diminishing both static and dynamic energy.

The decoding scheme is shown as Equation 5.5, to construct the input after approximation.

$$b_{ax,i}[t] = B_i[t] \odot b_{ax,i}[t-1], \ \forall i \in [0, n-1] \tag{5.5}$$

The encoding/decoding scheme described here exhibits linear complexity $O(n)$, and is amenable to a low-overhead implementation in either SW or HW. Additionally, the number of approximated Least Significant Bits (LSBs), denoted as l, is adjustable,

enabling real-time adjustment of the trade-off between approximation and energy conservation. With l approximated bits, the maximum encoding error of ACME is given by $E_{max} = 2^l - 1$, and assuming a uniform distribution of LSBs [271], the expected error is $E_{max}/2$.

Hardware Implementation

The implementation of ACME encoding and decoding can be achieved effortlessly using simple and cost-effective digital hardware circuits. The schematic for an N-bit implementation is illustrated in Fig. 5.4.

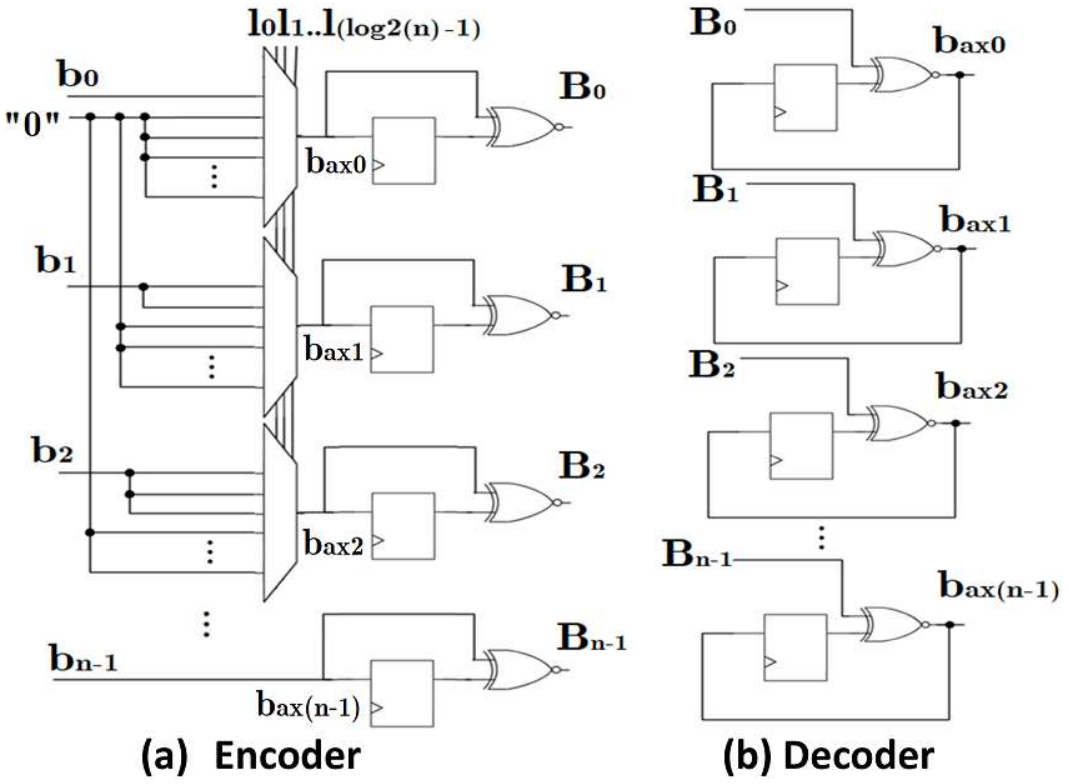


Fig. 5.4 Hardware implementation of N-bit ACME.

The encoder (Fig. 5.4a) executes the floor operation via an array of multiplexers. The selection signal for these multiplexers is l, representing the number of approximated LSBs, encoded on $log_2(n)$ wires, where n denotes the word length. Each multiplexer forwards a constant "0" to the subsequent encoder stage for every approximated bit, replacing the corresponding input word bit b_i. For instance, when $l_i = 0,\ \forall i \in [0,\ log_2(n) - 1],\ l_0 = 1$, corresponding to l being value-1 for approxi-

mated LSBs, b_0 is substituted with 0, while b_i, $\forall i \in [0, n-1]$, are selected. Following saturation, an array of n XNOR gates is employed to execute the bitwise equality operation between the current word and the previous one, stored in a corresponding set of flip-flops.

The decoder (Fig. 5.4b) consists of just one flip-flop and one XNOR gate for each bit. The XNOR gates perform a bitwise equality comparison between corresponding bits of the current codeword and the previously decoded word, implementing Equation 5.5. For example, if $B_i[t] = 0$, the negated value of $b_{ax,i}[t-1]$ is generated at the output as $b_{ax,i}[t]$, otherwise the affirmed $b_{ax,i}[t-1]$ is copied. These hardware implementations of the encoder and decoder achieve a throughput of 1 encoded/decoded word per cycle, with a single clock cycle latency.

Experimental Results

To assess the performance of ACME on the applications involving smart sensors, a variety of datasets containing common sensor data are utilized to emulate the data exchanged with diverse error-resilient smart sensors via I2C connections. In particular, the experimental results show the evaluation of two data types: accelerometer samples and audio data [272, 273].

A SPICE model of the I2C bus calculates the energy consumption associated with various encoding schemes. Initially, the appropriate I2C operating mode for each input was selected, based on the required throughput: Standard-Mode is employed for Accelerometer data, while Fast-Mode is chosen for Image and Audio data. Subsequently, we set the bus frequency f to the maximum allowed by each operating mode, specifically 100kHz and 400kHz, respectively. It is worth noting that although a lower bus frequency might suffice for certain inputs, opting for the maximum frequency notably enhances the dynamic/static energy consumption ratio. Given that ACME's primary advantage over comparison baselines lies in its superior static energy reduction, this configuration represents a worst-case scenario for the algorithm, with potential savings potentially increasing at lower frequencies.

Refer to [274], the bus voltage was set to $V_{CC} = 3.3V$ and the bus capacitance to $C_b = 150pF$. Subsequently, R_p was derived from C_b using the plots provided in the I2C specification [269], obtaining $R_p = 7.8k\Omega$ and $R_p = 2.3k\Omega$ for Standard- and Fast-Mode, respectively.

Initially, all encodings are implemented and simulated in Python. Subsequently, to assess the area and power of a hardware implementation, the ACME encoder and decoder were synthesized from an RTL description in VHDL, targeting a 45nm CMOS library from STMicroelectronics. Synthesis is conducted using Synopsys Design Compiler and verified using Mentor Questa Simulator.

Fig. 5.5 depicts the Pareto curves generated by ACME and the three comparison baselines (ADE [271], Serial T0 [168], AXSERBUS [159]) in error vs. energy space for one Standard-mode (f = 100kHz, R_p = 7.8kω) and one Fast-mode (f = 400kHz, R_p = 2.3kω) dataset, namely Accelerometer and Audio data, respectively. The x-axes represent the average absolute error achieved by each method after encoding and decoding, normalized to a percentage of the full-scale value of each dataset1. Different points are obtained by varying the approximation parameters of the encodings. For ACME, the number of floored LSBs, denoted as l, is varied. All other encodings also feature a single approximation parameter, as described in the respective papers, except for Axserbus [159], which employs 2 parameters (d_0 and d_m). For Axserbus, a grid search is conducted over d_0 and d_m values with a step of powers of 2 (while ensuring d_m is always larger than d_0) to extract the Pareto fronts.

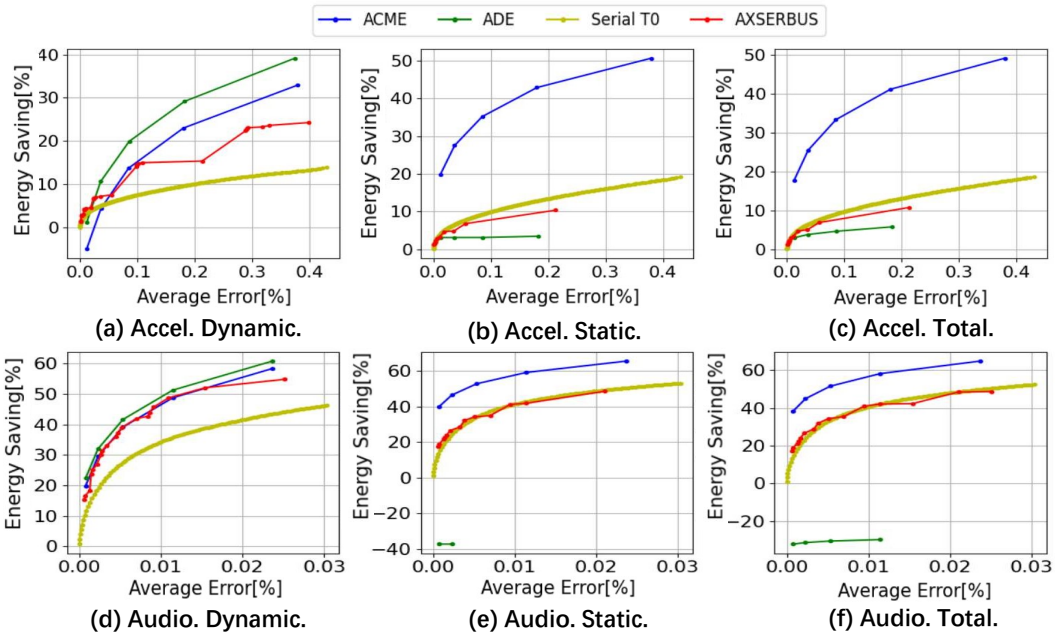


Fig. 5.5 The Pareto curves in terms of average error vs. energy saving obtained by ACME and comparison baselines on accelerometer and audio samples.

The y-axes depict the dynamic, static, and total energy savings concerning the transmission of raw (unencoded) data over the same I2C bus for each encoding and approximation parameter setting. Although ACME competes favorably in terms of dynamic energy reduction, the results indicate that it is slightly surpassed by ADE on both datasets. This outcome is attributed to the lower average error and fewer inter-word transitions achieved by ADE, which are attributable to its employment of LSB rounding rather than flooring.

In contrast, ACME outperforms all other encodings (especially ADE) in terms of static energy reduction, achieving up to 30% (for accelerometer data) and 10% (for audio data) higher savings for the same error level. This superiority stems from its explicit attempt to increase the number of transmitted logic-1s when permitting more error. Conversely, Serial-T0 and Axserbus achieve a similar increase, but solely due to a "secondary effect" of their more frequent utilization of the all-1 special pattern [168, 159]. Conversely, ADE actually transmits fewer logic-1s for larger errors, resulting in the generation of Pareto-dominated points (hence not shown in the plots) with higher static energy when increasing the amount of approximation. The total energy saving graphs closely resemble the static ones, highlighting the dominance of the static component in I2C buses. Consequently, ACME emerges as superior to its comparison baselines for these connections. In particular, by setting the average error constraint as 0.1%, ACME achieves a 54.65% average total energy reduction on these datasets with different data types.

Table 5.2 presents the synthesis outcomes of the ACME encoder and decoder depicted in Fig. 5.4 on a 45nm CMOS technology. Both circuits are synthesized with the aim of achieving the same clock frequency as the encoder/decoder from [271] for direct comparison, even though this frequency does not correspond to an operating frequency for I2C. Additionally, the bit-width is set to 12 bits. Power results encompass both leakage and dynamic consumption and are provided both at the targeted frequency and at 100kHz (i.e., the maximum frequency of I2C Standard mode), enabling comparison with ADE.

The findings indicate that while ACME significantly outperforms in reducing static energy on the bus, it also yields a hardware implementation with similar area and power requirements. Moreover, for practical I2C frequencies, the power overheads associated with the encoding/decoding circuitry are entirely negligible compared to the bus consumption.

Table 5.2 Comparison of synthesis results for the 12-bit encoder and decoder of ACME and ADE

Circuit	f [GHz]	Enc.	Area $[\mu m^2]$	Pow.@f $[mW]$	Pow.@100kHz $[mW]$
Encoder	1.67	ACME	130.18	0.27	0.001045
		ADE	170.40	0.33	-
Decoder	2.50	ACME	105.84	0.61	0.00084
		ADE	101.61	0.60	-

5.4 Conclusion and Discussion

Hardware-level optimizations of the entire smart sensor systems focused on both the Processing Unit and Communication are considered, aiming to achieve optimal energy-efficient smart sensors.

Thanks to the customized core with ISA extension, the proposed smart IR sensor prototype enables SIMD operations to significantly speed up the inference of 4-bit and 8-bit quantized NNs, meanwhile, achieving up to 15.38× energy reduction at iso-accuracy.

Furthermore, focusing on I2C, one of the most common serial bus protocols involved in smart sensor systems, a novel approximate serial bus ACME encoding specifically tailored for I2C buses was proposed. ACME offers a runtime-controllable energy saving vs error trade-off and simultaneously reduces both dynamic and static energy by eliminating value transitions and increasing the number of transmitted logic-1s. Despite its simplicity, this mechanism achieves superior total energy savings compared to those of state-of-the-art approximate encodings, with negligible hardware implementation overheads.

Software-hardware co-design in IR-based applications. Considering the applications in reality, the proposed hardware optimizations can collaborate with algorithm-level design, to maximize energy optimization in a smart sensor system. In fact, the MAUPITI IR sensor system containing the customized RISC-V core for low-precision DNN inference in Section 5.2 can efficiently support quantized DL models described in Chapter 4, targeting people counting and social distancing task. In Table 5.1, the DL models used in the evaluation of the proposed sensor prototype started with the architectures proposed in

Chapter 4, and it is clear that the proposed hardware platform achieves less energy consumption compared to running on commercial STM32 MCUs.

On the other hand, it is worth noting that these DL-enabled tasks based on IR sensors are error-resilient. For example, the slight changes in pixels of hot profiles of objects in one frame will not impact the final counts of present people. Therefore, the I2C bus in the systems can be combined with the approximate bus encoding hardware, to achieve trade-offs between energy consumption and errors when using I2C communication.

To prove the effectiveness of combining ACME approximate I2C bus encoding with the previous works, an additional experiment was conducted on the LINAIGE dataset, which is the proposed dataset used for IR-based applications targeting social distancing and people counting. In particular, the I2C bus was set to Fast mode as introduced in Section 5.3.2. To close to practical scenarios, 31036 IR frames with both "hard-to-label" and "non-hard-to-label" from the LINAIGE dataset are used to represent the data transmission from/to the IR sensor by I2C. In particular, each pixel is converted into a 12-bit two's complement, based on the precision of the sensor device [172] used in data collection. The average error was calculated by all frames under the specific l approximated LSBs. By setting average error constraints as 0.05% compared to a fully-accurate setting, the results show that ACME achieves total energy saving (including both dynamic energy saving and static energy saving) as 53%. Therefore, when combining with ACME approximate bus with IR smart sensors, the gain of extra energy saving could be obtained when data communication is needed.

Impact of AxC on reliability. As a promising paradigm in energy-efficient systems, approximate computing may impact the reliability of the sensor systems in multiple aspects. Since the design of the target sensor systems often relies on error-tolerant applications, approximate computing may potentially improve overall reliability in certain contexts by making the system more robust to minor faults and transient errors. For instance, after approximate encoding and decoding with appropriate level settings, the IR frames may have some image quality drops. However, and the same time, the noise may be removed to some degree as well, which affects final prediction results in positive way. On the other hand, sensor systems can not only benefit from

energy efficiency achieved by approximate computing but also less thermal stress and a potentially extended lifespan.

Obviously, the negative impacts of approximate computing can be considered on their intentional inaccuracies in computing, especially when these inaccuracies accumulate to lead to significant errors. This drawback becomes critical in applications requiring high accuracy and service qualities. Moreover, in complex systems where intermediate results are used for further calculations, errors introduced by approximate computing may propagate to lead task fails. Therefore, the users have to configure the approximation levels carefully based on the specific application requirements and may introduce additional error detection and correction schemes to catch significant deviations in the target systems, to mitigate the negative impacts of approximate computing.

Chapter 6

Electronic Design Automation for Smart Sensors

This chapter introduced a graph-based topology recognition approach in automation placement flow for AMS ICs. Based on a deterministic algorithm namely VF2 for subgraph matching, the pre-defined topologies can be recognized in the target netlists. In particular, based on real-world industrial netlists, this work introduced several post-processing steps, further improving the precision of topology recognition. This work is from the AMBEATion project [113], and preliminary results are shown and discussed in this chapter.

6.1 Motivation

As smart sensors become increasingly complex, integrating multiple distinct functionality blocks including data acquisition, data processing, and the communication interface, a complex design of heterogeneous assembly of components is required. AMS ICs play a pivotal role in this integration, as they intricately interweave the Analog and Digital domains, offering a platform for closed collaboration among various system elements.

EDA tools help manage such complexity by providing design automation solutions for different stages of the smart sensor design process, significantly reducing time to market, especially considering the need for iterative design cycles and rigor-

ous testing. On the other hand, such IC designs are typically highly dependent on human experts. While the complexity of design has been increasing, it become more and more difficult to perform complete design space exploration to achieve optimization across multiple dimensions such as power consumption, performance, resource occupation, etc. EDA tools enable designers to explore the design space efficiently, evaluating different design configurations to explore Pareto-optimal solutions in terms of a set of trade-offs.

However, developing robust EDA tools for AMS ICs to generate ideal results is not easy. In fact, for a given AMS IC design, existing EDA tools have achieved high-quality digital design automation, meanwhile, transitioning from standardized design descriptions to the physical design strategies for analog components still heavily rely on human expertise. Real-world AMS systems encompass a diverse array of circuit designs and parameters, even for the same functionality, which are heavily influenced by individual designer expertise and preferences.

6.2 Analog Mixed-Signal Topology Recognition in a Real-World Industrial Flow

To address the limitation that exists in the EDA tools for AMS ICs, robust topology recognition is crucial within a generalizable AMS layout automation framework. This involves identifying the main elementary functional topologies within the schematic, such as transistors and capacitors, to appropriately assign constraints to the layout placer in subsequent modules. These topologies can be categorized into analog topologies (e.g., current mirrors, and differential pairs) and digital functions (e.g., inverters, NOR gates).

However, the topologies within AMS circuits, especially analog ones, often manifest themselves in a multitude of variants tailored by different designers. For instance, a cascade current mirror may vary in transistor sizing and the number of stages to customize performance. Additionally, designers may employ unique approaches by adjusting biasing networks or compensation techniques to achieve desired performance characteristics. Given the multifaceted nature and complexity of design strategies used in large AMS circuits, implementing accurate autonomous topology recognition for all potential variants poses a significant challenge.

6.3 The State-of-the-art Topology Recognition Solutions

The state-of-the-art topology recognition in the literature can be divided into several categories based on applied strategies:

Library-based. [275–277] perform topology recognition based on pre-defined templates, and iterate through template matching in a large dataset. This method suffers the limitation of recognition of new topology variants that do not exist in the dataset;

Rule-based. [278–281] apply pre-defined rules to recognize circuits. However, the extracted rules are highly dependent on human designer experts, and similarly, it is hard to generalize to all variants, especially novel topology variants that never appeared in previous designs;

ML-based. [282] utilized unsupervised learning of new block connections, but inherits the limitations of rule-based solutions while using specific rules. [283] applied a CNN-based approach to convert a matrix representation of a circuit into a collection of templates. More interesting ML/DL-based solutions are *graph*-based methods. [284] converted flattened netlists into bipartite graphs and applied GCN to recognize IP blocks as subgraphs. [285] presented a graph-based method for AMS circuit structure recognition in IP schematics. In particular, Graph ATtention networks (GATs) were applied to cluster analog circuit blocks. The similarities between the results and the industrial design at the same layer were calculated, and adaptations were proposed to modify the difference.

However, previous studies focus on small-scale structures containing fewer than 50 transistors (only [284] considered netlists with up to 500 transistors, but still far away from real-world scenarios), encountering difficulties when attempting to scale to larger netlists.

6.4 Proposed Method

The goal of topology recognition in AMS automation is to identify functional topologies present in the schematic, such as current mirrors and differential pairs. This identification process is crucial to accurately define constraints during the layout placement phase. Topology recognition involves solving a subgraph isomorphism problem between a template graph (e.g., a current mirror) and the target graph (e.g., an operational amplifier).

6.4.1 Graph Representation and Subgraph Isomorphism

The representation of an element-level circuit is accomplished through an undirected bipartite graph denoted as $G(V,E)$, wherein: V denotes two sets of vertices, signifying elements (devices such as resistors, etc.) and nets (such as VSS, etc.), respectively; E indicates edges connecting vertices in the graph, representing each connection between the terminals of elements and nets. Hence, both the netlist of the target circuit and the primitive templates can be transformed into bipartite graphs.

During the process of topology recognition, aided by subgraph isomorphism algorithms, each pattern graph (corresponding to each primitive template) within a library is matched against the target graph (pertaining to the target circuit).

One instance of subgraph matching between a pattern graph is depicted in Fig. 6.1 to accomplish topology recognition. In this depiction, an NMOS current mirror primitive (a) is represented by a pattern graph, where vertices on the left side represent devices M0 and M1 respectively and those on the right side represent nets D1, D2, and S. Similarly, a fully differential Telescopic operational transconductance amplifier (OTA) (b) is converted into a bipartite graph and one subgraph is matched the current mirror pattern graph, highlighted by red.

6.4.2 VF2-based Topology Recognition Approach

The matching of templates is achieved through the utilization of a deterministic algorithm known as *VF2*. This algorithm operates by seeking an isomorphism mapping between two graphs, $G1$ and $G2$, leveraging states generated throughout the execution of the algorithm. During each iteration, the algorithm identifies pairs

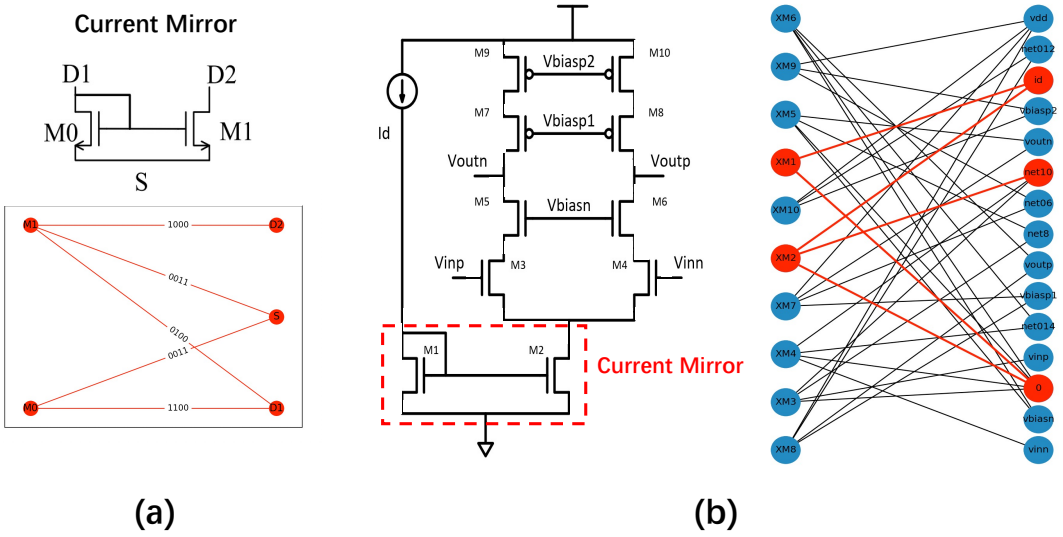


Fig. 6.1 Topology recognition example on one current mirror.

of vertices as potential candidates for the current state and evaluates whether to incorporate them into the mapping before advancing to the subsequent state. This mapping process continues recursively until a complete mapping is established.

In terms of implementation, the recognition process employing VF2 is realized in Python utilizing the *networkx* library.

To enhance accuracy, several novel postprocessing steps are implemented to filter matched results effectively. These steps include:

- The relationship between the length and width of instances can be defined in differential pairs (DPs). Topologies that do not conform to the specified length and width constraints are eliminated.
- Recognition of current mirrors (CMs) with an uncertain number of branches is conducted to address the limitation of having a few templates in the library, thereby enhancing the generalization of matching.
- Differential pairs (DPs) with a specific number of instances forming parallel structures are identified.

6.5 Experimental Results

The validation of topology recognition results involves the utilization of human-labeled netlists, where schematic devices are annotated manually, serving as golden values to assess the quality of the outcomes. The comprehensive validation results for seven distinct netlists, leveraging analog templates, are presented in Table. 6.1 Additionally, the table includes information on the execution time and memory usage of this component.

Table 6.1 Topology recognition validation results based on analog templates.

Netlist	Templates	Precision	Recall	Execution Time [s]	Memory [MB]
AGNDBuffer	CM	100%	100%	0.16	91
	DP	50%	100%		
	Tot.	75%	100%		
BOOST_ISENSE	CM	100%	100%	0.03	95
	DP	n.a.	0		
	Tot.	100%	75%		
DAC_TRIM	CM	100%	100%	25.42	91
	DP	n.a.	0		
	Tot.	100%	50%		
INswitch_bias	CascLV	100%	100%	0.21	92
	Tot.	100%	100%		
OPA_FAST_IN	CascLV	100%	100%	0.1	91
	CM	100%	100%		
	DP	100%	100%		
	Tot.	100%	100%		
INswitch_BUFF_1mA	CascLV	100%	100%	0.24	90
	DP	100%	100%		
	Tot.	100%	100%		
OPA_FAST_GO2	CascLV	100%	100%	0.31	90
	DP	100%	100%		
	Tot.	100%	100%		

Regarding the quality of results, Table. 6.1 presents metrics including precision and recall for the entire circuit, as well as for various types of topologies such as CMs, DPs, and cascaded current mirrors (CascLV).

Across the seven target netlists, topology recognition demonstrates high precision on average, reaching 100% for all circuits except one, where it attains 75%. Precision

remains high across different types of topology templates, indicating the algorithm's ability to accurately recognize topologies and mitigate false hits. Moreover, the recall metric exhibits promising results, predominantly reaching 100%, with only two exceptions. This demonstrates that the algorithm generally identifies all anticipated topologies, minimizing misses.

Specifically addressing the recognition of individual templates, the algorithm excels in CM and CascLV templates, whereas results are slightly worse for DP templates. It is noteworthy that the table exclusively displays templates identified by the algorithm or labeled by humans, and templates that do not appear in the netlist are omitted.

In terms of extra-functional metrics, the algorithm's execution time and memory consumption have been evaluated for all netlists, with results presented in the rightmost two columns of the table. The execution time consistently remains under 1 minute. However, comparing the results for DAC_TRIM (the largest netlist in the set) with the others shows a scalability issue with the algorithm's execution time concerning the number of devices. This will be a significant concern when transitioning to larger circuits. Importantly, it is evident that the algorithm necessitates significantly less time compared to manually labeling all topologies, although the latter is not easily quantifiable. Memory usage remains below 100MB, comfortably compatible with a single modern professional workstation.

6.6 Conclusion and Discussion

Advanced EDA tools can significantly reduce the design period for smart sensors, however, the placement automation of analog components involved in AMS ICs in smart sensor systems is facing challenges to obtain robust results. The graph-based solutions are one of the promising approaches to achieve topology recognition in schematics aiming to correctly place the recognized topologies into the layout, respecting the necessary constraints.

Integration into the industrial design flow. The final goal of the proposed topology recognition method is to achieve full integration within industrial design workflows through collaboration with leading EDA tool providers. Some considerations have already been implemented to achieve this goal. The first

consideration is the various industrial standards for schematic file formats, and Circuit Design Language (CDL) netlists are supported. Another consideration is that the proposed topology recognition supports user-defined template libraries, and user can add their templates easily and select the specific templates for the recognition step. In addition, it can be accessed both with a Command Line Interface (CLI) and with a simple Graphical User Interface (GUI). These user-friendly features further make it ready to integrate into industrial design flow.

Limitations and Future Works. The main limitation of the VF2 implementation lies in its complexity, implying a very long running time on large flattened netlists. In fact, it works well when applied independently and possibly in parallel to hierarchical blocks of the netlist, but scales poorly on large flattened netlists. However, in this way, topologies across hierarchies cannot be recognized and it decreases the quality of recognition results. In addition, the VF2 algorithm's template library for topology recognition had to be entirely rewritten once moved to a new technology, and new post-processing rules were necessary to avoid false positive matches. Therefore, the ML-based approach or combination of both may become a promising alternative to solve these problems to some degree. On the one hand, ML-based solutions can achieve fast inference to obtain results, even on large flattened netlists; On the other hand, ML-based solutions can use transfer learning to mitigate similar issues as in the VF2-based solution when moving from one technology to another.

Chapter 7

Conclusion and Future Work

Smart and intelligent sensors have become a promising paradigm as the pivotal link between the physical world and the Information Communication Technology (ICT) frameworks in the modern world. As sensor technologies continue to integrate with an expanding array of "smart" modules and achieve the benefits of increasingly advanced smart capabilities, the enlarged scale and increasingly complex systems have also posed new challenges for smart sensor design.

In this thesis, a comprehensive study of smart sensor designs has been conducted, encompassing the entire research process covering each step of the above design flow and its integration with practical applications in smart sensor design. The techniques introduced in this thesis are divided into three abstract levels: algorithm-, hardware-, and EDA-level. Focusing on these three pillars, several optimization techniques are introduced, detailed as the following:

Algorithm-level design for DL in smart sensors.

- **Challenge and motivation.** ML/DL has become a new paradigm in recent years and has begun a new era of artificial intelligence, smart and intelligent sensors have extended their conception by integrating with advanced ML-enabled components. However, there exists a gap between huge ML/DL algorithms which are designed to execute on high-performance platforms, and resource-limited and energy-sensitive sensor nodes. Edge sensor nodes are typically based on battery-powered platforms, and equipped with resource-limited MCUs. It is challenging

the effective deployment of ML/DL-based solutions on the sensor in terms of inference latency, energy consumption, and memory occupation. Moreover, different applications and sensor types may not suit all ML/DL algorithms optimally. For a chosen model, extensive exploration of architecture configuration and quantization formats is crucial to balance task performance with cost. Specifically, the required memory usage, latency, and energy consumption of ML/DL solutions often conflict, necessitating trade-offs based on the precise application scenario.

- **Proposed methods.** To overcome the limitations, several algorithm-level implementations and optimizations targeting specific sensor-based applications were proposed and discussed in this thesis. In particular, the compact architecture of CNNs and other DL families was introduced, and a complete comparative analysis targeting social distancing and people counting on ultra-low-resolution IR arrays was provided based on extensive architecture explorations. In addition, a compact RF implementation was introduced for human activity recognition targeting MCUs. In particular, the RF is represented by multiple C arrays, used instead of lists for better memory locality and simpler control flow. The main data structure representing the model is an array that contains the description of all nodes in the RF, and its elements are C "structs", where optimizations such as implicit left child were applied.

  To further reduce inference energy consumption, adaptive systems were implemented to adjust computation complexity based on input difficulty at runtime. Particularly, a cascaded system consisting of a threshold-based wake-up trigger and an 8-bit quantized CNN was implemented for social distancing tasks on IR sensors, and another three-staged inference system containing a lightweight RF and two tiny transformer models was introduced for gesture recognition tasks based on sEMG sensors. Based on these adaptive systems, original tasks were split into several stages, and the following stages would be invoked only when the previous stage equipped with simple models failed. This automatic adaptive inference focuses on "easy" tasks that occupy the most practical scenarios, avoiding energy consumption from models that are overly complex for the tasks. Importantly, these systems can provide a wide range of trade-offs

by configurable thresholds, etc., enhancing the flexibility of DL-based solutions on smart sensors.

Importantly, optimization techniques such as quantization were applied in all proposed ML/DL implementations to minimize the memory footprint. All the models were deployed and evaluated on the commercial and academic MCUs, to demonstrate their effectiveness on MCU-based edge sensor nodes.

- **Relevant resutls.** The experimental results of the proposed DL algorithms have demonstrated their effectiveness and energy efficiency in their specific applications. In particular, based on low-resolution IR arrays, for social distancing, the proposed efficient CNNs achieve 86.3% best balanced accuracy, significantly outperforming a state-of-the-art deterministic algorithm and a rich Pareto set of models obtained by extensive architecture exploration. The models were then quantized to 8 bits for optimization and deployed on an MCU proposed by ST, showing the proposed efficient CNNs can be implemented directly on MCU-based smart sensor nodes. The deployment results of the proposed adaptive system for the same task show that energy consumption can be reduced by 37-57% concerning a static CNN-based approach, with an accuracy drop of less than 2%. For the people counting task, with a wide architectural exploration of each model type, a rich set of Pareto-optimal solutions was obtained, spanning a wide range of accuracy scores, and providing an extensive cost evaluation of deployment on a commercial MCU.

  Based on sEMG sensors, based on tiny transformers achieved state-of-the-art accuracy on this task (67.04%), the proposed dynamic inference system achieved a 1.03×–1.35× energy reduction compared to standalone transformers. Based on acceleration sensors, experiments were performed with two RF variants on two datasets, public and proprietary respectively, by varying the number of trees and maximum depth. Pareto curves were generated to illustrate the trade-off between accuracy and cycles/memory, offering a range of Pareto optimal solutions for users.

Hardware-level optimization for DL-enabled sensors.

- **Challenge and motivation.** Hardware-level optimization for ML/DL-enabled smart sensors is driven by the need to efficiently handle the

conflict between the computational demands of DL-based algorithms and the resource-constrained environments in edge sensor platforms. It aims to enhance the high accuracy performance achieved by DL-based approaches while addressing key challenges such as limited processing power, memory constraints, and energy efficiency concerns. There exist several rooms to efficiently execute operations in DL models to achieve HW-SW co-design to implement optimized edge smart sensors. In this work, the low-precision formats in quantized DL models were considered and combined with SIMD features introduced in a RISC-V architecture. On the other hand, a large component of the energy budget involving sensors is communication. Therefore, energy-efficient bus communication has become an interesting topic to further contribute to efficient sensors. In general, optimized hardware not only improves the execution speed and responsiveness of DL algorithms but also extends battery life and reduces operational costs. However, achieving these optimizations requires careful balancing of computational complexity and area overheads.

- **Proposed methods.** Focusing on the discussed challenges and motivation of hardware-level design for smart sensors, a customized RISC-V core was introduced, and an approximate bus encoding mechanism was proposed. The former design aims at a 16x16 resolution IR sensor based on a RISC-V processor, targeting applications including people counting or social distancing. To better support efficient DL approaches proposed in algorithm-level design, an optimized RISC-V ISA core was adapted to provide efficient computation for quantized models. In particular, vectorized operations at multiple-bit widths have been considered. 8-bit and 4-bit vectorized operations support was implemented by an additional HW unit and the corresponding ISA extension.

  For serial bus communication in sensors, the first approximate serial bus encoding, namely ACME, was designed specifically for I2C connections. With a straightforward encoding/decoding scheme to maximize the number of logic-1 values in codewords, while

minimizing transitions, ACME concurrently reduces both static and dynamic energy on the bus.

- **Relevant results.** Depending on the deployment results of three different DL models on three different hardware platforms including commercial one, original RISC-V core, and customized RISC-V core, the proposed sensor prototype platform achieved the lowest energy consumption for all models. In particular, compared to the original core, the most significant gains in energy efficiency were achieved by the most accurate model in the proposed MAUPITI platform by efficient operations. Despite a 7% increase in area and a 2.2% increase in power consumption, MAUPITI achieves a notable reduction of up to 17.9% in energy consumption.
  On the other hand, the proposed approximate bus encoding ACME outperformed the three state-of-the-art approximate serial bus encodings, with negligible area overheads. An average energy saving on two different datasets of over 54% can be achieved while keeping the error constraint less than 0.1%.

Topology reconigiton in industrial EDA flow.

- **Challenge and motivation.** As smart sensors evolve with integrated functionality, they demand complex assemblies of heterogeneous components. In particular, AMS ICs play a crucial role by harmonizing these domains, enabling seamless collaboration among sensor system elements. However, optimizing AMS ICs remains challenging due to the intricate balance needed across power consumption, performance, and resource usage. While EDA tools facilitate efficient exploration of design alternatives, achieving optimal results of AMS IC design requires bridging the gap between automation capabilities and the expertise needed for analog design strategies. Therefore, to generate high-quality AMS design placement automatically, topology recognition for analog and digital primitives has become a crucial step and a challenge in industrial design flow.
- **Proposed methods.** In AMS automation, topology recognition aims to detect specific functional structures in the schematic, such as current mirrors and differential pairs. In this work, a template-

library-based approach was utilized. In particular, the template netlists and target netlist were converted into bipartite graphs, and a deterministic subgraph isomorphism algorithm namely VF2 was adopted, and generated recognition results contained instance information including device names, nets, etc., for following phases in the placement flow.

- **Relevant results.** The evaluation results on seven industrial netlists showed that the proposed topology recognition exhibits high precision, achieving 100% for all circuits except one, where it reaches 75%. This consistency across various topology templates underscores the method's accuracy and ability to minimize false positives. In addition, the recall metric shows strong performance, predominantly reaching 100%, with only two exceptions, indicating the algorithm's effectiveness in identifying nearly all expected topologies and minimizing misses. Moreover, the execution time and memory occupation for the proposed solution were reported, and it shows a promising compatibility in real-world design flow.

To sum up, this thesis extensively explores optimizing smart sensors through innovative ML/DL models, enhanced hardware architectures, and improved EDA tools. The main goal is to develop energy-efficient DL/ML-enabled smart sensors, addressing technological constraints and energy demands in smart sensor applications. The studies span three areas including algorithm-level, hardware-level, and EDA field. At the algorithm level, specific edge sensor applications are provided comprehensive analysis and assessments focusing on tailored ML/DL models, filling a gap in the specific field, especially considering practical scenarios. At the hardware level, a sensor prototype was implemented, and the corresponding hardware optimizations co-designed with the optimized algorithms demonstrated their effectiveness beyond the theoretical stage and filled the gap in technology implementation. In addition, a preliminary experiment targeting topology recognition in AMS system design is trying to answer the challenges of smart sensor design in this new era.

Smart sensor systems have shown the potential to fully replace traditional sensing systems, especially by combining rapidly developed AI techniques. The design of smart sensor systems is a vast subject and involves a variety of fields, working with the same goal of efficient and intelligent sensor nodes.

There exist multiple opportunities to achieve this objective. For instance, in terms of compact smart models, instead of manual architecture searching, many studies utilize automatic Neural Architecture Search (NAS) to efficiently generate optimal architecture targeting NN-based solutions, significantly reducing the design period. Moreover, other studies take advantage of knowledge distillation techniques to generate compact models by mimicking large models.

Another example in terms of HW-level optimization is hardware accelerators designed for smart sensors. By adding specific hardware features or customizing the entire hardware platform, smart sensors targeting specific tasks can achieve maximum effectiveness and efficiency to support smart models for target tasks.

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[3] Francesco Daghero, Chen Xie, Daniele Jahier Pagliari, Alessio Burrello, Marco Castellano, Luca Gandolfi, Andrea Calimera, Enrico Macii, and Massimo Poncino. Ultra-compact binary neural networks for human activity recognition on risc-v processors. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, pages 3–11, 2021.

[4] Francesca Palermo, Matteo Cognolato, Arjan Gijsberts, Henning Muller, Barbara Caputo, and Manfredo Atzori. Repeatability of grasp recognition for robotic hand prosthesis control based on sEMG data. In *2017 International Conference on Rehabilitation Robotics (ICORR)*, pages 1154–1159. IEEE, jul 2017.

[5] Alessio Burrello, Francesco Bianco Morghet, Moritz Scherer, Simone Benatti, Luca Benini, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Bioformers: Embedding transformers for ultra-low power semg-based gesture recognition. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1443–1448. IEEE, 2022.

[6] Matteo Risso, Chen Xie, Francesco Daghero, Alessio Burrello, Seyedmorteza Mollaei, Marco Castellano, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Hw-sw optimization of dnns for privacy-preserving people counting on low-resolution infrared arrays. *arXiv preprint arXiv:2402.01226*, 2024.

[7] Nam Ha, Kai Xu, Guanghui Ren, Arnan Mitchell, and Jian Zhen Ou. Machine Learning-Enabled Smart Sensor Systems. *Advanced Intelligent Systems*, 2(9), 2020.

[8] Xing Liu and Orlando Baiocchi. A comparison of the definitions for smart sensors, smart objects and Things in IoT. *7th IEEE Annual Information Tech-*

nology, Electronics and Mobile Communication Conference, IEEE IEMCON 2016, pages 1–4, 2016.

[9] IEEE Standards Association et al. The ieee 1451.4 standard for smart transducers. *http://standards. ieee. org/develop/regauth/tut/1451d4. pdf*.

[10] Sergey Y Yurish. Sensors: smart vs. intelligent. *Sensors & transducers*, 114(3):I, 2010.

[11] Zachary Ballard, Calvin Brown, Asad M Madni, and Aydogan Ozcan. Machine learning and computation-enabled intelligent sensor design. *Nature Machine Intelligence*, 3(7):556–565, 2021.

[12] Chen Xie, Daniele Jahier Pagliari, and Andrea Calimera. Energy-efficient and privacy-aware social distance monitoring with low-resolution infrared sensors and adaptive inference. In *2022 17th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*, pages 181–184. IEEE, 2022.

[13] Chen Xie, Francesco Daghero, Yukai Chen, Marco Castellano, Luca Gandolfi, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Privacy-preserving social distance monitoring on microcontrollers with low-resolution infrared sensors and cnns. In *Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISCAS 2022. IEEE, 2022.

[14] Chen Xie, Francesco Daghero, Yukai Chen, Marco Castellano, Luca Gandolfi, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Efficient deep learning models for privacy-preserving people counting on low-resolution infrared arrays. *IEEE IoT J.*, 2023.

[15] Chen Xie, Alessio Burrello, Francesco Daghero, Luca Benini, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Reducing the energy consumption of semg-based gesture recognition at the edge using transformers and dynamic inference. *Sensors*, 23(4):2065, 2023.

[16] Francesco Daghero, Alessio Burrello, Chen Xie, Luca Benini, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Adaptive random forests for energy-efficient inference on microcontrollers. In *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2021.

[17] Amira S Ashour, Amira El-Attar, Nilanjan Dey, Hatem Abd El-Kader, and Mostafa M Abd El-Naby. Long short term memory based patient-dependent model for fog detection in parkinson's disease. *Pattern recognition letters*, 131:23–29, 2020.

[18] Murtadha D Hssayeni, Joohi Jimenez-Shahed, Michelle A Burack, and Behnaz Ghoraani. Wearable sensors for estimation of parkinsonian tremor severity during free body movements. *Sensors*, 19(19):4215, 2019.

[19] Taylor R Mauldin, Marc E Canby, Vangelis Metsis, Anne HH Ngu, and Coralys Cubero Rivera. Smartfall: A smartwatch-based fall detection system using deep learning. *Sensors*, 18(10):3363, 2018.

[20] Zaher Al Barakeh, Philippe Breuil, Nathalie Redon, Christophe Pijolat, Nadine Locoge, and Jean-Paul Viricelle. Development of a normalized multisensors system for low cost on-line atmospheric pollution detection. *Sensors and Actuators B: Chemical*, 241:1235–1243, 2017.

[21] Gongbo Chen, Shanshan Li, Luke D Knibbs, Nicholas AS Hamm, Wei Cao, Tiantian Li, Jianping Guo, Hongyan Ren, Michael J Abramson, and Yuming Guo. A machine learning method to estimate pm2. 5 concentrations across china with remote sensing, meteorological and land use information. *Science of the Total Environment*, 636:52–60, 2018.

[22] Daniel Popa, Syed Zeeshan Ali, Richard Hopper, Ying Dai, and Florin Udrea. Smart cmos mid-infrared sensor array. *Optics letters*, 44(17):4111–4114, 2019.

[23] Veena Chidurala and Xinrong Li. Occupancy estimation using thermal imaging sensors and machine learning algorithms. *IEEE Sensors Journal*, 21(6):8627–8638, 2021.

[24] Shota Mashiyama, Jihoon Hong, and Tomoaki Ohtsuki. Activity recognition using low resolution infrared array sensor. In *2015 IEEE International Conference on Communications (ICC)*, pages 495–500. IEEE, 2015.

[25] Rui Zhu, Xiaoling Ji, Danyang Yu, Zhiyuan Tan, Liang Zhao, Jiajia Li, and Xiufeng Xia. Knn-based approximate outlier detection algorithm over iot streaming data. *IEEE Access*, 8:42749–42759, 2020.

[26] Tarik Saidi, Omar Zaim, Mohammed Moufid, Nezha El Bari, Radu Ionescu, and Benachir Bouchikhi. Exhaled breath analysis using electronic nose and gas chromatography–mass spectrometry for non-invasive diagnosis of chronic kidney disease, diabetes mellitus and healthy subjects. *Sensors and actuators B: chemical*, 257:178–188, 2018.

[27] Zhixin Liu, Ming Yang, Yazhou Yuan, and Kit Yan Chan. Fall detection and personnel tracking system using infrared array sensors. *IEEE Sensors Journal*, 20(16):9558–9566, 2020.

[28] Paul Brinkman, Niki Fens, Marianne A van de Pol, Peter J Sterk, and Anirban Sinha. Identification of exhaled volatile organic compounds (vocs) associated with loss of asthma control, 2016.

[29] Shui Jiang, Jun Wang, Yongwei Wang, and Shaoming Cheng. A novel framework for analyzing mos e-nose data based on voting theory: Application to evaluate the internal quality of chinese pecans. *Sensors and Actuators B: Chemical*, 242:511–521, 2017.

[30] Dedy Rahman Wijaya, Riyanarto Sarno, and Enny Zulaika. Noise filtering framework for electronic nose signals: An application for beef quality monitoring. *Computers and Electronics in Agriculture*, 157:305–321, 2019.

[31] Massimo F Rutolo, Daciana Iliescu, John P Clarkson, and James A Covington. Early identification of potato storage disease using an array of metal-oxide based gas sensors. *Postharvest Biology and Technology*, 116:50–58, 2016.

[32] Mojtaba Tohidi, Mahdi Ghasemi-Varnamkhasti, Vahid Ghafarinia, Mojtaba Bonyadian, and Seyed Saeid Mohtasebi. Development of a metal oxide semiconductor-based artificial nose as a fast, reliable and non-expensive analytical technique for aroma profiling of milk adulteration. *International dairy journal*, 77:38–46, 2018.

[33] Takayuki Kawashima, Yasutomo Kawanishi, Ichiro Ide, Hiroshi Murase, Daisuke Deguchi, Tomoyoshi Aizawa, and Masato Kawade. Action recognition from extremely low-resolution thermal image sequence. In *Proceedings of the 14th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 1–6, August 2017.

[34] Sakorn Mekruksavanich and Anuchit Jitpattanakul. Lstm networks using smartphone data for sensor-based human activity recognition in smart homes. *Sensors*, 21(5):1636, 2021.

[35] Xue-Bo Jin, Nian-Xiang Yang, Xiao-Yi Wang, Yu-Ting Bai, Ting-Li Su, and Jian-Lei Kong. Hybrid deep learning predictor for smart agriculture sensing based on empirical mode decomposition and gated recurrent unit group model. *Sensors*, 20(5):1334, 2020.

[36] Mondher Bouazizi, Chen Ye, and Tomoaki Ohtsuki. Low-resolution infrared array sensor for counting and localizing people indoors: When low end technology meets cutting edge deep learning techniques. *Information*, 13(3):132, 2022.

[37] Andres Gomez, Francesco Conti, and Luca Benini. Thermal Image-Based CNN's for Ultra-Low Power People Recognition. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, pages 326–331, New York, NY, USA, 2018. Association for Computing Machinery.

[38] Aly Metwaly, Jorge Peña Queralta, Victor Kathan Sarker, Tuan Nguyen Gia, Omar Nasir, and Tomi Westerlund. Edge Computing with Embedded AI: Thermal Image Analysis for Occupancy Estimation in Intelligent Buildings. In *Proceedings of the INTelligent Embedded Systems Architectures and Applications Workshop 2019*, INTESA2019, pages 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[39] Peter Kraft, Daniel Kang, Deepak Narayanan, Shoumik Palkar, Peter Bailis, and Matei Zaharia. Willump: A Statistically-Aware End-to-end Optimizer for Machine Learning Inference, 2019.

[40] Munkhjargal Gochoo, Tan-Hsu Tan, Shih-Chia Huang, Tsedevdorj Batjargal, Jun-Wei Hsieh, Fady S Alnajjar, and Yung-Fu Chen. Novel iot-based privacy-preserving yoga posture recognition system using low-resolution infrared sensors and deep learning. *IEEE Internet of Things Journal*, 6(4):7192–7200, 2019.

[41] Igor Morawski and Wen-Nung Lie. Two-stream deep learning architecture for action recognition by using extremely low-resolution infrared thermopile arrays. In Phooi Yee Lau and Mohammad Shobri, editors, *International Workshop on Advanced Imaging Technology (IWAIT) 2020*, volume 11515, pages 164–169. International Society for Optics and Photonics, SPIE, 2020.

[42] Ouday Hanosh, Rashid Ansari, Naoum P Issa, and A Enis Cetin. Convulsive movement detection using low-resolution thermopile sensor array. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 300–301, 2020.

[43] Huixiang Liu, Qing Li, Bin Yan, Lei Zhang, and Yu Gu. Bionic electronic nose based on mos sensors array and machine learning algorithms used for wine properties detection. *Sensors*, 19(1):45, 2018.

[44] Dongzhi Zhang, Jingjing Liu, Chuanxing Jiang, Aiming Liu, and Bokai Xia. Quantitative detection of formaldehyde and ammonia gas via metal oxide-modified graphene-based sensor array combining with neural network model. *Sensors and Actuators B: Chemical*, 240:55–65, 2017.

[45] Akbar Rahimi. Short-term prediction of no 2 and no x concentrations using multilayer perceptron neural network: a case study of tabriz, iran. *Ecological Processes*, 6:1–9, 2017.

[46] Deepak Ghimire, Dayoung Kil, and Seong Heum Kim. A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration. *Electronics (Switzerland)*, 11(6), 2022.

[47] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[48] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 1–6, 2021.

[49] Akihiro Hayashi, Kazuaki Ishizaki, Gita Koblents, and Vivek Sarkar. Machine-learning-based performance heuristics for runtime cpu/gpu selection. In *Proceedings of the principles and practices of programming on the Java platform*, pages 27–36. 2015.

[50] Angelo Garofalo, Giuseppe Tagliavini, Francesco Conti, Luca Benini, and Davide Rossi. Xpulpnn: Enabling energy efficient and flexible inference of quantized neural networks on risc-v based iot end nodes. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1489–1505, 2021.

[51] Jie Han and M Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Test Symposium (ETS), 2013 18th IEEE European*, pages 1–6, 2013.

[52] Kishor Kunal, Meghna Madhusudan, Arvind K Sharma, Wenbin Xu, Steven M Burns, Ramesh Harjani, Jiang Hu, Desmond A Kirkpatrick, and Sachin S Sapatnekar. Align: Open-source analog layout automation from the ground up. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–4, 2019.

[53] Biying Xu, Keren Zhu, Mingjie Liu, Yibo Lin, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z Pan. Magical: Toward fully automated analog ic layout leveraging human and machine intelligence. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[54] Xiaohan Gao, Chenhui Deng, Mingjie Liu, Zhiru Zhang, David Z Pan, and Yibo Lin. Layout symmetry annotation for analog circuits with graph neural networks. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 152–157, 2021.

[55] Hao Chen, Keren Zhu, Mingjie Liu, Xiyuan Tang, Nan Sun, and David Z Pan. Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1243–1248. IEEE, 2021.

[56] Kishor Kunal, Tonmoy Dhar, Meghna Madhusudan, Jitesh Poojary, Arvind K Sharma, Wenbin Xu, Steven M Burns, Jiang Hu, Ramesh Harjani, and Sachin S Sapatnekar. Gnn-based hierarchical annotation for analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[57] Keren Zhu, Hao Chen, Mingjie Liu, and David Z Pan. Automating analog constraint extraction: From heuristics to learning. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 108–113. IEEE, 2022.

[58] Edoardo Charbon, Enrico Malavasi, and Alberto Sangiovanni-Vincentelli. Generalized constraint generation for analog circuit design. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pages 408–414. IEEE, 1993.

[59] Michael Eick, Martin Strasser, Kun Lu, Ulf Schlichtmann, and Helmut E Graeb. Comprehensive generation of hierarchical placement rules for analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(2):180–193, 2011.

[60] Qingsheng Hao, Song Chen, Xianlong Hong, Yi Su, Sheqin Dong, and Zhiyi Qu. Constraints generation for analog circuits layout. In *2004 International Conference on Communications, Circuits and Systems (IEEE Cat. No. 04EX914)*, volume 2, pages 1334–1338. IEEE, 2004.

[61] Zhe Zhou, Sheqin Dong, Xianlong Hong, Qingsheng Hao, and Song Chen. Analog constraints extraction based on the signal flow analysis. In *2005 6th International Conference on ASIC*, volume 2, pages 825–828. IEEE, 2005.

[62] Po-Hsun Wu, Mark Po-Hung Lin, and Tsung-Yi Ho. Analog layout synthesis with knowledge mining. In *2015 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. IEEE, 2015.

[63] Mingjie Liu, Wuxi Li, Keren Zhu, Biying Xu, Yibo Lin, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z Pan. S 3 det: Detecting system symmetry constraints for analog circuits with graph similarity. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 193–198. IEEE, 2020.

[64] Hao Chen, Mingjie Liu, Xiyuan Tang, Keren Zhu, Nan Sun, and David Z. Pan. Challenges and opportunities toward fully automated analog layout design. *Journal of Semiconductors*, 41(11), 2020.

[65] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[66] Bin Jiang, Jiachen Yang, Zhihan Lv, and Houbing Song. Wearable vision assistance system based on binocular sensors for visually impaired users. *IEEE Internet of Things Journal*, 6(2):1375–1383, 2019.

[67] Z Zhou, X Chen, E Li, L Zeng, K Luo, and J Zhang. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[68] W Shi, J Cao, Q Zhang, Y Li, and L Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[69] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[70] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[71] Yu Liu, Hongyang Li, Junjie Yan, Fangyin Wei, Xiaogang Wang, and Xiaoou Tang. Recurrent Scale Approximation for Object Detection in CNN. *CoRR*, abs/1707.0, 2017.

[72] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[77] Hanan Hussain, PS Tamizharasan, and CS Rahul. Design possibilities and challenges of dnn models: a review on the perspective of end devices. *Artificial Intelligence Review*, pages 1–59, 2022.

[78] Gaowei Xu, Min Liu, Zhuofu Jiang, Dirk Söffker, and Weiming Shen. Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning. *Sensors (Switzerland)*, 19(5), 2019.

[79] Narendra Patwardhan, Stefano Marrone, and Carlo Sansone. Transformers in the Real World: A Survey on NLP Applications. *Information (Switzerland)*, 14(4), 2023.

[80] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3(September):111–132, 2022.

[81] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):87–110, 2023.

[82] Fahad Shamshad, Salman Khan, Syed Waqas Zamir, Muhammad Haris Khan, Munawar Hayat, Fahad Shahbaz Khan, and Huazhu Fu. Transformers in medical imaging: A survey. *Medical Image Analysis*, 88(March), 2023.

[83] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in Time Series: A Survey. *IJCAI International Joint Conference on Artificial Intelligence*, 2023-August:6778–6786, 2023.

[84] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[85] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[86] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[87] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 662–677. Springer, 2016.

[88] Lin Chen, Jianting Fu, Yuheng Wu, Haochen Li, and Bin Zheng. Hand gesture recognition using compact cnn via surface electromyography signals. *Sensors*, 20(3):672, 2020.

[89] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2077–2086, 2017.

[90] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[91] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

[92] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[93] Omid Rohanian, Mohammadmahdi Nouriborji, Samaneh Kouchaki, and David A Clifton. On the effectiveness of compact biomedical transformers. *Bioinformatics*, 39(3):btad103, 2023.

[94] Francesco Daghero, Alessio Burrello, Chen Xie, Luca Benini, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Low-overhead early-stopping policies for efficient random forests inference on microcontrollers. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, pages 25–47. Springer, 2021.

[95] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[96] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[97] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[98] Bahzad Charbuty and Adnan Abdulazeez. Classification Based on Decision Tree Algorithm for Machine Learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[99] B Xu, K Zhu, M Liu, Y Lin, S Li, X Tang, N Sun, and D Z Pan. MAGICAL: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence: Invited Paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[100] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93, 2014.

[101] Chao Guo, Yan Yang, Hong Pan, Tianrui Li, and Weidong Jin. Fault analysis of high speed train with dbn hierarchical ensemble. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2552–2559. IEEE, 2016.

[102] Zhaoli Wang, Chengguang Lai, Xiaohong Chen, Bing Yang, Shiwei Zhao, and Xiaoyan Bai. Flood hazard risk assessment model based on random forest. *Journal of Hydrology*, 527:1130–1141, 2015.

[103] H2o.ai.

[104] Xgboost.

[105] Marcelo Urbina, Tatiana Acosta, Jesus Lazaro, Armando Astarloa, and Unai Bidarte. Smart Sensor: SoC Architecture for the Industrial Internet of Things. *IEEE Internet of Things Journal*, 6(4):6567–6577, 2019.

[106] Paul Abbott. *Understanding Smart Sensors*, volume 18. 1998.

[107] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[108] ARM. Armv7-m architecture reference manual, 2021.

[109] Gerry Kane and Joe Heinrich. *MIPS RISC architectures*. Prentice-Hall, Inc., 1992.

[110] Calista Redmond. Risc-v: The open era of computing, 2021.

[111] Ryszard Świerczyński, Krzysztof Urbanski, and Artur Wymysłowski. Improving co-design of smart sensor front-ends. *Proceedings of the 21st International Conference on Mixed Design of Integrated Circuits and Systems, MIXDES 2014*, pages 236–240, 2014.

[112] Tonmoy Dhar, Kishor Kunal, Yaguang Li, Yishuang Lin, Meghna Madhusudan, Jitesh Poojary, Arvind K Sharma, Steven M Burns, Ramesh Harjani, Jiang Hu, Parijat Mukherjee, Soner Yaldiz, and Sachin S Sapatnekar. Machine Learning Techniques in Analog Layout Automation. In *Proceedings of the 2021 International Symposium on Physical Design*, ISPD '21, pages 71–72, New York, NY, USA, 2021. Association for Computing Machinery.

[113] Ambeation : Analog mixed-signal back-end design automation with machine learning and artificial intelligence techniques, 2020.

[114] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[115] Rahul Chauhan, Kamal Kumar Ghanshala, and RC Joshi. Convolutional neural network (cnn) for image detection and recognition. In *2018 first international conference on secure cyber computing and communication (ICSCCC)*, pages 278–282. IEEE, 2018.

[116] Md Anwar Hossain and Md Shahriar Alam Sajib. Classification of image using convolutional neural network (cnn). *Global Journal of Computer Science and Technology*, 19(D2):13–18, 2019.

[117] Joseph Bullock, Carolina Cuesta-Lázaro, and Arnau Quera-Bofarull. Xnet: a convolutional neural network (cnn) implementation for medical x-ray image segmentation suitable for small datasets. In *Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging*, volume 10953, pages 453–463. SPIE, 2019.

[118] Aijaz Ahmad Reshi, Furqan Rustam, Arif Mehmood, Abdulaziz Alhossan, Ziyad Alrabiah, Ajaz Ahmad, Hessa Alsuwailem, and Gyu Sang Choi. An efficient cnn model for covid-19 disease detection based on x-ray image classification. *Complexity*, 2021:1–12, 2021.

[119] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[120] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[121] Huaizu Jiang and Erik Learned-Miller. Face detection with the faster r-cnn. In *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*, pages 650–657. IEEE, 2017.

[122] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[123] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[124] Sharada Prasanna Mohanty, David Hughes, and Marcel Salathé. Inference of plant diseases from leaf images through deep learning. *Front. Plant Sci*, 7:1419, 2016.

[125] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[126] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[127] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and electronics in agriculture*, 145:311–318, 2018.

[128] Wei Xue, Xuejiao Hu, Zhong Wei, Xinlan Mei, Xingjian Chen, and Yangchun Xu. A fast and easy method for predicting agricultural waste compost maturity by image-based deep learning. *Bioresource Technology*, 290:121761, 2019.

[129] Nataliia Kussul, Mykola Lavreniuk, Sergii Skakun, and Andrii Shelestov. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5):778–782, 2017.

[130] Dongmei Song, Yaxiong Ding, Xiaofeng Li, Biao Zhang, and Mingyu Xu. Ocean oil spill classification with radarsat-2 sar based on an optimized wavelet neural network. *Remote Sensing*, 9(8):799, 2017.

[131] Alessio Burrello, Moritz Scherer, Marcello Zanghieri, Francesco Conti, and Luca Benini. A Microcontroller is All You Need: Enabling Transformer Execution on Low-Power IoT Endnodes. *2021 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2021*, pages 1–6, 2021.

[132] Alexander Wong, Mahmoud Famouri, Maya Pavlova, and Siddharth Surana. Tinyspeech: Attention condensers for deep speech recognition neural networks on edge devices. *arXiv preprint arXiv:2008.04245*, 2020.

[133] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[134] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

[135] Saad Abbasi, Mahmoud Famouri, Mohammad Javad Shafiee, and Alexander Wong. Outliernets: highly compact deep autoencoder network architectures for on-device acoustic anomaly detection. *Sensors*, 21(14):4805, 2021.

[136] Sinziana Mazilu, Michael Hardegger, Zack Zhu, Daniel Roggen, Gerhard Tröster, Meir Plotnik, and Jeffrey M Hausdorff. Online detection of freezing of gait with smartphones and machine learning techniques. In *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 123–130. IEEE, 2012.

[137] Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava. Machine Learning for Microcontroller-Class Hardware: A Review. *IEEE Sensors Journal*, 22(22):21362–21390, 2022.

[138] Pete Warden and Daniel Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

[139] Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet). *arXiv preprint arXiv:2201.08442*, 2022.

[140] Microsoft. Neural Network Intelligence, January 2021.

[141] Sridhar Gopinath, Nikhil Ghanathe, Vivek Seshadri, and Rahul Sharma. Compiling kb-sized machine learning models to tiny iot devices. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 79–95, 2019.

[142] Claudionor N Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, 3(8):675–686, 2021.

[143] Alessandro Capotondi, Manuele Rusci, Marco Fariselli, and Luca Benini. Cmix-nn: Mixed low-precision cnn library for memory-constrained edge devices. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(5):871–875, 2020.

[144] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[145] Manuele Rusci, Alessandro Capotondi, and Luca Benini. Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers. *Proceedings of Machine Learning and Systems*, 2:326–335, 2020.

[146] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

[147] Francesco Daghero, Daniele Jahier Pagliari, and Massimo Poncino. Energy-efficient deep learning inference on edge devices. In Shiho Kim and Ganesh Chandra Deka, editors, *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, volume 122 of *Advances in Computers*, chapter 8, pages 247–301. Elsevier, 2021.

[148] F Daghero, A Burrello, D J Pagliari, L Benini, E Macii, and M Poncino. Energy-Efficient Adaptive Machine Learning on IoT End-Nodes With Class-Dependent Confidence. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1–4, 2020.

[149] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 124–132, 2015.

[150] Hokchhay Tann, Soheil Hashemi, R. Iris Bahar, and Sherief Reda. Runtime configurable deep neural networks for energy-accuracy trade-off. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis - CODES '16*, pages 1–10, 2016.

[151] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks, 2018.

[152] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. LAPSE: Low-Overhead Adaptive Power Saving and Contrast Enhancement for OLEDs. *IEEE Transactions on Image Processing*, 27(9):4623–4637, 2018.

[153] M Parsa, P Panda, S Sen, and K Roy. Staged Inference using Conditional Deep Learning for energy efficient real-time smart diagnosis. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 78–81, 2017.

[154] Francesco Daghero, Daniele Jahier Pagliari, and Massimo Poncino. Two-stage human activity recognition on microcontrollers with decision trees and cnns. In *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 173–176, 2022.

[155] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. HydraNets: Specialized Dynamic Architectures for Efficient Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2018.

[156] Alessio Burrello, Daniele Jahier Pagliari, Pierangelo Maria Rapa, Matilde Semilia, Matteo Risso, Tommaso Polonelli, Massimo Poncino, Luca Benini, and Simone Benatti. Embedding temporal convolutional networks for energy-efficient ppg-based heart rate monitoring. *ACM Trans. Comput. Healthcare*, 3(2), mar 2022.

[157] DE Joseph Yiu. Introduction to the arm cortex-m55 processor. *Accessed: Feb*, 2020.

[158] Michael Gautschi et al. Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[159] Younghyun Kim, Setareh Behroozi, Vijay Raghunathan, and Anand Raghunathan. AXSERBUS: A quality-configurable approximate serial bus for energy-efficient sensing. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 1–6, 2017.

[160] S Ghosh, P Ghosal, N Das, S P Mohanty, and O Okobiah. Data Correlation Aware Serial Encoding for Low Switching Power On-Chip Communication. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pages 124–129, 2014.

[161] Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. SILENT: serialized low energy transmission coding for on-chip interconnection networks. In *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pages 448–451, nov 2004.

[162] Sabino Salerno, Alberto Bocca, Enrico Macii, and Massimo Poncino. Limited Intra-Word Transition Codes: An Energy-Efficient Bus Encoding for LCD Display Interfaces. In *Proceedings of the International Symposium on Low Power Electronics and Design*, volume 2004-Janua, pages 206–211, 2004.

[163] Jian Zeng, Jian-Yang Zhou, and Rung-Bin Lin. Transition inversion coding with parity check for off-chip serial transmission. In *Electronics, Circuits and Systems (ICECS), 2014 21st IEEE International Conference on*, pages 634–637, 2014.

[164] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Approximate Differential Encoding for Energy-Efficient Serial Communication. In *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '16, pages 421–426, New York, NY, USA, 2016. ACM.

[165] Xianglong Ren, Deyuan Gao, Xiaoya Fan, and Jianfeng An. Adaptive Low-Power Transmission Coding for Serial Links in Network-on-Chip. *Procedia Engineering*, 29:1618–1624, 2012.

[166] Massimo Poncino and Enrico Macii. Low-energy RGB color approximation for digital LCD interfaces. *IEEE Transactions on Consumer Electronics*, 52(3):1004–1012, 2006.

[167] Phillip Stanley-Marbell and Martin Rinard. Reducing Serial I/O Power in Error-tolerant Applications by Efficient Lossy Encoding. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 62:1—-62:6, New York, NY, USA, 2016. ACM.

[168] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Serial T0: Approximate Bus Encoding for Energy-efficient Transmission of Sensor Signals. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 14:1—-14:6, New York, NY, USA, 2016. ACM.

[169] Arun Venkatachar. Confluence of ai/ml with eda and software engineering. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 13–15. IEEE, 2021.

[170] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade

Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[171] Borivoje Nikolić. Ml for analog design: Good progress, but more to do. In *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, pages 53–54, 2022.

[172] Panasonic Industry. Grid-eye application note on social distancing. people detection and tracking with ceiling mounted sensors, 2020.

[173] STMicroelectronics. Stm32l476rg :ultra-low-power with fpu arm cortex-m4 mcu, 2024.

[174] STMicroelectronics. STM32L4A6ZG, Ultra-low-power Arm Cortex-M4 32-bit MCU. https://www.st.com/en/microcontrollers-microprocessors/stm32l4a6zg.html.

[175] Pasquale Davide Schiavone, Davide Rossi, Antonio Pullini, Alfio Di Mauro, Francesco Conti, and Luca Benini. Quentin: An ultra-low-power pulpissimo soc in 22nm fdx. In *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–3. IEEE, 2018.

[176] Chakravartula Raghavachari, V Aparna, S Chithira, and Vidhya Balasubramanian. A comparative study of vision based human detection techniques in people counting applications. *Procedia Computer Science*, 58:461–469, 2015.

[177] Wei Xi, Jizhong Zhao, Xiang-Yang Li, Kun Zhao, Shaojie Tang, Xue Liu, and Zhiping Jiang. Electronic frog eye: Counting crowd using wifi. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 361–369. IEEE, 2014.

[178] Kazuhiko Hashimoto, Katsuya Morinaka, Nobuyuki Yoshiike, Chihiro Kawaguchi, and Satoshi Matsueda. People count system using multi-sensing application. In *Proceedings of International Solid State Sensors and Actuators Conference (Transducers' 97)*, volume 2, pages 1291–1294. IEEE, 1997.

[179] Mamoona Birkhez Shami, Salman Maqbool, Hasan Sajid, Yasar Ayaz, and Sen-Ching Samson Cheung. People counting in dense crowd images using sparse head detections. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2627–2636, 2018.

[180] Akshaya D Shetty, B Shubha, K Suryanarayana, et al. Detection and tracking of a human using the infrared thermopile array sensor—"grid-eye". In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 1490–1495. IEEE, 2017.

[181] Saleh Basalamah, Sultan Daud Khan, and Habib Ullah. Scale driven convolutional neural network model for people counting and localization in crowd scenes. *IEEE Access*, 7:71576–71584, 2019.

[182] Valério Nogueira, Hugo Oliveira, José Augusto Silva, Thales Vieira, and Krerley Oliveira. Retailnet: A deep learning approach for people counting and hot spots detection in retail stores. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 155–162. IEEE, 2019.

[183] Sultan Daud Khan, Habib Ullah, Mohib Ullah, Nicola Conci, Faouzi Alaya Cheikh, and Azeddine Beghdadi. Person head detection based deep model for people counting in sports videos. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2019.

[184] Xiaoming Liu, Peter Henry Tu, Jens Rittscher, Amitha Perera, and Nils Krahnstoever. Detecting and counting people in surveillance applications. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 306–311. IEEE, 2005.

[185] Zoltán Vokó and János György Pitter. The effect of social distance measures on covid-19 epidemics in europe: an interrupted time series analysis. *GeroScience*, 42(4):1075–1082, 2020.

[186] Anna Trofimova, Andrea Masciadri, Fabio Veronese, and Fabio Salice. Indoor human detection based on thermal array sensor data and adaptive background estimation. *Journal of Computer and Communications*, 05:16–28, 03 2017.

[187] Christian Herrmann, Miriam Ruf, and Jürgen Beyerer. CNN-based thermal infrared person detection by domain adaptation. In Michael C. Dudzik and Jennifer C. Ricklin, editors, *Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything*, volume 10643, pages 38 – 43. International Society for Optics and Photonics, SPIE, 2018.

[188] Cristian Perra, Amit Kumar, Michele Losito, Paolo Pirino, Milad Moradpour, and Gianluca Gatto. Monitoring indoor people presence in buildings using low-cost infrared sensor array in doorways. *Sensors*, 21(12):4062, 2021.

[189] Hessam Mohammadmoradi, Sirajum Munir, Omprakash Gnawali, and Charles Shelton. Measuring people-flow through doorways using easy-to-install ir array sensors. In *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 35–43. IEEE, 2017.

[190] Hongfei Wang, Yanzhou Zhang, Zhanyuan Ye, Hengfa Liu, Xin Wei, and Juemin Tang. A lightweight people counting approach for smart buildings. In *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–5. IEEE, 2021.

[191] Saipriyati Singh and Baris Aksanli. Non-intrusive presence detection and position tracking for multiple people using low-resolution thermal sensors. *Journal of Sensor and Actuator Networks*, 8(3):40, 2019.

[192] Marwane Rezzouki, Safae Ouajih, and Guillaume Ferre. Monitoring Social Distancing in Queues Using Infrared Array Sensor. *IEEE Sensors Journal*, 22(17):16764–16771, 2022.

[193] Stefano Savazzi, Vittorio Rampa, Leonardo Costa, and Denis Tolochenko. Analysis of body-induced thermal signatures for social distancing monitoring. *Proceedings of IEEE Sensors*, 2020-October:20–23, 2020.

[194] Ramtin Rabiee and Johannes Karlsson. Multi-bernoulli tracking approach for occupancy monitoring of smart buildings using low-resolution infrared sensor array. *Remote Sensing*, 13(16):3127, 2021.

[195] Andres Gomez, Francesco Conti, and Luca Benini. Thermal image-based cnn's for ultra-low power people recognition. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 326–331, 2018.

[196] Aly Metwaly, Jorge Peña Queralta, Victor Kathan Sarker, Tuan Nguyen Gia, Omar Nasir, and Tomi Westerlund. Edge computing with embedded ai: Thermal image analysis for occupancy estimation in intelligent buildings. In *Proceedings of the INTelligent Embedded Systems Architectures and Applications Workshop 2019*, pages 1–6, 2019.

[197] Marek Kraft, Przemysław Aszkowski, Dominik Pieczyński, and Michał Fularz. Low-cost thermal camera-based counting occupancy meter facilitating energy saving in smart buildings. *Energies*, 14(15), 2021.

[198] Aisha Fahad Alraeesi, Hanan Fekri Kharbash, Jawaher Saif Alghfeli, Shamma Sultan Alsaedi, and Munkhjargal Gochoo. Privacy-Preserved Social Distancing System Using Low-Resolution Thermal Sensors and Deep Learning. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 66–71, 2021.

[199] STMicroelectronics. Stm32l412kb : Ultra-low-power with fpu arm cortex-m4 mcu 80 mhz with 128 kbytes of flash memory, usb, 2022.

[200] Lili Tao, Timothy Volonakis, Bo Tan, Ziqi Zhang, Yanguo Jing, and Melvyn Smith. 3D convolutional neural network for home monitoring using low resolution thermal-sensor array. In *3rd IET International Conference on Technologies for Active and Assisted Living (TechAAL 2019)*, pages 1–6, 2019.

[201] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[202] M. Abadi et al. Tesorflow: Large-scale machine learning on heterogeneous systems, 2015.

[203] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Dynamic Bit-width Reconfiguration for Energy-Efficient Deep Learning Hardware. In *Proceedings of the International Symposium on Low Power Electronics and Design*, ISLPED '18, pages 47:1—-47:6, New York, NY, USA, 2018. ACM.

[204] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.

[205] Koen Goetschalckx, Bert Moons, Steven Lauwereins, Martin Andraud, and Marian Verhelst. Optimized hierarchical cascaded processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):884–894, 2018.

[206] Louisa Lam and SY Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(5):553–568, 1997.

[207] Fatih Demir, Muammer Turkoglu, Muzaffer Aslan, and Abdulkadir Sengur. A new pyramidal concatenated cnn approach for environmental sound classification. *Applied Acoustics*, 170:107520, 2020.

[208] Qian Wu and Fei Wang. Concatenate convolutional neural networks for non-intrusive load monitoring across complex background. *Energies*, 12(8):1572, 2019.

[209] Tae-Young Kim and Sung-Bae Cho. Predicting residential energy consumption using cnn-lstm neural networks. *Energy*, 182:72–81, 2019.

[210] Jianfeng Zhao, Xia Mao, and Lijiang Chen. Speech emotion recognition using deep 1d & 2d cnn lstm networks. *Biomedical signal processing and control*, 47:312–323, 2019.

[211] Veronica Sciannameo, Alessia Goffi, Giuseppe Maffeis, Roberta Gianfreda, Daniele Jahier Pagliari, Tommaso Filippini, Pamela Mancuso, Paolo Giorgi-Rossi, Leonardo Alberto Dal Zovo, Angela Corbari, Marco Vinceti, and Paola Berchialla. A deep learning approach for Spatio-Temporal forecasting of new cases and new hospital admissions of COVID-19 spread in Reggio Emilia, Northern Italy. *Journal of Biomedical Informatics*, 132:104132, August 2022.

[212] Michal Stec, Viktor Herrmann, and Benno Stabernack. Multi-sensor-fusion system for people counting applications. In *2019 First International Conference on Societal Automation (SA)*, pages 1–4. IEEE, 2019.

[213] Daniel Olmeda, Cristiano Premebida, Urbano Nunes, Jose Maria Armingol, and Arturo de la Escalera. Pedestrian detection in far infrared images. *Integrated Computer-Aided Engineering*, 20(4):347–360, 2013.

[214] FLIR. Free flir thermal dataset for algorithm training, 2018.

[215] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection: Benchmark dataset and baselines. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[216] Rafael E. Rivadeneira, Angel D. Sappa, Boris X. Vintimilla, Lin Guo, Jiankun Hou, Armin Mehri, Parichehr Behjati Ardakani, Heena Patel, Vishal Chudasama, Kalpesh Prajapati, Kishor P. Upla, Raghavendra Ramachandra, Kiran Raja, Christoph Busch, Feras Almasri, Olivier Debeir, Sabari Nathan, Priya Kansal, Nolan Gutierrez, Bardia Mojra, and William J. Beksi. Thermal image super-resolution challenge - pbvs 2020. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[217] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2018.

[218] STMicroelectronics. X-CUBE-AI, AI expansion pack for STM32CubeMX. https://www.st.com/en/embedded-software/x-cube-ai.html.

[219] John D Kelleher, Brian Mac Namee, and Aoife D'arcy. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT press, 2020.

[220] STMicroelectronics. STM32CubeIDE, Integrated Development Environment for STM32. https://www.st.com/en/development-tools/stm32cubeide.htm.

[221] Pasquale Davide Schiavone, Davide Rossi, Antonio Pullini, Alfio Di Mauro, Francesco Conti, and Luca Benini. Quentin: an ultra-low-power pulpissimo soc in 22nm fdx. In *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–3. IEEE, 2018.

[222] Nazareno Bruschi, Germain Haugou, Giuseppe Tagliavini, Francesco Conti, Luca Benini, and Davide Rossi. Gvsoc: A highly configurable, fast and accurate full-platform simulator for risc-v based iot processors. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 409–416. IEEE, 2021.

[223] Angelo Garofalo, Manuele Rusci, Francesco Conti, Davide Rossi, and Luca Benini. PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2164):20190155, feb 2020.

[224] Ceiling heights in homes and offices, howpublished = https://realestate.wharton.upenn.edu/wp-content/uploads/2017/03/678.pdf, note = .

[225] Serkan Balli, Ensar Arif Sağbaş, and Musa Peker. Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. *Measurement and Control*, 52(1-2):37–45, 2019.

[226] Lin Fan, Zhongmin Wang, and Hai Wang. Human activity recognition model based on decision tree. In *2013 International Conference on Advanced Cloud and Big Data*, pages 64–68. IEEE, 2013.

[227] STMicroelectronics. inemo inertial module: always-on 3d accelerometer and 3d gyroscope. Website, 2019. www.st.com/resource/en/datasheet/lsm6dsox.pdf.

[228] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[229] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101, 2017.

[230] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(12):31314–31338, 2015.

[231] Akram Bayat, Marc Pomplun, and Duc A Tran. A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, 34:450–457, 2014.

[232] Valentina Bianchi, Marco Bassoli, Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, and Ilaria De Munari. Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet of Things Journal*, 6(5):8553–8562, 2019.

[233] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1533–1540, 2016.

[234] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[235] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[236] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[237] R. Meattini, S. Benatti, U. Scarcia, D. De Gregorio, L. Benini, and C. Melchiorri. An semg-based human–robot interface for robotic hands using machine learning and synergies. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2018.

[238] Zhiwen Zheng, Qingshan Wang, Dejun Yang, Qi Wang, Wei Huang, and Yinlong Xu. L-sign: Large-vocabulary sign gestures recognition system. *IEEE Transactions on Human-Machine Systems*, 52(2):290–301, 2022.

[239] Sakshi Sharma and Sukhwinder Singh. Vision-based hand gesture recognition using deep learning for the interpretation of sign language. *Expert Systems with Applications*, 182:115657, 2021.

[240] Debajit Sarma and Manas Kamal Bhuyan. Methods, databases and recent advancement of vision-based hand gesture recognition for hci systems: A review. *SN Computer Science*, 2(6):1–40, 2021.

[241] Manfredo Atzori, Arjan Gijsberts, Claudio Castellini, Barbara Caputo, Anne-Gabrielle Mittaz Hager, Simone Elsig, Giorgio Giatsidis, Franco Bassetto, and Henning Müller. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data*, 1:140053, dec 2014.

[242] Paul Kaufmann, Kevin Englehart, and Marco Platzner. Fluctuating emg signals: Investigating long-term effects of pattern matching algorithms. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 6357–6360. IEEE, 2010.

[243] Simone Benatti, Filippo Casamassima, Bojan Milosevic, Elisabetta Farella, Philipp Schönle, Schekeb Fateh, Thomas Burger, Qiuting Huang, and Luca Benini. A versatile embedded platform for emg acquisition and gesture recognition. *IEEE transactions on biomedical circuits and systems*, 9(5):620–630, 2015.

[244] Bojan Milosevic, Elisabetta Farella, and Simone Benatti. Exploring arm posture and temporal variability in myoelectric hand gesture recognition. In *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pages 1032–1037. IEEE, 2018.

[245] Yu Hu, Yongkang Wong, Wentao Wei, Yu Du, Mohan Kankanhalli, and Weidong Geng. A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition. *PloS one*, 13(10), 2018.

[246] Panagiotis Tsinganos, Bruno Cornelis, Jan Cornelis, Bart Jansen, and Athanassios Skodras. Deep learning in emg-based gesture recognition. In *PhyCS*, 2018.

[247] P Tsinganos et al. Improved gesture recognition based on semg signals and tcn. In *ICASSP 2019*, pages 1169–1173. IEEE, 2019.

[248] Eric Flamand, Davide Rossi, Francesco Conti, Igor Loi, Antonio Pullini, Florent Rotenberg, and Luca Benini. GAP-8: A RISC-V SoC for AI at the Edge of the IoT. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–4. IEEE, 2018.

[249] Joseph L. Betthauser, John T. Krall, Rahul R. Kaliki, Matthew S. Fifer, and Nitish V. Thakor. Stable Electromyographic Sequence Prediction during Movement Transitions using Temporal Convolutional Networks. *International IEEE/EMBS Conference on Neural Engineering*, 2019.

[250] Matteo Risso, Alessio Burrello, Daniele Jahier Pagliari, Simone Benatti, Enrico Macii, Luca Benini, and Massimo Poncino. Robust and Energy-Efficient PPG-Based Heart-Rate Monitoring. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2021.

[251] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[252] M Zanghieri et al. Robust real-time embedded emg recognition framework using temporal convolutional networks on a multicore iot processor. *IEEE transactions on biomedical circuits and systems*, 14(2):244–256, 2019.

[253] Wentao Wei, Qingfeng Dai, Yongkang Wong, Yu Hu, Mohan Kankanhalli, and Weidong Geng. Surface-electromyography-based gesture recognition by multi-view deep learning. *IEEE Transactions on Biomedical Engineering*, 66(10):2964–2973, 2019.

[254] Yongxiang Zou and Long Cheng. A transfer learning model for gesture recognition based on the deep features extracted by cnn. *IEEE Transactions on Artificial Intelligence*, 2(5):447–458, 2021.

[255] Lijun Han, Yongxiang Zou, and Long Cheng. A convolutional neural network with multi-scale kernel and feature fusion for semg-based gesture recognition. In *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 774–779. IEEE, 2021.

[256] Bernard Hudgins, Philip Parker, and Robert N Scott. A new strategy for multifunction myoelectric control. *IEEE transactions on biomedical engineering*, 40(1):82–94, 1993.

[257] K. Englehart and B. Hudgins. A robust, real-time control scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 50(7):848–854, July 2003.

[258] Claudio Castellini, Emanuele Gruppioni, Angelo Davalli, and Giulio Sandini. Fine detection of grasp force and posture by amputees via surface electromyography. *Journal of Physiology-Paris*, 103(3-5):255–262, 2009.

[259] Angkoon Phinyomark and Erik J. Scheme. Emg pattern recognition in the era of big data and deep learning. 2018.

[260] Simone Benatti, Elisabetta Farella, Emanuele Gruppioni, and Luca Benini. Analysis of robust implementation of an emg pattern recognition based control. In *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies-Volume 4*, pages 45–54. SCITEPRESS-Science and Technology Publications, Lda, 2014.

[261] Vinicius Horn Cene, Mauricio Tosin, Juliano Machado, and Alexandre Balbinot. Open Database for Accurate Upper-Limb Intent Detection Using Electromyography and Reliable Extreme Learning Machines. *Sensors (Basel, Switzerland)*, 19(8), 2019.

[262] Angelo Garofalo, Manuele Rusci, Francesco Conti, Davide Rossi, and Luca Benini. Pulp-nn: Accelerating quantized neural networks on parallel ultra-low-power risc-v processors. *arXiv preprint arXiv:1908.11263*, 2019.

[263] Ibex.

[264] Sifive e31.

[265] Liangzhen Lai, Naveen Suda, and Vikas Chandra. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*, 2018.

[266] Angelo Garofalo, Manuele Rusci, Francesco Conti, Davide Rossi, and Luca Benini. Pulp-nn: accelerating quantized neural networks on parallel ultra-low-power risc-v processors. *Philosophical Transactions of the Royal Society A*, 378(2164):20190155, 2020.

[267] STMicroelectronics. Stm32l4r9zi : Ultra-low-power with fpu arm cortex-m4 mcu, 2020.

[268] Chen Xie, Daniele Jahier Pagliari, Andrea Calimera, Enrico Macii, and Massimo Poncino. Acme: An energy-efficient approximate bus encoding for i 2 c. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2021.

[269] NXP. I2C-bus specification and user manual, 2014.

[270] Daniel Friesel and Olaf Spinczyk. I2c considered wasteful: saving energy with host-controlled pull-up resistors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pages 315–316, 2019.

[271] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Approximate Energy-Efficient Encoding for Serial Interfaces. *ACM Transactions on Design Automation of Electronic Systems*, 22(4):1–25, 2017.

[272] Muhammad Shoaib, Hans Scholten, and Paul JM Havinga. Towards physical activity recognition using smartphone sensors. In *2013 IEEE 10th international conference on ubiquitous intelligence and computing and 2013 IEEE 10th international conference on autonomic and trusted computing*, pages 80–87. IEEE, 2013.

[273] OSR.n.d. Open speech repository.

[274] Nicolae Patrascoiu. A low cost solution to monitor environmental parameters in industrial area perimeters. In *MATEC Web of Conferences*, volume 305, page 00003. EDP Sciences, 2020.

[275] Tobias Massier, Helmut Graeb, and Ulf Schlichtmann. The sizing rules method for cmos and bipolar analog integrated circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(12):2209–2222, 2008.

[276] Markus Meissner and Lars Hedrich. Feats: Framework for explorative analog topology synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(2):213–226, 2014.

[277] D Das, D Hsu, and W Wolf. Template matching for automatic analog circuit verification. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 396–403, 2001.

[278] Ramesh Harjani, Rob A Rutenbar, and L Richard Carley. A prototype framework for knowledge-based analog circuit synthesis. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pages 42–49, 1987.

[279] Po-Hsun Wu, Mark Po-Hung Lin, Tung-Chieh Chen, Ching-Feng Yeh, Xin Li, and Tsung-Yi Ho. A novel analog physical synthesis methodology integrating existent design expertise. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(2):199–212, 2014.

[280] S Chawla and KH Kwon. A rule-based approach for automatic analog circuit partitioning. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 580–584, 1997.

[281] Juergen Scheible and Jens Lienig. Automation of analog ic layout: Challenges and solutions. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, pages 33–40, 2015.

[282] Hao Li, Fanshu Jiao, and Alex Doboli. Analog circuit topological feature extraction with unsupervised learning of new sub-structures. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1509–1512. IEEE, 2016.

[283] Guan-Hong Liou, Shuo-Hui Wang, Yan-Yu Su, and Mark Po-Hung Lin. Classifying analog and digital circuits with machine learning techniques toward mixed-signal design automation. In *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 173–176. IEEE, 2018.

[284] K Kunal, T Dhar, M Madhusudan, J Poojary, A Sharma, W Xu, S M Burns, J Hu, R Harjani, and S S Sapatnekar. GANA: Graph Convolutional Network Based Automated Netlist Annotation for Analog Circuits. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 55–60, 2020.

[285] Mohamed Salem, Ali Deeb, Abdalrahman Ibrahim, Witesyavwirwa Vianney Kambale, Pichler Joachim, Fadi Al Machot, and Kyandoghere Kyamakya. A Vertical Systematic Generalization Towards Real IP Analog-Mixed-Signal AMS Schematics Structure Recognition. *2023 27th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, pages 283–288, 2023.