A Comprehensive Systematic Review of TinyML for Person Detection Systems

Yehia A. Soliman, Amr S. Ghoneim, Mahmoud M. Elkhouly

***Abstract*—Tiny Machine Learning (TinyML) enables the deployment of machine learning models on ultra-low-power and memory-constrained edge devices. This capability is crucial for person detection systems in applications such as smart homes, wearable health monitors, industrial safety, and wildlife surveillance. However, deploying person detection on microcontrollers poses significant challenges due to limited computation, memory, and energy resources. This paper presents a systematic literature review (SLR) of recent research in TinyML-based person detection from 2014 to 2024. We explore lightweight neural network architectures (e. g. , MobileNet, Tiny-YOLO), optimization techniques (e. g. , quantization, pruning, knowledge distillation), and performance metrics, including accuracy, latency, and energy efficiency. We also assess the suitability of edge hardware platforms such as ARM Cortex-M, ESP32, STM32, Jetson Nano, and Raspberry Pi. The review identifies current trends, highlights practical constraints, and proposes future directions involving adaptive models, federated learning, and privacy-preserving designs. This work serves as a reference for researchers and practitioners aiming to build efficient, scalable, and real-time TinyML-based person detection systems.**

***Index Terms*—Lightweight models, Person detection, Resource-constrained devices, TinyML**

I. Introduction

TINY Machine Learning (TinyML) represents a transformative shift in artificial intelligence, enabling the deployment of trained machine learning models directly on ultra-low-power, memory-constrained devices such as microcontrollers. This capability is particularly valuable in real-time, always-on applications where cloud connectivity is impractical, expensive, or privacy-sensitive [1]. Among the many applications of TinyML, person detection—the task of identifying the presence of individuals in images or video streams—has emerged as a critical function across domains such as smart homes, wearable health monitoring, industrial safety, environmental surveillance, and autonomous systems. Unlike traditional deep learning methods that rely on high-resource cloud servers or GPUs, TinyML-based person detection aims to perform on-device inference within the strict limits of computation, memory, and energy typical of embedded systems. [2]-[4].

Despite the growing interest in TinyML, deploying person detection models on edge devices presents numerous challenges. These include constrained memory (often limited to kilobytes), low clock speed CPUs, limited or absent accelerators, and strict power budgets. Consequently, achieving accurate and efficient person detection in such settings necessitates the use of lightweight neural network architectures (e. g., MobileNet, Tiny-YOLO, SqueezeNet) along with model optimization techniques such as quantization, pruning, knowledge distillation, and neural architecture search [5].

TinyML is an emerging subfield of machine learning that focuses on deploying models on resource-constrained edge devices—such as microcontrollers—with minimal power and memory requirements [6]. Devices like the ARM Cortex-M series, ESP32, and Arduino boards are low-cost and low-power, making them well-suited for applications that demand local, efficient, and real-time processing. Additionally, TinyML enables machine learning in remote or distributed environments without requiring continuous internet connectivity or access to cloud infrastructure. This capability is particularly valuable in scenarios that necessitate privacy, low latency, or offline functionality [5].

One of the key applications of TinyML is person detection [7]–[10], which involves identifying human presence in various environments [11]. Thus, implementing person detection on TinyML-enabled devices has practical value across multiple domains [12], [13]. For instance, in home automation, it can trigger lighting, climate control, or security systems based on occupancy. In industrial settings, it helps monitor hazardous areas, enhancing worker safety without the need for continuous supervision. Similarly, in health and fitness, it enables wearable devices to track activity patterns and support wellness monitoring. Furthermore, in agriculture and conservation, it can detect unauthorized human presence to protect restricted areas and wildlife habitats.

Person detection presents significant challenges for resource-constrained devices due to their limited processing power and memory capacity [14]. Although traditional machine learning and deep learning models—such as YOLO (You Only Look Once) and MobileNet—are widely employed for person detection, they typically require substantial computational resources [15]–[17].

Manuscript received April 9, 2025; revised September 7, 2025.

Yehia A. Soliman is a postgraduate student of Information Technology, Faculty of Computers and Artificial Intelligence, Helwan University, Egypt (corresponding author to provide e-mail: yahia.abuelkhair@fci.helwan.edu.eg).

Amr S. Ghoneim is an associate professor of Computer Science, Faculty of Computers and Artificial Intelligence, Helwan University, Egypt (e-mail: amr.ghoneim@fci.helwan.edu.eg).

Mahmoud M. Elkhouly is a professor of Information Technology, Faculty of Computers and Artificial Intelligence, Helwan University, Egypt (e-mail: elkhouly@fci.helwan.edu.eg).

Likewise, recent advancements in large language models (LLMs), while highly effective across various AI tasks, demand considerable computing power and are generally unsuitable for direct deployment on edge devices.

To address these limitations, TinyML adapts complex models through a range of optimization techniques, including model quantization, pruning, knowledge distillation, neural architecture search (NAS), weight sharing, low-rank factorization, and operator fusion. When combined with efficient architectural design and hardware-aware training, these methods enable the deployment of intelligent models on low-power, memory-constrained devices with minimal loss in accuracy [18].

This review adopts a rigorous methodology encompassing the selection, evaluation, and synthesis of peer-reviewed journal articles and conference papers published between 2014 and 2024. A total of 132 studies were initially screened, of which 50 met the inclusion criteria. The review focuses on key areas, including lightweight model architectures, optimization techniques such as quantization and pruning, and hardware-specific implementations across microcontrollers, edge devices, and IoT platforms. Additionally, evaluation metrics—such as accuracy, latency, and energy efficiency—were systematically analyzed to assess the performance of the reviewed approaches.

The goal of this paper is to provide a comprehensive and systematic literature review (SLR) of TinyML techniques specifically applied to person detection systems. By synthesizing insights from peer-reviewed publications between 2014 and 2024, we aim to:

1) Holistic Review of TinyML Person Detection Techniques: This review provides a comprehensive synthesis of recent literature on TinyML models applied to person detection, covering both vision-based and sensor-driven approaches. Moreover, it examines a wide range of embedded platforms, including microcontrollers and lightweight edge devices, to evaluate how TinyML techniques are practically implemented.

2) Comparative Analysis of Evaluation Metrics and Hardware Platforms: Key performance indicators such as accuracy, latency, model size, power consumption, and frames per second (FPS) are discussed in detail. Additionally, a comparative review of hardware platforms commonly used in TinyML applications is provided to highlight trade-offs and deployment considerations.

3) Identification of Research Challenges and Gaps: This review highlights key challenges in deploying person detection models on ultra-low-power devices, including constraints related to computational resources, generalization across diverse environmental conditions, lack of standardized evaluation benchmarks, and privacy concerns.

4) Direction for Future Research and Innovation: This review outlines a roadmap for future research, emphasizing the need for adaptive models, efficient federated learning, multi-sensor fusion, privacy-preserving techniques, and standardized datasets and evaluation frameworks. These insights aim to guide researchers toward building scalable, robust, and ethical TinyML-based person detection systems.

5) Inclusion of State-of-the-Art Studies: This review integrates findings from the most recent academic and industry contributions to ensure a current and forward-looking perspective on the evolving TinyML ecosystem.

The remainder of this paper is organized as follows: Section II describes the methodology for conducting this systematic review, including the criteria for selecting and analyzing relevant studies, as well as the technical requirements of person detection. Section III provides background on TinyML and the person detection task. Section IV categorizes and analyzes current approaches in terms of models, optimization methods, hardware platforms, and datasets. Section V discusses technical challenges, research gaps, and future directions. Finally, Section VI concludes the paper by summarizing findings and implications for future work.

II. Research Methodology

This section outlines the systematic approach used to identify, select, and analyze relevant literature on TinyML-based person detection systems. The review methodology is designed to ensure transparency, reproducibility, and comprehensive coverage of state-of-the-art research. A systematic literature review is a structured method used to identify, evaluate, and synthesize all relevant research on a specific topic to answer predefined research questions (RQs). To effectively assess the breadth of available evidence, a systematic and transparent approach is essential. Such reviews aim to provide a comprehensive and unbiased evaluation of a research domain through a rigorous, replicable, and auditable methodology. This approach addresses the need for researchers to consolidate existing knowledge on a given topic reliably and impartially. As a result, systematic reviews enable the formulation of generalizable conclusions beyond those of individual studies, offer insights into the current state of the field, and serve as a foundation for future investigations.

To conduct the SLR on TinyML-based person detection, a structured and rigorous approach was employed to ensure the completeness and reliability of the findings. Specifically, the methodology consisted of the following steps:

A. Research Questions Formulation

The review was structured around clearly defined research questions, which aimed to investigate the predominant approaches, optimization techniques, and existing challenges associated with TinyML-based person detection. The goal of this review is to address the following research questions (RQs):

1) RQ1: What are the key limitations and challenges in implementing TinyML-based person detection on resource-constrained devices?

2) RQ2: Which TinyML models are employed for person detection?

3) RQ3: What state-of-the-art techniques are applied in TinyML for person detection?

4) RQ4: Which performance metrics are most commonly reported?

5) RQ5: What datasets and benchmarks are commonly used for training and evaluating TinyML-based person detection models?

6) RQ6: Which hardware platforms are utilized for TinyML-based person detection, and what are their specific constraints?

7) RQ7: What hardware constraints are typically addressed, and what strategies are used to manage them?

8) RQ8: What future research directions and potential improvements have been proposed?

These questions guide the literature search and help frame the synthesis and discussion of findings.

B. Search Strategy

A comprehensive literature search was conducted using the following academic databases: IEEE Xplore, ACM Digital Library, SpringerLink, Scopus, Web of Science, and Google Scholar. The search covered publications between 2014 and 2024, reflecting the emergence and growth of TinyML research over the past decade.

Search queries included combinations of keywords such as: “TinyML, ” “person detection, ” “human presence recognition, ” “embedded object detection, ” “lightweight CNN, ” “on-device inference, ” “quantization, ” “pruning, ” “microcontroller, ” “edge AI, ” and “ultra-low-power vision.”

For example, one query used was “TinyML person detection” AND “edge computing” AND “low-power. ” To ensure the inclusion of recent advancements, the search was limited to articles published between 2014 and 2024. The strategy involved using Boolean operators to combine keywords, such as ("TinyML" OR "Tiny Machine Learning") AND ("person detection" OR "human detection") AND ("low-resource" OR "microcontroller" OR "IoT"), and incorporating synonyms or related terms like “embedded AI” or “edge AI” to broaden the scope of the search.

C. Inclusion and Exclusion

Studies were selected based on the predefined inclusion and exclusion criteria presented in Table I.

D. Screening and Selection Process

The study selection process was conducted in three stages to ensure the inclusion of high-quality and relevant literature. First, an initial screening reviewed the titles and abstracts of all identified studies to assess their relevance to TinyML in person detection systems. Next, a full-text review was performed on the selected studies to verify that they met the predefined inclusion criteria. Finally, a quality assessment evaluated each study’s methodological rigor, relevance to the research objectives, and completeness of reported data.

The study selection process in a SLR is a critical step to ensure the inclusion of relevant, high-quality, and focused papers [19]. The typical steps involved in this process, tailored to the domain of TinyML person detection, are outlined below. Initially, the literature search identified 132 research papers, as shown in Fig. 1. Subsequently, after reviewing titles, abstracts, and keywords, and removing duplicates, 79 papers were selected for further evaluation. Then, following the application of inclusion and exclusion criteria during the full-text review, a total of 50 papers were selected for detailed analysis in this survey. Overall, Fig. 1 illustrates the systematic process of literature selection conducted for this survey on TinyML and its related domains.

TABLE I
INCLUSION AND EXCLUSION CRITERIA FOR STUDY SELECTION

Inclusion criteria	Exclusion criteria
Articles focused on person detection using TinyML techniques.	General object detection studies without specific focus on person detection.
Evaluated models or algorithms on resource-constrained hardware platforms.	Papers lacking experimental validation or real-world implementation.
Provided performance metrics such as accuracy, latency, or energy efficiency.	Non-English publications unless translations are accessible.
Studies presenting models optimized for low-power or resource-constrained devices.	Studies not related to TinyML or person detection.
Papers published within a specific time frame (e.g., last 5–10 years).	Papers with only theoretical or simulation results without real-world application.
Peer-reviewed papers and reputable conferences (e.g., IEEE, ACM).	Studies focused on generic machine learning or AI but not on TinyML.
Papers focusing on TinyML applications for person detection.	Research addressing person detection on high-resource devices like GPUs or cloud platforms.
Studies involving low-resource hardware platforms like microcontrollers or IoT devices.	Publications without experimental validation or clear methodological details.
Research addressing hardware constraints (e.g., memory, power, computational resources).	Duplicates or papers with incomplete data.
Publications that propose or evaluate machine learning models suitable for resource-constrained environments.	
Papers published in peer-reviewed journals, conferences, or reputable technical platforms.	
Studies that provide benchmark datasets, metrics, or experimental evaluations.	

E. Data Extraction and Final Inclusion

A standardized data extraction form was employed to systematically collect key information from each study. This included publication details such as authors, year, and venue; the techniques or algorithms applied, including models like YOLO, MobileNet, and CNN; and dataset information covering widely used benchmarks like MS COCO, Pascal VOC, or custom datasets. Additionally, data were gathered on the model architectures and applied optimization techniques as well as the hardware platforms used for evaluation, for example, microcontrollers and edge AI chips. Furthermore, reported performance metrics such as accuracy, inference latency, and energy consumption were also collected to enable comprehensive analysis.

F. Data Synthesis and Analysis

The extracted data were synthesized to identify trends, categorize approaches, and compare performance metrics.

Quantitative metrics were aggregated for comparative analysis, while qualitative insights were drawn to highlight existing challenges and emerging opportunities.

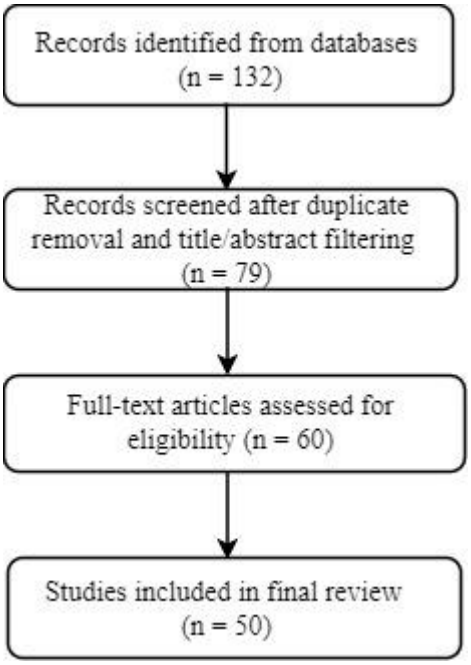

Fig. 1. Illustrates the screening process

The findings from the reviewed studies were organized into five key themes. First, Techniques and Algorithms revealed a broad range of TinyML approaches for person detection, including lightweight convolutional neural networks (CNNs), quantized models, and pruning strategies aimed at reducing model size and computational overhead. Second, Datasets and Benchmarking highlighted the frequent use of standard datasets such as MS COCO and Pascal VOC, while also exposing a significant gap in publicly available, real-world, small-scale datasets tailored for embedded person detection. Third, the theme of Challenges emphasized persistent issues such as limited memory capacity, constrained processing power, high inference latency, and the need for ultra-low power consumption on edge devices. Fourth, Hardware and Software Optimization explored commonly used microcontrollers and edge AI platforms, along with supporting frameworks like TensorFlow Lite for Microcontrollers and CMSIS-NN, which facilitate deployment on resource-limited hardware. Lastly, the theme of Future Directions outlined open research challenges, including improving model robustness under varying environmental conditions, enhancing detection accuracy without increasing resource demands, and minimizing computation for real-time, on-device inference.

G. Reliability and Validity

To enhance reliability, the study selection and data extraction processes were independently reviewed by multiple researchers. Any discrepancies were subsequently resolved through discussion in order to reach consensus and uphold methodological rigor. Moreover, the purpose of this literature review on TinyML is to capture the current state of development in TinyML applications, particularly those focused on person detection systems. To achieve this goal, it is crucial to collect data on various aspects, including the hardware platforms used, available memory, power consumption, software frameworks, and implemented algorithms. The insights obtained from this review will, therefore, serve as a foundation for the design and development of an efficient and reliable person detection system leveraging TinyML technology.

III. BACKGROUND

This section provides foundational context for the two core pillars of this review: Tiny Machine Learning (TinyML) and person detection. It outlines their definitions, relevance, and the motivation for their integration in embedded AI systems.

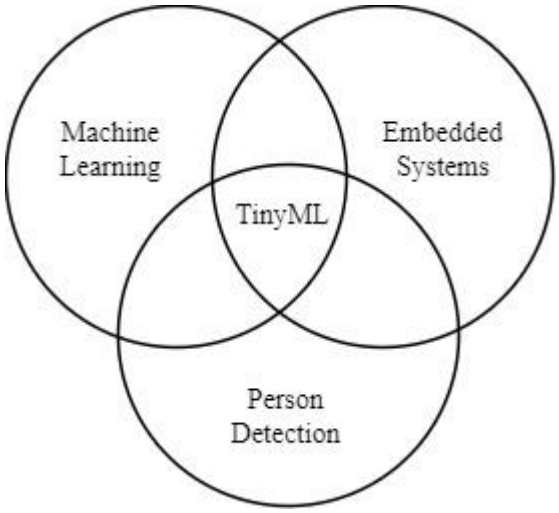

Fig. 2. Applying TinyML

A. Tiny Machine Learning (TinyML)

TinyML is a rapidly emerging subset of machine learning that focuses on deploying trained models on microcontrollers and other highly constrained embedded devices, typically with limited memory (in the range of tens to hundreds of kilobytes) and power consumption in the milliwatt range. The key advantage of TinyML lies in its ability to enable intelligent inference at the edge—without the need for continuous cloud connectivity—thereby reducing latency, improving privacy, and supporting real-time responses in distributed systems.

To operate effectively within the constraints of such hardware, TinyML relies heavily on [20]:

- Model compression techniques, such as quantization and pruning, to reduce model size and memory footprint;
- Lightweight neural architectures, such as MobileNet, Tiny-YOLO, and SqueezeNet, which balance efficiency and accuracy;
- Specialized deployment toolchains, including TensorFlow Lite for Microcontrollers (TFLM), CMSIS-NN, and Edge Impulse, to optimize inference at runtime.

Fig. 2 illustrates the interdisciplinary nature of Tiny Machine Learning (TinyML) within the context of person detection systems. TinyML exists at the intersection of three core domains: Machine Learning, Embedded Systems, and Person Detection. From a machine learning perspective,

TinyML leverages lightweight models and neural networks designed for inference on low-power devices. Embedded systems contribute the hardware and system-level constraints that shape how these models are deployed, typically on microcontrollers, SoCs, or edge devices with limited memory and computational capacity. The third domain, person detection, defines the target application, focusing on the recognition of human presence or activity through sensor data, often under real-time and privacy-sensitive conditions. The convergence of these three areas enables the development of intelligent, resource-efficient, and context-aware systems that can operate autonomously at the edge without relying on cloud infrastructure. This integration highlights the need for optimized algorithms, efficient hardware design, and robust sensing strategies to meet the stringent demands of TinyML-based person detection applications.

B. Person Detection

Person detection refers to the task of identifying human presence in various input modalities, including RGB images, video frames, and non-visual sensor data (e. g., thermal, radar, or PIR sensors). As a foundational function in computer vision and embedded AI, it enables a wide array of context-aware applications across smart environments, surveillance, healthcare, robotics, and automotive systems.

Traditionally, person detection has been addressed using large-scale convolutional neural networks (CNNs) such as YOLO, SSD, and Faster R-CNN, which provide robust accuracy but demand substantial computational resources. However, these models are impractical for resource-constrained embedded platforms due to their size, memory requirements, and power draw. [21].

To enable on-device person detection under TinyML constraints, several innovations have been adopted:

- Model simplification through the use of compact CNNs or binary neural networks (BNNs);
- Non-visual sensing techniques, such as passive infrared (PIR), ultrasonic, and low-resolution thermal imaging, which reduce input dimensionality;
- Hybrid architectures that combine classical computer vision techniques (e.g., Haar cascades) with lightweight deep learning inference for enhanced robustness.

Furthermore, sensor fusion strategies that combine inputs from visual and non-visual sources can improve detection accuracy under adverse conditions such as poor lighting, occlusion, or camera obstruction [22].

In the context of embedded AI, person detection must meet stringent performance benchmarks: high inference speed, low memory usage, real-time processing, and energy efficiency. Thus, optimizing person detection for TinyML platforms is an essential research challenge that drives innovation in edge-AI model design and hardware-aware learning.

IV. KEY RESULTS AND ANALYSIS

The sources for this review comprise 50 peer-reviewed journal articles and conference papers, which were identified through the methodology previously outlined. Fig. 3 illustrates the distribution of research papers related to TinyML over the past decade, spanning from 2014 to 2024. Specifically, the bar chart depicts the number of research papers published each year within this period. The x-axis represents the years, whereas the y-axis indicates the number of papers published annually. From the chart, it is evident that research activity in TinyML was minimal between 2014 and 2018.

However, a gradual increase began in 2019, marked by the publication of two papers. This upward trend continued through 2020 and 2021, with four papers published in each of those years. Furthermore, a more significant rise in research output is observed from 2022 onward. The number of papers increased to eight in 2022, followed by eleven in 2023, and peaked at twelve in 2024. This trend clearly suggests a growing interest and accelerating research activity in the field of TinyML, particularly from 2019 onward. Therefore, it may be concluded that TinyML is an emerging or rapidly developing area of study, gaining increasing attention from the research community.

The reviewed literature reveals several key research focuses, including the development of lightweight model architectures, the implementation of optimization techniques such as quantization and pruning, and the adaptation of machine learning models to hardware-constrained environments, particularly microcontrollers [23] and edge devices [24]. TinyML-based person detection has witnessed considerable progress, driven by its applicability in domains such as smart IoT devices [25], security systems, and wearable technologies [26]. Consequently, TinyML has emerged as a transformative paradigm for enabling intelligent functionalities on ultra-low-power embedded systems. Despite these advancements, existing studies also exhibit several limitations that impede broader deployment and limit generalizability across diverse operational contexts. Therefore, a critical evaluation of the core studies is warranted to elucidate their contributions while also identifying the methodological and practical constraints that must be addressed to advance the field further.

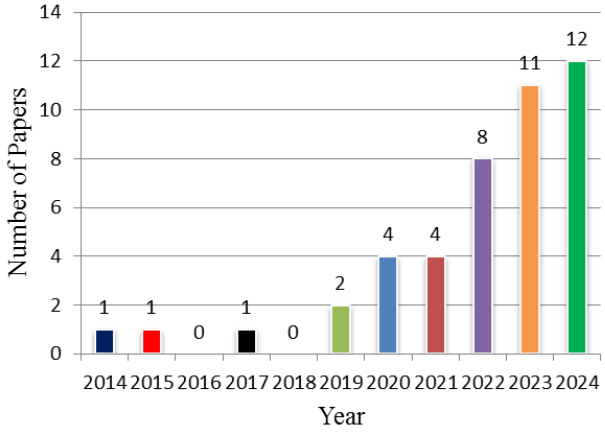

Fig. 3. Publication trend line

A. The Limitations and Challenges of Implementing Person Detection on TinyML Devices

Despite its significant potential, implementing person detection on TinyML devices presents several challenges [27], as illustrated in Fig. 4. One of the primary obstacles is memory and processing constraints. Microcontrollers

typically offer limited RAM and flash storage—often ranging from 64 to 512 KB of RAM and only a few megabytes of flash memory—which significantly restricts the size and complexity of models that can be deployed [28]. Moreover, TinyML devices are generally built around low-power microcontrollers, such as those from the ARM Cortex-M series, which operate at low clock speeds and lack dedicated hardware accelerators like GPUs or NPUs. As a result, the execution of complex neural networks becomes infeasible, necessitating extreme model compression techniques in order to maintain acceptable performance within these hardware limitations.

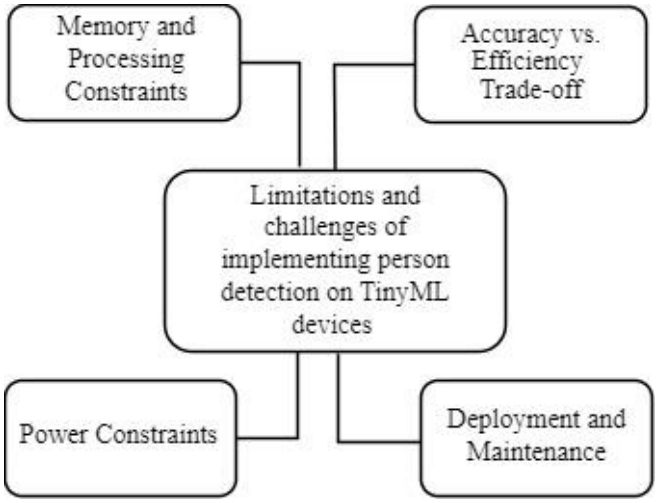

Fig. 4. Limitations and challenges

Power consumption is a critical concern for many TinyML devices, as they often rely on battery power; therefore, maintaining low power usage while performing frequent or real-time person detection poses significant challenges, particularly when utilizing cameras or processing continuous sensor streams. Consequently, person detection models must be highly efficient to prevent rapid battery depletion, especially in applications requiring continuous monitoring. Moreover, there is an inherent trade-off between model accuracy and efficiency: reducing model size to fit within hardware constraints often results in decreased detection accuracy, particularly under challenging conditions such as occlusions, varied poses, or complex backgrounds. Thus achieving robustness while maintaining efficiency remains a central challenge.

In addition, deployment and maintenance of models on TinyML devices present further difficulties due to limited connectivity and hardware interfaces, which complicate both initial deployment and subsequent updates in the field. Therefore, striking an optimal balance between accuracy, efficiency, and practical deployability is essential to enable effective and sustainable person detection on TinyML devices.

B. Several Lightweight Model for Person Detection

In the context of TinyML, lightweight models are essential for enabling person detection on devices constrained by limited computational resources, memory, and power. To address these constraints, several compact and efficient models have been developed specifically for deployment on TinyML devices. For instance, MobileNet variants (V1, V2, and V3) are widely adopted due to their use of depthwise separable convolutions, which provide an effective balance between accuracy and resource consumption. Similarly, Tiny-YOLO variants (such as Tiny-YOLOv3, v4, and v11) streamline the original YOLO architecture to support real-time person detection on low-power hardware. Moreover, models like SqueezeNet and ShuffleNet are designed to minimize model size and computational complexity, making them particularly well-suited for memory-constrained environments.

Additionally, Binary Neural Networks (BNNs), which rely on binary weights and activations, drastically reduce memory usage and computational demands, rendering them ideal for ultra-low-power applications. Furthermore, custom micro-CNNs and models optimized with CMSIS-NN are specifically tailored for ARM Cortex-M microcontrollers, achieving high efficiency in performing basic person detection tasks. Collectively, these lightweight models form the backbone of TinyML-enabled vision systems, where both computational efficiency and low power consumption are critical for practical deployment.

C. TinyML Leverages Several Techniques for Person Detection

Modern TinyML person detection techniques leverage lightweight neural network architectures specifically tailored for edge devices. Models such as MobileNet [29], EfficientNet-Lite, and YOLO-Tiny dominate the field, as they emphasize a balance between accuracy and computational efficiency. Furthermore, techniques such as quantization-aware training, model pruning, and hardware-specific optimizations significantly enhance the suitability of these models for resource-constrained environments, as illustrated in Table II. This table outlines several widely adopted methods aimed at improving model performance, reducing computational overhead, and facilitating deployment, particularly for edge and low-power devices. In addition, emerging approaches are increasingly exploring hybrid models that integrate traditional computer vision algorithms with neural networks to achieve improved real-time performance.

State-of-the-art techniques for enabling person detection in TinyML applications primarily focus on optimizing models for deployment on resource-constrained devices. To this end, model quantization and pruning are widely adopted to reduce model size and computational demands. These techniques achieve efficiency by converting models to lower-precision formats, such as 8-bit integers, and eliminating redundant parameters, all while preserving acceptable levels of accuracy. Moreover, efficient model architectures such as MobileNet, SqueezeNet, and Tiny YOLO are specifically engineered to function within the strict memory and power limitations of embedded systems, thereby making them particularly suitable for TinyML applications. In addition, on-device data processing techniques, including image filtering and resizing, are utilized to preprocess input data locally. This not only reduces the memory footprint but also alleviates the

computational burden during inference, further enhancing system efficiency.

TABLE II
TINYML TECHNIQUES FOR PERSON DETECTION

Technique	Description	Advantages	Limitations
Quantization	Reduces the precision of model weights and activations (e.g., 32-bit to 8-bit).	- Reduces memory and computational requirements. - Improves energy efficiency.	- May lead to a slight drop in accuracy. - Requires careful tuning.
Pruning	Removes redundant neurons or connections in the model. - Reduces model size and inference time. - Improves efficiency on edge devices.	- Can be challenging to implement. - May require retraining the model.	- It can be tricky to implement and often requires retraining the model to maintain accuracy.
Knowledge Distillation	Transfers knowledge from a larger, more accurate model to a smaller model	- Requires a pre-trained large model. - Reduces computational complexity. - Enables smaller models to achieve higher accuracy.	- Adds complexity to the training process. -Still needs resources during training
Model Architecture Optimization	Uses lightweight architectures like MobileNet, Tiny-YOLO, or SqueezeNet. - Designed for low-power devices.	- Balances accuracy and efficiency. - Smaller model size	- Limited by hardware constraints - May not achieve state-of-the-art accuracy.
Federated Learning	Trains models on decentralized data without sharing raw data.	- Enhances privacy and security. - Reduces data transfer costs. - Requires robust communication protocols.	- High burden on low-power TinyML devices - Local data may be biased or limited. - May increase training time.
Transfer Learning	Fine-tunes pre-trained models on specific person detection datasets. - Reduces training time and data requirements. - Improves accuracy for specific tasks.	- Requires a relevant pre-trained model. - May not generalize well to new domains.	- Pretrained models may still be too large - Still too intensive for most MCUs
On-device Training	Enables training or fine-tuning models directly on edge devices.	- Adapts to new data in real-time. - Reduces dependency on cloud services.	- Limited by device resources. - Increases energy consumption
Hybrid Models	Combines TinyML with traditional computer vision techniques (e.g., Haar cascades).	- Improves robustness and accuracy. - Leverages the strengths of both methods.	- Increases implementation complexity. - May require more computational resources.

D. Evaluation Metrics

Evaluation metrics for person detection are essential for assessing the performance, efficiency, and suitability of models, particularly in resource-constrained environments such as TinyML applications. Common accuracy-based metrics include precision, recall, and mean average precision (mAP), which evaluate how effectively a model identifies and localizes human figures while minimizing false positives and false negatives. Additionally, inference latency measures the time required for the model to make a prediction, a critical factor for real-time applications. Frames per second (FPS) indicate how many images the system can process per second, directly reflecting responsiveness.

Model size (measured in KB or MB) and memory footprint are crucial considerations in embedded deployments, where storage capacity and RAM are limited. Power consumption is another key metric, especially for battery-powered or always-on systems, and is typically measured in milliwatts (mW). Collectively, these metrics provide a holistic view of both the effectiveness and practicality of person detection models, thereby guiding developers in selecting and optimizing models for real-world edge deployments [30].

E. Datasets and Benchmarks

For training and evaluating TinyML person detection systems, researchers often utilize datasets and benchmarks that specifically focus on human detection while being compatible with the constraints of TinyML, such as low resolution, fewer labeled classes, and smaller dataset sizes, as illustrated in Fig. 5 Commonly used datasets for TinyML person detection include MLPerf Tiny [31], COCO [32], Visual Wake Words [33], Wake Vision [34], as well as proprietary datasets tailored to specific hardware constraints, as shown in Table III. This table compares several popular datasets and benchmarks used for training and evaluating person detection models, particularly within the context of TinyML on resource-constrained embedded and edge devices. Each dataset caters to different use cases and offers its own set of advantages and limitations, thereby guiding researchers in selecting the most appropriate resources for their specific applications.

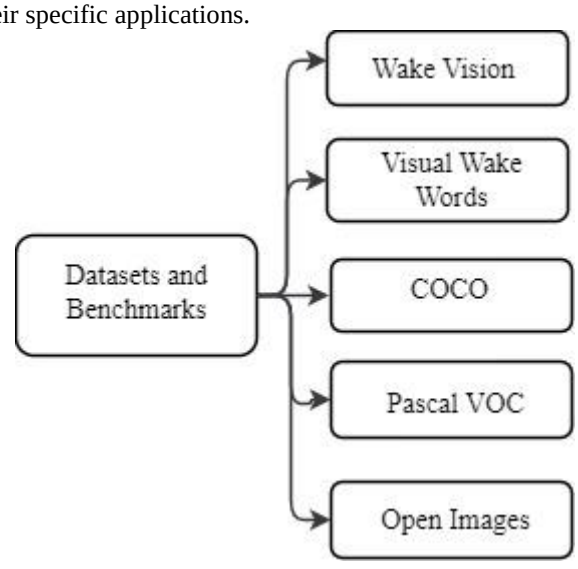

Fig. 5. Datasets and Benchmarks for person detection

In TinyML applications, the choice of dataset is critical. While large-scale datasets such as COCO and Open Images provide rich annotations and diverse data, they often demand extensive preprocessing and computational resources that exceed the capabilities of TinyML platforms. Conversely, smaller, purpose-built datasets like Wake Vision and Visual Wake Words offer a more practical foundation for training efficient models suitable for edge deployment. Additionally, custom datasets can provide domain-specific advantages; however, they require significant manual effort to create and maintain. Therefore, researchers must carefully balance model complexity, resource constraints, and specific application requirements when selecting an appropriate dataset.

Despite their usefulness, these datasets often lack the diversity or scale required for effective TinyML applications. Moreover, the absence of publicly available, standardized benchmarks that accurately reflect real-world scenarios in resource-constrained environments represents a critical gap in the field. To address this, efforts are currently underway to develop datasets specifically designed for edge AI applications, such as Edge AI Bench. Below are some of the most frequently used datasets.

A significant contribution to this field is the introduction of the Wake Vision dataset [34], a large-scale and diverse collection specifically tailored for TinyML person detection. Wake Vision comprises over 6 million images, representing a substantial increase compared to previous datasets, and has undergone thorough quality filtering. The dataset includes two training sets: Wake Vision (Large) and Wake Vision (Quality), with the latter offering higher-quality labels. Studies have demonstrated that models trained on Wake Vision (Quality) achieve greater accuracy, underscoring the importance of label quality in training data. Additionally, Wake Vision provides five detailed benchmark sets to evaluate model performance in challenging real-world scenarios, such as varying lighting conditions and distances from the camera.

The COCO (Common Objects in Context) dataset is one of the most widely used benchmarks for object detection [32], segmentation, and keypoint detection, offering a diverse collection of images labeled across multiple object categories, including people. However, in the context of TinyML, the dataset's large size often necessitates downsampling or selecting only the "person" class to align with the memory and processing limitations of edge devices. Moreover, researchers may crop or resize images further to accommodate the constraints of TinyML platforms. Despite these adaptations, COCO remains a valuable benchmark due to its rich variety of scenes, lighting conditions, and human poses, which helps evaluate model generalizability, although only subsets of the dataset are practical for TinyML applications.

The PASCAL Visual Object Classes (VOC) dataset [35] is a foundational benchmark in object detection, providing labeled images across various object categories, including "person." Compared to COCO, VOC contains fewer images and features simpler annotations, which makes it particularly suitable for TinyML applications. Its smaller dataset size and straightforward labeling structure facilitate proof-of-concept development and enable rapid model deployment on resource-constrained devices. Consequently, VOC is frequently used as a baseline in TinyML research, allowing for quicker training and inference on edge hardware while still providing meaningful performance evaluation.

TABLE III
THE DATASETS AND BENCHMARKS FOR TINYML-BASED PERSON DETECTION SYSTEMS

Dataset /Benchmark	Description	Use Cases	Limitations
Wake Vision [34]	It is a state-of-the-art person detection dataset specifically created for TinyML applications.	- Training and evaluating machine learning models for efficient person detection on embedded and edge devices. - Wake Vision also includes a fine-grain benchmark suite for evaluating the robustness of TinyML models.	- Large dataset size - Limited Diversity
COCO [32]	A large-scale dataset with 80 object categories, including person detection.	- General object and person detection. - Benchmarking TinyML models.	- Large dataset size may not be suitable for TinyML training. - Requires significant preprocessing. - High computational resources needed for training.
Visual Wake Words [33]	Identifying whether a person is present in the image or not.	- Provides a realistic benchmark for tiny vision models.	- Simplistic Classifications - Lack of Diversity - Small for training larger, more complex models.
Pascal VOC [35]	A dataset with 20 object categories, including person detection.	- Benchmarking lightweight models. - Transfer learning for TinyML.	- Smaller dataset size may limit training diversity. - Less complex than COCO.
Open Images Dataset [36]	A large dataset with over 600 object categories, including person detection.	- Training and evaluating TinyML models. - Multi-class detection tasks.	- Large dataset size may not be suitable for TinyML. - Requires significant computational resources.
Custom Datasets	Domain-specific datasets created for specific applications (e.g., industrial, home).	- Tailored to specific use cases. - Improves accuracy in targeted environments.	- Requires significant effort to collect and annotate. - May lack diversity.

The Open Images Dataset (OID) [36] is a large-scale collection containing millions of images annotated with bounding boxes and labels for a wide range of object classes, including "person." It is significantly larger than both COCO and VOC but includes pre-labeled subsets that facilitate focused use in person detection tasks. In TinyML applications, researchers typically extract and preprocess specific subsets containing human figures, often downsampling or cropping images to meet the memory and computational constraints of edge devices. Despite the need for preprocessing, OID's scale and diversity make it a valuable resource for robust benchmarking and for evaluating the generalization performance of TinyML models.

Several Edge AI companies, such as Google, have released specialized datasets specifically designed for person detection in embedded environments, with Google's person detection dataset being a notable example. These datasets are optimized for real-world TinyML applications by featuring low-resolution images and minimal feature sets suitable for microcontroller-based deployment. Moreover, their preprocessed nature aligns well with the memory, processing, and power limitations of TinyML devices, thereby eliminating the need for extensive adaptation. Consequently, these datasets serve as highly relevant benchmarks, providing a realistic assessment of model performance in actual embedded and edge scenarios, and are instrumental in advancing practical TinyML solutions.

For TinyML, the challenge lies in finding datasets that balance richness, which ensures model robustness, with simplicity, which accommodates hardware constraints. Common approaches include using subsets of larger datasets, downsizing images, and creating custom datasets tailored to the deployment environment. By benchmarking TinyML person detection models on these datasets, researchers can effectively assess model performance in real-world, low-power applications. Consequently, this process provides a solid foundation for improving and deploying person detection systems on TinyML platforms.

F. Hardware Platforms

TinyML person detection relies on highly resource-constrained hardware platforms, typically microcontroller-based, which prioritize low power consumption and limited processing capabilities. These platforms are specifically designed to perform real-time inference while maintaining a balance between memory usage, computational demands, and energy efficiency [37]–[40]. Table IV presents a comparative overview of the most commonly used hardware platforms for TinyML person detection, highlighting their individual constraints and characteristics.

Each hardware platform varies in terms of computational power, memory capacity, connectivity options, and suitability for specific machine learning workloads. Microcontrollers such as the Arduino Nano 33 BLE Sense, ESP32, and STM32 are well suited for lightweight, low-power ML tasks, especially in IoT and wearable applications [41]. In contrast, more powerful platforms like the Jetson Nano provide significantly higher computational capabilities, making them suitable for real-time inference with more complex models. However, this comes with increased power consumption and cost. Therefore, selecting the appropriate platform depends on various application-specific requirements, including model complexity, latency tolerance, connectivity needs, power constraints, and budget considerations.

TABLE IV
THE HARDWARE PLATFORMS USED FOR TINYML-BASED PERSON DETECTION

Hardware Platform	CPU / Memory	Power Use	ML Support	Suitability
Arduino Nano 33 BLE Sense	ARM Cortex-M4 / 256 KB RAM	< 0.5 W	TensorFlow Lite Micro	Basic tasks, wearable devices
ESP32	Dual-core 240 MHz / 520 KB RAM	~1 W	Edge Impulse, TFLM	Low-power IoT detection
STM32 H7	Cortex-M7 / 1–2 MB RAM	< 1 W	CMSIS-NN	Intermediate TinyML tasks
Jetson Nano	Quad-core CPU + 128-core GPU/ up to 8 GB RAM.	5–10 W	TensorRT, PyTorch, TFLite	Complex models with real-time requirements
Raspberry Pi 4	Quad-core / up to 8 GB RAM.	5–15 W	TFLite, OpenCV DNN	Flexible prototyping, high-resource models

The ARM Cortex-M series microcontrollers, including models such as Cortex-M4, M7, M33, and the more recent Cortex-M55 [42], are widely adopted in TinyML applications due to their strong balance between performance and power efficiency. These microcontrollers typically operate at frequencies of up to 480 MHz and offer memory capacities ranging from 128 KB to several megabytes of RAM and flash, making them well-suited for deploying lightweight, optimized machine learning models. Notably, the Cortex-M55 features ARM Helium technology, which introduces vector processing capabilities that significantly enhance ML performance. In addition, with ultra-low power consumption, typically under 1 watt, and support for deep sleep modes, Cortex-M devices are particularly ideal for low-power person detection systems in embedded environments. However, their constrained computational and memory resources present challenges for deploying more complex models. Consequently, techniques such as model quantization, pruning, and architecture tuning are essential to achieve efficient and effective operation on these platforms.

The ESP32 and ESP32-S series microcontrollers, developed by Espressif Systems, are popular low-cost platforms that feature integrated Wi-Fi and Bluetooth [43], making them well-suited for IoT and edge AI applications, including person detection. These devices typically include dual-core processors operating at up to 240 MHz, with limited onboard SRAM (approximately 520 KB) and support for external PSRAM (up to 8 MB) in some variants, along with flash storage of up to 4 MB. Their moderate

power consumption, combined with support for various sleep modes, enables energy-efficient operation, particularly in battery-powered scenarios. ESP32 devices are frequently used with lightweight person detection models, especially in connected systems where local inference is followed by communication with the cloud. However, the limited RAM significantly restricts the deployment of complex models, thus requiring aggressive optimization techniques such as quantization, pruning, and image downsampling to fit within the available hardware resources.

STM32 microcontrollers, developed by STMicroelectronics, encompass a versatile family of devices, including the STM32F, STM32H, and STM32L series [44]. These are designed to balance performance and power efficiency for embedded AI applications. With clock speeds reaching up to 400 MHz in the STM32H series and RAM ranging from 64 KB to several megabytes, these microcontrollers provide sufficient resources for deploying lightweight machine learning models. The integrated STM32Cube. AI toolchain offers a dedicated environment for converting and optimizing neural networks for STM32 hardware, thereby facilitating efficient TinyML deployment. The STM32L series, in particular, is known for its ultra-low power consumption, making it ideal for battery-powered person detection systems. Overall, these microcontrollers are well-suited for small-scale, simple person detection tasks. However, limited memory and processing power continue to pose challenges, especially when aiming for real-time performance with more complex model architectures. This necessitates the use of compact and highly optimized models.

The NVIDIA Jetson Nano is a compact yet powerful computing platform designed for edge AI applications [45]. It features a quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU, enabling the execution of more complex AI models compared to typical microcontrollers. With 4 GB of RAM and support for expandable storage, the Jetson Nano can accommodate higher-capacity models and offers greater deployment flexibility. While it delivers real-time performance and is particularly well-suited for person detection tasks in applications such as surveillance and smart monitoring, its power consumption, which ranges from 5 to 10 watts, makes it significantly more power-intensive than traditional TinyML platforms. As a result, it is less suitable for ultra-low-power or battery-operated scenarios. However, it excels in edge environments where moderate power availability is acceptable and high accuracy and speed are critical.

The Arduino Nano 33 BLE Sense [46], [47] is a compact microcontroller designed for TinyML and IoT applications. It features an ARM Cortex-M4 CPU running at 64 MHz and is equipped with onboard sensors such as an accelerometer, gyroscope, and microphone. With 256 KB of SRAM and 1 MB of flash storage, it offers sufficient resources for deploying highly compact neural networks, making it suitable for basic person detection tasks that do not require high-resolution input or complex model architectures. Its low power consumption and Bluetooth Low Energy (BLE) capability make it ideal for wearable and battery-powered devices. However, the limited memory and processing power pose significant challenges, necessitating the use of extremely small, quantized models and careful optimization to ensure efficient on-device inference.

The Raspberry Pi family [48], including the Raspberry Pi Zero and Raspberry Pi 4, is widely used in TinyML applications due to its affordability and flexible computing capabilities, particularly for prototyping and educational purposes. While the Raspberry Pi Zero features a single-core ARM CPU with 512 MB of RAM, making it suitable only for simple, low-power person detection tasks, the Raspberry Pi 4 offers a quad-core processor and up to 8 GB of RAM, enabling it to run more complex models and support real-time inference. Storage is typically handled via SD cards, providing ample space for larger datasets and models compared to microcontrollers. However, power consumption ranges from 5 to 15 W, which is higher than typical TinyML platforms and can present challenges in power- or thermally-constrained environments. Despite this, the Raspberry Pi platform remains a popular choice for person detection tasks that require a balance between computational capability and development flexibility, especially when moderate power usage is acceptable.

The primary constraints across these platforms include limited memory, low clock speeds, and strict power budgets. To run person detection models on such devices, researchers typically rely on model optimizations such as quantization, pruning, and using low-resolution input data. Platforms like ARM Cortex-M and ESP32 are ideal for ultra-low-power applications, whereas devices like the Jetson Nano and Coral Edge TPU cater to higher-performance needs where power consumption is less constrained. Ultimately, balancing accuracy, latency, and energy efficiency is key when selecting a hardware platform for TinyML person detection.

G. Hardware Constraints

TinyML enables machine learning models to run on resource-constrained devices, such as microcontrollers and IoT devices. These devices often encounter several hardware limitations, including limited memory, low computational power, restricted storage capacity, and stringent power efficiency requirements. Therefore, addressing these constraints is essential for the effective deployment of TinyML applications.

Among these challenges, memory constraints are particularly fundamental, as microcontrollers typically provide limited RAM and flash storage. This limitation restricts both the size of deployable models and the volume of data that can be processed simultaneously. To overcome this, several optimization strategies have been developed. First, model quantization, which reduces the precision of parameters (for example, from 32-bit floating-point to 8-bit integers), significantly lowers memory usage with minimal loss in accuracy. This technique is supported by frameworks such as TensorFlow Lite [49]. Second, model pruning reduces memory demands by eliminating unimportant weights or neurons from the network, thereby compressing the model. Moreover, employing efficient model architectures like MobileNet and SqueezeNet, which are specifically designed for low-memory environments, allows the deployment of capable models within the tight memory

budgets typical of TinyML platforms [50].

Computational power constraints represent a significant limitation in TinyML, as devices often operate at low clock speeds and lack the processing capacity required for complex computations. To address this challenge, researchers and developers have adopted several strategies. One effective approach is the use of efficient model architectures, such as MobileNet, which utilizes depthwise separable convolutions to drastically reduce the number of required computations. In addition, hardware-specific optimizations, such as the use of digital signal processing (DSP) extensions or hardware accelerators (when available), can significantly enhance inference speed on certain microcontrollers. Furthermore, a particularly promising solution is the deployment of binary neural networks (BNNs), which employ binary weights and activations. This approach not only minimizes computational complexity but also reduces memory usage, making BNNs especially well-suited for resource-constrained TinyML environments.

Storage constraints pose a critical challenge in TinyML, as embedded systems often have limited onboard storage for housing models and datasets. To address this issue, developers frequently use model compression techniques such as quantization and pruning, which significantly reduce model size while maintaining acceptable levels of accuracy. These methods enable complex models to fit within the limited flash memory typically available on microcontrollers. In cases where additional capacity is needed, external storage solutions, such as SD cards or flash modules, can be employed to extend available storage. However, this may introduce additional latency during model loading or data access, particularly in real-time applications.

Power efficiency is a critical concern in TinyML, as many devices are battery-powered and must operate for extended periods without frequent recharging or replacement. To address this challenge, developers adopt low-power model architectures, such as Tiny YOLO, which are specifically designed to perform inference efficiently while minimizing energy consumption. Additionally, strategies such as duty cycling and event-driven processing help conserve energy by keeping the device in a low-power state and activating it only in response to specific events or triggers, such as motion detection. Furthermore, many microcontrollers support ultra-low-power modes, which significantly reduce energy usage during idle periods. These approaches collectively enable the sustainable, long-term deployment of TinyML applications in power-constrained environments.

Latency constraints are a major challenge in TinyML, particularly for real-time applications such as person detection, where rapid response is essential despite limited computational resources. To mitigate latency issues, developers increasingly rely on Edge AI accelerators, which are specialized hardware components integrated into some microcontrollers and edge devices. These accelerators enable faster inference by offloading and accelerating AI computations. Additionally, on-device processing optimizations, such as fixed-point quantization, help reduce computational overhead and enhance execution speed. These strategies allow TinyML models to meet real-time performance requirements while still operating within the limitations of resource-constrained hardware.

Connectivity and data transmission constraints present significant challenges in TinyML systems, particularly in scenarios with limited or low-bandwidth connections. These limitations can hinder real-time communication and increase energy consumption, especially when large amounts of data are transmitted frequently. To address this, on-device processing is commonly employed, allowing for local inference so that only essential results, rather than raw data, are transmitted. Additionally, data compression techniques are used to reduce the size of the output before transmission, further minimizing bandwidth requirements. For wireless communication, low-power protocols such as Bluetooth Low Energy (BLE) are preferred, as they support efficient and energy-conscious data exchange, making them well suited for resource-constrained TinyML applications.

Addressing these hardware constraints requires a combination of model optimizations, efficient hardware utilization, and system-level strategies to ensure the effective deployment of TinyML applications.

V. FUTURE RESEARCH DIRECTIONS AND DISCUSSION

As TinyML continues to evolve, several promising directions can enhance the effectiveness, scalability, and robustness of person detection systems deployed on resource-constrained platforms. This section outlines key research opportunities and unresolved challenges that the academic and industrial communities must address to push the frontier of edge AI. While many existing studies provide valuable experimental results, they often lack insights into practical deployment scenarios, particularly regarding hardware heterogeneity, privacy implications, and ethical considerations [51]–[53].

One major area for future exploration is the development of advanced model compression techniques, such as quantization-aware training and pruning strategies, to reduce the computational and memory demands of person detection models without significantly sacrificing performance. In parallel, novel TinyML-specific architectures optimized for low-power microcontrollers and neuromorphic hardware are essential to support increasingly complex edge applications. Additionally, there is a growing need to integrate multi-sensor data, combining visual inputs with thermal, radar, or acoustic signals, to improve detection accuracy and reliability under challenging conditions such as low light or occlusions. Furthermore, adaptive and self-optimizing models that dynamically respond to changes in environmental conditions or hardware constraints represent a promising direction for enhancing model resilience and efficiency [54].

To further support scalability and reliability, collaborative edge computing, where multiple edge devices share computational workloads, can improve processing efficiency and fault tolerance. Federated learning frameworks, particularly those adapted for TinyML devices, enable distributed training while preserving data privacy and minimizing communication overhead. Standardization is another critical priority. Establishing shared benchmarks,

datasets, and evaluation protocols will facilitate consistent and fair comparisons across different TinyML systems, fostering collaborative development and accelerating progress in the field. Moreover, privacy and security must be treated as integral components of system design. Techniques such as differential privacy, on-device anonymization, and defenses against adversarial attacks are essential for ethical and secure deployments.

Looking ahead, hybrid optimization strategies that combine software-level improvements, such as efficient model architectures, with hardware-level innovations, including custom accelerators and energy-aware scheduling, will be key to meeting the strict constraints of edge environments. Additionally, the creation of diverse real-world datasets, capturing variability in lighting, occlusion, motion, and background clutter, will be essential for training models with greater generalizability.

In summary, TinyML for person detection represents a convergence of machine learning, embedded systems, and edge computing, offering the promise of intelligent, privacy-aware systems deployed ubiquitously in the real world. By addressing current gaps, particularly in scalability, adaptability, and ethics, future research can pave the way for more robust, efficient, and responsible TinyML solutions with broad societal impact.

VI. CONCLUSION

Tiny Machine Learning (TinyML) has emerged as a transformative paradigm that enables intelligent data processing directly on ultra-low-power and memory-constrained devices. As the demand for real-time, privacy-aware, and energy-efficient systems continues to grow, the integration of TinyML into person detection applications has become both a critical challenge and a compelling opportunity.

This paper presented a systematic literature review of TinyML-based person detection systems published between 2014 and 2024. Through the analysis of 50 peer-reviewed studies, we identified and categorized a wide range of lightweight neural network architectures, optimization techniques, datasets, and hardware platforms suitable for deploying person detection on the edge. The review highlighted prominent models such as MobileNet, Tiny-YOLO, and SqueezeNet, as well as key techniques including quantization, pruning, and knowledge distillation. It also examined deployment across various microcontrollers and single-board computers, including the ARM Cortex-M family, ESP32, STM32, Jetson Nano, and Raspberry Pi.

Despite notable progress, several open challenges remain. These include the limited availability of standardized benchmarks for embedded settings, trade-offs between model accuracy and resource usage, and the difficulty of achieving robustness in diverse environmental conditions. Additionally, power consumption, latency, and memory constraints continue to shape the design and deployment of TinyML models.

Looking ahead, future research must focus on adaptive architectures, federated and on-device learning, multi-sensor fusion, and privacy-preserving inference strategies to enhance system resilience and scalability. Furthermore, collaborative development of standardized evaluation protocols and diverse real-world datasets will be crucial for advancing reproducibility and generalizability in the field.

In conclusion, this review consolidates the current state of TinyML for person detection, identifies existing research gaps, and offers actionable insights for researchers and practitioners. As TinyML technologies mature, their role in enabling intelligent, secure, and efficient person-aware systems at the edge is poised to become increasingly central across a wide range of real-world applications.

REFERENCES

[1] Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., ... & Yadav, P, "Benchmarking tinyml systems: Challenges and direction," arXiv preprint arXiv:2003.04821. 2020.

[2] Tang, Fan, Fang Yang, and Xianqing Tian, "Long-distance person detection based on YOLOv7," Electronics 12.6,1502, 2023.

[3] Xu, K., Zhang, H., Li, Y., Zhang, Y., Lai, R., & Liu, Y, "An ultra-low power tinyml system for real-time visual processing at edge," IEEE Transactions on Circuits and Systems II: Express Briefs, 70(7), 2640-2644, 2023.

[4] Chen, Y., Zheng, B., Zhang, Z., Wang, Q., Shen, C., & Zhang, Q, "Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions," ACM Computing Surveys (CSUR), 53(4), 1-37, 2020.

[5] Banbury, C., Zhou, C., Fedorov, I., Matas, R., Thakker, U., Gope, D., ... & Whatmough, P, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," Proceedings of machine learning and systems, 3, 517-532, 2021.

[6] Ray, Partha Pratim, "A review on TinyML: State-of-the-art and prospects," Journal of King Saud University-Computer and Information Sciences 34.4, 1595-1623, 2022.

[7] Vandersteegen, M., Reusen, W., Beeck, K. V., & Goedemé, T, "Person Detection Using an Ultra Low-resolution Thermal Imager on a Low-cost MCU," In International Conference on Image and Vision Computing New Zealand. Cham: Springer Nature Switzerland, 33-47, 2022.

[8] Yu, X., Gong, Y., Jiang, N., Ye, Q., & Han, Z, "Scale match for tiny person detection," In Proceedings of the IEEE/CVF winter conference on applications of computer vision, 1257-1265, 2020.

[9] Jiang, N., Yu, X., Peng, X., Gong, Y., & Han, Z, "SM+: Refined scale match for tiny person detection," In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1815-1819, 2021.

[10] Pimpalkar, Anway S., and Deeplaxmi V. Niture, "Towards Contactless Elevators with TinyML using CNN-based Person Detection and Keyword Spotting," arXiv preprint arXiv:2405,13051,2024.

[11] Pavan, M., Caltabiano, A., & Roveri, M, "TinyML for UWB-radar based presence detection," In 2022 International Joint Conference on Neural Networks (IJCNN), 1-8, 2022.

[12] Tsoukas, V., Gkogkidis, A., Boumpa, E., & Kakarountas, A, "A Review on the emerging technology of TinyML," ACM Computing Surveys, 56(10), 1-37, 2024.

[13] Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S, "A comprehensive survey on tinyml," IEEE Access, 11, 96892-96922, 2023.

[14] Yang, Y., Han, X., Wang, K., Yu, X., Yu, W., Wang, Z., ... & Jiao, J, "NRPerson: A Non-Registered Multi-Modal Benchmark for Tiny Person Detection and Localization," Electronics, 13(9), 1697, 2024.

[15] Guo, L., Liu, H., Pang, Z., Luo, J., & Shen, J, "Optimizing YOLO Algorithm for Efficient Object Detection in Resource-Constrained Environments," In 2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI), 1358-1363, 2024.

[16] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[17] Elgendy, Mohamed, "Deep learning for vision systems. Manning," 2020.

[18] Yue, M., Zhang, L., Huang, J., & Zhang, H, "Lightweight and efficient tiny-object detection based on improved YOLOv8n for UAV aerial images," Drones, 8(7), 276, 2024.
[19] Han, H., & Siebert, J, "TinyML: A systematic review and synthesis of existing research," In 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 269-274, 2022.
[20] Tsoukas, V., Gkogkidis, A., Boumpa, E., & Kakarountas, A, "A Review on the emerging technology of TinyML," ACM Computing Surveys, 56(10), 1-37, 2024.
[21] Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y, "Review of image classification algorithms based on convolutional neural networks," Remote Sensing, 13(22), 4712, 2021.
[22] Hao, Y. W., Min, C. C., Sabri, R. M., Fat, R. L. T., Wei, S. H., Halim, H., & Fadhlullah, S. Y, "Implementation Research of Tiny ML on RISC-V Platform Using Efinix Platform for Real Time Person Detection," In 2025 21st IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 233-238, 2025.
[23] Immonen, R., & Hämäläinen, T, "Tiny Machine Learning for Resource-Constrained Microcontrollers," Journal of Sensors, 2022(1), 7437023, 2022.
[24] Alajlan, N. N., & Ibrahim, D. M, "TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications," Micromachines, 13(6), 851, 2022.
[25] Kim, K., Jang, S. J., Park, J., Lee, E., & Lee, S. S, "Lightweight and energy-efficient deep learning accelerator for real-time object detection on edge devices," Sensors, 23(3), 1185, 2023.
[26] Elhanashi, A., Dini, P., Saponara, S., & Zheng, Q, "Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices," Electronics, 13(17), 3562, 2024.
[27] Lin, J., Zhu, L., Chen, W. M., Wang, W. C., & Han, S, "Tiny machine learning: Progress and futures [feature]," IEEE Circuits and Systems Magazine, 23(3), 8-34, 2023.
[28] Lin, J., Zhu, L., Chen, W. M., Wang, W. C., Gan, C., & Han, S, "On-device training under 256kb memory," Advances in Neural Information Processing Systems, 35, 22941-22954, 2022.
[29] Capogrosso, L., Cunico, F., Cheng, D. S., Fummi, F., & Cristani, M, "A machine learning-oriented survey on tiny machine learning," IEEE Access, 12, 23406-23426, 2024.
[30] Mittal, P, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," Artificial Intelligence Review, 57(9), 242, 2024.
[31] Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., ... & Xuesong, X, "Mlperf tiny benchmark," arXiv preprint arXiv:2106.07597, 2021.
[32] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L, "Microsoft coco: Common objects in context," In Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, Springer International Publishing, proceedings, part v 13,740-755, 2014.
[33] Chowdhery, A., Warden, P., Shlens, J., Howard, A., & Rhodes, R, "Visual wake words dataset," arXiv preprint arXiv:1906.05721, 2019.
[34] Banbury, C., Njor, E., Garavagno, A. M., Stewart, M., Warden, P., Kudlur, M., ... & Reddi, V. J, "Wake Vision: A Tailored Dataset and Benchmark Suite for TinyML Computer Vision Applications," arXiv preprint arXiv:2405.00892, 2024.
[35] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A, "The pascal visual object classes challenge: A retrospective," International journal of computer vision, 111, 98-136, 2015.
[36] Google Research, "Open Images Dataset," [Online]. Available:https://storage.googleapis.com/openimages/web/index.html , Accessed June 3, 202.
[37] Alajlan, N. N., & Ibrahim, D. M, "DDD TinyML: a TinyML-based driver drowsiness detection model using deep learning," Sensors, 23(12), 5696, 2023.
[38] Moosmann, J., Giordano, M., Vogt, C., & Magno, M, "Tinyissimoyolo: A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers," In 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), 1-5, 2023.
[39] El Mrabet, A., Tber, A., Benaly, M., Hlou, L., & El Gouri, R, "Unlocking doors: A tinyml-based approach for real-time face mask detection in door lock systems," Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 11(2), 515-527, 2023.
[40] Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., & López, O. L, "TinyML: Tools, applications, challenges, and future research directions," Multimedia Tools and Applications, 83(10), 29015-29045, 2024.
[41] Chang, Y. H., Wu, F. C., & Lin, H. W, "Design and implementation of esp32-based edge computing for object detection" Sensors, 25(6), 1656, 2025.
[42] Maciá-Lillo, A., Barrachina, S., Fabregat, G., & Dolz, M. F, "Optimising Convolutions for Deep Learning Inference On ARM Cortex-M Processors," IEEE Internet of Things Journal, 2024.
[43] Espressif Systems, "ESP32 Series," Espressif, [Online]. Available: https://www.espressif.com/en/products/socs/esp32, Accessed: June 3, 2025.
[44] STMicroelectronics, "STM32 32-bit Arm Cortex MCUs," STMicroelectronics, [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html, Accessed: June 3, 2025.
[45] NVIDIA, "Jetson Nano Product Development," NVIDIA, [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, Accessed: June 3, 2025.
[46] Arduino, "Nano 33 BLE Sense," Arduino Documentation, [Online]. Available: https://docs.arduino.cc/hardware/nano-33-ble-sense/, Accessed: June 3, 2025.
[47] Warden, Pete, and Daniel Situnayake, "Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers," O'Reilly Media, 2019.
[48] Raspberry Pi, "Raspberry Pi Product Series Explained," Raspberry Pi, May 6, 2024. [Online]. Available: https://www.raspberrypi.com/news/raspberry-pi-product-series-explained/, Accessed: June 3, 2025.
[49] TensorFlow, "Person Detection with TensorFlow Lite Micro," TensorFlow Documentation, [Online]. Available: https://www.tensorflow.org/hub/tutorials/object_detection, Accessed: June 3, 2025.
[50] Lokhande, H., & Ganorkar, S. R, "Object detection in video surveillance using MobileNetV2 on resource-constrained low-power edge devices," Bulletin of Electrical Engineering and Informatics, 14(1), 357-365, 2025.
[51] Seng, Kah Phooi, and Li-Minn Ang, "Embedded intelligence: State-of-the-art and research challenges," IEEE Access, 59236-59258, 2022.
[52] Lê, M. T., Wolinski, P., & Arbel, J, "Efficient neural networks for tiny machine learning: A comprehensive review," arXiv preprint arXiv:2311.11883, 2023.
[53] Lê, Minh Tri, Pierre Wolinski, and Julyan Arbel, "Efficient neural networks for tiny machine learning: A comprehensive review," arXiv preprint arXiv:2311.11883, 2023.
[54] Alaa, R., Hussein, E. A., & Al-libawy, H, "OBJECT DETECTION ALGORITHMS IMPLEMENTATION ON EMBEDDED DEVICES: CHALLENGES AND SUGGESTED SOLUTIONS," Kufa Journal of Engineering, 15(3), 2024.