




Review

# On-Device Object Detection for More Efficient and Privacy-Compliant Visual Perception in Context-Aware Systems

Ivan Rodriguez-Conde <sup>1,\*</sup>, Celso Campos <sup>2</sup> and Florentino Fdez-Riverola <sup>3,4</sup>

- <sup>1</sup> Department of Computer Science, University of Arkansas at Little Rock, 2801 South University Avenue, Little Rock, AR 72204, USA
- <sup>2</sup> Department of Computer Science, ESEI—Escuela Superior de Ingeniería Informática, Universidade de Vigo, 32004 Ourense, Spain; ccampos@uvigo.es
- <sup>3</sup> CINBIO, Department of Computer Science, ESEI—Escuela Superior de Ingeniería Informática, Universidade de Vigo, 32004 Ourense, Spain; riverola@uvigo.es
- <sup>4</sup> SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur), SERGAS-UVIGO, 36213 Vigo, Spain
- \* Correspondence: irconde@ualr.edu; Tel.: +1-501-442-4859

**Abstract:** Ambient Intelligence (AmI) encompasses technological infrastructures capable of sensing data from environments and extracting high-level knowledge to detect or recognize users' features and actions, as well as entities or events in their surroundings. Visual perception, particularly object detection, has become one of the most relevant enabling factors for this context-aware user-centered intelligence, being the cornerstone of relevant but complex tasks, such as object tracking or human action recognition. In this context, convolutional neural networks have proven to achieve state-of-the-art accuracy levels. However, they typically result in large and highly complex models that typically demand computation offloading onto remote cloud platforms. Such an approach has security- and latency-related limitations and may not be appropriate for some AmI use cases where the system response time must be as short as possible, and data privacy must be guaranteed. In the last few years, the on-device paradigm has emerged in response to those limitations, yielding more compact and efficient neural networks able to address inference directly on client machines, thus providing users with a smoother and better-tailored experience, with no need of sharing their data with an outsourced service. Framed in that novel paradigm, this work presents a review of the recent advances made along those lines in object detection, providing a comprehensive study of the most relevant lightweight CNN-based detection frameworks, discussing the most paradigmatic AmI domains where such an approach has been successfully applied, the different challenges arisen, the key strategies and techniques adopted to create visual solutions for image-based object classification and localization, as well as the most relevant factors to bear in mind when assessing or comparing those techniques, such as the evaluation metrics or the hardware setups used.

**Keywords:** on-device; object detection; ambient intelligence; deep learning; convolutional neural networks



**Citation:** Rodriguez-Conde, I.; Campos, C.; Fdez-Riverola, F. On-Device Object Detection for More Efficient and Privacy-Compliant Visual Perception in Context-Aware Systems. *Appl. Sci.* **2021**, *11*, 9173. <https://doi.org/10.3390/app11199173>

Academic Editors: Dalila Durães and Jason J. Jung

Received: 10 August 2021  
Accepted: 29 September 2021  
Published: 2 October 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Today ambient intelligence (AmI) systems are experiencing an unprecedented growth momentum essentially due to the thrust and fast development of two of their main enabling technological forces: the so-called Internet of Things (IoT), focused on the exploitation of networked sensor infrastructures to remotely gather data and enable data exchange between several distributed ends and Artificial Intelligence (AI), more oriented to the use of gathered data and the subsequent distillation of knowledge necessary to make AmI systems adaptable and “aware” of their surroundings. Both fields, IoT and AI have experienced outstanding progress almost side by side in the last two decades, a fact in no case fortunate, but rather the result of a continuous interaction that is still going on nowadays.

Methods and technologies developed within AmI under the IoT paradigm facilitate communication and data exchange among the different devices that comprise the so-called intelligent environments, enabling the creation and proper exploitation of new network architectures based on connected sensing instruments and resource-constrained energy-efficient end devices (i.e., embedded devices and mobile devices). Efforts made in that regard have been primarily aimed both at the design and deployment of faster and more efficient network infrastructures, as well as at the development of more accurate sensing platforms and more powerful computing hardware, enabling the collection and storage of large volumes of data and consequently, the support of increasingly sophisticated AI techniques, from traditional machine learning (ML) methods, all the way to more recent deep learning (DL) approaches.

Modern advanced algorithmic solutions can process sensed data and derive high-level knowledge to allow intelligent computational systems to recognize and understand both user activity and behavior, as well as the different phenomena around them to extract context-tailored knowledge and consequently, provide more effective support. Artificial visual perception techniques, particularly object detection methods, have emerged as some of the most relevant enabling factors of that user-centered intelligence, providing a better understanding of the environment and the entities in it, thus constituting a fundamental pillar for vision tasks in the AmI context, such as object tracking [1] and human action recognition [2,3], among others.

Despite the qualitative leap forward that the recent wide adoption of convolutional neural networks (CNNs) brought accuracy-wise and the countless research efforts mobilized in the last decade in that direction, object detection remains a highly complex problem. CNNs, as a specialized type of DL networks, are computationally expensive [4,5] and require a large memory footprint. Although cloud platforms have shown to be scalable and powerful enough to meet those needs, leveraging such infrastructures typically involves deploying neural networks as remote services, demanding a virtually persistent connection with the server-side. That usually leads to latency (network speed + processing time) and security limitations that make those solutions unsuitable for conventional AmI scenarios where the system response time must be as short as possible, or data privacy needs to be ensured.

As an alternative to the cloud, many authors have focused their research on local processing strategies in response to the limitations mentioned, resulting in a novel DL trend called on-device machine learning. Such an approach pursues more compact and efficient models directly deployable into IoT devices, reducing the server-side computational load, the data traffic between endpoints, and the associated latency. Thus, it makes it possible to incorporate a layer of intelligence into the different end devices in AmI systems and as a result of the latter, it enables a more fluid experience better tailored to end users' needs, without compromising the privacy of their data.

While AmI is a mature discipline that has attracted in the last fifteen years a great deal of attention of scientists and practitioners from various fields, such as human-computer interaction (HCI), AI, and communication networks, the search for on-device lightweight detection solutions is a still-emerging area of study that has barely been around for five years. Both of them, however, have yielded a vast scientific production, as evidenced by the important number of surveys currently present in the related literature. Specifically concerning AmI it is possible to find a wide range of studies: (i) papers of a general nature that overview the field, introducing key concepts, common use cases, and state-of-the-art techniques [6–11]; (ii) works circumscribed only to a subset of relevant related matters, such as security and privacy [12–15], sensing, computing, and networking technologies [16,17], as well as ethical implications and social issues [18], and (iii) research focused on particular application domains (e.g., architecture [19], workplaces [20], educational environments [21,22], and smart cities [18]), with healthcare being the one that has shown greater treatment [12,23–29]. For its part, the on-device object detection corpus is limited to literature published over the last five years (2017–2021). It encompasses (i) surveys that cover

the on-device ML field from a generic perspective following a schema similar to the one indicated for AmI surveys [30–34], (ii) works exclusively circumscribed to the object detection problem [35–41], and last but not least, (iii) research on problem-specific matters, such as good-performant network architectures [37,39], appropriate training strategies [36,39], and techniques to enhance the representational power of neural networks [40].

Beyond the works referenced, the intersection of AmI and AI has already resulted in a substantial number of review articles [17,42–45]. They summarize the progress both fields have experienced jointly, detailing the several approaches explored for creating intelligent and context-aware environments (e.g., human activity and behavior recognition techniques [44], and classification methods for more accurate disease prediction and diagnosis [45]). Within such body of works, although some papers can be considered part of the on-device literature since they analyze different IoT communication architectures, as well as hardware technologies for local data processing, there is not any equivalent yet dealing either with the algorithmic part or, in any case, providing an overview of the advanced sensing techniques required for the implementation of the intelligence, context awareness, user adaptability and user privacy preservation principles that, according to Augusto et al. [46], should drive the building process of any AmI system.

To fill that gap, this paper reviews some of the most relevant research works recently produced in different AmI application areas of interest, aimed at the design and development of efficient, compact, locally executable, and consequently, privacy- and data security-friendly object detection frameworks. In particular, this work is exclusively focused on CNNs, a deep learning algorithm family that has shown to be the current state-of-the-art in relation to the target problem [36,38,39]. More specifically, the paper analyzes and provides the reader with a structured presentation of the most salient related approaches and techniques, giving details on both the different challenges or issues that the authors in the field have faced and the various CNN architectures exploited for that purpose, also discussing the specific strategies or approaches adopted and the different configurations or frameworks used to evaluate their performance.

The scope of the study is restricted to CNN networks conceived according to the on-device paradigm. The paradigm itself and its relatively short lifetime represent a highly constraining filtering criterion on the source search process carried out. MobileNets' [47] year of publication (2017) was taken as the primary point in time reference, given that, to the best of the authors' knowledge, it is widely considered the very first compact CNN architecture explicitly designed to be deployed on low-powered devices, in particular, on mobile devices. Thus, the survey covers the major progress made in that direction in the last five years, only considering still image processing techniques and being restricted to application domains and use cases traditionally associated with the AmI field drawn from existing literature, such as [42,43]. Domain names together with terms, such as "on-device", "object detection", "convolutional neural networks" or its acronym "CNN" were used as keywords in the search for pertinent resources, being such search performed using the Google Scholar engine. The list of results originally obtained was refined later on through an iterative process, discarding research papers outside beyond the scope previously outlined, initially after merely reviewing the abstract and in a subsequent step, after an in-depth reading of the work.

The rest of the document is organized as follows. Section 2 provides context by characterizing the most relevant studies in the field. Section 3 analyzes the different lightweight detection frameworks conceived, detailing the different strategies or approaches adopted, as well as the specific challenges or typical problems addressed in such context. Section 4 discusses the factors and metrics considered when evaluating the performance of the applied solutions or comparing them with existing alternatives. Finally, Section 5 summarizes the observations drawn from the state of the art and identifies research challenges to be addressed in future work.

## 2. On-Device Object Detection for Context Awareness in Ambient Intelligence Systems

Following a top-down approach, we start the review by first creating a taxonomy of the main AmI application domains where lightweight sensor-based solutions have been recently explored for building context-aware intelligent systems. Such a type of system can sense the environment and as a result, both acquire static images and record videos. Thanks to an “augmented” visual perception layer (that assembles at the same level what is traditionally known as the perception layer in IoT, together with the application layer), they can successfully process such input data and extract the knowledge required to better understand the surroundings, as well as to interact with the entities (including users) that exist there in a more effective manner. The ability to detect such entities without offloading computation in external third-party systems, endows AmI solutions with great flexibility and versatility, as evidenced by the significant number of domains and applications where on-device object detection has been explored, shown in Table 1.

It is possible to create a first classification in which we can distinguish, on the one hand a number of general-scope works [48–66] and on the other hand a larger group of publications that present domain-specific research [67–110]. The first group includes studies that, far from focusing on a particular application area, propose more open research involved in the search for efficient and lightweight detection solutions, oriented to AmI scenarios. Overall, authors pursue advances in object detection within the on-device paradigm, trying, in general terms, to satisfy the requirements that the implementation or deployment of state-of-the-art detection techniques in resource-limited devices inevitably entail, but putting particular interest in, if not achieving real-time performance [49,50,54,64,66], obtaining at least an adequate trade-off between speed and accuracy [52,56,57,60,65]. The experimentation proposed along those lines is typically contextualized in a given AmI scenario or application. Such contextualization, however, has nothing to do with addressing the specificities of an AmI area or sub-field. It is used instead for practical purposes, just as a use case for quantitatively analyzing the performance of the algorithms and methods devised, as well as for assessing the feasibility of their deployment given the constraints of the hardware platforms used; merely having implications on the datasets (on the classes or object types considered) exploited for training, fine-tuning, and testing the CNNs designed. There exists a prevalence in that regard, of widespread and cross-domain AmI applications on mobile or embedded devices, such as face detection [48,58,60–62] (biometric security, surveillance), and vehicle and pedestrian detection (security, surveillance, autonomous vehicles, smart cities) both in cars [56,64,65] and unmanned aerial vehicles (UAV) [49,52–55,57,59].

**Table 1.** Recent research works exploring the use of on-device object detection techniques for building AmI applications.

Work	Year	Application	Domain	Vision Task It Supports	High-Level Requirements
[48]	2017	Face detection	Generic	–	<ul style="list-style-type: none"> <li>• Deployment in resource-limited devices</li> </ul>
[67]	2018	Human detection	Surveillance	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[68]	2018	Landing marker detection from UAV	Intelligent transportation	–	<ul style="list-style-type: none"> <li>• Small target detection</li> </ul>
[69]	2018	Vehicle detection from UAV	Intelligent transportation	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> <li>• Energy efficiency</li> </ul>
[70]	2018	Object detection using depth imaging	Surveillance	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> <li>• Privacy awareness</li> </ul>
[71]	2018	Smile detection	Human emotion recognition	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> <li>• Deployment in resource-limited devices</li> </ul>

Table 1. Cont.

Work	Year	Application	Domain	Vision Task It Supports	High-Level Requirements
[72]	2019	Waste detection	Smart cities	<ul style="list-style-type: none"> <li>Distance to target estimation</li> <li>Target size estimation</li> </ul>	–
[49]	2019	Vehicle detection from UAV	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[50]	2019	Generic object detection	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[51]	2019	Face detection	Generic	–	<ul style="list-style-type: none"> <li>Trade-off between accuracy and model size</li> </ul>
[52]	2019	Vehicle and pedestrian detection from UAV	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[53]	2019	Object detection from UAV	Generic	–	<ul style="list-style-type: none"> <li>Energy efficiency</li> </ul>
[54]	2019	Vehicle detection from UAV	Generic	–	<ul style="list-style-type: none"> <li>Higher execution speed</li> </ul>
[55]	2019	Object detection from UAV	Generic	–	<ul style="list-style-type: none"> <li>Deployment in resource-limited devices</li> </ul>
[73]	2019	Disease symptoms detection	Healthcare	–	<ul style="list-style-type: none"> <li>Deployment in highly portable devices</li> </ul>
[74]	2019	Disease symptoms detection	Healthcare	–	–
[56]	2019	Vehicle and pedestrian detection	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[57]	2019	Object detection in UAV imagery	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[75]	2019	Human detection	Surveillance	–	<ul style="list-style-type: none"> <li>Low-cost infrastructure</li> </ul>
[76]	2019	Fruit detection	Smart farming	<ul style="list-style-type: none"> <li>Crop-load estimation</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[77]	2019	Face detection + Mobile phone detection	Intelligent transportation	<ul style="list-style-type: none"> <li>Driver distraction detection</li> </ul>	<ul style="list-style-type: none"> <li>Robustness</li> </ul>
[58]	2019	Face detection	Generic	–	<ul style="list-style-type: none"> <li>Better model generalization</li> </ul>
[78]	2019	Open and closed eyes detection	Intelligent transportation	<ul style="list-style-type: none"> <li>Driver drowsiness detection</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[79]	2019	Vehicle and pedestrian detection from large trucks	Intelligent transportation	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>Robustness</li> </ul>
[80]	2019	Plant disease detection	Smart farming	–	–
[59]	2019	Object detection in UAV imagery	Generic	–	<ul style="list-style-type: none"> <li>High accuracy</li> </ul>
[81]	2019	Object detection in indoor environment	Robotics	–	<ul style="list-style-type: none"> <li>Robustness</li> </ul>
[82]	2019	Vehicle detection	Intelligent transportation	<ul style="list-style-type: none"> <li>Traffic flow estimation</li> </ul>	<ul style="list-style-type: none"> <li>Trade-off between accuracy and execution speed</li> </ul>

Table 1. Cont.

Work	Year	Application	Domain	Vision Task It Supports	High-Level Requirements
[60]	2019	Face detection	Generic	<ul style="list-style-type: none"> <li>• Video face analysis</li> </ul>	<ul style="list-style-type: none"> <li>• Trade-off between accuracy and execution speed</li> </ul>
[83]	2019	Ship parts detection	Smart logistics	<ul style="list-style-type: none"> <li>• Ship's identity classification</li> </ul>	<ul style="list-style-type: none"> <li>• Robustness</li> </ul>
[61]	2019	Face detection	Generic	–	<ul style="list-style-type: none"> <li>• Trade-off between accuracy and model size</li> </ul>
[84]	2019	Obstacle detection in unmanned airships	Defense	–	<ul style="list-style-type: none"> <li>• Trade-off between accuracy and execution speed</li> </ul>
[62]	2020	Face detection	Generic	–	<ul style="list-style-type: none"> <li>• High accuracy</li> </ul>
[63]	2020	Ordinary object detection	Generic	–	<ul style="list-style-type: none"> <li>• High accuracy</li> </ul>
[85]	2020	Eye detection	Intelligent transportation	<ul style="list-style-type: none"> <li>• Driver drowsiness detection</li> </ul>	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[86]	2020	Ship detection in radar images	Defense	–	<ul style="list-style-type: none"> <li>• Deployment in resource-limited devices</li> </ul>
[87]	2020	Fruit detection	Smart farming	<ul style="list-style-type: none"> <li>• Crop-load estimation</li> </ul>	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[88]	2020	Obstacle and object detection	Healthcare	<ul style="list-style-type: none"> <li>• Distance to target computation</li> <li>• Navigation assistance</li> </ul>	–
[89]	2020	Human activity recognition	Surveillance	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[90]	2020	Human detection	Surveillance	–	–
[91]	2020	Vineyard trunks detection	Smart farming	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[92]	2020	Vehicle and pedestrian detection	Intelligent transportation	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> <li>• High accuracy</li> </ul>
[64]	2020	Vehicle and pedestrian detection	Generic	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[93]	2020	Livestock (pigs) monitoring	Smart farming	<ul style="list-style-type: none"> <li>• Livestock counting</li> <li>• Abnormalities detection</li> </ul>	<ul style="list-style-type: none"> <li>• Trade-off between accuracy and model size</li> <li>• Low-cost infrastructure</li> </ul>
[94]	2020	Dynamic targets detection	Robotics	<ul style="list-style-type: none"> <li>• Visual SLAM</li> </ul>	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[95]	2020	Human detection	Surveillance	–	<ul style="list-style-type: none"> <li>• Trade-off between accuracy and model size</li> </ul>
[96]	2020	Vehicle detection	Intelligent transportation	–	<ul style="list-style-type: none"> <li>• Real-time execution</li> </ul>
[97]	2020	Wound localization	Healthcare	<ul style="list-style-type: none"> <li>• Wound segmentation</li> <li>• Wound classification</li> </ul>	–

Table 1. Cont.

Work	Year	Application	Domain	Vision Task It Supports	High-Level Requirements
[98]	2020	Drone detection	Surveillance	–	<ul style="list-style-type: none"> <li>Trade-off between accuracy and model size</li> </ul>
[99]	2020	Barcode detection	Smart logistics	<ul style="list-style-type: none"> <li>Barcode scanning</li> </ul>	<ul style="list-style-type: none"> <li>Energy efficiency</li> <li>High accuracy</li> </ul>
[100]	2020	Waste detection	Smart cities	–	<ul style="list-style-type: none"> <li>High accuracy</li> </ul>
[101]	2020	Bus passenger detection	Smart cities	<ul style="list-style-type: none"> <li>Passenger counting</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[65]	2020	Vehicle and pedestrian detection	Generic	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[102]	2020	Vehicle detection	Intelligent transportation	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[103]	2020	Fruit detection	Smart farming	<ul style="list-style-type: none"> <li>Crop-load estimation</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[104]	2020	Human action detection from autonomous drones	Surveillance	–	–
[105]	2020	Vehicle and pedestrian detection	Intelligent transportation	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[106]	2020	Traffic sign recognition	Intelligent transportation	–	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[107]	2020	Plant disease detection	Smart farming	–	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>
[108]	2020	Garbage truck supervision	Smart cities	–	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[66]	2020	Calibration of object tracking systems	Generic	<ul style="list-style-type: none"> <li>Object tracking</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[109]	2020	Face mask monitoring	Surveillance	–	<ul style="list-style-type: none"> <li>Real-time execution</li> </ul>
[110]	2021	Navigation in open surgeries	Healthcare	<ul style="list-style-type: none"> <li>3D object localization</li> </ul>	<ul style="list-style-type: none"> <li>Real-time execution</li> <li>High accuracy</li> </ul>

Furthermore, as far as domain-specific studies are concerned, it is possible to categorize them into five groups, intelligent transportation [68,69,77–79,82,85,92,96,102,105,106], surveillance [67,70,75,89,90,95,98,104,109], smart farming [76,80,87,91,93,103,107], healthcare [73,74,88,97,110] and smart cities [72,100,101,108], all of them, scenarios where constant and real-time object detection is necessary for enabling context-awareness on end devices. While further information on each of those domains will be incorporated into the discussion in successive paragraphs to draw a clearer picture, it should be noted first that additional application areas, albeit almost residually with only one or two related works identified, have emerged in the analysis: (i) robotics [81,94], a domain where vision represents one of the most important communication channels with the environment, and where object detection has traditionally shown to be critical for the perception, modeling, planning, and understanding of unknown terrains [94]; (ii) defense, where object detection constitutes a major factor for controlling UAVs [84] and detecting ships in radar images [86]; (iii) smart logistics, with two distinct but equally representative examples of the use of sensing technologies, one on embedded platforms (in situ detection and recognition of ships for more

efficient port management) [83], and the second one on mobile devices (barcode detection) [99] and finally, (iv) human emotion recognition based on facial expression detection, as reported in [71]. Once the less representative options are covered, the remainder of this section will deal with the most common domains that have emerged in the analysis.

Intelligent transportation systems (ITS) have lately attracted considerable research interest driven by the modernization of transportation infrastructures and the development of autonomous driving and its supporting technologies. The reviewed AmI literature is good evidence of that. Although the works specifically identified along those lines [64,68,69,77–79,82,85,96,102,105,106] cover only a limited subset of the broad spectrum of applications resulting from the technological progress in the field, they provide good intuition on the relevance of object detection as a key factor for making both vehicles and infrastructures safer, more efficient, comfortable and reliable. In particular, in-vehicle ITS systems [64,77–79,85,96,102,105,106] (commonly known as Advanced Driving Assistance Systems or ADAS) stand out in the analysis as the central focus of interest. Such systems embed detection solutions conceived as safety mechanisms for monitoring both driver operations [77,78,85], preventing distractions, and ultimately, the loss of control of the vehicle, as well as on-road events [64,79,96,102,105,106], being the latter mainly implemented nowadays in the form of a warning instrument triggered in situations of potential collisions [79,96] or infractions [106], but also designed towards decision-making support in future autonomous vehicles [64,102,105]. For its part, regarding conventional ITS alternatives [68,69,82], the different approaches identified are aimed at road traffic control and early response to emergencies, applications in which the use of UAVs has been intensively explored [68,69] due to their capabilities to record and communicate information in a non-intrusive way, as well as to recognize areas inaccessible by other means. A key challenge in this context lies in UAVs' built-in computing components since they have to both process the acquired images with almost no latency to make critical navigation decisions [68] while consuming as little power as possible to minimize its impact on the battery life and hence on the system's flight time [69].

As with ITS, surveillance systems have similarly undergone a dramatic development, pushed by the advancement of information and communication technologies. Like the former, they require a real-time capacity to perform key tasks, such as monitoring and target tracking. In this regard, the studies analyzed within the domain [67,70,75,89,90,95,98,104,109] reveal a large body of works focused on the exploitation of efficient CNNs for human detection in a wide range of scenarios: street monitoring in urban areas [67], home surveillance [75], subway [95] and railway security [90], people search and rescue in natural disasters [104], and recognition of specific human behaviors [89]. Besides low latency, there is a couple of additional factors to be considered when designing such systems that have emerged in the review: privacy [70] and cost [75]. Individuals' privacy and anonymity should be a central tenet of surveillance systems. However, RGB cameras commonly used for such purposes are too invasive, and for that reason, we see how authors, such as Mithun et al. [70] explore the exploitation of alternatives, such as raw depth data. Furthermore, surveillance has traditionally been extremely important for business and home security, and while there is a decent number of technological solutions in the market, most of them rely on expensive proprietary cameras. Low-cost IoT hardware platforms, as suggested by [75], might be a perfectly valid approach to building solutions tailored to more austere budgets. Surveillance and development of low-cost systems can be also observed in smart farming literature, the third of the five main AmI domains identified. In particular, both of those elements are present in Seo et al.'s work [93]. Detection frameworks are devised to automate pig monitoring, a use case where implementing solutions with low cost is tremendously important given the potential need for large-scale deployment of such systems and the more than likely high turnover rate due to their rapid physical deterioration. Nevertheless, except for Seo et al.'s work, the different observed approaches are all related to activities of the agricultural sector, a context where detection techniques stand out as an effective method to recognize plant diseases [80,107], as well as an essential service integrated into



the control software of agricultural robots [76,87,91,103]. Specifically regarding the last point, leveraging mobile agricultural robots makes it possible to automate a considerable part of traditional farmers' tasks, eminently repetitive, such as fruit counting [87,103], harvesting, and picking [76]. Fruit detection, as well as the detection of any other typical element in such contexts like trunks or branches, is exploited not only for the indicated purpose but also to provide the robot with information on the environment necessary to successfully navigate through the crop fields [91], often irregular or located in areas where the GPS signal is not reliable enough.

Finally, to conclude this first analysis focused on the current landscape of object-detection-based AmI applications for low-power devices, we add to the discussion the two application domains not covered yet from the group of five with greater representation in the study: healthcare [73,74,88,97,110] and the so-called smart cities [72,100,101,108]. In regard to healthcare, on-device detection techniques are shown to be effective in extending healthcare spaces beyond the traditional scenario of closed clinical environments, bringing the capabilities of (i) disease diagnosis [73,74], (ii) wound or injury zone delimitation [97] and (iii) patient monitoring and support [88], (available only in typically complex and expensive configurations until recently) to low-cost portable devices. Object detection, however, is not the final intended application. Detection solutions are conceived instead as enabling services for more sophisticated vision tasks, as shown in Table 1. In addition, as far as smart cities are concerned, detection mechanisms are exploited to provide support to more complex tasks as well, but, in this particular case, seeking more resource-efficient management in everyday services in urban areas, such as public transportation [101] and garbage collection [72,100,108].

### 3. Compact and Efficient Vision-Based Detection Frameworks

Nowadays, whether we tackle the cross-domain on-device space involving compact and efficient solutions or more specific research focused just on the current AmI landscape, dealing with the design of vision-based object detection frameworks involves the discussion of CNN architectures. While statistical classifiers, such as Support Vector Machines (SVM) [111], Random Forest [112], Adaboost [113] and classical neural networks were for many years the computer vision (CV) standard and played a dominant role in object detection tasks, the advent of DL techniques has unquestionably represented a step forward compared to traditional detectors. The exploration of new approaches and the design of new CNN architectures, capable of automating the extraction of representative features, boosted by challenges, such as Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [114] and Pascal Visual Object Classes (VOC) [115], has resulted in a plethora of novel methods with increasingly higher performance in visual recognition tasks over the last decade, producing more robust frameworks capable of addressing object localization and classification tasks in highly complex scenarios.

Such challenges and the tremendous interest generated in the scientific community have pushed CNNs to an unprecedented evolution. Such progress, however, has led to increasingly complex architectures, as indicated in the introduction of the paper. Models, such as VGG-16 (138 M parameters) [5] or RetinaNet (built on ResNet-152 [116], with 60 M parameters) [117], while achieving high levels of accuracy, are generally based on bulky structures and multidimensional parameter spaces, resulting in a high volume of both computed intermediate products and output values. Towards a response to the reality described, a fair number of investigations have been focused on producing more efficient and effective detection frameworks during the last five years. The reduction of the computational cost of traditional detectors and the preservation of accuracy have driven the solution search process carried out by CV experts (both scientists and practitioners), resulting in a fair number of techniques and methods that, built upon existing restrained-complexity CNNs (e.g., SSD [118], the YOLO family [119–122], and Faster R-CNN [123]), have shown promising performance in detection tasks and have also proven to be applicable and relevant in different disciplines, including AmI. With the support of Tables in 3, we provide an

overview of the various detection frameworks devised within AmI along the described lines, covering both architectural details and traditionally related challenges together with the most relevant solutions explored.

### 3.1. Architectures

Intuitively, upon a first look at the data reported in Table 2, it is possible to observe the predominance of one-stage detectors (39 out of a total of 42) over the two-stage alternative [83,95,99]. Two-stage and single-stage detection frameworks (the latter also called unified detectors) are the two main categories typically considered for the classification of modern object detection pipelines. Two-stage detectors have typically featured high accuracy in both target localization and classification tasks thanks to the exploitation of a Region Proposal Network (RPN) [123] dedicated to producing Region of Interest (RoI) proposals, i.e., candidate object bounding boxes. However, region proposal generation has shown to be an insurmountable bottleneck when trying to produce less costly models. Therefore, many authors have focused their research on exploring unified detection strategies, seeking more efficient solutions. Such an approach models object detection as a single regression problem, addressing in a single step the prediction in the form of bounding boxes of both the areas where potentially interesting objects might be found (localization) and the various classes they belong to (classification). The result is a simplified and smaller architecture, conceived as a single feedforward neural network, suitable to derive high-inference-speed detection models.

**Table 2.** Summary of the compact CNN-based object detection frameworks used in the research works considered.

Detector's Name	Framework Type	Baseline	Backbone	Pre-Training	Enhancement Emphasis	Used in
–	One-stage	Tiny YOLOv2	Darknet-19	–	–	[110]
–	One-stage	Tiny YOLOv3	Darknet-19	Detector	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[58,96,97,103,108,109]
–	One-stage	YOLOv3	Darknet-53	Backbone	–	[66,77,106]
–	One-stage	YOLOv3	MobileNetV2	Detector	<ul style="list-style-type: none"> <li>• Lower memory overhead</li> </ul>	[107]
FRDet	One-stage	YOLOv3	Custom	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> <li>• Lower memory overhead</li> </ul>	[105]
–	One-stage	SSD	MobileNetV1	Detector	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[63,65,72,75,78,80,88,91]
PPt-YOLOv3	One-stage	Tiny YOLOv3	Darknet-19	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[102]
–	One-stage	SSD	MobileNetV2	Detector	–	[65,87,100]
M7	One-stage	Tiny YOLO	–	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[101]
Tiny-BDN	Two-stage	Faster-RCNN	VGG	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[99]
TIB-Net	One-stage	Extremely Tiny Face Detector	–	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[98]

Table 2. Cont.

Detector's Name	Framework Type	Baseline	Backbone	Pre-Training	Enhancement Emphasis	Used in
Tiny MetroNet	Two-stage	MetroNet	SqueezeNet	Backbone	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[95]
–	One-stage	YOLOv4	–	–	–	[94]
Embedded PigYOLO	One-stage	Tiny YOLO	–	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> </ul>	[93]
–	One-stage	SSDLite	MobileNetV3 Small	–	<ul style="list-style-type: none"> <li>• Higher accuracy</li> </ul>	[64]
Tiny Gaussian YOLOv3	One-stage	Tiny YOLOv3	Darknet-19	–	<ul style="list-style-type: none"> <li>• Higher accuracy</li> </ul>	[92]
YOLOv3-PDN	One-stage	YOLOv3	Darknet-53	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[90]
T-TINY-YOLO	One-stage	Tiny YOLO	–	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[89]
Lira-YOLO	One-stage	YOLO	LiraNet	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> </ul>	[86]
Fast Eye-CPU	One-stage	–	–	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> </ul>	[85]
Multi-Task FaceBoxes	One-stage	FaceBoxes	–	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[62]
NYOLO-tiny	One-stage	Tiny YOLOv3	Darknet-19	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Higher accuracy</li> </ul>	[84]
Extremely Tiny Face Detector	One-stage	–	–	–	<ul style="list-style-type: none"> <li>• Higher accuracy</li> </ul>	[61]
DCNet	Two-stage	–	–	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> </ul>	[83]
EagleEye	One-stage	–	VGG	–	–	[60]
Concatenated Feature Pyramid Network	One-stage	YOLOv3	–	Detector	<ul style="list-style-type: none"> <li>• Higher robustness</li> <li>• Lower overhead</li> </ul>	[82]
–	One-stage	YOLOv2	Darknet-19	Detector	<ul style="list-style-type: none"> <li>• Higher robustness</li> </ul>	[81]
–	One-stage	SSD	–	Detector	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[79]
M3	One-stage	YOLO9000	–	Detector	<ul style="list-style-type: none"> <li>• Higher accuracy</li> </ul>	[76]
Modular Feature Fusion Detector	One-stage	–	–	–	–	[56]
–	One-stage	SSDLite	MobileNetV2	–	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[74]
Kankanet	One-stage	SSD	MobileNetV1	–	–	[73]
SlimYOLOv3-SPP3	One-stage	YOLOv3	–	Detector	<ul style="list-style-type: none"> <li>• Lower overhead</li> <li>• Lower memory overhead</li> </ul>	[55]
UAV-Net	One-stage	SSD	ZynqNet	–	<ul style="list-style-type: none"> <li>• Higher robustness</li> </ul>	[54]

Table 2. Cont.

Detector's Name	Framework Type	Baseline	Backbone	Pre-Training	Enhancement Emphasis	Used in
–	One-stage	SSD	PeleeNet	–	• Higher robustness	[52]
DupNet-Tinier-YOLO	One-stage	IFQ-Tinier-YOLO	–	–	• Higher accuracy	[51]
–	One-stage	Tiny YOLO	–	Backbone	• Lower overhead • Lower memory overhead	[50]
ShuffleDet	One-stage	SSD	ShuffleNet	Backbone	• Higher robustness	[49]
–	One-stage	SSD	VGG	Detector	• Lower overhead • Lower memory overhead	[70]
DroNet	One-stage	TinyYOLO	–	–	–	[69]
LCDet	One-stage	YOLO	–	Detector	–	[48]
D-RFBNet300	One-stage	RFBNet300	MobileNetV1	Backbone	• Higher accuracy	[59]
–	One-stage	–	MobileNetV1	Backbone	–	[71]
LightDenseYOLO	One-stage	YOLOv2	LightDenseNet	Backbone	–	[68]

The unified approach, crystallized in compact architectures considered a standard nowadays, such as SSD [118] and YOLO in its different versions [119–122], constitute the foundation on which modern lightweight object detection architectures have been built in AmI [48,49,52,54,55,63,65,66,68,70,72,73,75,77–82,86–88,90,91,94,100,105–107]. Such compact architectures, however, resulted from an eminently moderate structural optimization in a first attempt to make CNNs more manageable, prioritizing accuracy preservation over structural reduction and inference speed increase. Thus, although it is possible to state that those detectors represented an effective evolution towards more efficient techniques, especially if we compare them with two-stage frameworks, they were still insufficient for running such a type of detection system on limited-resource devices mainly due to their considerable computational complexity and size. It has not been until recently when works within the on-device paradigm have actually delved into the compression and simplification of detection models and their underlying architectures, achieving a more pronounced decrease of both the number of parameters (and accordingly, the storage required) and the inference time of the detectors produced. Such research has resulted in significantly more efficient detection frameworks, making them directly deployable on low-power and low-memory hardware platforms but, on the flip side, causing a dramatic accuracy degradation. Despite the latter, these novel models, e.g., Tiny YOLO [119,120] and SSDLite [118], are already considered standard benchmarking options when designing new lightweight detectors in AmI [50,58,64,69,74,84,89,92,93,96,97,101–103,108–110].

The framework type constitutes the master blueprint adopted to build the detector. The building process is driven by the different design decisions aimed at modeling and tailoring the underlying CNN to the specific problem addressed. Such tasks can be tackled by directly reusing an existing standard network or designing a new detection pipeline from scratch. The utilization of off-the-shelf CNNs, whose implementation and weights are available in ML development frameworks, such as TensorFlow [124] and PyTorch [125], simplifies the development of new detection solutions considerably.

In this regard, pre-trained detectors can be either exploited right away or, on the contrary, be tuned for a different second target task (typically, more specific) through transfer learning. This transfer learning approach enables reusing already existing models [48,55,63,70,75,76,78–82,87,91,100,103,107], previously trained on public standard object detection benchmarks, such as Pascal VOC [115] (used in [48]), and Microsoft COCO [126]

(used in [63,75,78,80,91,107]). Still, as shown in Table 2, it is a common practice as well to exploit not the entire detection model but merely the backbone [49,50,59,68,71,95,106,109], being this a CNN embedded in the detection framework, responsible for extracting from some given input images the different feature maps subsequently exploited by the deeper layers of the detector for predicting the several classes and bounding boxes produced as output. In any case, either globally or just circumscribed to the backbone, weights are initialized with values taken directly from pre-trained models, and then, through a fine-tuning process, the detector is re-trained on an application-specific dataset in order to adjust it to the specific use case to be addressed.

As reflected in the “Baseline” column of Table 2, leaving aside some few works [56,61,83,85] that have alternatively explored the creation of new architectures from scratch, the most common approach along those lines involves, however, using an existing architecture as a baseline and then either replacing the backbone [49,52,54,86] or introducing changes on the detector’s architecture or the training process parameterization [55,58,62,69,84,89,90,93,96,98,99,101,104,108,110]. Such tweaks and their focus, as seen if the “Baseline” and “Enhancement Emphasis” columns are analyzed jointly, are closely related to the very nature of the detection framework adopted as a reference. Both the “classical” moderate single-stage detection frameworks and the modern on-device models constitute a valuable starting point that simplifies and speeds up the creation of the new detectors. However, overall, they do not represent proper solutions for developing real-world AmI applications in austere technological environments. In particular, the more accuracy-preserving approaches, as just pointed out, fall short when it comes to optimizing the underlying network architecture and therefore, require efforts mainly aimed at minimizing the memory and computational resources demanded [48,55,70,79,82,85,86,90,107]. For its part, lightweight detectors are the result of more aggressive compression and optimization strategies. A two-fold effort is required to move towards a better speed-accuracy trade-off, on the one hand exploiting techniques capable of mitigating the accuracy degradation caused by the use of compact architectures, and on the other hand with approaches producing more expressive networks with greater detection capacity [51,58,59,62,64,76,84,92,93,96–98,102,103,108,109].

Furthermore, in addition to the dual path of improvement indicated, there is a third line of further refinement that has pursued the creation of more robust detectors [49,52,54,58,62,63,65,72,75,78,80–84,88,91,93,96–98,102,103,105,108,109] no matter what their nature is. Overall, CNNs have typically shown limitations in generalizing their predictions beyond the training domain or the distribution of the used dataset. Consequently, tasks considered reasonably straightforward for humans, such as detecting faces in the wild (with different head poses and face expressions), might be very challenging for vision-based detectors. Generalization and robustness against variations have been traditionally relevant in object detection, but they have shown to be even more sensitive for on-device detection solutions conceived for AmI. On the one hand as made explicit in Section 2, AmI is a domain that covers a fair number of applications where a bad-performant detector can have actual implications or impact on the physical integrity, safety, and wellbeing of the users (e.g., in autonomous driving [92,102,105]), and even on their personal finances (e.g., in smart farming [76,80,87,91,93,103,107]); for that reason, detection systems must ideally ensure consistency in their behavior. On the other hand, on-device detection solutions are, by definition, lightweight DL models directly deployable on hardware platforms with limited resources, a category where we can find portable devices, such as smartphones and machinery with mobility, such as robots and UAVs. Those devices have been reportedly used and proved to be helpful in different real-world scenarios where there are good chances of having dynamic and challenging conditions that might alter how targets look, contributing in that way to higher intra-class variability.

Below we list, in decreasing order of occurrence, the various factors pinpointed in the AmI papers analyzed as usual obstacles against detection robustness: (i) illumination [58,65,75,78,80,81,83,103,105,109], that has an impact on images at the pixel level,

altering object colors and even producing sense of occlusion in shadowed areas; (ii) total or partial target occlusion [58,80,81,83,103], including also object overlapping and truncation [102]; (iii) viewpoint [63,65,75,80,84], which makes a single object look completely different depending on the angle of view; (iv) weather conditions [83,84,105,109], that lead to illumination variations in the scene, also encompassing phenomena (e.g., fog and rain) that cause distortion or blurring effects on the acquired image; (v) object pose and orientation [58,78,109], usually upward in the real world, but not necessarily so in implementations, such as mobile augmented reality applications [63], and (vi) background complexity [65,83,98], which usually arises in crowded or cluttered scenes and makes it difficult to separate targets located in the foreground from non-relevant objects located in the background.

### 3.2. Challenges and Explored Approaches

Environment variation is still one of the common challenges that need to be addressed when developing detectors, and so they are when addressing the object detection problem within AmI while pursuing more efficient and compact solutions, as we have just seen. Although a fair number of such conditions have been successfully modeled by gathering representative samples as part of the datasets used for training the detection models, it is impossible to anticipate all the conditions that can potentially occur in the wild. Those context variations represent, however, only one of the various challenges observed in the AmI literature reviewed. Table 3 presents all the challenges resulting from the analysis, together with the different strategies or methods also noted. As far as the challenges are concerned, it is possible to categorize them into three distinct groups: AmI-related challenges, a group that comprises the environment-variation-specific issues we mentioned early on, but also both intra-class variation and scale variation challenges; performance objectives inherent to the on-device paradigm, such as model size and computational load decrease, as well as accuracy boost and finally, data-specific issues, including scarcity and low quality, both of them classical DL challenges that arise when dealing with specific applications or domains. In addition, concerning the various solutions reported, we can group them into four categories or approaches: (i) design of novel CNN-based architectures, (ii) model compression and acceleration, (iii) improved ad hoc datasets creation, and (iv) hyperparameter tuning. Below we discuss those categories in greater detail, providing information on the specific techniques and methods explored, putting the focus on the potential benefits that have contributed to addressing the challenges previously mentioned effectively.

**Table 3.** Summary of the challenges faced in the reviewed studies and the explored strategies when developing on-device detection frameworks.

	Model Size	Computational Burden	Accuracy	Environment Variation	Scale Variation	Intra-Class Variation	Data Scarcity	Low Quality Data
Lighter backbone	[49]	[49]	-	-	-	-	-	-
More efficient operations	[60,67,101]	[49,60,67,82,101]	-	-	-	-	-	-
Smaller conv kernel size	[95]	[95]	-	-	-	-	-	-
Multiple conv kernel sizes	-	-	-	-	[95]	-	-	-
More efficient modules	[95,105]	[95,105]	-	-	-	-	-	-
FC layers replaced with conv layers	[95]	[95]	-	-	-	-	-	-
Smaller input size	[50,90]	[50,90]	-	-	-	-	-	-
Larger input size	-	-	[69,102]	-	[102]	-	-	-
Fewer channels	[53,86,90]	[53,86,90]	-	-	-	-	-	-
Higher # channels	-	-	[102]	-	-	-	-	-
Fewer kernels	[50,69,89,95]	[50,69,89,95]	-	-	-	-	-	-
Shallower network	[76,79,82]	[76,79,82]	-	-	-	-	-	-
Deeper network	-	-	[84,96,107]	-	-	-	-	-
Early downsampling	[60]	[60]	-	-	-	-	-	-
Late downsampling	-	-	[56]	-	-	-	-	-
Residual connections	-	[86]	[49,105]	-	-	-	-	-
Feature fusion	-	-	[98]	-	[53,56,57,62,64,68,82,98,102–105]	-	-	-

Table 3. Cont.

	Model Size	Computational Burden	Accuracy	Environment Variation	Scale Variation	Intra-Class Variation	Data Scarcity	Low Quality Data
Attention mechanisms	–	–	[62,98]	–	[98]	–	–	–
Larger receptive field	–	–	[60,79]	–	[79]	–	–	–
Various receptive field sizes	–	–	–	–	[55,59]	–	–	–
Pruning	–	[50,54,55,70,90,93,99,101]	–	–	–	–	–	–
Quantization	[48,87,99]	[48,87,99]	–	–	–	–	–	–
Knowledge distillation	–	[64]	–	–	–	–	–	–
<i>Ad hoc</i> dataset	–	–	–	[50,64,68,77–80,84,86,91,95,98,100,103,109]	–	[50,64,68,77–80,84,86,91,95,98,100,103,109]	–	–
Data augmentation	–	–	–	–	–	–	[48,50,52,53,57,60,61,64,70,71,73,74,76,81,84,85,87,97,98,100,106]	[64]
Anchor box filtering	–	[79]	–	–	–	–	–	–
Anchor box sizing strategy	–	–	–	–	[60,95]	[95]	–	–
Loss function formulation	–	–	[92,105,107]	–	–	–	–	[53,60,64]

### 3.2.1. Design of Novel Lightweight CNN-Based Architectures

The design of novel lightweight CNN architectures stands out in the review as the dominant line of work, represented by the first 20 rows out of the 29 in Table 3. As its name suggests, this approach aims primarily at creating more compact and efficient network architectures (as reflected by the number of cells resulting from the intersection of the corresponding row in Table 3 with the first two columns), but it also seeks to improve accuracy in order to achieve an appropriate trade-off between speed and accuracy. Size and latency reduction go hand in hand. The more model weights, the higher the number of operations involved and consequently, the computational complexity and inference time required. Leaving aside a few exceptions in [49,82,86], in the AmI research analyzed, several strategies can be observed that pursue in the first place the reduction of the number of parameters, but that also affects the computational cost of the detector. In particular, it is possible to distinguish among them techniques eminently focused on the internal configuration of the layers in the network, as well as methods that, on the contrary, address global-scope factors focused on the organization of the layers or building blocks within the architecture. More specifically, within the first group, two different types of approaches have been observed: (i) a set of practices dealing with properties at the convolution kernel level (kernel number decrease, whether this is network-wide [69,95] or more selective [50,89]; size reduction in the spatial dimensions [95], and channel number decrease, applied, again, both indiscriminately to all the layers in the network [53], and particularly to specific stages [86] or layers [90]) and (ii) a second subset of works exploring the use of less expensive operations [49,60,67,82,95,101], focused primarily on finding lighter alternatives to standard convolution operations, such as point-wise convolutions [95], depth-wise separable convolutions [60,67,101], and grouped convolutions [49].

For its part, with regard to the group dealing with higher-level matters, we find a greater heterogeneity of approaches that includes: (i) the design and integration into an existing architecture of building blocks, such as two-way dense connection modules [86], based on residual shortcut connections, or the so-called Fire modules [95,105], consisting of a compression layer and an expansion layer, jointly capable of reducing the number of parameters of the models produced while preserving accuracy; (ii) the replacement of heavyweight backbones, such as VGG [5] by lighter networks, such as ShuffleNet [49]; (iii)

the replacement of costly layers, such as fully connected layers, in which each neuron of a given layer is connected to each of the neurons in its preceding, by alternatives with fewer connections and weights, such as convolution layers [95]; (iv) shallow layer subsampling (reduction of spatial dimensions) [60], by using convolution layers with a stride value greater than 1 instead of utilizing pooling layers which result in a higher loss of detailed information. Finally, it also identified in this group more simplistic techniques, such as reducing the input dimensions of the network [50,90] or eliminating some of its layers [76,79,82].

Regarding detection accuracy, an additional fair number of approaches can be observed in the literature analyzed. They are methods and techniques that pursue better object classification performance but, eminently, go after a detection accuracy boost, especially when dealing with multiscale or tiny targets. Thus, two distinct strategies can be distinguished along those lines: broader-scoped CNN-related methods addressing classification-related concerns and object-detection-specific techniques primarily aimed at achieving higher localization accuracy. In particular, the first group encompasses solutions devised to increase the representational capacity of the networks, improving both the network's learning capacity and the accuracy yielded. Although most of the actions reported in that direction, such as increasing networks' depth [84,96,107] and input size [69,102], late subsampling (i.e., reducing deep layers' size to obtain or maintain large receptive fields, thus enabling the coding of a higher volume of information [60,79]), or increasing the number of channels in convolution kernels [102], may seem misplaced within the on-device landscape, their underlying philosophy remains entirely pertinent for the search of lightweight solutions aimed at mitigating accuracy degradation. Furthermore, those methods are presented in the Aml body of works complemented by the exploitation of more versatile techniques, such as the utilization of residual shortcut connections, a building block that has proven to be effective at enabling a better information flow throughout the network, thus mitigating the vanishing gradient problem [49,105], and the exploitation of both convolution kernels [95] and receptive fields [55,59] with multiple sizes, aimed explicitly at achieving higher detection accuracy when targeting multiscale objects.

Target scale variation poses a challenge that commonly requires the fusion of low-level high-resolution features and high-level semantic features to obtain greater semantic richness. Data reported in this respect through Table 3, draw a scenario where the fusion of multiscale feature maps constitutes the dominant approach [53,56,57,62,64,68,82,98,102–105], with this substantiated in a considerable variety of strategies: channel concatenation [53], which provides a "trainable" way of fusing feature maps from different stages in the network; the exploitation of the pyramidal hierarchy naturally shaped by the characteristic size-descending feature maps in CNNs [102–104]; the use of  $1 \times 1$  convolution kernels right after larger convolutions [56], which is equivalent to the linear combination of the corresponding pixels along the channels in the kernel, and the exploitation of Feature Pyramid Networks (FPN) [68,105], or any other variant, such as Deep Feature Pyramid Network (DFPN) [57], FPNLite [31], or Concatenated Feature Pyramid Network (CFPN) [82], as part of the detection framework, fusing semantic information from multiscale feature maps through an architecture comprised of an upstream and a downstream path, and the downstream path being responsible for building higher resolution feature maps using a semantically rich map as a starting point. Lastly, in addition to the several techniques just mentioned, a secondary approach has been noted in this respect, focused on investigating novel blocks or architectural elements to facilitate information transfer throughout the network underlying the detection framework. More specifically, the analyzed studies report using the Squeeze-and-Excitation blocks [62], which explicitly model channel relationships and interdependencies and include a form of self-attention on channels, and spatial attention blocks [98], which can emphasize smaller targets' features, thus capturing more important information and filtering out noise.



### 3.2.2. Model Compression and Acceleration

While architectural-tweak-based techniques and methods for designing more compact object detection frameworks have been widely studied in the on-device literature, they do not always lead to solutions suitable for austere environments. Depending on how those techniques perform, but also on the target hardware platform's capabilities, they might produce models too costly to be deployed in such environments. In this context, CNN compression has proven to be an effective approach orthogonal to lightweight CNN architecture design, accelerating and reducing the size of already existing networks. This approach, shown in Table 3 as the second one with more occurrences, encompasses three distinct observed techniques: pruning [50,54,55,70,90,93,99,101], quantization [48,87,99], and knowledge distillation [48,87,99].

Pruning consists of a typically incremental process that compresses a trained model, eliminating redundant or non-informative elements of the network, being these either individual neural connections or larger structures within the network itself. It commonly entails three distinct stages: evaluating how important the parameters considered are, trimming the less relevant ones, and finally, finetuning the resulting model to recover part of the accuracy lost. Although this pipeline remains unchanged in the reviewed investigations, it is possible to observe in those studies two different pruning modalities: unstructured [99] and structured [50,54,55,70,90,93,101]. Unstructured pruning eliminates connections with lower-valued weights. Although conceptually intuitive, it results in network architectures that present an irregular structure, potentially unsuitable for exploitation in practical applications. Structured pruning, on the other hand is capable of producing more formal standard network structures as output, resulting in a structured sparsity, particularly beneficial for saving computational resources. In this modality, in turn, two types of methods are observed in the collection of papers reviewed: kernel pruning [50,54,70,93] and channel pruning [55,90,101]. Particularly concerning kernel pruning, three different techniques have been identified. Two of them, motivated by the intuition that kernels with lower-valued weights tend to produce feature maps with weaker activations, assess the relevance of every kernel present in each layer by computing the absolute sum of its weights [50,54], while the third one leverages a kernel clustering algorithm to get rid of the kernels in the network that extract similar features [93]. For its part, as far as channel pruning is concerned [55,90,101], as its name suggests, it is an approach that reduces the number of channels in convolutional kernels, thus accelerating convolutions. Each channel is assigned a scale factor representing its relevance in the first place, being the network trained later on in a sparse manner to delete channels with weight values below a given threshold.

The second main compression technique identified, data quantization, reduces the number of bits used to represent network parameters, reducing the size and computational complexity of the model, but resulting in a significant accuracy loss. In particular, 8-bit quantization, a standard practice in on-device ML, is exploited in three of the AmI research works analyzed [48,87,99]. Parameter conversion, from 32-bit floating-point precision to 8-bit precision, is shown to be a mechanism effective for compressing DL models, as well as necessary to obtain an implementation suitable for modern hardware accelerators, such as DSPs [48]. Specifically, in [87], quantization is applied to both kernel weights and data in the activation functions, [48] the authors report the application of quantization directly on the pre-trained model.

Finally, while both parameter pruning and quantization address the inherent parameter redundancy in DL models by deleting the less critical or relevant parameters, knowledge distillation is a training strategy able to generate compact neural networks but capable of producing an output similar to the one provided by more complex networks. This latter approach, represented in Table 3 by a single paper [64], goes beyond a mere compression technique in the strict sense of the word, producing lightweight models with better performance. Instead of training a compact network from scratch, it relies on a refinement strategy in which a lightweight model, namely the "student", is trained guided

by a more complex and sophisticated “teacher” model, resulting in a transfer of knowledge between the two through the matching of certain statistics.

### 3.2.3. Improved Ad Hoc Datasets

Datasets represent a key factor when conceiving DL-based detection frameworks. The exploitation of datasets is an unavoidable consideration for model training and evaluation, as well as for comparing different algorithmic alternatives. Indeed, the access to massive collections of annotated images and videos through the Internet has certainly played a significant role in the development of those models. The widespread availability of the so-called big data and the ubiquitous nature of modern information access technologies have enabled the compilation of extensive datasets, such as the well-known Pascal VOC [115] or Microsoft COCO [126], representative of a fair number and diversity of objects, boosting object detection in the direction of increasingly complex problems and more sophisticated solutions.

Such a scenario responds, however, only to general-purpose object detection. In domains, such as AmI, where object detection is applied to specific problems, the reality differs significantly from the one described in the previous paragraph. Overall, mainstream datasets are unable to capture the singularities of the distinct class instances considered for particular use cases (especially when they are very numerous or similar), and they are not able either to do the same with the context or environment where they can commonly be found. Particularly in AmI, as seen in Section 2, there is a fair number of paradigmatic use cases that have been extensively studied, where, consequently, well-known and widely proved public databases can be found, e.g., KITTI [127] for autonomous detection systems, VisDrone [128] for object detection on images taken from UAVs, or WIDER FACE [129] for face detection. However, they do not cover in any case the broad spectrum of existing related applications in the field, and overall, they lack proper size and quality, especially when compared to general-purpose detection benchmarks.

The study revealed two different approaches devised in response to the noted data-related shortcomings: the creation of ad hoc datasets for the problem addressed [50,64,68,77–80,84,86,91,95,98,100,103,109], and data augmentation [48,50,52,53,57,60,61,64,70,71,73,74,76,81,84,85,87,97,98,100,106]. As shown in Table 3, both solutions have been widely adopted by the authors, a fact quite surprising considering the substantial effort typically required to address the various tasks involved in creating new datasets (image and video acquisition and homogenization, but fundamentally, the creation of annotations). In particular, the creation of customized datasets is featured in 15 of the 62 works reviewed, proving to be highly effective in dealing with the variations among the different objects within the same class and the changing environmental conditions that might affect their visual appearance. For its part, data augmentation is reported to be used in 22 studies, supplementing datasets by directly increasing their number of samples, thus avoiding classical problems in DL, such as model overfitting and class imbalance.

Newly created AmI-specific datasets reportedly aggregate contents to cover as many variations as feasible (see a detailed list of common variations in AmI domains at the end of Section 3.1). The majority of those datasets are image sets created from scratch, either by actively taking pictures in real-world scenes [50,77,80,84,86,91,98,100,103,109] or by extracting frames from previously recorded videos [64,95] but there are also three exceptions that reuse images already present in other existing datasets [68,78,79]. As far as the covered domains are concerned, intelligent transport stands out in the study as the one where the creation of ad hoc datasets has aroused the most significant interest, with datasets covering both the surroundings [64,79] and the passenger compartment of vehicles [77,78]. Complementarily, data augmentation techniques have also been extensively exploited, extending datasets with additional samples produced in a straightforward manner thanks to relatively effortless transformations on the original images. In this regard, in the AmI literature reviewed, it has been reported that a fair number of operations can be classified into two distinct groups: operations at the pixel level and geometric

transformations. The pixel-level category involves operations that affect image pixel values, producing alterations in some properties, such as hue [50,53], saturation [50,53], contrast [76], brightness [76], exposure [50,53], color [60,61,64,76,85], even incorporating into images noise [76], or blurring [64,97], while geometric transformations includes practices, such as image scaling [48,71,76,85,100,106], translation [81], horizontal and vertical flipping [53,57,60,61,64,71,73,74,76,81,84,85,87,97,98,100], random cropping [50,52,53,60,61,73,74,85,100], target-centered cropping [48,106], rotations [50,53,70,74,76,81,84,87,97,100], resizing [76], and perspective warping [76].

#### 3.2.4. Hyperparameter Tuning

This fourth and last approach encompasses a set of methods that, although barely observed in the body of works studied, has traditionally had a significant impact on ML and DL-based solutions, pursuing the proper parametrization of certain attributes either of the underlying network architectures or the specific algorithms exploited for training in order to yield a more comprehensive performance improvement of the models produced. Particularly in AmI, it has been reported in this regard, on the one hand two different approaches that involve the configuration of the anchor boxes used for the prediction tasks performed by object detection frameworks [60,61,79,95] and on the other hand a better-represented approach that explores more appropriate or representative loss functions [53,60,64,92,95,105,107].

Anchor boxes or “priors” represent a concept intrinsic to modern CNN-based object detection frameworks. Thus, for every image provided as input, detectors typically produce a broad sample of anchor boxes, adopted as candidate regions examined by the algorithm to determine whether or not they contain any of the objects of interest considered, with this decision being made according to the overlapping of the anchor boxes themselves with the ground-truth boxes (a metric known as Intersection over Union, IoU). That said, a particular case is assumed to be positive when the computed overlap value is greater than a predefined threshold, triggering the refinement of the candidate region boundaries to fit the enclosed object better and thus, provide a more accurate prediction of the output bounding box. In that sense, although each feature map in the underlying CNN is likely to have several candidate regions, potentially only a small percentage of them actually contains objects, leading to a clear imbalance between positive and negative samples and consequently, to a waste of resources. Anchor box filtering [79] addresses this problem by using an object-priority-based labeling mechanism to filter out a significant chunk of the negative samples. Hence, it reduces the object search space considerably, also mitigating the detector’s computational overhead. Furthermore, anchor boxes’ dimensions and their aspect ratio largely determine the size and shape of the objects a detector is able to detect, to the point that if the priors do not match the different objects in the exploited dataset (for instance, because they are tiny or irregularly shaped), the detector might omit them. Motivated by this issue, two research works exploring different anchor box scaling mechanisms were identified in the analysis of on-device detectors within AmI: [59] which proposes a multiscale-box-based strategy for predicting targets with different scales and [95] that explores the utilization of small anchors to detect small and partially occluded targets.

Lastly, with regards to the loss function, which drives the learning process of deep neural networks (DNN) and so it does with CNN-based object detection frameworks, it is a well-known element that has shown a significant impact on the accuracy level of the predictions produced by such algorithmic solutions. In particular, when it comes to the object detection frameworks studied, several enhancement strategies have been observed on two different types of loss: the classification loss, necessary to train detectors on object class prediction tasks, and the localization loss, required as well, to do the same, in this case, with bounding box regression. It is precisely concerning the latter, the location loss, that we find various specific optimization techniques within the AmI space. Three of them pursue a better performance in terms of accuracy, exploring possible loss factors more representative of the particular problems addressed [92,105,107]. Both in [92] and [105],

unlike the deterministic-regression-based approach usually employed in mainstream object detection frameworks, authors opt to introduce uncertainty in localization prediction by modeling bounding box vertices using Gaussian parameters, while in [107] they explore a metric alternative to the standard IoU with better capability for representing two-object overlapping. Furthermore, accuracy boost aside, a more suitable definition of the loss function has proved to effectively mitigate the class imbalance problem. In particular, representing this line of action, we can find in Table 3, three works [53,60,64] that incorporate a focal loss factor into the loss function, filtering in that way simple samples automatically and thus, increasing the weight of complex samples.

#### 4. Study and Evaluation

To conclude the present revision of research on the design of on-design object detection solutions in the context of AmI, it is necessary to provide a brief overview of the most significant aspects involved in the evaluation and comparison of such techniques. The study and evaluation of the models, in the same way as, for example, the design, coding, tuning, or training of the networks, constitute an unavoidable step when implementing a DL or ML solution. Creating a robust and reliable test setup is a fundamental task, not only to ensure the adequate performance of the detection framework conceived but also to guide the iterative refinement or improvement typically involved in the design of such complex models. It is, therefore, necessary to present, even if only briefly, the main factors or elements to consider both when creating a test setup and when analyzing the results obtained (besides the detector or the several detectors of interest and the various datasets selected for benchmarking). Accordingly, Table 4 collects the several hardware platforms commonly used in the considered context of the study, a fundamental decision when assessing the performance of the solutions in real-world scenarios with memory and computational constraints, together with the main aspects or values representing detection frameworks' performance, as well as the most widely used associated metrics.

**Table 4.** Summary of evaluation frameworks used for assessing on-device HAR models' performance in the reviewed studies.

Work	Test Device	Hardware Acceleration	Accuracy	Speed	Model Size	Computational Complexity	Real Time
[48]	–	• DSP	• TPR	• FPS	• Weight	FLOPS	Yes
[67]	• RSPB 3 B	No	• FPR • FNR	• FPS	• Weight	–	No
[68]	• ANDRD	–	• P • R	–	–	–	Yes
[69]	• XU4 • RSPB 3 B	No	• TPR • P	• FPS	–	–	No
[70]	• NVD TX1	• GPU	• mAP	• FPS	• Params • Weight	–	No
[71]	• NVD TX2	• GPU	• ACC • AUC-ROC	• FPS	• Params	FLOPS	Yes
[72]	• RSPB	–	–	–	–	–	–
[49]	• NVD TX2	• GPU	–	• FPS	–	FLOPS	Yes
[50]	• NAO	• No	• mAP	–	–	FLOPS	–

Table 4. Cont.

Work	Test Device	Hardware Acceleration	Accuracy	Speed	Model Size	Computational Complexity	Real Time
[51]	–	–	• DR	–	• Weight	FLOPS	–
[52]	• NVD TX1	• GPU	• mAP	• FPS	–	–	No
[53]	• NVD TX2	• GPU	• mIoU	• FPS	• Params	FLOPS	Yes
[54]	• NVD TX2	• GPU	• mAP	• FPS	• Params • Weight	–	Yes
[55]	–	• Desktop GPU	• P • R • F1-S • mAP	• FPS • IT	• Params • Weight	FLOPS	Yes
[73]	• ANDRD	–	–	–	–	–	–
[74]	• ANDRD	• No	• mAP	• IT	• Params	MAdds	No
[56]	• NVD TX2	• GPU	• mAP	• IT	• Weight	–	No
[57]	• NVD TX2	• GPU	• mAP	• FPS	–	–	Yes
[75]	• RSPB 3 UP2	• VPU (NCS) • FPGA	• mAP	• FPS	–	–	No
[76]	• NVD TX2	• GPU	• F1-S	• FPS	–	–	Yes
[77]	• NVD TX2	• GPU	• ACC • FP	• IT	–	–	Yes
[58]	• DSKTP	–	• mAP	–	–	–	–
[78]	• RSPB 3 ANDRD	• No	• mAP	• FPS	–	–	No
[79]	• DSKTP	–	• mAP	• FPS	–	–	Yes
[80]	• ANDRD	–	• mAP • F1-S	–	–	–	–
[59]	• DSKTP	• GPU	• AP	• IT	–	–	Yes
[81]	–	–	–	–	–	–	–
[82]	• NVD TX2	• GPU	• mAP	• FPS	–	–	Yes
[60]	• RSPB 3 B+	• No	• mAP	• FPS	• Params • Weight • Memory Footprint	FLOPS	Yes

Table 4. Cont.

Work	Test Device	Hardware Acceleration	Accuracy	Speed	Model Size	Computational Complexity	Real Time
[83]	• NVD TX2	• GPU	• mAP	• FPS	–	–	Yes
[61]	• DSKTP	• GPU	• mAP	• FPS	• Params	MAdds	–
[84]	–	–	• mAP	• FPS	–	–	Yes
[62]	• DSKTP	• No	• AP	• FPS	• Params	–	Yes
[63]	• ANDRD	–	• AP	–	–	–	No
[85]	• DSKTP	• No	• AP	• FPS	• Params	FLOPS	Yes
[86]	• DSKTP	• GPU	• mAP	–	• Weight	FLOPS	–
[87]	• ANDRD	• No	• TPR • FNR • FDR	• IT	• Weight	–	No
[88]	–	–	–	–	–	–	–
[89]	• NVD TX2	• GPU	• mAP • R • P	–	–	–	Yes
[90]	–	–	• AP	• IT	• Params	FLOPS	No
[91]	• NVD NN • DSKTP	–	• AP • F1-S	• IT	–	–	Yes
[92]	• NVD AGX	• GPU	• mAP	• FPS	–	FLOPS	Yes
[64]	–	–	• mAP	• IT	• Weight	FLOPS	Yes
[93]	• NVD TX2 • NVD NN	• GPU	• AP	• FPS	–	–	Yes
[94]	• NVD TX2	• GPU	–	–	–	–	–
[95]	• RK3399	• GPU	• FNR	• IT	• Params	–	Yes
[96]	–	–	• P • R • F1-S • mAP	• IT	–	–	No
[97]	–	–	• P • R • F1-S • mAP	–	–	–	–
[98]	–	–	• mAP	• IT	• Weight	–	No

Table 4. Cont.

Work	Test Device	Hardware Acceleration	Accuracy	Speed	Model Size	Computational Complexity	Real Time
[99]	• IOS	• No	• P • R • mAE	–	• Params	–	No
[100]	• RSPB 3 B+	• VPU (NCS V2)	• AP	• IT • FPS	• Params • Weight	–	No
[101]	–	–	• AP	• IT • FPS	• Weight	–	No
[65]	• NVD NN	• GPU	• P • R • ACC	• IT	–	–	No
[102]	–	–	• P • R • mAP	• FPS	• Weight	–	Yes
[103]	• RSPB 3 B+ • NVD NN • NVD AGX	• VPU (NCS V1/V2) • GPU	• P • R • mAP	• FPS	–	–	Yes
[104]	–	–	• mAP	–	–	–	–
[105]	• NVD AGX	• GPU	• mAP	• FPS	• Weight	FLOPS	Yes
[106]	• NVD AGX	• GPU	• P • R	• FPS	–	–	Yes
[107]	–	–	• F1-S • mAP	• FPS	• Weight	–	Yes
[108]	• RK3399 pro	• NPU	–	–	–	–	Yes
[66]	• NVD TX2	• GPU	• F1-S	• IT	–	–	No
[109]	• DSKTP	–	• P • R • F1-S • mAP	• FPS	• Params • Weight	FLOPS	Yes
[110]	• MS HLENS	No	• AP	–	–	–	No

Abbreviations: RSPB, Raspberry Pi board; ANDRD, Android smartphone; XU4, Odroid XU4 board; NVD TX1, Nvidia Jetson TX1; NVD TX2, Nvidia Jetson TX2; NAO, NAO robot; UP<sup>2</sup>, Up Squared board; NCS, Intel Neural Compute Stick; DSKTP, Desktop system; NVD NN, Nvidia Jetson Nano; CORAL USB, Google Coral USB Accelerator; NVD AGX, Nvidia Jetson AGX Xavier; RK3399, RK3399 Rockchip board; IOS, iOS smartphone; MS HLENS, Microsoft HoloLens; DSP, Digital Signal Processors; GPU, Graphics Processing Unit; TPU, Tensor Processor Unit; VPU, Vision Processing Unit; NPU, Neural Processing Unit; TPR, True Processing Rate; FPR, False Positive Rate; FNR, False Negative Rate; P, precision; R, recall; mAP, mean Average Precision; ACC, accuracy; AUC-ROC, area under the ROC curve; DR, detection rate; mIoU; F1-S, F1 score; FP; AP, Average Precision; FDR, False Detected Rate; mAE, mean Area Error; FPS, Frames Per Second; IT, inference time; FLOPS, Floating Point Operations Per Second; MAdds, Multiply-Adds operations.

#### 4.1. Hardware Platforms

In the last few years, a plethora of new cutting-edge mobile and embedded devices have appeared in the market as potential AI-supporting hardware platforms suited for the implementation and continuous improvement of on-device DL models due to their ever-increasing computational power and modest energy consumption compared to the more traditional server or desktop alternatives. In particular, mobile SoCs (System on a Chip), adopted in today's mass-produced embedded devices, such as mainstream single-board computers, have been and still are, definitely, one of the main drivers of the recent on-device trend. Their physical design based on the tight integration of computing, memory, and communication components in a single integrated circuit not only optimizes the internal communication among components and maximizes energy efficiency but also minimizes chips' waste heat, as well as their die area, enabling the miniaturization of the devices built on its basis.

Table 4, in particular, the field tagged as "Test device", lists the several devices used in the AmI research analyzed for evaluation purposes. It is possible to classify those devices into three different groups: high-performance single-board computers [49,52–54,56,57,65,66,70,71,75–77,82,83,89,91–94,103,105,106,108], low-power single-board computers [60,67,69,72,75,78,95,100,103], and mobile devices [63,68,73,74,80,87,99]. This categorization, however, does not contemplate papers where the hardware setup used for model deployment is not properly detailed or detailed at all [48,51,55,64,81,84,88,90,96–98,101,102,104,107], nor the ones that report using desktop systems for that purpose [58,59,61,62,79,85,86,109]. Moreover, in addition to those omissions, there are two punctual investigations also excluded from the previous classification where authors propose detectors specially designed for specific-purpose hardware: humanoid robots programmed to play soccer [50], and a mixed reality headset used to overlay virtual information [110] on the real world.

The group of high-performance single-board computers includes low-energy embedded devices designed specifically for accelerating ML applications through a dedicated built-in processor. As the data presented in Table 4 reflect, this category is monopolized by Nvidia's Jetson family. Up to 23 of the 47 papers detailing the evaluation hardware setup exploited report using at least one device belonging to the Jetson hardware platform for AI processing on edge, being the latter a distributed computation paradigm based on an intermediate layer of resource-constrained devices located physically closer to data source where computation is partially offloaded, thus mitigating latency and reducing response times comparing to the cloud alternative. In particular, the specific devices observed in this respect, listed in ascending order according to their computational power, are: Nvidia Jetson Nano [65,91,93,103], Nvidia Jetson TX1 [52,70], Nvidia Jetson TX2 [49,53,54,56,57,66,71,76,77,82,83,89,93,94] and Nvidia Jetson AGX Xavier [92,103,105,106]. All of them are based on CPUs with ARM architecture, ranging from four cores in the Jetson Nano and Jetson TX series to eight cores in the Nvidia Jetson AGX Xavier. However, their actual AI processing capability lies in the powerful GPUs they embed, delivering performance equivalent to desktop graphics chipsets of recent past years. Their physical design, based on a fair number of cores (128 in the Jetson Nano, 256 in the Jetson TX1 and Jetson TX2, and 512 in the Jetson AGX Xavier), makes it possible to parallelize repetitive operations and thus perform matrix computation more efficiently than general-purpose CPUs. They present, however, more limited memory than their desktop counterparts, insufficient for unrolling and converting convolution operations into matrix operations. For that reason, as a rule of thumb, models trained using desktop GPUs must be converted into models suitable for mobile GPUs, transforming matrix multiplications into dot product operations.

Still, within the high-performance single-board computers group, two alternatives to the Nvidia devices have been observed: the RK3399 pro [108] and Up Squared [75] boards. Although more modest than the Jetson series as far as CPU and GPU are concerned, the RK3399 pro integrates, unlike them, a Neural Processor Unit (NPU) as the primary AI



inference acceleration component. Those NPUs, as we will see for other devices, depending on the manufacturer, can present different commercial names, such as Tensor Processing Unit (TPU), Neural Network Processor (NNP), Intelligence Processing Unit (IPU), or Vision Processing Unit (VPU). They all refer to specialized circuits that implement all the control logic and arithmetic necessary to execute ML algorithms. In the particular case of RK3399 pro's SoC, the CPU embeds an NPU with power enough to support both 8-bit and 16-bit data, yielding a computing performance of up to 3.0 TOPS (Tera Operations per Second), at the cost of power consumption, according to the manufacturer, of only 1% of that of the GPU. For its part, the Up Squared board is the only device with an x64 architecture among all the ones identified in the review, standing out because it integrates an FPGA, an acceleration solution that, although less costly than NPUs or GPUs, has shown more potential than the latter in DL tasks, with higher flexibility to operate with different types of data (binary, ternary and even customized), and strong capacity to deal with the usual irregular parallelism that characterizes sparse DNN-based algorithms.

In the second category of hardware platforms exploited for evaluation, we find single-board embedded devices again, but, in this case, governed by more modest SoCs, embedding general-purpose mobile CPUs and GPUs with significantly lower cost and power consumption. Raspberry Pi (in its different versions) [60,67,69,72,75,78,100,103], Odroid XU4 [69], and the standard version of the RK3399 board [95], the three representative devices observed in the Aml literature analyzed, show, in general terms, limited hardware capabilities, especially if we compare them with the typically demanding requirements of DL tasks, yielding a performance far from the pursued real-time. Hence, with the end goal of compensating to some extent such shortcomings, some plug-and-play ML-accelerating devices have recently emerged on the market, which, typically connected to the host device through a USB port, act as a dedicated coprocessor, enabling a high-speed ML inference on a wide range of conventional systems. The Intel Neural Compute Stick, in its two versions, is shown in Table 4 as the most popular alternative [75,100,103]. Built on Intel's Movidius Myriad X VPU, it incorporates the inference capability of 16 programmable shaving cores and a dedicated neural computation engine into the end device. Still, if we take as a reference the data collected in Table 4, it can be stated that its use remains merely testimonial, and its impact on the resulting final performance obtained is not sufficient yet.

Finally, the third group includes only mobile devices, in the strict sense of the term. Android systems practically monopolize this last category (there is only one iOS device [99] in the eight papers where smartphones are used for evaluation), with Qualcomm and its Snapdragon SoC [68,74,78,87] being the preferred hardware configuration. Essentially through its 8 series, Snapdragon has not only led the "flagship" smartphone market in recent years but has also been one of the main spearheads of the technological development in the mobile world, progressively incorporating in the last few years new components specialized in AI: from the Qualcomm Hexagon DSP (Digital Signal Processor) introduced in the Snapdragon 835 [68], all the way to the recent and more sophisticated Qualcomm AI Engine embedded in the Snapdragon 865 [87], where the Qualcomm Hexagon Tensor Accelerator DSP, capable of performing 15 trillion operations per second, is jointly exploited with the Adreno GPU and the Kryo CPU cores as a comprehensive acceleration solution. DSPs can perform part of the computations involved in ML and DL processes with high efficiency, alleviating the workload of the other cores and thus reducing power consumption.

#### 4.2. Evaluation Metrics

The analysis of on-device models goes beyond the mere evaluation of detector performance (typically modeled in terms of accuracy and speed), additionally covering cost-specific aspects (i.e., model size and computational complexity), which are fundamental to assessing the feasibility of such solutions in resource-limited devices. Although cost is a concern that can be considered inherent to the on-device paradigm, that premise is not reflected in the data presented in Table 4. Instead, they largely omit information regarding the size [49,50,52,57–59,63,65,66,68,69,72,73,75–84,88,89,91–94,96,97,103,104,106,108,110]

and complexity of the models produced [52,54,56–59,62,63,65–70,72,73,75–84,87–89,91,93–104,106–108,110], with no values in those respect reported in more than half of the papers. On the contrary, speed and accuracy are addressed in virtually all the AmI studies analyzed [48,52–57,59–62,64–67,69–71,74–79,82–85,87,90–93,95,96,98,100–103,105–107,109], revealing a trend consistent with the experimentation traditionally conducted for evaluating conventional ML and DL techniques.

As far as accuracy is concerned, the related metrics that emerge in Table 4 as the most popular options are average precision (AP) and its variant, the mean average precision (mAP) [50,52,54–64,70,74,75,78–80,82–86,89–93,96–98,102–105,107,109,110], distantly followed by precision (P) [55,65,68,69,89,96,97,99,102,103,106,109], recall (R) [55,65,68,89,96,97,99,102,103,106,109], and F1 score [55,66,76,80,91,96,97,107,109]. AP is computed by selecting the 11 best predictions produced by a detection framework for a single object. Specifically, given the number of true positives (TP), for each of the 11 selected predictions, both precision (P) and recall (R) values are calculated, being TP every correct detection present in the ground truth, represented as a bounding box. For its part, precision is the ratio between the number of TP and the total number of detections, representing the ability of a model to identify only the relevant objects, while recall is the percentage of TPs over the total number of detections present in the ground truth, being the result indicative of the model's ability to find all relevant cases (all ground-truth bounding boxes). Lastly, the F1 score is computed as the weighted average of precision and recall, resulting in a value between 0 and 1, where 1 represents the highest precision.

Furthermore, the object detection framework's speed is the time it requires to process a given image or video frame to output the various bounding boxes representing the location of the objects of interest together with the labels identifying the class that objects are associated with. Such inference speed is usually denoted as the number of frames processed per second [48,49,52–55,57,60–62,67,69–71,75,76,78,79,82–85,92,93,100–103,105–107,109], although, as shown in the "Speed" column of Table 4, there are a good number of authors who choose to directly report the average time spent in the inference process [55,56,59,64–66,74,77,87,90,91,95,96,98,100,101] (typically measured in milliseconds).

Accuracy and speed are closely related to the complexity of the trained model: a superior model representation capacity makes it possible to produce more accurate predictions, but, on the flip side, it generally results in a longer inference time. Such complexity, as done by a fair number of authors, might be intuitively evaluated in terms of model size, reporting the results produced either in the form of the number of parameters [53–55,60–62,70,71,74,85,90,95,99,100,109], their weight [48,51,54–56,60,64,67,70,86,87,98,100–102,105,107,109], or their memory footprint [60]. The number of parameters responds to the number of model weights learned during the training process, essentially considering in that respect the parameters in convolution and fully connected layers. Weight, in turn, stands for the size of the parameters counted, while the memory footprint, although sometimes erroneously used as a synonym of the former, corresponds to the estimation of the space in main memory required by the model at runtime, thus constituting the measure that best characterizes the requirements of such nature.

Model size, particularly the number of parameters and their weight, are straightforward metrics capable of providing a first intuition on its resource requirements at a glance. However, when discussing an algorithmic solution's computational complexity in a more rigorous fashion, it is necessary, especially within the on-device paradigm, to know the way computations are carried out and their cost in terms of demanded resources. In this regard, despite the scarcity shown in the AmI works reviewed, it is possible to observe additional representative metrics, commonly reported in DL studies to supplement the accuracy and inference information provided, denoting a model's computational complexity by the number of operations involved, either directly as MAdds (number of multiplication-addition operations) [61,74], or indirectly as FLOPS (number of floating-point operations per second) [48–51,53,55,60,64,71,85,86,90,92,105,109].

## 5. Conclusions and Future Work

This paper reviews the more relevant ambient intelligent recent research focused on the study and exploitation of lightweight object detection frameworks capable of address the inference process locally, thus ensuring more robust data security and better user privacy within intelligent environments. Specifically, the study carried out provides a comprehensive analysis of such frameworks, discussing in an organized and schematic way (i) the application domains where those techniques have proven to be particularly useful, (ii) the various CNN architectures designed explicitly for devising more efficient and compact detection systems, (iii) the challenges associated with the design process of such systems, together with (iv) the different approaches explored in response to them, and lastly, (v) the hardware setups and metrics adopted by researchers and AI practitioners for assessing the solutions proposed.

Ultra-compact detectors, such as Tiny YOLO in its different versions or SSDLite, emerge in the reviewed literature as the most salient on-device options among the several lightweight detectors exploited in AmI. Adopting a unified-detection-pipeline-based architecture or model as a starting point, primarily oriented towards higher inference speed, alongside the integration of a more refined and simplified network model as the backbone, makes it possible to build fast detection systems. Such an approach, however, delivers processing times that fail to achieve real-time performance in many scenarios and what is even worse, it derives in a dramatic accuracy loss in comparison with existing state-of-the-art alternatives. In that sense, most of the research efforts observed have been aimed, overall, at finding mechanisms and strategies for a better accuracy-speed trade-off, focusing on accuracy or speed depending on the nature of the architecture or model adopted as a baseline, but also depending on the particular application pursued.

Networks resulting from more aggressive compression and optimization approaches have proven to fall short in use cases where accuracy is crucial (e.g., autonomous driving, and robotic systems for task automation), thus demanding methods capable of producing more expressive CNNs, either through richer feature hierarchies, such as the fusion of features extracted at different levels of the network, deeper network architecture or the exploitation of building blocks, such as the attention mechanism. On the other hand, detection frameworks based on compact networks but more conservative in terms of the accuracy provided typically feature an excessive size and resource consumption, which, while not hindering their deployment in austere systems or execution environments, do weigh down their performance significantly by degrading response speed. Therefore, they require optimization efforts to reduce the size of the models produced, as well as the computational power and memory space required for their execution. In this regard, architecture-tweaking approaches, such as exploiting more efficient convolution operations or decreasing the number of convolution kernels per layer constitute today's mainstream practices. However, techniques such as pruning and quantization have shown to be complementary widespread improvement options.

In addition to speed- and accuracy-specific challenges and solutions, both closely related to object detection on low-power end devices, a collection of domain-specific issues arise when developing AmI applications. The reuse of models previously trained on large public data sets leads in the first instance to general-purpose solutions that, overall, do not fit the specificities of the AmI scenarios observed, resulting in less accurate and robust detectors. In this regard, transfer learning has proved to be effective in dealing with that gap, successfully tailoring a given model to a particular use case by further training it on a new dataset more representative of the new target task. Creating such datasets involves, however, a tedious and costly process, typically based on manual annotation and thus prone to human error. That usually results in datasets that, on the one hand are not big enough for DL training, and on the other hand present quality deficiencies, such as class imbalance. Furthermore, a fair number of authors have focused part of their research efforts on acquiring and annotating images representative of the actual context where the resulting detection systems would be used. Many produced datasets have been reportedly

insufficient to cover the typical highly-changing environmental conditions in the wild (e.g., brightness, viewpoint and weather) or deal with the intra-class variance problem, no matter the latter is due to the very nature of the targets (e.g., different facial expressions in face detection, and different clothing and poses in human detection[109]) or to alterations caused either by the environment itself or by the rest of entities present in it (e.g., total or partial occlusion).

The selection of a proper lightweight detection framework, jointly with the implementation of improvement tweaks eminently oriented to both the reduction of size and complexity of the underlying CNN network and the creation of a more robust and adequately sized dataset, have yielded promising results as can be seen in the data presented in the column tagged as “Real-time” in Table 4. Nevertheless, although some works confirm the feasibility of real-time detection solutions on mobile and embedded devices in AmI contexts, an overwhelming majority of them either fail to achieve such efficiency or present important gaps in the evaluation report that degrade the robustness of the results. In particular, with respect to the latter point, the table reveals (i) research that did not consider execution speed as an evaluation metric and that only report accuracy values; (ii) studies that omit information about the materials used for testing the models’ performance on inference tasks, reporting in many of those cases only the hardware setups used for training and (iii) works that report achieving real-time performance, but on high-end GPU-powered desktop systems far from the strong memory and computational limitations that characterize the on-device paradigm.

Future research should address the deficiencies mentioned above, dealing with the computational complexity of the solutions devised, and also addressing hardware-specific concerns that may affect their final performance, generalizing the study of energy consumption (approached punctually in [53,57,83]), as well as other interesting related matters, such as the proper use of parallelism strategies or how to exploit jointly modern multi-core architectures and AI acceleration hardware in a proper way. Likewise, in order to overcome data scarcity, it will be necessary to either explore techniques to alleviate or streamline the dataset creation process (e.g., synthetic data generation based on Generative Adversarial Networks [130,131], or image and video acquisition in simulated environments [132]), or devise DL alternatives that demand a smaller volume of data (e.g., the so-called few-shot learning techniques [133,134]). Finally, although the body of works considered in the study represents a broad spectrum of applications within ambient intelligence, it does not cover paradigmatic scenarios in the field, such as workplaces, educational centers, or smart homes. Vision-based methods and techniques, such as those presented, featuring offline solving capabilities, will undoubtedly push forward the implementation of intelligent systems in such contexts, where data security and individuals’ privacy are hard requirements. It will, however, be necessary to approach the specificities of each of them in order to maximize the performance of the proposed solutions.

**Author Contributions:** Conceptualization, methodology, and investigation, I.R.-C., C.C. and F.F.-R.; writing—original draft creation—and figure creation, I.R.-C.; writing—review and editing, supervision and project administration, C.C. and F.F.-R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Consellería de Educación, Universidades e Formación Profesional (Xunta de Galicia) under the scope of the strategic funding ED431C2018/55-GRC Competitive Reference Group and the “Centro singular de investigación de Galicia” (accreditation 2019-2022) funded by the European Regional Development Fund (ERDF)-Ref. ED431G2019/06.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Wang, Z.; Zheng, L.; Liu, Y.; Li, Y.; Wang, S. Towards Real-Time Multi-Object Tracking. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 107–122.
2. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 172–186. [\[CrossRef\]](#)
3. Zhang, H.B.; Zhang, Y.X.; Zhong, B.; Lei, Q.; Yang, L.; Du, J.X.; Chen, D.S. A comprehensive survey of vision-based human action recognition methods. *Sensors* **2019**, *19*, 1005. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
5. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
6. Dunne, R.; Morris, T.; Harper, S. A Survey of Ambient Intelligence. *ACM Comput. Surv.* **2021**, *54*, 1–27. [\[CrossRef\]](#)
7. Sadri, F. Ambient intelligence: A survey. *ACM Comput. Surv.* **2011**, *43*, 1–66. [\[CrossRef\]](#)
8. Remagnino, P.; Foresti, G.L. Ambient Intelligence: A New Multidisciplinary Paradigm. *IEEE Trans. Syst. Man Cybern. —Part A Syst. Hum.* **2005**, *35*, 1–6. [\[CrossRef\]](#)
9. Gandodhar, P.S.; Chaware, S.M. Context Aware Computing Systems: A survey. In Proceedings of the 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 30–31 August 2018; pp. 605–608.
10. Augusto, J.; Aztiria, A.; Kramer, D.; Alegre, U. A Survey on the Evolution of the Notion of Context-Awareness. *Appl. Artif. Intell.* **2017**, *31*, 613–642. [\[CrossRef\]](#)
11. Cook, D.J.; Augusto, J.C.; Jakkula, V.R. Review: Ambient intelligence: Technologies, applications, and opportunities. *Pervasive Mob. Comput.* **2009**, *5*, 277–298. [\[CrossRef\]](#)
12. Mawela, T. Ambient Intelligence Systems for the Elderly: A Privacy Perspective. In Proceedings of the Computational Science and Its Applications—ICCSA 2020, Cagliari, Italy, 1–4 July 2020; Springer: Cham, Switzerland, 2020; pp. 875–888.
13. Friedewald, M.; Vildjiounaite, E.; Punie, Y.; Wright, D. The Brave New World of Ambient Intelligence: An Analysis of Scenarios Regarding Privacy, Identity and Security Issues. In Proceedings of the Security in Pervasive Computing, York, UK, 18–21 April 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 119–133.
14. Theoharidou, M.; Marias, G.; Dritsas, S.; Gritzalis, D. The ambient intelligence paradigm A review of security and privacy strategies in leading economies. In Proceedings of the 2006 2nd IET International Conference on Intelligent Environments—IE 06, Athens, Greece, 5–6 July 2006; IET.: London, UK, 2006; pp. 213–219.
15. Caire, P.; Moawad, A.; Efthymiou, V.; Bikakis, A.; Le Traon, Y. Privacy challenges in Ambient Intelligence systems. *J. Ambient. Intell. Smart Environ.* **2016**, *8*, 619–644. [\[CrossRef\]](#)
16. Gomez, C.; Chessa, S.; Fleury, A.; Roussos, G.; Preuveneres, D. Internet of Things for enabling smart environments: A technology-centric perspective. *J. Ambient. Intell. Smart Environ.* **2019**, *11*, 23–43. [\[CrossRef\]](#)
17. Cai, Y.; Genovese, A.; Piuri, V.; Scotti, F.; Siegel, M. IoT-based Architectures for Sensing and Local Data Processing in Ambient Intelligence: Research and Industrial Trends. In Proceedings of the 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Auckland, New Zealand, 20–23 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
18. Streitz, N.; Charitos, D.; Kaptein, M.; Böhlen, M. Grand challenges for ambient intelligence and implications for design contexts and smart societies. *J. Ambient. Intell. Smart Environ.* **2019**, *11*, 87–107. [\[CrossRef\]](#)
19. Abhari, M.; Abhari, K. Ambient Intelligence Applications in Architecture: Factors Affecting Adoption Decisions. In Proceedings of the Advances in Information and Communication, Cham, Switzerland, 9–11 September 2020; pp. 235–250.
20. Röcker, C. Perceived Usefulness and Perceived Ease-of-Use of Ambient Intelligence Applications in Office Environments. In Proceedings of the Human Centered Design, San Diego, CA, USA, 19–24 July 2009; Springer: Berlin/Heidelberg, Germany; pp. 1052–1061.
21. Hasanov, A.; Laine, T.H.; Chung, T.-S. A survey of adaptive context-aware learning environments. *J. Ambient. Intell. Smart Environ.* **2019**, *11*, 403–428. [\[CrossRef\]](#)
22. Kanagarajan, S.; Ramakrishnan, S. Ubiquitous and Ambient Intelligence Assisted Learning Environment Infrastructures Development—A review. *Educ. Inf. Technol.* **2018**, *23*, 569–598. [\[CrossRef\]](#)
23. Karthick, G.S.; Pankajavalli, P.B. Ambient Intelligence for Patient-Centric Healthcare Delivery: Technologies, Framework, and Applications. In *Design Frameworks for Wireless Networks*; Das, S.K., Samanta, S., Dey, N., Kumar, R., Eds.; Springer: Singapore, 2020; pp. 223–254.
24. Haque, A.; Milstein, A.; Fei-Fei, L. Illuminating the dark spaces of healthcare with ambient intelligence. *Nature* **2020**, *585*, 193–202. [\[CrossRef\]](#)
25. Uddin, M.Z.; Khaksar, W.; Torresen, J. Ambient Sensors for Elderly Care and Independent Living: A Survey. *Sensors* **2018**, *18*, 2027. [\[CrossRef\]](#)

26. El murabet, A.; Abtoy, A.; Touhafi, A.; Tahiri, A. Ambient Assisted living system's models and architectures: A survey of the state of the art. *J. King Saud Univ. —Comput. Inf. Sci.* **2020**, *32*, 1–10. [[CrossRef](#)]
27. Ramkumar, M.; Catharin, S.S.; Nivetha, D. Survey of Cognitive Assisted Living Ambient System Using Ambient intelligence as a Companion. In Proceedings of the 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 29–30 March 2019; pp. 1–5.
28. Calvaresi, D.; Cesarini, D.; Sernani, P.; Marinoni, M.; Dragoni, A.F.; Sturm, A. Exploring the ambient assisted living domain: A systematic review. *J. Ambient. Intell. Humaniz. Comput.* **2017**, *8*, 239–257. [[CrossRef](#)]
29. Salih, A.S.M. A Review of Ambient Intelligence Assisted Healthcare Monitoring. *Int. J. Comput. Inf. Syst. Ind. Manag.* **2014**, *5*, 741–750.
30. Dhar, S.; Guo, J.; Liu, J.; Tripathi, S.; Kurup, U.; Shah, M. On-Device Machine Learning: An Algorithms and Learning Theory Perspective. *arXiv* **2019**, arXiv:1911.00623.
31. Guo, K.; Zeng, S.; Yu, J.; Wang, Y.; Yang, H. A Survey of FPGA-Based Neural Network Accelerator. *arXiv* **2017**, arXiv:1712.08934.
32. Deng, L.; Li, G.; Han, S.; Shi, L.; Xie, Y. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proc. IEEE* **2020**, *108*, 485–532. [[CrossRef](#)]
33. Cheng, J.; Wang, P.; Li, G.; Hu, Q.; Lu, H. Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. *Front. Inf. Technol. Electron. Eng.* **2018**, *19*, 64–77. [[CrossRef](#)]
34. Qin, H.; Gong, R.; Liu, X.; Bai, X.; Song, J.; Sebe, N. Binary neural networks: A survey. *Pattern Recognit.* **2020**. [[CrossRef](#)]
35. Wu, X.; Sahoo, D.; Hoi, S.C.H. Recent advances in deep learning for object detection. *Neurocomputing* **2020**, *396*, 39–64. [[CrossRef](#)]
36. Chahal, K.; Dey, K. A Survey of Modern Object Detection Literature using Deep Learning. *arXiv* **2018**, arXiv:1808.07256.
37. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection With Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)]
38. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A survey of deep learning-based object detection. *IEEE Access* **2019**, *7*, 128837–128868. [[CrossRef](#)]
39. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M.; Wang, X.; Fieguth, P.; Chen, J.; et al. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [[CrossRef](#)]
40. Khan, A.; Sohail, A.; Zahoor, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**. [[CrossRef](#)]
41. Sultana, F.; Sufian, A.; Dutta, P. A review of object detection models based on convolutional neural network. *Adv. Intell. Syst. Comput.* **2020**, *1157*, 1–16. [[CrossRef](#)]
42. Gams, M.; Gu, I.Y.-H.; Härmä, A.; Muñoz, A.; Tam, V. Artificial intelligence and ambient intelligence. *J. Ambient. Intell. Smart Environ.* **2019**, *11*, 71–86. [[CrossRef](#)]
43. Ramos, C. Ambient Intelligence—A State of the Art from Artificial Intelligence Perspective. In Proceedings of the Progress in Artificial Intelligence, Guimarães, Portugal, 3–7 December 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 285–295.
44. Patel, A.; Shah, J. Sensor-based activity recognition in the context of ambient assisted living systems: A review. *J. Ambient. Intell. Smart Environ.* **2019**, *11*, 301–322. [[CrossRef](#)]
45. Bansal, A.; Ahirwar, M.K.; Shukla, P.K. A Survey on Classification Algorithms Used in Healthcare Environment of the Internet of Things. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 883–887. [[CrossRef](#)]
46. Augusto, J.C.; Callaghan, V.; Cook, D.; Kameas, A.; Satoh, I. "Intelligent Environments: A manifesto". *Hum. -Cent. Comput. Inf. Sci.* **2013**, *3*, 12. [[CrossRef](#)]
47. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
48. Tripathi, S.; Dane, G.; Kang, B.; Bhaskaran, V.; Nguyen, T. LCDet: Low-Complexity Fully-Convolutional Neural Networks for Object Detection in Embedded Systems. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 411–420.
49. Azimi, S.M. ShuffleDet: Real-Time Vehicle Detection Network in On-Board Embedded UAV Imagery. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2019; pp. 88–99.
50. Keefe, S.O.; Villing, R. Evaluating pruned object detection networks for real-time robot vision. In Proceedings of the 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Torres Vedras, Portugal, 25–27 April 2018; pp. 91–96.
51. Gao, H.; Tao, W.; Wen, D.; Liu, J.; Chen, T.; Osa, K.; Kato, M. DupNet: Towards Very Tiny Quantized CNN With Improved Accuracy for Face Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019; pp. 168–177.
52. Ünel, F.Ö.; Özkalayci, B.O.; Çiğla, C. The Power of Tiling for Small Object Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019; pp. 582–591.
53. Deng, J.; Shi, Z.; Zhuo, C. Energy-Efficient Real-Time UAV Object Detection on Embedded Platforms. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2020**, *39*, 3123–3127. [[CrossRef](#)]

54. Ringwald, T.; Sommer, L.; Schumann, A.; Beyerer, J.; Stiefelwagen, R. UAV-Net: A Fast Aerial Vehicle Detector for Mobile Platforms. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019; pp. 544–552.
55. Zhang, P.; Zhong, Y.; Li, X. SlimYOLOv3: Narrower, faster and better for real-time UAV applications. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 27–28 October 2019; pp. 37–45.
56. Liu, Y.; Cao, S.; Lasang, P.; Shen, S. Modular Lightweight Network for Road Object Detection Using a Feature Fusion Approach. *IEEE Transaction on Systems, Man, and Cybernetics: Systems* **2019**, *51*, 4716–4728. [[CrossRef](#)]
57. Vaddi, S.; Kumar, C.; Jannesari, A. Efficient Object Detection Model for Real-Time UAV Applications. *arXiv* **2019**, arXiv:1906.00786.
58. Yang, Z.; Xu, W.; Wang, Z.; He, X.; Yang, F.; Yin, Z. Combining Yolov3-tiny Model with Dropblock for Tiny-face Detection. In Proceedings of the 2019 IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 16–19 October 2019; pp. 1673–1677.
59. Han, S.; Kwon, J.; Kwon, S. Real-time Small Object Detection Model in the Bird-view UAV Imagery. In Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, Vancouver, BC, Canada, 26–28 August 2019. Article 47.
60. Zhao, X.; Liang, X.; Zhao, C.; Tang, M.; Wang, J. Real-Time Multi-Scale Face Detector on Embedded Devices. *Sensors* **2019**, *19*, 2158. [[CrossRef](#)]
61. Yoo, Y.J.; Han, D.; Yun, S. EXT-D: Extremely tiny face detector via iterative filter reuse. *arXiv* **2019**, arXiv:1906.06579.
62. Qi, S.; Yang, J.; Song, X.; Jiang, C. Multi-Task FaceBoxes: A Lightweight Face Detector Based on Channel Attention and Context Information. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 4080–4097. [[CrossRef](#)]
63. Li, X.; Tian, Y.; Zhang, F.; Quan, S.; Xu, Y. Object detection in the context of mobile augmented reality. In Proceedings of the 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Porto de Galinhas, Brazil, 9–13 November 2020.
64. Zhao, Y.; Wang, L.; Hou, L.; Gan, C.; Huang, Z.; Hu, X.; Shen, H.; Ye, J. Real Time Object Detection for Traffic Based on Knowledge Distillation: 3rd Place Solution to Pair Competition. In Proceedings of the 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), London, UK, 6–10 July 2020; pp. 1–6.
65. Barba-Guaman, L.; Eugenio Naranjo, J.; Ortiz, A. Deep Learning Framework for Vehicle and Pedestrian Detection in Rural Roads on an Embedded GPU. *Electronics* **2020**, *9*, 589. [[CrossRef](#)]
66. Liu, M.; Ding, X.; Du, W. Continuous, Real-Time Object Detection on Mobile Devices without Offloading. In Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 976–986.
67. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.; Faughnan, T. Smart Surveillance as an Edge Network Service: From Harr-Cascade, SVM to a Lightweight CNN. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 256–265.
68. Nguyen, P.H.; Arsalan, M.; Koo, J.H.; Naqvi, R.A.; Truong, N.Q.; Park, K.R. LightDenseYOLO: A Fast and Accurate Marker Tracker for Autonomous UAV Landing by Visible Light Camera Sensor on Drone. *Sensors* **2018**, *18*, 1703. [[CrossRef](#)]
69. Kyrkou, C.; Plastiras, G.; Theocharides, T.; Venieris, S.I.; Bouganis, C. DroNet: Efficient convolutional neural network detector for real-time UAV applications. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 967–972.
70. Mithun, N.C.; Munir, S.; Guo, K.; Shelton, C. ODDS: Real-Time Object Detection Using Depth Sensors on Embedded GPUs. In Proceedings of the 2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Porto, Portugal, 11–13 April 2018; pp. 230–241.
71. Ghazi, P.; Happonen, A.P.; Boutellier, J.; Huttunen, H. Embedded Implementation of a Deep Learning Smile Detector. In Proceedings of the 2018 7th European Workshop on Visual Information Processing (EUVIP), Tampere, Finland, 26–28 November 2018.
72. Melinte, D.O.; Dumitriu, D.; Mărgăritescu, M.; Ancuța, P.-N. Deep Learning Computer Vision for Sorting and Size Determination of Municipal Waste. In Proceedings of the International Conference of Mechatronics and Cyber-MixMechatronics—2019, Cham, Switzerland; 2020; pp. 142–152.
73. Yang, A.; Bakhtari, N.; Langdon-Embry, L.; Redwood, E.; Grandjean Lapierre, S.; Rakotomanga, P.; Rafalimanantsoa, A.; De Dios Santos, J.; Vigan-Womas, I.; Knoblauch, A.M.; et al. Kankanet: An artificial neural network-based object detection smartphone application and mobile microscope as a point-of-care diagnostic aid for soil-transmitted helminthiasis. *PLoS Negl Trop Dis.* **2019**, *13*, e0007577. [[CrossRef](#)]
74. Pang, S.; Wang, S.; Rodriguez-Paton, A.; Li, P.; Wang, X.; Rodriguez-Patón, A.; Li, P.; Wang, X. An artificial intelligent diagnostic system on mobile Android terminals for cholelithiasis by lightweight convolutional neural network. *PLoS ONE* **2019**, *14*, e0221720. [[CrossRef](#)]
75. Lage, E.S.; Santos, R.L.; Junior, S.M.T.; Andreotti, F. Low-Cost IoT Surveillance System Using Hardware-Acceleration and Convolutional Neural Networks. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 931–936.
76. Bresilla, K.; Perulli, G.D.; Boini, A.; Morandi, B.; Corelli Grappadelli, L.; Manfrini, L. Single-Shot Convolution Neural Networks for Real-Time Fruit Detection Within the Tree. *Front. Plant Sci.* **2019**, *10*. [[CrossRef](#)] [[PubMed](#)]

77. Xiong, Q.; Lin, J.; Yue, W.; Liu, S.; Liu, Y.; Ding, C. A Deep Learning Approach to Driver Distraction Detection of Using Mobile Phone. In Proceedings of the 2019 IEEE Vehicle Power and Propulsion Conference (VPPC), Hanoi, Vietnam, 14–17 October 2019; pp. 1–5.
78. Shakeel, M.F.; Bajwa, N.A.; Anwaar, A.M.; Sohail, A.; Khan, A.; Haroon-ur-Rashid. Detecting Driver Drowsiness in Real Time Through Deep Learning Based Object Detection. In Proceedings of the International Work-Conference on Artificial Neural Networks, Gran Canaria, Spain, 12–14 June 2019; Springer: Cham, Switzerland, 2019; pp. 283–296.
79. Xiao, D.; Li, H.; Liu, C.; He, Q.; Alonso-Betanzos, A. Large-Truck Safety Warning System Based on Lightweight SSD Model. *Intell. Neurosci.* **2019**, *2019*, 10. [[CrossRef](#)]
80. Ramcharan, A.; McCloskey, P.; Baranowski, K.; Mbilinyi, N.; Mrisho, L.; Ndalaha, M.; Legg, J.; Hughes, D.P. A Mobile-Based Deep Learning Model for Cassava Disease Diagnosis. *Front. Plant Sci.* **2019**, *10*. [[CrossRef](#)]
81. Wang, F.; Tan, J.T.C. Improving Deep Learning Based Object Detection of Mobile Robot Vision by HSI Preprocessing Method and CycleGAN Method Under Inconsistent Illumination Conditions in Real Environment. In Proceedings of the 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Hong Kong, China, 8–12 July 2019; pp. 583–588.
82. Chen, P.; Hsieh, J.; Gochoo, M.; Wang, C.; Liao, H.M. Smaller Object Detection for Real-Time Embedded Traffic Flow Estimation Using Fish-Eye Cameras. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 2956–2960.
83. Zhao, H.; Zhang, W.; Sun, H.; Xue, B. Embedded Deep Learning for Ship Detection and Recognition. *Future Internet* **2019**, *11*, 53. [[CrossRef](#)]
84. Ding, S.; Long, F.; Fan, H.; Liu, L.; Wang, Y. A novel YOLOv3-tiny network for unmanned airship obstacle detection. In Proceedings of the Proceedings of 2019 IEEE 8th Data Driven Control and Learning Systems Conference, DDCLS 2019, Dali, China, 24–27 May 2019; pp. 277–281.
85. Putro, M.D.; Nguyen, D.L.; Jo, K.H. Fast Eye Detector Using CPU Based Lightweight Convolutional Neural Network. In Proceedings of the 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea, 13–16 October 2020; pp. 12–16.
86. Long, Z.; Suyuan, W.; Zhongma, C.; Jiaqi, F.; Xiaoting, Y.; Wei, D. Lira-YOLO: A lightweight model for ship detection in radar images. *J. Syst. Eng. Electron.* **2020**, *31*, 950–956. [[CrossRef](#)]
87. Zhou, Z.; Song, Z.; Fu, L.; Gao, F.; Li, R.; Cui, Y. Real-time kiwifruit detection in orchard using deep learning on Android™ smartphones for yield estimation. *Comput. Electron. Agric.* **2020**, *179*, 105856. [[CrossRef](#)]
88. Khaled, N.; Mohsen, S.; El-Din, K.E.; Akram, S.; Metawie, H.; Mohamed, A. In-Door Assistant Mobile Application Using CNN and TensorFlow. In Proceedings of the 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 12–13 June 2020; pp. 1–6.
89. Ji, H.; Zeng, X.; Li, H.; Ding, W.; Nie, X.; Zhang, Y.; Xiao, Z. Human abnormal behavior detection method based on T-TINY-YOLO. In Proceedings of the 5th International Conference on Multimedia and Image Processing, Nanjing, China, 10–12 January 2020; pp. 1–5.
90. Han, C.; Zhu, J.; Li, F.; Wan, S.; Chen, H. Design of lightweight pedestrian detection network in railway scenes. *J. Phys. Conf. Ser.* **2020**, *1544*, 012053. [[CrossRef](#)]
91. Pinto de Aguiar, A.S.; Neves dos Santos, F.B.; Feliz dos Santos, L.C.; de Jesus Filipe, V.M.; Miranda de Sousa, A.J. Vineyard trunk detection using deep learning—An experimental device benchmark. *Comput. Electron. Agric.* **2020**, *175*, 105535. [[CrossRef](#)]
92. Choi, J.; Chun, D.; Lee, H.; Kim, H. Uncertainty-based Object Detector for Autonomous Driving Embedded Platforms. In Proceedings of the 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genova, Italy, 31 August–2 September 2020; pp. 16–20.
93. Seo, J.; Ahn, H.; Kim, D.; Lee, S.; Chung, Y.; Park, D. EmbeddedPigDet—Fast and Accurate Pig Detection for Embedded Board Implementations. *Appl. Sci.* **2020**, *10*, 2878. [[CrossRef](#)]
94. Ai, Y.B.; Rui, T.; Yang, X.Q.; He, J.L.; Fu, L.; Li, J.B.; Lu, M. Visual SLAM in dynamic environments based on object detection. *Def. Technol.* **2020**. [[CrossRef](#)]
95. Guo, Q.; Liu, Q.; Wang, W.; Zhang, Y.; Kang, Q. A fast occluded passenger detector based on MetroNet and Tiny MetroNet. *Inf. Sci.* **2020**, *534*, 16–26. [[CrossRef](#)]
96. Gong, J.; Zhao, J.; Li, F.; Zhang, H. Vehicle detection in thermal images with an improved yolov3-tiny. In Proceedings of the 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 28–30 July 2020; pp. 253–256.
97. Anisuzzaman, D.M.; Patel, Y.; Niezgoda, J.; Gopalakrishnan, S.; Yu, Z. A Mobile App for Wound Localization using Deep Learning. *arXiv* **2020**, arXiv:2009.07133.
98. Sun, H.; Yang, J.; Shen, J.; Liang, D.; Ning-Zhong, L.; Zhou, H. TIB-Net: Drone Detection Network with Tiny Iterative Backbone. *IEEE Access* **2020**, *8*, 130697–130707. [[CrossRef](#)]
99. Jia, J.; Zhai, G.; Ren, P.; Zhang, J.; Gao, Z.; Min, X.; Yang, X. Tiny-BDN: An Efficient and Compact Barcode Detection Network. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 688–699. [[CrossRef](#)]
100. Melinte, D.O.; Travediu, A.-M.; Dumitriu, D.N. Deep Convolutional Neural Networks Object Detector for Real-Time Waste Identification. *Appl. Sci.* **2020**, *10*, 7301. [[CrossRef](#)]



101. Zhang, S.; Wu, Y.; Men, C.; Ren, N.; Li, X. Channel Compression Optimization Oriented Bus Passenger Object Detection. *Math. Probl. Eng.* **2020**, 2020. [[CrossRef](#)]
102. Wang, X.; Wang, S.; Cao, J.; Wang, Y. Data-Driven Based Tiny-YOLOv3 Method for Front Vehicle Detection Inducing SPP-Net. *IEEE Access* **2020**, *8*, 110227–110236. [[CrossRef](#)]
103. Mazzia, V.; Khaliq, A.; Salvetti, F.; Chiaberge, M. Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application. *IEEE Access* **2020**, *8*, 9102–9114. [[CrossRef](#)]
104. Mishra, B.; Garg, D.; Narang, P.; Mishra, V. Drone-surveillance for search and rescue in natural disaster. *Comput. Commun.* **2020**, *156*, 1–10. [[CrossRef](#)]
105. Oh, S.; You, J.-H.; Kim, Y.-K. FRDet: Balanced and Lightweight Object Detector based on Fire-Residual Modules for Embedded Processor of Autonomous Driving. *arXiv* **2020**, arXiv:2011.08061.
106. Etxeberria-Garcia, M.; Ezaguirre, F.; Plazaola, J.; Muñoz, U.; Zamalloa, M. Embedded object detection applying Deep Neural Networks in railway domain. In Proceedings of the 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia, 26–28 August 2020; pp. 565–569.
107. Liu, J.; Wang, X. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. *Plant Methods* **2020**, *16*, 83. [[CrossRef](#)]
108. Zhang, X.; Gao, Y.; Xiao, G.; Feng, B.; Chen, W. A Real-Time Garbage Truck Supervision and Data Statistics Method Based on Object Detection. *Wirel. Commun. Mob. Comput.* **2020**, 2020. [[CrossRef](#)]
109. Roy, B.; Nandy, S.; Ghosh, D.; Dutta, D.; Biswas, P.; Das, T. MOXA: A Deep Learning Based Unmanned Approach For Real-Time Monitoring of People Wearing Medical Masks. *Trans. Indian Natl. Acad. Eng.* **2020**, *5*, 509–518. [[CrossRef](#)]
110. von Atzigen, M.; Liebmann, F.; Hoch, A.; Bauer, D.E.; Snedeker, J.G.; Farshad, M.; Fünstahl, P. HoloYolo: A proof-of-concept study for marker-less surgical navigation of spinal rod implants with augmented reality and on-device machine learning. *Int. J. Med Robot. Comput. Assist. Surg.* **2021**, *17*, 1–10. [[CrossRef](#)] [[PubMed](#)]
111. Kazemi, F.M.; Samadi, S.; Poorreza, H.R.; Akbarzadeh-T, M.R. Vehicle recognition using curvelet transform and SVM. In Proceedings of the International Conference on Information Technology-New Generations, ITNG 2007, Las Vegas, NV, USA, 2–4 April 2007; pp. 516–521.
112. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
113. Wu, S.; Nagahashi, H. Parameterized adaboost: Introducing a parameter to speed up the training of real adaboost. *IEEE Signal Process. Lett.* **2014**, *21*, 687–691. [[CrossRef](#)]
114. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
115. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [[CrossRef](#)]
116. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [[CrossRef](#)]
117. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [[CrossRef](#)] [[PubMed](#)]
118. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; 9905 LNCS. pp. 21–37. [[CrossRef](#)]
119. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the Proceedings—30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
120. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
121. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
122. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
123. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [[CrossRef](#)]
124. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
125. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the NIPS 2017 The Thirty-first Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
126. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the European conference on computer vision, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014; pp. 740–755.
127. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [[CrossRef](#)]

128. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Hu, Q.; Ling, H. Vision Meets Drones: Past, Present and Future. *arXiv* **2020**, arXiv:2001.06303.
129. Yang, S.; Luo, P.; Loy, C.C.; Tang, X. WIDER FACE: A face detection benchmark. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 5525–5533.
130. Kukreja, V.; Kumar, D.; Kaur, A.; Geetanjali; Sakshi. GAN-based synthetic data augmentation for increased CNN performance in Vehicle Number Plate Recognition. In Proceedings of the 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 5–7 November 2020; pp. 1190–1195.
131. Bhattarai, B.; Baek, S.; Bodur, R.; Kim, T.K. Sampling Strategies for GAN Synthetic Data. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 2303–2307.
132. Qiu, W.; Yuille, A. UnrealCV: Connecting Computer Vision to Unreal Engine. In Proceedings of the Computer Vision—ECCV 2016 Workshops, Amsterdam, The Netherlands, 8–10, 15–16 October 2016; Springer: Cham, Switzerland, 2016; pp. 909–916.
133. Pérez-Rúa, J.M.; Zhu, X.; Hospedales, T.M.; Xiang, T. Incremental Few-Shot Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 13843–13852.
134. Fan, Q.; Zhuo, W.; Tang, C.K.; Tai, Y.W. Few-Shot Object Detection With Attention-RPN and Multi-Relation Detector. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 4012–4021.