Neural Computing and Applications (2022) 34:10469–10501
https://doi.org/10.1007/s00521-021-06830-w

S. I. : EFFECTIVE AND EFFICIENT DEEP LEARNING

Optimized convolutional neural network architectures for efficient on-device vision-based object detection

Ivan Rodriguez-Conde[1] · Celso Campos[2] · Florentino Fdez-Riverola[3,4]

Received: 31 March 2021 / Accepted: 4 December 2021 / Published online: 27 December 2021

Abstract
Convolutional neural networks have pushed forward image analysis research and computer vision over the last decade, constituting a state-of-the-art approach in object detection today. The design of increasingly deeper and wider architectures has made it possible to achieve unprecedented levels of detection accuracy, albeit at the cost of both a dramatic computational burden and a large memory footprint. In such a context, cloud systems have become a mainstream technological solution due to their tremendous scalability, providing researchers and practitioners with virtually unlimited resources. However, these resources are typically made available as remote services, requiring communication over the network to be accessed, thus compromising the speed of response, availability, and security of the implemented solution. In view of these limitations, the on-device paradigm has emerged as a recent yet widely explored alternative, pursuing more compact and efficient networks to ultimately enable the execution of the derived models directly on resource-constrained client devices. This study provides an up-to-date review of the more relevant scientific research carried out in this vein, circumscribed to the object detection problem. In particular, the paper contributes to the field with a comprehensive architectural overview of both the existing lightweight object detection frameworks targeted to mobile and embedded devices, and the underlying convolutional neural networks that make up their internal structure. More specifically, it addresses the main structural-level strategies used for conceiving the various components of a detection pipeline (i.e., backbone, neck, and head), as well as the most salient techniques proposed for adapting such structures and the resulting architectures to more austere deployment environments. Finally, the study concludes with a discussion of the specific challenges and next steps to be taken to move toward a more convenient accuracy–speed trade-off.

Keywords Convolutional neural networks · Object detection · On-device machine learning · Efficient architectures

1 Introduction

Despite being widely studied over the last three decades, object detection still represents a highly complex problem and remains an uphill challenge of great interest in research. Nowadays, classification and localization of specific targets or object instances on images and videos transcend computer vision and constitute both a central research topic within the scientific community, and a technical approach is increasingly explored and exploited by industry. It is possible to find a fair amount of related works in the existing literature on application domains as diverse as posture estimation [1], pedestrian [2], and face detection [3], or human behavior recognition and analysis [4], among others. Moreover, outside academia, the recent adoption of cutting-edge object detection methods and

✉ Florentino Fdez-Riverola
riverola@uvigo.es

1 Department of Computer Science, University of Arkansas at Little Rock, 2801 South University Avenue, Little Rock, AR 72204, USA

2 Department of Computer Science, ESEI – Escuela Superior de Ingeniería Informática, Universidade de Vigo, 32004 Ourense, Spain

3 CINBIO, Department of Computer Science, ESEI – Escuela Superior de Ingeniería Informática, Universidade de Vigo, 32004 Ourense, Spain

4 SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur), SERGAS-UVIGO, Vigo, Spain

techniques has enabled professionals in the corporate field to improve the efficiency and effectiveness of robots and cyber-physical systems in general, due to a more sophisticated perception of the environment. It has also had a very significant impact on a considerable number of practical use cases closer to the general public, and primarily associated with mobile applications and real-time support as well as monitoring systems such as advanced driving assistance systems (ADAS) [5] or drone surveillance systems [6].

Although first presented in 1989 by LeCun et al. [7], convolutional neural networks (CNNs) have emerged over the last decade as a major driver of progress in image analysis and computer vision, delivering state-of-the-art results in terms of accuracy. Though statistical classifiers, such as support vector machines (SVM) [8], Random Forest [9], Adaboost [10], or traditional neural networks, were considered the standard in computer vision for many years and had a leading role in object detection tasks, and the relatively recent breakthrough of deep learning (DL) techniques represents an unquestionable leap over previous object detection research, enabling not only the detection of objects in more complex situations but also the simplification of the design process of pursued algorithmic solutions. In this regard, there has been a clear paradigm transition from a handcrafted approach for the conception and design of detection techniques with a strong focus on feature engineering to a streamlined model based on fully automatic feature extraction. Thus today, CNNs represent a comprehensive detection solution that, due to their ability to exploit both spatial and temporal correlation of input data, enables feature representation learning to be carried out directly with no need of domain expertise, an essential requirement to design feature extraction algorithms such as shift invariant feature transform (SIFT) [11], histogram of oriented gradients (HOG) [12], or local binary patterns (LBP) [13], which are omnipresent among the more classical approaches.

The aforementioned qualitative leap forward brought by the advent of CNNs in computer vision [14] did not come without cost. As has been the case with essentially all DL techniques, convolution-based networks are also exceptionally computationally demanding and require a large memory footprint. The exploration of innovative vision approaches in general and the design of novel CNN architectures in particular, promoted by challenges such as the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [15] and Pascal VOC [16], have been primarily aimed at achieving better performance in visual recognition tasks, concentrating efforts on outperforming the latest state-of-the-art accuracy. This has meant a tremendous boost for the field and a historically unprecedented evolution of CNN. On the other hand, it has also led to increasingly complex architectures. CNN models such as VGG-16 (138M parameters) [17] or RetinaNet (built on ResNet-152 [18], with 60M parameters) [19] have been able to achieve high accuracy levels, but they typically rely on complex structures and work with multidimensional parameter spaces, resulting in a large amount of computed intermediate products and output values. The path toward more accurate CNNs has also involved, throughout much of the last decade, the design of progressively deeper architectures and hence an ever-greater number of feature maps, resulting not only in a dramatic increase in the number of parameters, as just noted, but also in the number of multiply-adds (MAdd) operations [17, 20].

However, recent hardware progress has shown adequate power to successfully meet the above-mentioned needs. Cloud-infrastructure-based configurations powered by graphics processing units (GPUs) clusters have become the standard in DL research. Conceived as highly scalable systems with access to potentially unlimited resources, they can accelerate both CNN inference and training on dedicated servers, fully or partially taking on the required computational load and thus relegating user-level devices to mere data-entry and result-presentation terminals [21, 22]. Nevertheless, this model presents certain limitations in terms of response speed, availability, and security [23–31], which is why it might be inadequate in scenarios where system response time must be as short as possible, in austere contexts with limited communication or computational resources, or even in cases where data privacy is a hard requirement. More specifically:

- Increase in latency, i.e., the response time of the system, does not harm detection accuracy, but it can lead to significant degradation of the experience [24]. Offloading processing tasks to a remote machine adds to inference time—that is, trained model execution time—the amount of time devoted to image encoding and transfer, consequently generating the perception among end-users of higher slowness in the whole process [23, 25, 28].
- Connectivity between a terminal device and the server-side is crucial. A decent number of today's applications and cyber-physical systems, the latter designed as distributed infrastructures and oriented to mobility contexts, are based on terminal devices that strongly rely on wireless communications. While providing ubiquity to detection solutions, this type of connection is less robust than wired alternatives [27]. Thus, depending on factors such as the reception signal strength, the environmental conditions, or the location of the device itself, among others, the connection can experience substantial speed fluctuations or, in the

worst case, connection drops, hindering and interrupting computations offloading on the cloud [30].
- Client–server data exchange inherently involves security and user data privacy concerns [26, 31], both in data transmission, which can be affected by illicit captures, and data persistence on the cloud infrastructure, a potential open gate to unauthorized access.

In recent years a novel DL trend named On-Device Machine Learning has emerged as an alternative to the cloud approach, constituting a potential solution to the different problems just pointed out. This trend can be framed, from a technological perspective, within the so-called edge computing paradigm, and encompasses a whole new set of techniques and technologies that enable DL solutions to be deployed on low-power, energy-constrained terminal devices. Although we include both next-generation mobile and embedded devices in the latter, for brevity, we will refer to them jointly as edge devices throughout the paper, following [32]. Inference is offloaded on the terminal device, alleviating the computational load on the server-side, the data traffic between endpoints and the associated latency, but also incorporating to terminal devices a layer of intelligence capable of providing users with a smoother and better-tailored experience, without compromising data integrity. This demand for higher performance in models deployed on edge devices is reflected in the need of more sophisticated hardware systems and more complex CNN models.

Deep neural network (DNN) acceleration hardware such as application-specific integrated circuits (ASICs) [33, 34], GPUs [35, 36], and field programmable gate arrays (FPGAs) [37–40] are already part of the technological landscape of mobile and embedded devices and have proven to considerably speed up mathematical computations in the latter while providing a good balance with respect to their intrinsic power consumption constraints. However, the performance achieved in most cases is still insufficient and far from the pursued real-time, revealing the need for an alternative approach focused on the software side. Previous studies carried out along these lines have focused on the possibility of bringing artificial intelligence closer to devices with limited resources. Those works explore a possible decrease in both the inference time and memory size of DNN models, not only through better exploitation of device hardware capabilities but also, and mainly, through the design of more compact and efficient models despite the limitations. With specific regard to this last point, there has been a very prolific scientific production during the last 5 years that has resulted in a great diversity of reported approaches. Those approaches can be classified into two distinct categories:

- *CNN compression* Mainstream practice focused on the optimization and size decrease in an existing network by removing potential redundancy in model parameters. This approach comprises widely known model compression methods such as data quantization [41, 42], network sparsification [43, 44], network pruning [45] and knowledge distillation [46].
- *Lightweight CNN design* Relatively more recent approach [47] that, as the name itself suggests, pursues the creation of new efficient architectures from the ground up [48–52], based on novel less costly convolution operation types including: (i) methods that operate at the filter channel level and can reduce the number of parameters, such as group convolutions [50, 53–55] and depth-wise separable convolutions [47, 48], and (ii) techniques that act on the spatial dimension of filters to improve parameter efficiency, such as low-rank filters.

Moreover, due to the remarkable ongoing scientific efforts, it is already possible to find a fair number of surveys in the computer literature aimed at introducing, analyzing, and comparing the more relevant related research, in order to shed light and put into perspective the extraordinary amount of recently published contributions. We identify studies published over the last 4 years (2017–2020) from different levels of abstraction, explore the use of machine learning techniques on terminal devices [32, 38, 56–58], and also publications explicitly circumscribed to the detection problem, that comprehensively review the most salient issues concerning the current state-of-the-art [59–65]. More specifically, in the first group, we can find summary-oriented works that provide an overview of recent progress in DNN acceleration focused on both (i) effective methods for network compression and optimization tasks and (ii) the different hardware solutions and software frameworks jointly conceived for that purpose. The second group brings together, as pointed out, publications that review the most significant recent DL-based efforts—more specifically, based on CNN—in object detection, introducing and analyzing the detection algorithms and frameworks that have emerged as milestones in the search for higher accuracy, and also covering topics such as (i) underlying architectures [61, 63] (ii) topology-specific innovations for increasing the representational capacity of CNNs [64], (iii) challenges still pending in the field [60, 63], (iv) methodologies and strategies best suited for training [60, 63] (v) common use cases or application domains [61], and (vi) evaluation metrics [60, 63].

Although the referenced works address a broad spectrum of relevant topics regarding both the search toward more efficient DL techniques and the design of more accurate detection solutions, there is still no work of this

nature in the literature providing a detailed overview of recent object-detection-oriented compact CNN architectures that, beyond simply pursuing higher accuracy, have been conceived from scratch within the on-device paradigm and aim to bring the detection process closer to resource-limited hardware platforms. This paper provides a comprehensive review of the foremost lightweight CNN architectures specifically designed for generating efficient models, directly deployable on edge devices and characterized by a proper speed-accuracy balance and contained energy consumption on inference time. More specifically, the main objective of this work is to provide the reader with a structured and detailed presentation of the main approaches and techniques developed in that line, as well as the primary features and more significant design decisions that have guided the conception of such lightweight architectures. The aim is to provide machine learning professionals interested in developing specific detection-based applications for edge devices, an overview to be used as a starting point or help guide; and to delve into the techniques and structural principles underlying the most popular current compact CNN architectures, in order to create a solid foundation, among researchers and in academics, necessary for a steady progress toward true real-time performance.

In particular, the review focuses on object detection frameworks that, conceived as convolutional neural networks (CNNs), have been designed from scratch according to the on-device paradigm. In the literature, it is possible to find alternative approaches such as Transformers [66] and Multi-layer Perceptron [67] that tackle the object detection problem as well and have proven to yield high performance in this regard. However, to the best of the authors' knowledge, such approaches are very recent, and they have not yet found their way into the on-device corpus. For this reason, the study remains limited to detectors built on CNN architectures. The on-device paradigm itself and its relatively short lifetime represent highly restrictive filtering considerations in the source discovery process and have shaped the keyword list used to find related articles and the inclusion/exclusion criteria adopted for determining the eligibility or degree of interest of such works. As far as the keywords are concerned, they were carefully selected according to the research objectives, seeking to obtain eminently relevant papers while ensuring none of them were left out of the search results. Specifically, for the various queries made, the term "object detection" was used together with the following keywords: "on-device machine learning", "embedded machine learning", "on-device intelligence", "on-device AI", "TinyML", "resource-constrained machine learning", "edge AI", "mobile machine learning", "embedded AI", "compact neural network", "portable neural network", "energy-efficient deep learning". The list of results initially obtained was further refined, excluding works based on the application of compression techniques to existing models rather than on the design of compact and efficient architectures deployable on low-resource devices, and also omitting contributions outside the scope of interest either because the approach proposed fell outside the on-device paradigm (lack of experimentation or results reported in that direction), or because of the nature of the data handled (3D point clouds [68, 69], and RGB-D images [70, 71]). Lastly, after an in-depth reading, 37 papers were selected and finally used as the core of this study.

The rest of the paper is structured as follows. Section 2 provides context to the study carried out, briefly presenting some of the most relevant milestones in the recent evolution experienced by CNN-based object detectors, characterizing the different components that integrate the underlying architecture of those systems and thus establishing the theoretical foundations necessary for a better understanding of the rest of the document. Section 3 provides a comprehensive analysis of the architecture of the different lightweight detection frameworks present in the literature, with special emphasis on the review of ultra-compact CNN networks due to their particular relevance as the backbone of detectors. Finally, Sect. 4 summarizes the observations drawn from state-of-the-art and identifies research challenges to be addressed in future work.

2 Toward efficient CNN-based object detection

The design of more efficient and effective detection frameworks has become one of the main objectives pursued in object detection over the last 5 years. Computational cost reduction in traditional detectors and accuracy preservation have guided the search for solutions carried out in recent years by the community of computer vision experts, scientists, and other professionals. This has led to the development of several techniques and methods based on CNN—for instance, Single Shot MultiBox Detector (SSD) [72], You Only Look Once (YOLO) [73–77], Faster R-CNN [78], Deeply Supervised Object Detectors (DSOD) [79], RetinaNet, RefineDet [80], or CornerNet [81]—which have shown promising performance in image-based target localization and classification tasks.

In this regard, two-stage detectors have maintained a leading role in the object detection application domain almost since R-CNN was first presented in 2014 [82], in large part due to the remarkable results provided by frameworks such as Faster R-CNN. More specifically, the integration of a Region Proposal Network (RPN) [78], responsible for generating Region of Interest (RoI)

proposals in the inner configuration of detectors, has made it possible to reach a very high level of accuracy. However, despite multiple efforts to increase detection speed, such as Light-Head R-CNN [83], by simplifying or reducing RoI computation and thus lightening the detection head, region proposal generation has been an insurmountable bottleneck for obtaining less computationally expensive models. For this reason, many authors interested in the search for more efficient solutions have turned their work toward the exploration of unified detection strategies instead of intensifying efforts and further deepening the optimization of the different components that assemble region-based pipelines.

This unified or single-stage approach gives its name to a set of techniques that model object detection as a simple regression problem, bringing together, in a single step, the prediction of both the areas of potential interest in the form of bounding boxes (localization) and the class names of the different searched objects (classification). Adopting such a simpler and more efficient localization mechanism has led to a transition to a more structurally reduced overall pipeline configuration, resulting in a less bulky architecture, conceived as a single feedforward neural network capable of clearly surpassing the inference speed offered by two-stage detectors, albeit at the cost of a substantial accuracy reduction. Despite this negative impact, architectures such as SSD [72] and the original version of YOLO [84], both leading exponents of this paradigm, represent a recent milestone of particular relevance as far as macroarchitectural design and detection pipeline simplification is concerned, and have been enormously successful at conquering much of the space of interest occupied by two-stage frameworks until relatively recently.

The one-stage pipeline model has undoubtedly been a major improvement in terms of efficiency, successfully alleviating the complexity of previous state-of-the-art DL-based detection alternatives. However, the structural optimization carried out for this purpose has been relatively conservative, and special care has been taken to avoid harming the accuracy of the resulting frameworks. Thus, despite effectively reducing the latency associated with the detection process, the unified pipeline represents an evolution of detectors still unsatisfactory for deploying this type of system in low-power target devices due to the significant computational complexity and large size of the derived model. As shown by the analysis carried out in Sect. 3, this approach constitutes the primary reference or base on which modern lightweight object detection architectures are built.

To that end, it is necessary to strengthen efforts to move toward a better speed-accuracy trade-off, exploring and developing techniques able to soften the aforementioned negative effect on the accuracy values derived from the use of compact architectures, as well as approaches that make progress in conceiving more expressive and thereby capable detection-oriented networks. Furthermore, it will be just as relevant as the nature or focus of the required modifications to comprehensively approach the structural optimization process, bearing in mind both the different specificities of the object detection problem and the singularities of the three components that comprise the detection frameworks.

Requirements of techniques designed ad hoc for domain-specific vision applications, including object detection, have traditionally received a treatment that could be qualified as auxiliary, subject to the work and progress made on general-purpose approaches. Although leveraging and operating on CNN models have made it possible to surpass the traditional performance of vision-based systems, due to their sophisticated ability to learn rich representations from image data, those models are typically algorithmic solutions based on vision-generic approaches mainly aimed at improving accuracy and speed in classification tasks. Consequently, they might show significant mismatches with respect to the needs derived from the object detection process, in some cases even leading to conflicts between the design principles underlying the object detectors and those characterizing the more generic classification-oriented networks.

Object detection relies particularly on the standard CNN-based approach [82], built on the design principles introduced by the seminal work of Lecun et al. in 1998 [85], such as the gradual decrease in the spatial dimensions in feature maps as the network deepens, or the generation of high-level features due to the feedforward communication of high-resolution convolutions located in shallow layers of the network, with low-resolution convolutions embedded in deeper areas. In turn, detection extends object classification with several localization-specific challenges such as scale variation or small object detection. Those challenges, in general terms, require a higher degree of expressiveness from the CNNs that define the inner structure of detectors almost entirely. Moreover, regarding features, greater expressiveness also implies a demand for multilevel feature processing, necessary for accurate visual recognition, or the exploitation of high-level features with better semantic quality to enrich low-level features.

Beyond an optimal trade-off able to successfully satisfy the different needs derived from the duality of the object detection problem, it will be equally necessary to explore the structural and operational particularities of the three elements that compose the architecture of the detection framework: (i) the backbone, also called “base network” in SSD [72], responsible for the extraction of semantic features from the images supplied as input to the detector; (ii) the neck, introduced in the architecture as an intermediate

element intended for the refinement of the properties extracted in the backbone, mainly through multilevel properties fusion; and lastly, (iii) the head, responsible for class prediction the bounding box regression. Figure 1 provides a schematic representation of the standard architecture of modern unified detection systems, depicting how the three components are arranged together and indicating which role they play in the detection pipeline. In addition, for each of the components under consideration, details are given on which properties of the resulting detector are impacted the most by their configuration as well as which aspects of the latter are more relevant.

As its name suggests, the backbone is the most relevant element in the architecture of an object detection system, mainly because of its predominance within the overall structure of the system but also because of the impact that some of its aspects or characteristics have on the overall performance of the detector. The backbone's ability to extract representative properties and its expressiveness contribute significantly to the general framework's accuracy, while salient topology-specific features, such as network depth or layer size in both the spatial and channel dimensions, bound or set the requirements for the resulting detection system in terms of computational cost and memory space. Moreover, the backbone's appeal as an individually capable CNN network, able to perform classification tasks on its own, transcends object detection domain, and constitutes a valuable approach for a fair number of vision-based problems such as instance segmentation or object tracking, among others. That is why, traditionally, and still today, computer vision researchers have commonly directed their efforts to the exploration and design of new CNN architectural alternatives regardless of the specific problem addressed. While the majority of these alternative approaches have been conceived as generic solutions, they are also used, nearly straightforwardly, as techniques and methods to devise and improve detection systems' backbone.

With regard to the architecture, the neck is also a CNN designed to expand or refine the features initially extracted by the backbone in order to mitigate the mismatch or gap in terms of power representation between the features generated by the latter and those required to obtain an adequate level of accuracy in object detection. Recent approaches address this problem and explore the integration in detection frameworks of new building blocks or multi-scale subnetworks, initially conceived in diverse vision-related application domains such as human pose estimation, face recognition, or instance segmentation, to improve the network's spatial awareness. More specifically, those new structural components are assembled into the neck to obtain higher resolution and semantically richer representations, not only to enable multi-scale object detection but also to improve the detection—primarily localization—of such objects in complex situations by leveraging the use of feature maps with different scales and the fusion of high and low-level features. Existing literature includes a considerable number of works [72, 79, 86–88] that address the problem described and propose different methods of joint exploitation of multiple CNN layers to improve detection accuracy. Among them, SSD [72] and feature pyramid network (FPN) [86] probably stand out as the two most paradigmatic approaches.

The very nature of CNNs and the gradual subsampling performed across their layers as the network depth increases produce a pyramid-shaped multiscale feature map structure in which higher layers have both larger receptive field size and higher semantic richness, while

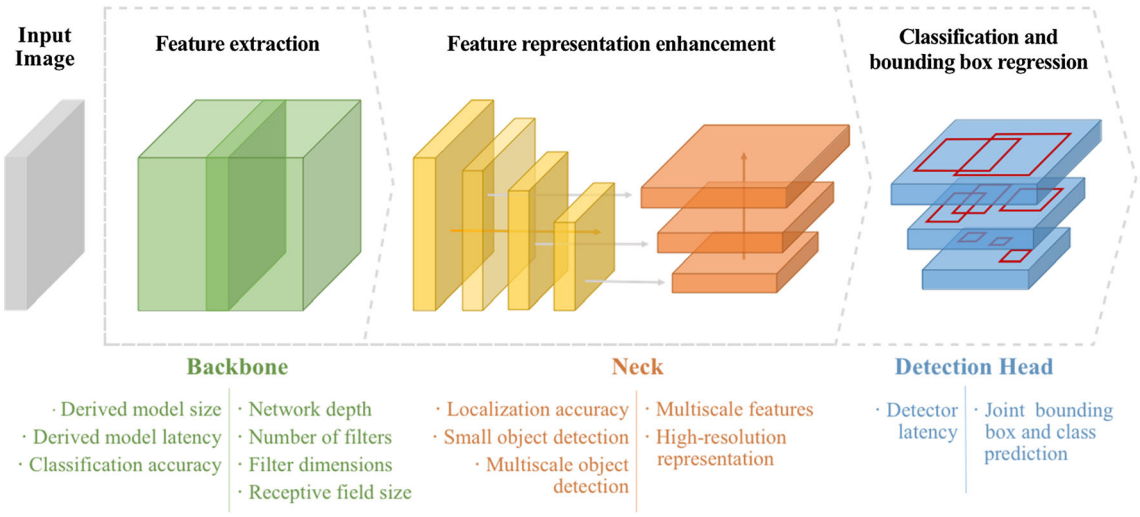

Fig. 1 Standard architecture of modern one-stage object detection frameworks

lower layers have a smaller receptive field size and, therefore, higher resolution but lower semantic sensitivity. SSD extends the base CNN network acting as backbone within the detector's architecture by appending a stack of additional convolution layers of different scales, where each layer is devoted to target-related predictions for a particular scale, thus enabling multiscale detection. It has shown relatively poor performance, however, in small object detection. High-level feature maps with a large receptive field are used to predict large targets; in contrast, low-level feature maps have a small receptive field and are able to predict small objects. The absence of a feature fusion mechanism makes it impossible to take advantage of the complementary nature of the features coming from different layers, which decreases the semantic richness necessary for small target detection.

FPN [86] is able to integrate semantic information in multi-scale feature maps due to an architecture composed of a bottom-up and a top-down path. In particular, the top-down pathway is responsible for building higher resolution feature maps from an initial semantically rich map. Newly generated feature maps, though semantically strong, are not appropriate for an accurate localization due to the negative impact of both the upsampling and downsampling operations used in the process. For this reason, as a mechanism to help with or improve localization, lateral connections are then added between the reconstructed feature maps and the corresponding original ones. However, although FPN has proven to be an effective and straightforward option, layer-by-layer feature fusion is not an optimal strategy and penalizes network efficiency in situations where the number of feature maps to be fused is very high. Fortunately, some design alternatives have emerged, intending to mitigate that deficiency by pursuing more efficient fusion strategies. An example of those alternative approaches is FSSD [89], which replaces FPN's layer-by-layer fusion with a single aggregation of the different multiscale feature maps available initially based on a less costly concatenation operation.

Finalizing the typical workflow in detection systems, the features extracted by the backbone network and then refined by the neck are passed as input to a series of prediction layers in charge of classification and bounding box regression tasks, both necessary for generating the output of the detection process. Whereas components involved in feature extraction have experienced a rapid evolution, resulting from significant research efforts that have traditionally led to more accurate convolutional architectures and, more recently, to more compact and efficient networks, quite the opposite has happened with the detection head, which is still today relegated to a minor role in the spotlight. That does not mean, however, that there has been no progress. The transition to a more compact architectural style, driven by the advent of single-stage detection models, has also been reflected in the organization and nature of the prediction layers attached to the end of the detector. Specifically, the head has been gradually slimmed down, transitioning from a traditional structure [78] consisting of two sibling branches, each with expensive fully connected layers, to a lighter architecture consisting entirely of convolutional layers capable of jointly performing class estimation and bounding box regression.

The enhancement of a single part within the detection system is typically insufficient to boost the joint potential of the three component assembly. A single-component-focused search for new strategies or specific improvements may lead to a result not necessarily appropriate [90]. For this reason, an ill-fitted joint configuration of the backbone and the detection head, for example, or even the typical detector creation approach based on CNN designed for classification tasks, may not be optimal for object detection. Aware of this, authors have commonly approached the design process of new lightweight detection frameworks comprehensively, exploring different approaches at the structural and operational level in order to improve performance and reduce the complexity of some, if not all, of the architecture components, in compliance with the well-known memory and computational limitations imposed by edge hardware platforms. Pursued solutions go beyond the sole adoption of a one-stage-detector macroarchitecture as a starting point or the integration of a shallower and simplified network model as the backbone in the detection pipeline. They are built instead upon an in-depth analysis exercise at both the micro and macroarchitectural levels, diversifying efforts to tweak or fine-tune the components in the detection system throughout a continuous process of search for balance between latency and accuracy.

3 Object detectors for resource-constrained devices

This section provides a detailed review of the main milestones or most representative approaches developed in recent years to bring the process of object localization and classification to devices with limited computational and memory resources. We will start this review with a holistic view of the main detection frameworks collected from the related literature. In total, Table 1 lists thirty detectors conceived as CNNs with small size (number of parameters) and modest computational complexity (computational volume). Specifically, for each detection framework analyzed, we will examine in detail the different techniques or methods adopted at the architectural level for their construction, not only pointing out which main building blocks

Table 1 Summary of most relevant architectural aspects of CNN-based lightweight frameworks for object detection

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
MobileNet+ SSD [47]	2017	SSD	MobileNet	–	FPN	–	SSD head	–	Accuracy Computational cost Model size	–	–
SqueezeDet [118]	2017	YOLO	SqueezeNet	–	x	x	YOLO head	FC layers replaced with conv layers	Accuracy Computational cost Model size Speed	Desktop GPU	x
MobileNetV2 SSDLite [48]	2018	SSDLite	MobileNetV2	–	SSD neck	Standard conv replaced with depth-wise separable conv	SSD head	Conv replaced with depth-wise separable conv	Accuracy Computational cost Model size Speed	Mobile single-core CPU	x
Pelee [107]	2018	SSD	PeleeNet	–	SSD neck	Largest feature map discarded One residual block per feature map Smaller kernel size for prediction	SSD head	–	Accuracy Computational cost Model size Speed	Mobile single-core CPU Embedded GPU	–
Tiny-DSOD [95]	2018	DSOD	DSOD backbone	Standard conv replaced with depth-wise conv # channels linearly increased with network depth	FPN	Depth-wise conv used for downsampling and reverse upsampling paths	SSD head	–	Accuracy Computational cost Model size Speed	Desktop dedicated GPU	✔
Fire SSD [108]	2018	SSD	SqueezeNet	–	SSD neck	Conv layers replaced with improved Fire modules Expand 3x3 and 1x1 conv layer in Fire module replaced with group conv layer Shortcut connections added to connect upper and lower Fire modules	SSD head	–	Accuracy Computational cost Model size Speed	Desktop x64 CPU Desktop integrated GPU USB VPU accelerator	✔

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
Tiny SSD [96]	2018	SSD	SqueezeNet	Nonuniform # filters, optimized for each Fire module	SSD neck	Nonuniform # filters, optimized for each Fire module	SSD head	–	Accuracy Computational cost Model size	–	–
ShuffleDet [97]	2019	SSD	ShuffleNet [54]	–	SSD neck	Standard conv is replaced with Reduction-B module [119] 1x1 conv is added to Reduction-B module 3x3 standard conv replaced with 3x3 depth-wise conv	SSD head	Residual block added, combination of 1x1 conv and 3x3 deformable conv	Accuracy Computational cost Model size Speed	Embedded GPU	✔
Improved YOLO v3-Tiny [109]	2019	YOLOv3-Tiny	Darknet-19 [120]	–	YOLOv3-Tiny neck	Third output with new scale added, resulting from feature fusion of a deep and a shallow map	YOLOv3-Tiny head	–	Accuracy Speed	Desktop x64 CPU Desktop dedicated GPU	✔
Mini-YOLOv3 [98]	2019	YOLOv3	Darknet-53 [77]	Standard conv replaced with depth-wise separable conv 1x1 conv replaced with 1x1 pointwise group conv Channel shuffle following group conv Shortcut connection used for residual structure	FPN	Encoder–decoder (ED) module for multiscale feature generation Features adaptatively aggregated using Squeeze-and-Excitation (SE) attention [121]	YOLOv3 head	Less # layers	Accuracy Computational cost Speed	–	✔
Pvalite CLN [110]	2019	Faster R-CNN	PVANET [122]	# shallower conv layers dramatically decreased Max pooling layer removed from early stage	Hypernet's hyper feature extraction network [87] + classification and localization (CLN) layer	–	Faster R-CNN head	–	Accuracy Computational cost Model size Speed	Embedded ARM-based chipset Embedded GPU	x

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
NAS-FPNLite MobileNetV2 [111]	2019	RetinaNet	MobileNetV2	–	NAS-FPN	Standard conv replaced with depth-wise separable conv	RetinaNet head	–	Accuracy Computational cost Model size Speed	Mobile single-core CPU	✔
ThunderNet [91]	2019	Light-Head R-CNN	ShuffleNet V2	3x3 depth-wise conv replaced with 5x5 depth-wise conv # channels increased in early stages	RPN	3x3 standard conv replaced with a 5x5 depth-wise separable conv Context enhancement module (CEM) that merges feature maps with three different scales, based on 2 1x1 conv and a FC layer Spatial attention module	Light-Head R-CNN head	# channels in the FC layer reduced by 50%	Accuracy Computational cost Model size Speed	Mobile x64 ARM CPU Desktop x64 CPU Dedicated desktop GPU	✔
MobileNetV3 + SSDLite [112]	2019	SSDLite	MobileNetV3	3 expensive layers dropped at the end of the network # filters in first conv layer halved Nonlinearity ReLU replaced with hard swish	SSDLite neck	–	SSDLite head	–	Accuracy Computational cost Model size Speed	–	✔
Mobile-YOLO [113]	2019	YOLOv3	MobileNetV2	–	FPN	–	YOLOv3 head	–	Accuracy Computational cost Model size Speed	Embedded ARM x64 CPU Dedicated desktop GPU	x

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
CornerNet-Squeeze [93]	2020	CornerNet	Hourglass [88]	Higher # of shallower and lighter modules Residual blocks replaced with Fire modules Second 3x3 conv layer replaced with 3x3 depth-wise separable convolution Downsampling layer added before hourglass modules and removed from the module	Custom neck	Attention mechanism for both locations and coarse object scales generation Attention maps prediction based on standard conv (3x3 conv followed by 1x1 conv)	CornerNet head	3x3 filters replaced with 1x1 filters	Accuracy Computational cost Model size Speed	–	✔
MobileNet-SSDv2 [99]	2020	SSD	MobileNetV2	4 inverted residual modules added to extract feature maps at 4 different scales	FPN	Inverse residual module added	SSD head	–	Accuracy Model size Speed	Embedded GPU	✔
FRDet [100]	2020	YOLOv3	–	Fire modules enhanced with residual shortcut connections Optimum squeeze ratio defined as 2^4	FPN	–	YOLOv3 head	–	Accuracy Computational cost Model size Speed	Embedded GPU	✔
Mixed YOLOv3-LITE [114]	2020	YOLO-LITE	Darknet-19	Residual blocks with shortcut connections added	HRNet	–	YOLOv3 head	–	Accuracy Computational cost Model size Speed	Embedded GPU	✔

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
MAOD [92]	2020	FCOS	MobileNetV3	Depth-wise dilated conv introduced in the linear inverted residual bottleneck Light spatial-channel (LSC) attention module that focuses on both spatial and channel info [123]	FPN	Depth-wise and 1x1 conv added to produce feature levels Hard-swish non-linear activation after each conv LSC attention module inserted Downsampling based on the huge stride value (128) is removed	FCOS head	Linear inverted residual bottleneck applied in each module Extra confidence sub-branch added to classification branch	Accuracy Computational cost Model size Speed	Desktop x64 CPU Dedicated desktop GPU	✔
RefineDetLite [101]	2020	RefineDet	Res2Net [124]	Bottleneck layer included at the beginning of each stage for downsampling, except for the first and the last one Same # input and output channels for Res2Net blocks	–	–	RefineDet head	3x3 conv layers replaced with a bottleneck structure and a 1x1 conv layer	Accuracy Speed	Desktop x64 CPU	✔
BMNet [102]	2020	SSD	PeleeNet	Use of Tiny-DSOD's stem block: 3x3 standard conv replaced with 3x3 depth-wise conv Standard 3x3 conv replaced with 3x3 depth-wise separable in the main building block (2WDB)	D-FPN	Attention module (attention prediction block) that focuses on both spatial and channel info [123]	–	–	Accuracy Computational cost Model size	–	–

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
LightDet [94]	2020	–	ShuffleNet V2	Novel stem block It starts with a 1x1 conv Multi-branch structure: a branch with standard conv, the rest with two dilated conv per branch Conv with larger dilation in each branch is modeled as a residual block The outputs of the branches are merged by concatenation	FPN	For lateral inputs, 1x1 conv is replaced with a residual block (#1): 1x1 in one of the paths for # channel reduction and 3x3 depth-wise conv in the other one to incorporate more context features Two paths in #1 merged by concatenation For top-down feature fusion, 1x1 conv is replaced with a residual block (#2): 3x3 depth-wise conv in one of the paths	Custom head	Input evenly split by channel A sequence of 1x3 and 3x1 conv applied to each branch Two parts merged by concatenation Channel shuffle applied 1x1 conv layer used for classification and localization	Accuracy Computational cost Speed	Dedicated desktop GPU	✔
Tinier-YOLO [103]	2020	YOLOv3-Tiny	Darknet-19	Integration of five Fire modules in the middle of the network Dense connections between Fire modules added Total network depth increased	YOLOv3-Tiny neck	Passthrough layer to merge previous feature maps before the first detection layer Fire modules added to process the aggregated feature maps Dense connections between Fire modules added	YOLOv3-Tiny head	–	Accuracy Computational cost Model size Speed	Embedded GPU	✔
FFBNet [115]	2020	SSD	MobileNet	–	FSSD	Feature pyramid reconstructed using dense connections, given a set of strong feature maps generated by an FSSD-like block	SSD head	–	Accuracy Model size Speed	Dedicated desktop GPU	✔

Table 1 (continued)

Detector	Year	Base detection framework	Backbone		Neck		Head		Evaluation		Real-time
			Base network	Adjustments	Base network	Adjustments	Base network	Adjustments	Metrics	Test hardware	
YOLOv4-tiny [106]	2020	YOLOv4	VoVNet	Residual structure in one-shot aggregation (OSA) blocks replaced with a Cross Stage Partial (CSP) structure [125]	YOLOv3-Tiny neck	Residual structure replaced with CSP structure	YOLOv3 head	–	Accuracy Computational cost Speed	Embedded GPU	✔
L-Net [104]	2021	–	ShuffleNet V2	3x3 depth-wise conv replaced with 5x5 depth-wise conv Input layer size increased	FPN	Channel attention module (attention pyramid module)	YOLOv3 head	–	Accuracy Computational cost Speed	Dedicated desktop GPU	✔
Lightweight SSD [117]	2021	SSD	MobileNetV2	Spatial attention module introduced in the linear inverted residual bottleneck Application of 3x3 and 5x5 kernels to depth-wise separable conv	FPN	Intermediate layers circularly used to produce the fused output information	SSD head	–	Accuracy Model size	Dedicated desktop GPU	✔
Micro-YOLO [116]	2021	YOLOv3-Tiny	Darknet-19	Standard conv replaced with depth-wise separable conv Standard conv replaced with linear inverted residual bottleneck with SE block [112]	YOLOv3-Tiny neck	–	YOLOv3-Tiny head	–	Accuracy Computational cost Model size Speed	Dedicated desktop GPU	✔
RSANet [105]	2021	–	LCNet	–	FPN	Channel attention module (residual semantic-guided attention mechanism)	–	–	Accuracy Computational cost Model size Speed	–	✔

or network architectures are chosen as the main constituent elements for each of the different stages or components of its structure, but also comprehensively analyzing which specific topological adjustments or improvements were applied to each of the parts to achieve the desired efficiency-accuracy trade-off.

Given the relevance of the backbone as the detection system's key structural element, we will spend a large part of the section studying the main architectural approaches and design principles that have led to the development of lightweight yet expressive CNNs appropriate for the extraction of features with the quality required to effectively perform bounding box regression and class prediction, both of which are tasks involved in the detection process. However, this review will not be limited to the backbone networks listed in Table 1. Supported by a second table (Table 2), we will extend the discussion of those networks beyond the items presented in the first table, analyzing the most relevant general-purpose CNN architectures designed from scratch for mobile or embedded devices. Although, as we will see, most of them have not been used so far as part of any detection framework, they all represent perfectly valid approaches for this purpose. Moreover, due to their convolutional nature, they are based on structures and topological principles similar to those comprising the foundation of detectors, so their incorporation into the analysis will complement the global discussion, providing further relevant information, both at the micro and macroarchitectural level.

3.1 Lightweight object detection frameworks

The data collected in Table 1 provide context to the current mobile scenario, chronologically locating recent research efforts focused on studying and creating lightweight object detectors in the last 5 years. In a first superficial inspection of the works analyzed in the table, focusing exclusively on the first three fields that provide more general data, it is possible to identify certain aspects of interest that outline the evolution of this new trend in the last few years. Specifically, the increasing number of related papers published (from only two in 2017, to fifteen in the last year and a half) clearly highlights a significant growing interest in the application of this new on-device paradigm to object detection. Those numbers further confirm the massive adoption of a single-stage pipeline configuration as the predominant architectural model, with ThunderNet [91] being the only two-stage detection framework of all the lightweight detectors and base detection frameworks listed.

Maintaining the same level of abstraction, but extending the analysis on Table 1 to the columns that contribute with specific data regarding each of the components that comprise the architecture of the different detection systems considered, we see that there is a marginal number of papers, namely MAOD [92], CornerNet-Squeeze [93], and LightDet [94], that explore the joint application of adjustments on backbone, neck, and head. The remaining majority is evenly split between work that explores enhancements on two of the elements that form the detection system in its different permutations [91, 95–106], and approaches that choose to focus on just one component [48, 107–117]. The main object of interest in the latter case is the neck, and, to a lesser extent, the backbone. If we delve deeper into this classification and extract the number of studies per individual component examined, it is possible to establish a ranking or prioritization of the three based on the level of attention they received in the different studies considered. The resulting list, in decreasing order of interest, is as follows: neck > backbone > head. Therefore, it is clear both that the emphasis on the development of specific approaches is aimed at improving the neck and the relative absence of actions focused on the detection head, whose structure, in general terms, is directly defined by the detection framework used as base macroarchitecture. The remainder of the section will include the main contributions made in relation to the three components in the last few years.

3.1.1 Neck-specific design considerations

We will now increase the level of detail of the analysis to focus the discussion on Table 1, dealing with specific architectural aspects of the different networks used as neck within the several ultra-compact detectors studied.

3.1.1.1 Classification according to the multiscale-detection-enabler mechanism used We start the discussion with the *Base Network* field, which contains the most relevant CNN microarchitectures adopted as base structure for designing the final actual neck architecture. Setting aside the RPN intended for the synthesis of RoI within two-step detection frameworks and not for the enhancement of the representational power of the features involved, it is possible to group those microarchitectures into three differentiated categories or approach types according to the type of multiscale-detection-enabler mechanism used: (i) the exploitation of a pyramidal feature hierarchy, (ii) the recovery of high-resolution representations from low-resolution representations, and (iii) the maintenance of high-resolution representations throughout the entire network. This classification does not include HyperNet [87], more specifically, the Hyper Features extraction network used as neck. Although it relies on the fusion of different feature maps, thus being excluded from (i), the performed feature aggregation does not involve the generation of higher resolution representations, making it also non-

Table 2 Efficient CNN architectures for mobile and embedded vision applications

Network	Year	Base network	Novel building blocks	Adjustments	Emphasis	Architectural scope	Evaluation metrics	Test hardware	Real-time
SqueezeNet [49]	2016	AlexNet [14]	Fire module	3x3 conv replaced with 1x1 conv # input channels decreased for 3x3 conv Layers with stride > 1 toward the end of the network Last fully-connected layers replaced with conv layers	Smaller model size	Macroarchitecture Microarchitecture	Accuracy Model size	–	–
MobileNet [47]	2017	–	–	3x3 conv replaced with 3x3 depth-wise conv	Less computational complexity Smaller model size	Microarchitecture	Accuracy Computational cost Model size	–	–
ShuffleNet [54]	2017	ResNet	ShuffleNet unit	1x1 conv replaced with 1x1 group conv 3x3 conv replaced with 3x3 depth-wise conv Channel shuffle following group conv	Less computational complexity	Microarchitecture	Accuracy Computational cost Speed	Mobile single-core CPU	x
CondenseNet [55]	2017	DenseNet [137]	–	Redundant connections removed in favor of learned group convolutions Exponentially increasing growth rate Input layers connected to all subsequent layers whether they are within the same block or not	Smaller model size	Macroarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU	x
ShiftNet [128]	2018	ResNet	Conv-Shift-Conv (CSC) module	Spatial conv replaced with shift operations Shift operations used together with point-wise conv for spatial information mixing across channels	Smaller model size	Microarchitecture	Accuracy Computational cost Model size	–	–
MobileNetV2 [48]	2018	MobileNet	Inverted residual with linear bottleneck	ReLU non-linearity replaced with linear bottleneck layer Shortcut connections directly inserted between bottlenecks Conv replaced with depth-wise conv in Expansion layer	Less computational complexity	Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU	x
ShuffleNet V2 [52]	2018	ShuffleNet	ShuffleNet v2 block	1x1 group convolutions replaced with novel channel split operation Conv with the same # input channels and # output channels Successive element-wise operations (concat, channel shuffle, channel split) merged into a single operation	Less memory access cost	Macroarchitecture Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU Dedicated desktop GPU	✔

Table 2 (continued)

Network	Year	Base network	Novel building blocks	Adjustments	Emphasis	Architectural scope	Evaluation metrics	Test hardware	Real-time
SqueezeNext [136]	2018	SqueezeNet	SqueezeNext block	Depth-wise separable conv avoided Squeeze layer replaced with a more aggressive two-stage squeeze module 3x3 conv replaced with 3x1 and 1x3 separable convs Bottleneck layer added right before the last fully-connected layer	Smaller model size	Macroarchitecture Microarchitecture	Accuracy Computational cost Model size Speed Energy consumption	Software emulator	–
PeleeNet [107]	2018	DenseNet	Two-way dense layer	Regular conv used instead of depth-wise separable conv Two-way dense layer with different # filters to get receptive fields with different scales Two-way stem block for increased # channels or increased growth rate Dynamic # channels in bottleneck based on the input shape Transition layers with the same # input channels and # output channels	More efficient architecture based on well-known efficiently-implemented operations	Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU Embedded GPU	✔
clcNet [130]	2018	MobileNet	Channel local convolution (CLC) block	CLC group-like conv where and output channel depends on an arbitrary subset of input channels Depth-wise separable conv replaced with CLC-based blocks # input channels should be the same as the size of the subset of input channels that the output channel depends on	Less computational complexity	Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU	x
AddressNet [50]	2019	ShuffleNet	Address-based module	Channel shift used for info exchange among channel groups Address shift used to fuse spatial information Shortcut shift used for memory optimization of channel concatenation	Less computational complexity	Microarchitecture	Accuracy Computational cost Model size Speed	Dedicated desktop GPU	✔
ANTNet [132]	2019	MobileNetV2	Attention NesTed block (ANTBlock)	1x1 conv replaced with 1x1 group conv in projection layer Channel attention mechanism applied to output of depth-wise layer All the inverted residual blocks replaced with ANTBlocks	Higher accuracy	Macroarchitecture Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU Desktop x64 CPU	x

Table 2 (continued)

Network	Year	Base network	Novel building blocks	Adjustments	Emphasis	Architectural scope	Evaluation metrics	Test hardware	Real-time
VoVNet [129]	2019	DenseNet	One-shot aggregation (OSA) block	All-previous-features aggregation at every layer is replaced with a single all-features concatenation in the last feature map	Less computational complexity	Macroarchitecture	Accuracy Computational cost Model size Speed	Dedicated desktop GPU	✔
GhostNet [133]	2020	MobileNetV3	Ghost module	Bottlenecks replaced with Ghost bottlenecks (two stacked Ghost modules) Both expansion and compression layers replaced with Ghost modules Regular conv replaced with subset of conv followed by simpler linear operations Linear operations used to generate additional intrinsic feature maps SE module applied to the residual layer in some Ghost bottleneck h-swish nonlinearity function avoided	Smaller model size	Microarchitecture	Accuracy Computational cost Model size Speed	Mobile single-core CPU	✔
Dilated-MobileNet [134]	2020	MobileNet	–	conv replaced with dilated conv with an expansion rate of 2 in shallow layers	Higher accuracy	Microarchitecture	Accuracy	Dedicated desktop GPU	–
DiCENet [135]	2020	ShuffleNet V2	DiCE unit	Dimension-wise convolutions (DimConv) learn local dimension-wise representations Point-wise conv replaced with Dimension-wise Fusion (DimFuse) for information fusion and global information incorporation DiCE unit factorizes standard conv using DimConv and DimFuse 3x3 depth-wise separable conv replaced with DiCE units	Less computational complexity	Microarchitecture	Accuracy Computational cost Speed	Embedded GPU Dedicated desktop GPU	✔
Res2NetLite [101]	2020	ResNet	–	Series of 3x3 conv layers replaced with bottleneck layer followed by a sequence of several Res2Block blocks [124] Same # input channels and # output channels for Res2Blocks Maximum # groups of 2 for all group conv in Bottleneck and Res2Blocks	Higher accuracy	Macroarchitecture Microarchitecture	–	–	–

Table 2 (continued)

Network	Year	Base network	Novel building blocks	Adjustments	Emphasis	Architectural scope	Evaluation metrics	Test hardware	Real-time
HGCNets [53]	2020	CondenseNet	Hierarchical group convolution (HGC) module	1x1 conv in bottleneck replaced with HGC module 3x3 conv replaced with depth-wise separable conv Optional integration of a SE block in to the HGC module to recalibrate channel-wise features	Higher accuracy	Microarchitecture	Accuracy Computational cost Model size	–	–
CSPNet [125]	2020	DenseNet	Partial dense block Partial transition layer (PTL)	Feature map in the block's first layer separated into two parts Only one part goes through the dense block Both parts fused at the end of the block by the PTL	Less computational complexity Higher accuracy	Macroarchitecture	Accuracy Computational cost Model size Speed	Embedded GPU Dedicated desktop GPU	✔
ChannelNets [131]	2021	MobileNetV2	Group module (GM) Group channel-wise module (GCWM)	Channel-wise conv used as a sparse 1x1 conv Group channel-wise conv used as fusion layer right after channel-wise conv Depth-wise separable channel-wise conv replaces the depth-wise separable conv Classification output layers (global pooling + fully connected layer) replaced with convolutional classification layer	Smaller model size	Macroarchitecture Microarchitecture	Accuracy Computational cost Model size	–	–
LCNet [105]	2021	VGG	Constant channel module (CCM) Down sampling module	3x3 conv replaced with depth-wise conv Shortcut connection added for residual structure Same # input channels and # output channels in CCM Two-branch downsampling strategy: depth-wise conv with stride 2+ max pooling	Less computational complexity Less memory access cost	Macroarchitecture Microarchitecture	–	–	–

categorizable in (ii) or (iii). Setting aside this exception, we will now discuss the specific features that characterize each of the three approach types indicated, and we will also address specific aspects concerning the different related architectures to clarify the underlying design principles.

i. *Pyramidal feature hierarchy* This is the simplest approach of the three categories identified, purely focused on detecting objects of interest with different sizes due to the exploitation of pyramid-shaped multiscale feature maps. It is represented in Table 1 by the SSD and SSDLite [48] detectors. SSD, as mentioned in Sect. 2, constitutes one of the more paradigmatic architectures within this approach, while SSDLite simply represents a lighter version of SSD, explicitly conceived for mobile devices by replacing conventional convolution operations with depth-wise separable convolutions, a practice that has been extensively adopted for alleviating the computational complexity of traditional CNNs, as shown below. Aside from this particular point, SSDLite does not introduce additional architectural concepts of interest beyond those already materialized in the original SSD architecture.

ii. *Recovery of high-resolution representations* This approach involves the fusion of multiscale feature maps to solve the lack of accuracy problem when detecting small objects, upsampling low-resolution representations to recover high-resolution representations progressively. This category possibly encompasses the most widely used neck-specific methods among those reviewed, covering mainstream pyramidal architectures such as FPN [86] or Hourglass [88], more compact alternatives such as Depth-wise FPN (D-FPN) [95]—which incorporates more efficient depth-wise convolutions into regular FPN—or YOLOv3-Tiny's neck [120], and even more differentiated proposals such as NAS-FPN [111]—exploiting feature-fusion building blocks automatically derived using Neural Architecture Search (NAS)—or FSSD [89]—an improved version of the SSD architecture. Among the approaches just mentioned, FPN is undoubtedly the most representative architecture for pyramid-like feature representation generation in object detection. According to the data presented in Table 1, FPN represents the architectural option that has been selected by more authors (ten papers out of a total of fifteen, including the D-FPN and NAS-FPN variants) as the foundation for building the neck part in the lightweight detection proposals.

D-FPN and YOLOv3-Tiny's neck are particularly interesting, since both of them follow the current on-device trend of exploring computer vision solutions tailored to low-power devices. D-FPN shares the same dual-path architecture as FPN (an initial downsampling stage followed by a second inverse stage) but it also succeeds in reducing upsampling path's computational complexity by exploiting a more optimal structure consisting of a bilinear interpolation layer followed by a depth-wise convolution. With regard to YOLOv3-Tiny's, the final implemented network results from a profound structural simplification, as is the case with the detector's global architecture. That structural simplification is performed by means of aggressive optimization practices such as the significant reduction in both the number of considered scales and integrated layers on the original YOLOv3 network [77], or the fusion of only single-scale features, which is certainly to the detriment of the semantic richness of the extracted features and, ultimately, the detection accuracy.

iii. *Maintenance of high-resolution representations* Along the lines of the previous approach, this strategy pursues a convolutional architecture design aimed again at the generation of high-resolution representations; in this case, however, communicating those representations throughout the entire detection network in order to avoid transitions between high and low resolutions, common in multiscale approaches. Specifically, High-Resolution Net (HRNet) [126], the only architecture listed in Table 1 corresponding to this paradigm, goes beyond multilevel fusion and, as an alternative, proposes taking a high-resolution convolution stream directly as a starting point, subsequently connecting in parallel one stream per considered resolution, thus exchanging information between multiple streams. In this way, multi-resolution fusion can be performed recurrently, resulting in high-resolution representations with great semantic richness and spatial precision.

3.1.1.2 Classification according to the enhancement type produced

The architectural solutions space just discussed clearly indicates a strong presence of pyramidal CNN models, originally designed as building blocks of standard unified detectors, halfway between the current on-device approach and the more complex traditional architectures. Those models, although able to produce better representations than those generated by ultra-compact networks, typically feature both a complexity and size impracticable for systems with modest capabilities, as well as an accuracy level lower than what is commonly reached by two-step detection frameworks. In that regard, the use of a pre-existing base CNN architecture, even though it is a practice that can lighten and, in certain occasions, completely bypass the study and design of specific solutions, streamlining the design of new architectures for the neck, does not itself constitute an optimal solution. As noted in Sect. 2, a twofold effort to advance in the direction of an improved speed-accuracy trade-off, with emphasis on techniques for size and computational complexity reduction so as to consequently reduce latency (i), but also exploring methods toward more expressive networks and therefore with greater detection capacity (ii). Next, we present the key

strategies and methods adopted in both directions to obtain a structure compliant to on-device paradigm's efficiency principles, omitting overly specific design details in order to keep a desired level of abstraction to facilitate the applicability of the adjustments required in different architectures or future cases.

i. *Size and latency reduction* This approach pursues building network architectures that could result in models with fewer parameters and lower computational complexity, that is, with smaller size and higher inference speed. The use of factorized convolutional filters represents the most paradigmatic mechanism related to this approach [48, 91, 92, 95, 97, 108, 111], constituting in itself a CNN-compression-specific subcategory of techniques that encompasses several lighter and faster variants of the standard convolution operation such as depth [48, 91, 92, 95, 97, 111] and group convolutions [108]. In addition, this group also embraces techniques based on the integration into the architecture of building blocks such as attention modules, used in [93] to both reduce the number of pixels to be processed in region-based detection and thereby increase the speed of object detectors, and Fire modules, explored in [96, 108] to, again, lower the number of parameters while preserving accuracy. Along the same lines, supplementing the exploitation of such blocks, Wang et al. [106] report the application of the recent CSP design [125] to the various structural components of a detector as a highly beneficial alternative to the more traditional residual connections able to reduce the number of parameters, the computations and the inference time. Finally, this category also includes more simplistic techniques such as the direct reduction of the quantity of weights in the network, for instance, removing larger feature maps as in [107]; the use of layers based on 1x1 filters instead of fully connected layers to perform predictions [107]; or the simple optimization of the number of filters used, even if that involves breaking the ruling microarchitectural homogeneity in CNN building blocks, as in [96].

ii. *Detection performance increase* This is a significantly more heterogeneous approach than the one presented in (i) aimed at achieving better classification performance but mainly focused on increasing detection accuracy, especially in complex applications such as small target detection. It is possible, therefore, to identify, two differentiated strategy types: general-purpose methods [91, 92, 97, 107, 108] applicable for any CNN and grounded in concepts that shall emerge again in the discussion of the backbone; and object-detection-specific methods [91, 94, 97–99, 102–105, 109, 115, 117], primarily focused on improving localization tasks.

Since the emergence of CNNs, there has been a well-known and long-standing interest, regarding general-purpose methods, in improving the performance of vision-based systems in classification tasks, exploring solutions aimed primarily at increasing the representation capacity of the built networks and, consequently, improving learning and accuracy. Although many of the actions taken to that end, such as making the network deeper, are largely impractical in the on-device context, the underlying philosophy remains completely applicable, and it also constitutes the foundation of a considerable body of more specific approaches or subcategories seeking lightweight solutions. As shown in Table 1, various studies can be found in the literature, such as: [108], which explicitly seeks to provide CNNs with a better and more efficient representation learning capacity by leveraging group convolutions, as in the referenced work; studies that perform more simplistic practices, for instance, exploiting larger convolution filters [91] and removing subsampling layers [92], to achieve or maintain a large-sized receptive field, enabling the subsequent encoding of a larger volume of information; approaches [107, 108] that, for example, rely on the addition of shortcut connections (residual blocks) in the network architecture in order to alleviate the vanishing gradient problem; strategies for increasing nonlinearity, such as the use of 1x1 pointwise convolution operations [97] or the use of the Hard Swish (h-swish) activation function instead of a more standard option such as Rectified Linear Unit (ReLU) [92]; and, finally, mechanisms for better information flow, such as the aforementioned shortcut connections [107, 108].

In terms of detection-specific enhancement solutions, these are usually methods that, based on multiscale feature maps [98, 109] (essential for the detection of multiple targets with different sizes), aggregate low-level high-resolution features with high-level semantic features to achieve greater semantic richness [91, 94, 97–99, 102–105, 109, 115, 117] as a result. Apart from two specific contributions that propose efforts directly related to the exploitation of multiscale features—increasing the number of different scale levels considered for the output [98] and using encoder–decoder structures for feature generation at different levels [109]—we identify table methods in the lightweight-detection-architecture-devoted that are essentially located in the space of solutions aimed at obtaining more valuable features, semantically speaking. More specifically, data presented in the table in this respect create a scenario where the fusion of multiscale feature maps [94] constitutes the dominant approach and where related works primarily focus both on different information transfer and exchange structures, namely dense connections [103, 115] and inverted residual blocks [99, 104, 105], and on attention mechanisms, an approach primarily aimed at extracting more discriminative features, mainly channel-wise [98, 104, 105] but also simultaneously at the spatial and channel level [102]. Additionally, several other approaches that also seek to

improve the network's expressiveness can be identified, but, in this case, they are achieved by enriching intermediate features due to the use of the convolution on dimension-reduction blocks [97] or by using attention modules to adjust the feature distribution and thus facilitate the distinction between background and front features (spatial attention) [91].

3.1.2 Detection head architectural principles

Twenty-one [47, 95, 96, 99, 100, 102–117] out of the thirty papers reviewed show no action explicitly focused on the design of the head for lightweight detectors other than the study and selection of the base network architecture to be used. It is possible to go a little further and even state that there is no trace of apparent activity in this regard since the different microarchitectures used for this purpose (at least, the ones reported in Table 1) are merely the structures proposed as detection head of the corresponding base detection frameworks. This is even extended beyond this group of publications that do not address head-specific enhancements and remains as a constant throughout all the works listed, with the dual exception of BMNet, which does not provide head-specific information at all, and LightDet, which proposes its own microarchitecture conceived from scratch.

Regarding the different architectural alternatives used as a reference for the design of the head in lightweight object detectors, Table 1 shows a general scenario very similar to the one described in the previous point for the neck, dominated by networks initially conceived as structural elements of unified detectors, but a scenario that, in this particular case, features a slightly broader range of options. An initial superficial exploration of the data collected in the table makes it possible to infer a predominant network type or profile characterized not only by its unified architecture but also by its ability to detect objects with different scales and aspect ratios due to multiscale feature processing and the use of anchor boxes in the detection process. Thus, fitting the profile outlined, a total of six base head designs (originally part of SSD, SSDLite, YOLOv3, YOLOv3-Tiny, RetinaNet, and RefineDet) adopted in eighteen of the twenty-five works studied can be found in Table 1. There are also microarchitectural alternatives that do not, albeit almost marginally, conform to the well-known anchor-based approach, either because they have been implemented as a part of a two-stage detection pipeline (Faster R-CNN and Light-Head R-CNN), despite relying on anchors, or simply because they have been conceived as part of non-anchor-based detectors (YOLO, CornerNet, and FCOS [127]).

Regarding any modifications applied to the base architecture, the two-fold approach already identified during the neck-related discussion in Sect. 3.1.1 (and present in the backbone analysis as well) emerges again. Thus, it is possible to classify the neck-specific enhancement methods into the same two categories or approach types: a first group focused on size and latency reduction (i) and a second body of techniques with an emphasis on increasing or at least maintaining detection accuracy to make up for the potential harm caused in this respect by the techniques in the first category (ii). Overall, there is a major gap in terms of prevalence distribution between neck-related adjustments and those targeting the detection head. More specifically, the data presented in Table 1 show that there is an evident polarization of head-centered techniques into two distinct groups that did not emerge in the analysis of neck-related approaches. Thus, except for a couple of papers that can be simultaneously associated with the two different approach types considered [92, 94], every single modification can be located in one of the two indicated solution spaces. That divergence becomes even more pronounced if we take into account the size of those spaces: quite even for the two groups when it comes to the neck while significantly uneven when talking about the head. Furthermore, regarding the latter, the subgroup of methods that seek to lower computational and memory cost have an evident prominence (approach embodied by five publications [48, 91, 93, 98, 118] referenced in Table 1) compared to the method that encompasses accuracy-centric modifications (with only two representative works [97, 101] in the table).

Turning now to specific approaches, we identify in group (i) strategies that are mainly aimed at reducing the number of parameters in the models produced, and thus are able to initially reduce the models' size, and consequently in many cases, their computational complexity as well. Among them we can distinguish techniques eminently focused on the inner configuration of layers and, therefore, on the modifications of filter-specific aspects: the use of depth-wise separable convolutions instead of standard convolutions [48], the decrease in the number of channels [91], or the use of smaller-sized convolutional filters [93, 94]. Also included in this parameter-reduction-oriented subgroup are methods that address more general layer-related considerations, such as replacing fully connected layers with convolutional layers [118] or simply omitting a subset of the layers that can be found in the original architecture [98]. Finally, to complement the different approaches just mentioned, we also associate to group (i) a different subcategory or approach type that directly pursues computational complexity reduction, represented in Table 1 by a single paper [92] that proposes the addition of dedicated layers for removing the background of the given input image (suppression of non-useful

information) to reduce the number of pixels to be processed.

The second head-specific- category or group (ii) is monopolized by the exploitation of residual blocks as a constituent part of the head's structure [92, 94, 97, 101]. Such an approach is able to both increase the detection accuracy and contribute to reducing the resulting network's memory requirements. Used as an enhancement mechanism also for the neck's architecture, as indicated in Sect. 3.1.1, this type of block integrates the so-called shortcut connections to allow the flow of information between shallow and late-stage layers and thus keeps the semantic richness of features [97]. The value of residual connection goes beyond its accuracy-enhancement ability; it has also been adopted as the base structure for conceiving architectural alternatives equally advantageous in terms of detection accuracy, such as the bottleneck residual block [101], capable of fusing high-level multi-scale features, the inverted residuals and linear bottlenecks [92] that enable increasing the representational power of channel-wise nonlinear transformations, and a lighter version [94] that, inspired by group convolutions and comprised of two different branches, leverages channel shuffle to allow information exchange between branches.

3.1.3 Efforts for a more efficient backbone

Specifically, concerning the architectures integrated as backbone in the detection frameworks under examination, the related data collected in the table confirm the predominant, but not exclusive, use of simplified CNN architectures. Excluding FRDet [100]—with no representative data reported in Table 1 about the network or architecture used as backbone—twenty-four frameworks of a total number of thirty use a lightweight subnetwork as backbone. Furthermore, in that group we can identify just ten distinct alternatives, a number that could be even lower if grouped into families of detectors: MobileNets [47, 48, 92, 99, 111–113, 115, 117], ShuffleNets [91, 94, 97, 104], SqueezeNet [96, 108, 118], PeleeNet [102, 107], and DarkNet-19 [103, 109, 114, 116]. Google's MobileNets emerges as the most dominant lightweight architectural solution. This observation, although it ignores configuration or structural efficiency matters (they will be addressed in the next section), is entirely consistent with the evolutionary sequence of on-device vision models reported in the literature, where MobileNets, first introduced in 2017 [47] and with three different versions, stands as the most mature compact alternative as well as one of the main drivers of the growing attention generated by ultra-compact vision models in the research community during the last few years. Finally, regarding the rest of the backbone-specific architectures referred to in the table, apart from the lightweight alternatives, it is possible to identify a second group that encompasses five standard CNN architectures [93, 95, 98, 101, 110], where, beyond the mere intuition of a more specific nature, it is not possible to infer any pattern that might be of interest in this analysis.

In a joint review of the several architectures adopted as backbone and the adjustments applied to them, it is possible to extract observations that, while not backed by specific metrics and measurements, provide valuable intuition about the performance and, in general, the suitability of the architectural solutions proposed. In that sense, even though lightweight CNN architectures have been designed from scratch, bearing in mind the hardware limitations of the target devices, and have largely succeeded in deriving models of extremely reduced size and complexity, they may still be insufficient or inadequate solutions depending on various factors such as the hardware platform and the application domain. This reality is reflected in Table 1, where few studies report directly employing compact CNN architectures as backbone of the detector [47, 48, 97, 105, 107–109, 111, 113, 118], while a fair majority proposes specific enhancements or optimizations [91, 92, 94, 96, 99, 102–104, 106, 112, 114, 116, 117] for the architectures previously selected. Going into more detail, a closer look at the data on such modifications allows us to identify an approach that is fundamentally oriented at obtaining greater precision [91, 92, 94, 99, 103, 104, 112, 114, 117], which confirms the need to make up for the accuracy degradation typically resulting from the structural simplification or miniaturization of the network.

Limiting our focus to the modifications applied to the architecture selected as the starting point for building the backbone, it is possible to categorize the strategies and methods listed in the table into the same two groups we considered for this purpose in both Sects. 3.1.1 and 3.1.2. Hence, we identify once again a group of techniques on one side of the table that respond to a size and latency reduction approach, and, on the other side, a collection of methods focused on preserving ad increasing accuracy.

The first group contains techniques basically aimed at reducing the computational cost of the network. As we pointed out in relation to the mechanisms designed for enhancing the head's structure, it is possible to identify two different types of solutions within this group according to the architectural level they operate on. In particular, in a first microarchitectural subgroup, we find (i) approaches based again on the exploitation of more efficient variants of the convolution operation, such as depth-wise convolutions [95, 98, 102], depth-wise separable convolutions [93, 116], and group convolutions [98]; and in the second group we find (ii) strategies that have a direct effect on the configuration of the convolution filters used in layers or blocks of the CNN, i.e., both methods targeting the number of

filters—reducing the number of filters of the first layer of the network [112], selecting an optimal number of filters specific per building block [96], and the choice of a better compression rate for Fire modules [100]—and additional techniques focused on the number of channels—linearly increasing the number of channels as the network deepens [95], or assigning the same number of channels both to the input and output of residual blocks such as Res2Net [101]. Finally, in addition to the slimming strategies pointed out, there is a second collection of solutions that address the same problem, but, through a macro lens, exploring different design options such as the removal of some layers present in the original architecture, the CSP-ization of the network [106], the use of Fire modules [93, 103] (with greater ability to reduce the number of parameters), the thinning of layers and building blocks [93], and more appropriate distribution of subsampling layers across the network [93, 101].

Closing this review of the specific tweaks performed on the backbone, we can also identify in Table 1 a substantial number of studies that explore alternatives in the search for greater accuracy in object detection. Among the options listed, the residual block structure, based on shortcut connections, is revealed as the most versatile approach in this regard, constituting an effective solution for increasing accuracy both in classification and detection tasks and also the preferred option [98–100, 103, 114] among the several related alternatives presented in the table. Interest in the ability of residual connections to enable better feature propagation and guarantee maximum information flow across the network goes beyond the residual block. Thus, that type of connection has been successfully incorporated into other building blocks, being used, for example, as an upgrade of Fire modules [100] or as an integral part of the inverted residual blocks exploited in [99] to achieve better multi-scale detection. In addition to the detection-specific techniques for better accuracy just mentioned, the accuracy-focused approach also encompasses a second collection of methods primarily designed to provide better class predictions. In the table, we can distinguish the following: (i) approaches that seek to increase the size of the receptive field due to, among other practices, the use of larger convolution filters [91, 104, 117], the insertion of bottleneck layers for subsampling at different stages of the network [101], and the use of dilated convolutions in the network stem [92, 94]; (ii) studies such as [94] or [91], which by using either dilated convolutions or convolutional filters with a higher number of channels in early stages of the network, seek to extract and preserve more low-level features; and, finally, (iii) alternatives of a more punctual nature, already mentioned in the two previous analysis made correspondingly on the neck and head-oriented modifications, such as the use of the h-swish activation function [112], the inclusion of attention modules to increase the representation power of the network [92, 117], or the application of channel shuffle after group convolutions to enable information exchange between groups.

3.2 High-efficient CNN architectures for backbone build

The analysis of the mechanisms and design strategies adopted for conceiving lightweight detection frameworks reveals a constant interest in finding a better trade-off between accuracy and detection speed. In this context, the backbone constitutes the key component within the object detector architecture, not only because it lay downs the structural guidelines for detectors but also because it is the component responsible for processing input images in the first stage of the detection pipeline in order to extract the features that are supplied later on to the two remaining components of the detector. Backed by the data collected in Table 2, we extend the analysis performed in Sect. 3.1.3 with additional lightweight CNN architectures that, despite not having been used to date for building detection frameworks in the on-device context, have been entirely conceived under the design principles of this paradigm. As in the different subsections included in Sect. 3.1, we will address the structural specificities of the different CNN architectures considered, focusing our efforts on identifying the principal techniques and methods applied in each case.

In a first superficial review of the data included in the table, which was focused only on the first four columns, it is possible to derive several general points that add further detail to the on-device scenario so far presented. The architectural developments, with the exception of the study from 2016 by Iandola et al. [49], are temporally located between 2017 and 2021, just like the different detection frameworks above analyzed. Once again, it confirms the chronological parallelism between lightweight-CNN-specific and ultra-compact-detector-specific development approaches already pointed out in Sect. 3.1.3, and it also reinforces the key role that recent general computer vision progress has played in developing ultra-compact detection systems. Regarding the CNN architectures used as a reference for conceiving algorithmic solutions deployable on low-powered devices, except for a handful of authors working on conventional CNNs [47, 49, 54, 55, 101, 105, 107, 125, 128, 129], the mainstream focus has been on exploiting lightweight architectures as the starting point. Moving on to the detail of the specific architectures used for that purpose, a family-based grouping of the several approaches considered can be easily observed (MobileNets [48, 130–134], ShuffleNets [50, 52, 135], SqueezeNet [136], and CondenseNet [55]) bearing a strong similarity to the approach laid out in the

previous section, emerging again as the most used lightweight architectural solution the MobileNets family. The residual structure also stands out as the design pattern with the more significant presence in these architectures, either directly as part of the base standard CNNs [54, 55, 101, 107, 125, 128, 129] or as the base structure of the novel building blocks resulting from the enhancement techniques and methods applied.

If we turn our attention to the various adjustments made, the first thing that stands out is the significantly higher number of enhancements listed in Table 2 for each study if we compare it with the number of entries that we can see for the same concept in Table 1. Those data highlight the complexity of the miniaturization and rational simplification of CNN architectures for use on edge devices, as well as the enormous research efforts undertaken in this line in recent years, which has made it possible to enhance and streamline the design of specific on-device solutions in different vision-related application domains, such as object detection. There is, however, a gap between the predominant backbone-focused adjustment type found in Table 1 and the type derived from Table 2. More specifically, in the first case, we mostly find techniques and methods that emphasize achieving greater precision for producing a lightweight backbone design (mainly defined by the base CNN architecture), yet with an effective expression capacity to properly act as a structuring element in detection systems. In contrast, for the adjustments listed in Table 2, the focus is placed on obtaining more efficient CNN architectures (in particular, in seventeen [47–50, 52, 54, 55, 104, 107, 125, 128–131, 133, 135, 136] of the twenty-one works under study), also considering new avenues of exploration in this respect such as the reduction of memory access cost or the exploitation of more efficient optimized-implementation-based operations at the code level.

In terms of scope at the architectural level, we can make a first classification of the enhancement techniques and methods considered in two different types of approach: those operating at the microarchitectural level, i.e., at the inner level of layers and modules; and those working at the macroarchitecture level, defining arrangement-specific aspects regarding the different modules or layers within the CNN architecture. Beyond the data collected in the *Architectural scope* field in Table 2, which aims to capture the general essence of the several related works listed, a more detailed analysis of required adjustments provides a much more accurate picture of the trend in terms of structural design, especially with such an important body of information as the one presented in the table. Thus, although a majority of microarchitectural adjustments can already be noted from the data in the *Architectural scope* field, that becomes even more evident when the data included in the *Adjustments* column are incorporated into the study. Numerically speaking, only fifteen macroarchitectural adjustments are identified compared to the fifty observed at the microarchitecture level. More specifically, the first group of approaches encompasses strategies to enhance the CNN's overall architecture by (i) replacing a certain type of layers or building blocks with more lightweight alternatives [49, 131, 132] or variants with greater capacity to maintain or even increase the expressiveness of the network [101], (ii) implementing guidelines governing how certain network properties or elements evolve as it becomes deeper [49, 55], and (iii) appropriately configuring the connections between layers or modules [48, 55, 105, 125, 129]. Within the group of micro approaches, we find a wide range of options that can be categorized into two distinct subgroups: an initial collection of techniques that focus on convolutional-filter-specific aspects or properties such as the number of filters [107], the size of these in the spatial dimension [49], the number of channels [49, 52, 101, 105, 107, 130], the communication between them [50, 54], or the number of channel groups [101]; and a second subgroup encompassing methods targeting the internal structure of layers or modules such as the exploitation of alternative operations to convolution [47, 48, 50, 52, 54, 105, 107, 128, 130, 131, 133–136], the replacement [48] or omission [133] of nonlinearity, or the application of an attention mechanism [53, 132, 133].

Keeping structural consistency with the different subsections in Sect. 3.1, we establish a second categorization of the adjustments under consideration, according to the targeted network-accuracy-specific features or aspects. Thus, we identify techniques that respond to a size and latency reduction approach and as well as methods focused on preserving and increasing accuracy as much as possible.

In the first category, the usage of less costly convolutions—depth-wise convolution [47, 48, 54, 105], separable convolution [136], and depth-wise separable convolution [53]—stands out again as the most common approach, extended in this case by the exploration of other practices that revolve around additional efficient operations: replacing costly standard convolutions with memory shift operations [50, 128] for information fusion, information exchange between channels, and channel concatenation; replacing 1x1 group convolutions with a less complex channel split operation [52]; using simpler linear operations for partially generating feature maps [133] instead of fully using convolutions for that; or designing a novel building block to encode spatial and channel information with higher efficiency than depth-wise separable convolutions [135]. Precisely in relation to channels and, specifically, to the introduction of sparsity in the connections, we identify a second large group of adjustments that encompasses some of the strategies already observed in previous analyses such as the replacement of pointwise convolutions

with group convolutions [54, 132], but also unseen related techniques, such as the replacement of pointwise convolutions with channel wise more sparse convolutions [131], or the conception of a novel type of convolution that extends group convolution and, in contrast to the latter, allows an output channel to depend on an arbitrary subset of input channels, thus obtaining greater computational efficiency and reducing the number of parameters. Also related to channels, but in this case, focused on the number of channels handled, we identify additional enhancement strategies that pursue channel reduction [49, 136] and some others that lead to interesting guidelines about what the ratio between the number of input and output channels should be in order to lower the computational cost [105, 107] or the memory access cost [52]. Rounding this collection of efficiency-focused adjustments, there are several solutions of a more precise nature, such as exploiting convolutions with a more efficient software implementation [107, 136], using more efficient residual structures [125, 129] or downsampling strategies [105], omitting h-swish nonlinearity due to its high latency [133], merging successive element-wise operations, and thus lowering this type of costly operation in terms of memory access [52], removing redundant connections [55], using smaller-sized convolutional filters [49], or replacing heavy layers with a lighter alternative [49].

Finally, regarding refinement approaches expressly designed to preserve or increase accuracy, it is possible to distinguish a considerable range of different techniques, which are, however, practically evenly distributed. Specifically, except for just one of the adjustments in consideration, it is possible to cluster the options listed into seven distinct groups, each comprising two specific strategies or methods. We have identified the following groups in the table: (i) approaches aiming to prevent feature map size reduction in order to avoid harming the network expressiveness, either by delaying subsampling, i.e., moving subsampling layers or blocks to deeper stages of the network [49], or by using transition layers—composed of convolution and pooling operations—without compression [107]; (ii) methods that, like channel shuffle [54] already introduced in Sect. 3.1.3, enable information exchange between channels, either via more efficient memory shift operations [128], or in a more straightforward way replacing point-wise group convolutions with alternatives that do not block the above-mentioned information exchange between groups [53]; (iii) techniques based on the exploitation of the residual block structure, adding to the network architecture not only the already well-known shortcut connections [48], but also dense connections to boost feature reuse [55]; (iv) strategies that rely on the gradual increase of the growth rate in dense-connection-based networks [55, 107] to cost-effectively increase feature expressiveness; (v) approaches focused on the receptive field that, in line with some of the practices already identified for neck and head refinement, aim to both increase its size [134] and also generate variations with different scales [107]; (vi) the integration of attention mechanisms [132, 133] to boost representational power; and even (vii) simpler practices such as removing nonlinearities in shallow layers to preserve the representativity of the network as well [48].

4 Learned lessons and conclusions

This paper provides a review of the leading recent research efforts aimed at bringing historically demanding object localization and classification tasks to terminal devices with limited memory and computational resources. In particular, the study provides a comprehensive analysis of the main CNN architectures specifically designed to generate efficient and compact vision models directly deployable on mobile and embedded devices as part of real-time object detection software solutions. That is why we cover in this study the most relevant architecture strategies and techniques that have been used not only to enhance the design and configuration of the different components that comprise this type of solutions—backbone, neck, and head—but also and primarily to make them suitable for more austere deployment environments, are contemplated in the study.

The backbone is the most critical element in the detector's architecture because of its predominance in the general structure of the detector as well as its impact on the performance of the "wrapping" framework. Specifically, the detection system's accuracy significantly depends on the expressiveness of the CNN used as backbone and on the ability of the latter to extract representative properties. Likewise, aspects of the backbone's topology, such as the depth or size (in both the spatial and channel dimensions) of the layers, have a strong influence on the final computational complexity and memory requirements of the resulting detector. However, even though it has a tremendous influence on the detector's accuracy and efficiency, the direct translation of general-purpose CNN networks to object detection by merely replacing the final classification layers with a detection head does not constitute an optimal solution.

Ultra-compact detectors, for instance, MobileNet+ SSD [47] and Mobile-YOLO [113], emerged precisely from the replacement of the backbone network integrated into single-stage detectors (SSD [47] and YOLOv3 [77], respectively) with an even smaller and lighter CNN architecture, such as MobileNet [47] and MobileNetV2 [48], explicitly conceived for mobile vision applications. The adoption as a

starting point of a macroarchitecture based on a unified pipeline primarily oriented toward higher inference speed, jointly with the adoption as backbone of a slimmer and simplified network model, enables building high-speed detection systems. This simplistic approach, however, delivers processing times that fall short of real-time performance and, worse yet, an accuracy level far below the values reported for state-of-the-art approaches.

The design of appropriate architectural solutions requires a more comprehensive approach, based on the exploration and application of techniques and methods compliant to the singularities of each of the components in the detection pipeline, aimed at increasing accuracy and reducing both complexity and size in order to achieve a more desirable latency-accuracy trade-off. In this regard, the present work evidences the strong attention the neck has attracted from the research community, an interest that sharply contrasts with the lack of contributions focused explicitly on the detection head structure, usually derived directly from the detection framework adopted as the base architecture. The role played by the neck in modern object detection frameworks is particularly relevant for localization tasks. More specifically, it is responsible for the generation of high-resolution representations [86, 88, 89, 95, 111, 120], the exploitation of multi-scale feature maps [48, 72], and the fusion of the latter to initially extend and refine the features extracted by the backbone, and consequently, enable the detection of both objects with different scale values and small-sized targets.

In addition to the overview on the several architectural solutions that shape current lightweight detection pipelines, a vast number of significant techniques and factors regarding the design process have emerged as a result of the analysis performed throughout Sect. 3 on dozens of publications focused on the construction of both detection systems and, in general, on ultra-compact CNNs networks. Among the collection of architectural modifications discussed, it is possible to pinpoint a fair number of points and practices that remain a shared research focus for the three different components in an object detection framework: (i) the joint exploitation of low-level features, extracted in early stages of the network and critical for localization, and high-level features, extracted in late stages and fundamental for classification [91,94,97–99,102,103,109,115]; (ii) the proper sizing of the receptive field for learning high-resolution features [91, 92, 94, 101, 107, 134]; (iii) the configuration of the number and dimensions of convolutional filters [49, 91, 93, 94, 96, 104, 105, 107, 112, 117], with both factors having a substantial impact on the network accuracy as well as on the size and computational complexity of the derived models, and with the decomposition of convolutional filters [48, 91, 92, 95, 97, 108, 111] being particularly relevant in this respect; (iv) the design of operations more efficient than standard convolutions [48,50,91–93,95,97,98,102,105,108,111,116,128]; and (v) the exploitation of channel correlation, an approach comprising some of the most popular methods such as the use of shortcut connections [48,98–100,103,105–108,114,125,129] and group convolutions [54, 108, 132].

The rational choice of a detection framework and a base backbone architecture, both small-sized, together with the introduction of tweaks eminently oriented to the reduction of the size and complexity of the underlying CNN network (while also accuracy-compliant), has already produced promising results, as reflected by the evaluation data collected in Tables 1 and 2. Even though there are some works such as [52, 91, 97, 99, 100, 103, 107, 111, 114, 125, 133, 135] that report very low latency values, thereby confirming the feasibility of real-time object detection on edge devices, there is still an overwhelming majority of them that either fail to achieve such efficiency or present significant gaps in the evaluation that somewhat blur the results presented above. Regarding the latter point, the tables list: (i) works that have not considered execution speed as a metric for efficiency assessment, relying merely on the number and computational cost of the operations involved [47, 49, 53, 96, 101, 128, 131, 134]; (ii) studies that provide no information about the devices used for the execution and testing of the models, reporting in many of those cases only the hardware configuration used for training [47, 49, 53, 93, 96, 98, 101, 102, 105, 112, 128, 131]; and even (iii) publications that report real-time performance but only for solutions tested on desktop devices with high-powered graphics, and, therefore, do not experience the hard memory and computational constraints that characterize the on-device paradigm [50, 92, 94, 95, 101, 108, 109, 115, 129].

These absences do not tarnish the remarkable work done so far on the miniaturization of DL-based detection solutions. However, they do represent and direct us to highly relevant issues that need to be addressed in future work. We conclude the present study by briefly discussing those challenges, identifying specific research problems, and pointing out possible approaches to be explored.

With particular regard to the challenges that still lie ahead, there is, generally speaking, a notable absence of specific discussion on factors closely related to the support hardware configurations used in lightweight detection systems, precisely in a domain where hardware limitations constitute the principal point to consider for building proper detection frameworks. Instead, with the exception of two specific studies that include memory access cost [52] and energy consumption [136] in their analysis, researchers have primarily focused their efforts on evaluating how efficient and robust the conceived approaches are, eminently adopting the number of operations required

and, less frequently, the execution speed of the derived models as efficiency metrics. Moreover, the portfolio of DL-oriented hardware acceleration solutions has grown in the last few years as a result of advances in the mobile sector and also the development of novel cutting-edge chipsets designed ad hoc accelerating artificial intelligence tasks. Although GPUs have become a de facto standard when it comes to real-time DL inference on edge devices, there is still an important gap in the study and exploitation of modern AI accelerators.

Leaving hardware-specific issues aside, several other issues emerge from the research performed, in this case, related to the design and assessment of detection frameworks and their underlying architecture. Thus, although CNNs have meant a tremendous step forward in computer vision, enabling the automation of feature extraction and thereby reducing the need for human intervention in the solution-making process, the design of both lightweight detection frameworks and, in general, of CNN architectures optimized for low-power hardware platforms is usually the result of an in-depth research effort, derived from expert knowledge and mostly driven by an exhaustive exploration of the myriad of existing operations and design alternatives, all of which makes the process of finding an optimal configuration costly. When evaluating the solutions found in the literature reviewed, a significant disparity can be intuitively observed in the decisions made as part of the protocols used to evaluate and compare the proposed approaches. The heterogeneity observed on matters such as the dataset used, the hyperparameters configuration, or the testing hardware, although it does not invalidate the reported results, definitely introduces a certain degree of uncertainty. Indeed, it is for this reason that, despite being a standard practice, we have avoided using specific accuracy or efficiency values in the present study to substantiate the observations derived.

Despite the recent tremendous interest and subsequent progress on DL object detection models deployable on mobile and embedded devices, we still observe a fair number of substantial questions that remain to be answered to reach a desirable level of maturity. Therefore, we have outlined some potential future lines of work that could lead to further steps toward such maturity:

- Incorporating to the evaluation of additional hardware-related aspects for on-device solutions that may affect their final performance, by either generalizing the study of energy consumption and memory access cost mentioned above or by considering other relevant factors, for instance, parallel processing or how to use current multi-core architectures effectively.
- Exploring the suitability of next-generation AI acceleration devices and electronic components, not just for the sake of determining which alternative is more optimal for the problem of interest but also to explore additional questions of interest such as: (i) to what extent their joint exploitation is possible and, if so, what specific benefits this would bring in terms of efficiency or the extent to which it allows for easing computational constraints and thus increasing accuracy; or (ii) how much their use impacts energy consumption.
- Assessing whether the lightweight-detector-related methods discussed are able to exploit the capabilities of the modern acceleration hardware and, if necessary, further investigating new techniques for devising models that can effectively make use of such specific hardware configurations.
- Simplifying and streamlining the design of architectures for on-device vision-based detection solutions. In recent years, a number of studies have been developed to automate the search for solutions, given a space of potentially desirable optimization techniques and parameterizations. Such techniques, collectively known as NAS, have already shown enormous potential [112, 138–144] and are undoubtedly a promising approach to automatically synthesize more optimal network designs, incorporating hardware-specific metrics such as latency and power consumption into the objective function that guides the solution search.
- Benchmarking the performance of light-weight object detectors in a greater-fairness context or configuration. While it may be impractical to compare all recently proposed detectors, we believe it would be of great interest to the research community to establish a common evaluation framework for the most representative detectors in order to execute a unified comparison.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. Funding for open access charge: Universidade de Vigo/CISUG. This work was partially supported 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.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Cao Z, Hidalgo G, Simon T, Wei SE, Sheikh Y (2021) OpenPose: realtime multi-person 2D pose estimation using part affinity fields. IEEE Trans Pattern Anal Mach Intell 43:172–186. https://doi.org/10.1109/TPAMI.2019.2929257
2. Dollár P, Wojek C, Schiele B, Perona P (2012) Pedestrian detection: an evaluation of the state of the art. IEEE Trans Pattern Anal Mach Intell 34:743–761. https://doi.org/10.1109/TPAMI.2011.155
3. Yang S, Luo P, Loy CC, Tang X (2016) WIDER FACE: a face detection benchmark. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), 27–30 June 2016, pp 5525–5533. https://doi.org/10.1109/CVPR.2016.596
4. Zhang HB, Zhang YX, Zhong B, Lei Q, Yang L, Du JX, Chen DS (2019) A comprehensive survey of vision-based human action recognition methods. Sensors (Switzerland) 19:1005. https://doi.org/10.3390/s19051005
5. Wei J, He J, Zhou Y, Chen K, Tang Z, Xiong Z (2020) Enhanced object detection with deep convolutional neural networks for advanced driving assistance. IEEE Trans Intell Transp Syst 21:1572–1583. https://doi.org/10.1109/TITS.2019.2910643
6. Mishra B, Garg D, Narang P, Mishra V (2020) Drone-surveillance for search and rescue in natural disaster. Comput Commun 156:1–10. https://doi.org/10.1016/j.comcom.2020.03.012
7. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1:541–551. https://doi.org/10.1162/neco.1989.1.4.541
8. Kazemi FM, Samadi S, Poorreza HR, Akbarzadeh-T MR (2007) Vehicle recognition using curvelet transform and SVM. In: Fourth international conference on information technology (ITNG'07), 2–4 April 2007, pp 516–521. https://doi.org/10.1109/ITNG.2007.205
9. Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
10. Wu S, Nagahashi H (2014) Parameterized adaboost: Introducing a parameter to speed up the training of real adaboost. IEEE Signal Process Lett 21:687–691. https://doi.org/10.1109/LSP.2014.2313570
11. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60:91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
12. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: 2005 IEEE Computer society conference on computer vision and pattern recognition (CVPR'05), 20–25 June 2005, vol 881, pp 886–893. https://doi.org/10.1109/CVPR.2005.177
13. Ojala T, Pietikäinen M, Mäenpää T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24:971–987. https://doi.org/10.1109/TPAMI.2002.1017623
14. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60:84–90. https://doi.org/10.1145/3065386
15. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115:211–252. https://doi.org/10.1007/s11263-015-0816-y
16. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The pascal visual object classes (VOC) challenge. Int J Comput Vis 88:303–338. https://doi.org/10.1007/s11263-009-0275-4
17. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: 3rd International conference on learning representations (ICLR), San Diego, CA, USA, 7–9 May 2015
18. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), 27–30 June 2016, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
19. Lin TY, Goyal P, Girshick R, He K, Dollar P (2020) Focal loss for dense object detection. IEEE Trans Pattern Anal Mach Intell 42:318–327. https://doi.org/10.1109/TPAMI.2018.2858826
20. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), 7–12 June 2015, pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
21. Kang Y, Hauswald J, Gao C, Rovinski A, Mudge T, Mars J, Tang L (2017) Neurosurgeon: collaborative intelligence between the cloud and mobile edge. ACM SIGPLAN Not 52:615–629. https://doi.org/10.1145/3037697.3037698
22. Teerapittayanon S, McDanel B, Kung HT (2017) Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International conference on distributed computing systems (ICDCS), 5–8 June 2017, pp 328–339. https://doi.org/10.1109/ICDCS.2017.226
23. Chinchali S, Sharma A, Harrison J, Elhafsi A, Kang D, Pergament E, Cidon E, Katti S, Pavone M (2021) Network offloading policies for cloud robotics: a learning-based approach. Auton Robot. https://doi.org/10.1007/s10514-021-09987-4
24. Jauro F, Chiroma H, Gital AY, Almutairi M, SiM A, Abawajy JH (2020) Deep learning architectures in emerging cloud computing architectures: recent development, challenges and next research trend. Appl Soft Comput 96:106582. https://doi.org/10.1016/j.asoc.2020.106582
25. Wu H, Li X, Deng Y (2020) Deep learning-driven wireless communication for edge-cloud computing: opportunities and challenges. J Cloud Comput 9(1):21. https://doi.org/10.1186/s13677-020-00168-9
26. Qayyum A, Ijaz A, Usama M, Iqbal W, Qadir J, Elkhatib Y, Al-Fuqaha A (2020) Securing machine learning in the cloud: a systematic review of cloud machine learning security. Front Big Data 3(43):587139. https://doi.org/10.3389/fdata.2020.587139
27. Wu H, Zhang Z, Guan C, Wolter K, Xu M (2020) Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. IEEE Internet Things J 7(9):8099–8110. https://doi.org/10.1109/JIOT.2020.2996784
28. Choi H, Bajić IV (2018) Deep feature compression for collaborative object detection. In: 25th IEEE International conference on image processing (ICIP), 7–10 Oct 2018, pp 3743–3747. https://doi.org/10.1109/ICIP.2018.8451100
29. Ishakian V, Muthusamy V, Slominski A (2018) Serving deep learning models in a serverless platform. In: 2018 IEEE International conference on cloud engineering (IC2E), 17–20 April 2018, pp 257–262. https://doi.org/10.1109/IC2E.2018.00052
30. Varghese B, Buyya R (2018) Next generation cloud computing: new trends and research directions. Futur Gener Comput Syst 79:849–861. https://doi.org/10.1016/j.future.2017.09.020

31. Wang J, Zhang J, Bao W, Zhu X, Cao B, Yu PS (2018) Not just privacy: improving performance of private deep learning in mobile cloud. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, London, United Kingdom, 2018. Association for Computing Machinery, pp 2407–2416. https://doi.org/10.1145/3219819.3220106
32. Dhar S, Guo J, Liu J, Tripathi S, Kurup U, Shah M (2019) On-device machine learning: an algorithms and learning theory perspective. arXiv preprint arXIv:1911.00623
33. Chen T, Du Z, Sun N, Wang J, Wu C, Chen Y, Temam O (2014) DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In: Proceedings of the 19th international conference on architectural support for programming languages and operating systems, Salt Lake City, Utah, USA, 2014, pp 269–284. https://doi.org/10.1145/2541940.2541967
34. Chen YH, Yang TJ, Emer JS, Sze V (2019) Eyeriss v2: a flexible accelerator for emerging deep neural networks on mobile devices. IEEE J Emerg Sel Top Circuits Syst. https://doi.org/10.1109/JETCAS.2019.2910232
35. Yin X, Chen L, Zhang X, Gao Z (2018) Object detection implementation and optimization on embedded GPU system. In: 2018 IEEE International symposium on broadband multimedia systems and broadcasting (BMSB), 6–8 June 2018, pp 1–5. https://doi.org/10.1109/BMSB.2018.8436848
36. Andargie FA, Rose J, Austin T, Bertacco V (2017) Energy efficient object detection on the mobile GP-GPU. In: 2017 IEEE AFRICON, 18–20 Sept 2017, pp 945–950. https://doi.org/10.1109/AFRCON.2017.8095609
37. Wai YJ, Yussof ZM, Irwan S, Salim M (2019) A scalable FPGA based accelerator for Tiny-YOLO-v2 using openCL. Int J Reconfigurable Embed Syst (IJRES) 8:206–214. https://doi.org/10.11591/ijres.v8.i3.pp206-214
38. Guo K, Zeng S, Yu J, Wang Y, Yang H (2019) [DL] A survey of FPGA-based neural network inference accelerators. ACM Trans Reconfigurable Technol Syst 12(1):2. https://doi.org/10.1145/3289185
39. Zhang C, Li P, Sun G, Guan Y, Xiao B, Cong J (2015) Optimizing FPGA-based accelerator design for deep convolutional neural networks. In: Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, Monterey, California, USA, 2015. Association for Computing Machinery, pp 161–170. https://doi.org/10.1145/2684746.2689060
40. Kaarmukilan SP, Poddar S (2020) FPGA based deep learning models for object detection and recognition comparison of object detection comparison of object detection models using FPGA. In: 2020 Fourth international conference on computing methodologies and communication (ICCMC), 11–13 March 2020, pp 471–474. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00088
41. Wu J, Leng C, Wang Y, Hu Q, Cheng J (2016) Quantized convolutional neural networks for mobile devices. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), 27–30 June 2016, pp 4820–4828. https://doi.org/10.1109/CVPR.2016.521
42. Simons T, Lee D-J (2019) A review of binarized neural networks. Electronics 8(6):661. https://doi.org/10.3390/electronics8060661
43. Bhattacharya S, Lane ND (2016) Sparsification and separation of deep learning layers for constrained resource inference on wearables. Paper presented at the Proceedings of the 14th ACM conference on embedded networked sensor systems (SenSys), Stanford, CA, USA. https://doi.org/10.1145/2994551.2994564
44. Fedorov I, Adams RP, Mattina M, Whatmough PN (2019) SpArSe: sparse architecture search for CNNs on resource-constrained microcontrollers. arXiv preprint https://arxiv.org/abs/1905.12107
45. Yang TJ, Chen YH, Sze V (2017) Designing energy-efficient convolutional neural networks using energy-aware pruning. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), 21–26 July 2017, pp 6071–6079. https://doi.org/10.1109/CVPR.2017.643
46. Zhang L, Song J, Gao A, Chen J, Bao C, Ma K (2019) Be your own teacher: improve the performance of convolutional neural networks via self distillation. In: 2019 IEEE/CVF International conference on computer vision (ICCV), 27 Oct–2 Nov 2019, pp 3712–3721. https://doi.org/10.1109/ICCV.2019.00381
47. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint https://arxiv.org/abs/1704.04861
48. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-CC (2018) MobileNetV2: inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 4510–4520. https://doi.org/10.1109/CVPR.2018.00474
49. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and < 0.5 MB model size. arXiv preprint https://arxiv.org/abs/1602.07360
50. He Y, Liu X, Zhong H, Ma Y (2019) AddressNet: shift-based primitives for efficient convolutional neural networks. In: 2019 IEEE Winter conference on applications of computer vision (WACV), 7–11 Jan 2019, pp 1213–1222. https://doi.org/10.1109/WACV.2019.00134
51. Mehta S, Rastegari M, Shapiro L, Hajishirzi H (2019) ESPNetv2: a light-weight, power efficient, and general purpose convolutional neural network. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 15–20 June 2019, pp 9182–9192. https://doi.org/10.1109/CVPR.2019.00941
52. Ma N, Zhang X, Zheng H-T, Sun J (2018) ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer vision—ECCV 2018. Springer International Publishing, Cham, pp 122–138. https://doi.org/10.1007/978-3-030-01264-9_8
53. Xie X, Zhou Y, Kung SY (2020) Exploring highly efficient compact neural networks for image classification. In: 2020 IEEE International conference on image processing (ICIP), 25–28 Oct 2020, pp 2930–2934. https://doi.org/10.1109/ICIP40778.2020.9191334
54. Zhang X, Zhou X, Lin M, Sun J (2018) ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 6848–6856. https://doi.org/10.1109/CVPR.2018.00716
55. Huang G, Liu S, Maaten Lvd, Weinberger KQ (2018) CondenseNet: an efficient DenseNet using learned group convolutions. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 2752–2761. https://doi.org/10.1109/CVPR.2018.00291
56. Deng L, Li G, Han S, Shi L, Xie Y (2020) Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc IEEE 108(4):485–532. https://doi.org/10.1109/JPROC.2020.2976475
57. Qin H, Gong R, Liu X, Bai X, Song J, Sebe N (2020) Binary neural networks: a survey. Pattern Recogn 105:107281. https://doi.org/10.1016/j.patcog.2020.107281
58. Cheng J, Wang P-s, Li G, Hu Q-h, Lu H-q (2018) Recent advances in efficient computation of deep convolutional neural networks. Front Inf Technol Electron Eng 19(1):64–77. https://doi.org/10.1631/FITEE.1700789

59. Wu X, Sahoo D, Hoi SCH (2020) Recent advances in deep learning for object detection. Neurocomputing 396:39–64. https://doi.org/10.1016/j.neucom.2020.01.085
60. Chahal K, Dey K (2018) A survey of modern object detection literature using deep learning. arXiv preprint https://arxiv.org/abs/1808.07256
61. Zhao Z, Zheng P, Xu S, Wu X (2019) Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst 30(11):3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865
62. Jiao L, Zhang F, Liu F, Yang S, Li L, Feng Z, Qu R (2019) A survey of deep learning-based object detection. IEEE Access 7:128837–128868. https://doi.org/10.1109/ACCESS.2019.2939201
63. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M (2020) Deep learning for generic object detection: a survey. Int J Comput Vision 128:261–318. https://doi.org/10.1007/s11263-019-01247-4
64. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53(8):5455–5516. https://doi.org/10.1007/s10462-020-09825-6
65. Sultana F, Sufian A, Dutta P (2020) A review of object detection models based on convolutional neural network. In: Mandal JK, Banerjee S (eds) Intelligent computing: image processing based applications. Springer Singapore, Singapore, pp 1–16. https://doi.org/10.1007/978-981-15-4288-6_1
66. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer vision—ECCV 2020. Springer International Publishing, Cham, pp 213–229. https://doi.org/10.1007/978-3-030-58452-8_13
67. Tolstikhin I, Houlsby N, Kolesnikov A, Beyer L, Zhai X, Unterthiner T, Yung J, Steiner A, Keysers D, Uszkoreit J, Lucic M, Dosovitskiy A (2021) MLP-Mixer: an all-MLP architecture for vision. arXiv preprint https://arxiv.org/abs/2105.01601
68. Ullah S, Kim D (2020) Benchmarking Jetson platform for 3D point-cloud and hyper-spectral image classification. In: 2020 IEEE International conference on big data and smart computing (BigComp), 19–22 Feb 2020, pp 477–482. https://doi.org/10.1109/BigComp48618.2020.00-21
69. Qi CR, Litany O, He K, Guibas L (2019) Deep hough voting for 3D object detection in point clouds. In: 2019 IEEE/CVF International conference on computer vision (ICCV), 27 Oct–2 Nov 2019, pp 9276–9285. https://doi.org/10.1109/ICCV.2019.00937
70. Wang Y, Zell A (2021) Yolo+FPN: 2D and 3D fused object detection with an RGB-D camera. In: 2020 25th International conference on pattern recognition (ICPR), 10–15 Jan 2021, pp 4657–4664. https://doi.org/10.1109/ICPR48806.2021.9413066
71. Zhou T, Fan D-P, Cheng M-M, Shen J, Shao L (2021) RGB-D salient object detection: a survey. Comput Vis Media 7(1):37–69. https://doi.org/10.1007/s41095-020-0199-z
72. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: single shot MultiBox detector. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision—ECCV 2016. Springer International Publishing, Cham, pp 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
73. Huang R, Pedoeem J, Chen C (2018) YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In: 2018 IEEE International conference on big data (Big Data), 10–13 Dec 2018, pp 2503–2510. https://doi.org/10.1109/BigData.2018.8621865
74. He W, Huang Z, Wei Z, Li C, Guo B (2019) TF-YOLO: an improved incremental network for real-time object detection. Appl Sci 9(16):3225. https://doi.org/10.3390/app9163225
75. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), 21–26 July 2017, pp 6517–6525. https://doi.org/10.1109/CVPR.2017.690
76. Kyrkou C (2020) YOLOpeds: efficient real-time single-shot pedestrian detection for smart camera applications. IET Comput Vis 14:417–425. https://doi.org/10.1049/iet-cvi.2019.0897
77. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. arXiv preprint https://arxiv.org/pdf/1804.02767.pdf
78. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
79. Shen Z, Liu Z, Li J, Jiang YG, Chen Y, Xue X (2020) Object detection from scratch with deep supervision. IEEE Trans Pattern Anal Mach Intell 42(2):398–412. https://doi.org/10.1109/TPAMI.2019.2922181
80. Zhang S, Wen L, Bian X, Lei Z, Li SZ (2018) Single-shot refinement neural network for object detection. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 4203–4212. https://doi.org/10.1109/CVPR.2018.00442
81. Law H, Deng J (2020) CornerNet: detecting objects as paired keypoints. Int J Comput Vis 128(3):642–656. https://doi.org/10.1007/s11263-019-01204-1
82. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on computer vision and pattern recognition, 23–28 June 2014, pp 580–587. https://doi.org/10.1109/CVPR.2014.81
83. Li Z, Peng C, Yu G, Zhang X, Deng Y, Sun J (2017) Light-head R-CNN: in defense of two-stage object detector. arXiv preprint https://arxiv.org/abs/1711.07264
84. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), 27–30 June 2016, pp 779–788. https://doi.org/10.1109/CVPR.2016.91
85. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
86. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), 21–26 July 2017, pp 936–944. https://doi.org/10.1109/CVPR.2017.106
87. Kong T, Yao A, Chen Y, Sun F (2016) HyperNet: towards accurate region proposal generation and joint object detection. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), 27–30 June 2016, pp 845–853. https://doi.org/10.1109/CVPR.2016.98
88. Newell A, Yang K, Deng J (2016) Stacked Hourglass networks for human pose estimation. In: 2016 European conference on computer vision (ECCV). Springer International Publishing, Cham, pp 483–499. https://doi.org/10.1007/978-3-319-46484-8_29
89. Li Z, Zhou F (2017) FSSD: feature fusion single shot Multibox detector. arXiv preprint https://arxiv.org/abs/1712.00960
90. Li Z, Peng C, Yu G, Zhang X, Deng Y, Sun J (2018) DetNet: design backbone for object detection. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer vision—ECCV 2018. Springer International Publishing, Cham, pp 339–354

91. Qin Z, Li Z, Zhang Z, Bao Y, Yu G, Peng Y, Sun J (2019) ThunderNet: towards real-time generic object detection on mobile devices. In: 2019 IEEE/CVF International conference on computer vision (ICCV), 27 Oct–2 Nov 2019, pp 6717–6726. https://doi.org/10.1109/ICCV.2019.00682
92. Chen D, Shen H (2020) MAOD: an efficient anchor-free object detector based on MobileDet. IEEE Access 8:86564–86572. https://doi.org/10.1109/ACCESS.2020.2992516
93. Law H, Teng Y, Russakovsky O, Deng J (2020) CornerNet-Lite: efficient keypoint based object detection. In: 31st British machine vision conference 2020 (BMVC), Virtual Event, UK, 7–10 Sept 2020
94. Tang Q, Li J, Shi Z, Hu Y (2020) Lightdet: a lightweight and accurate object detection network. In: ICASSP 2020–2020 IEEE International conference on acoustics, speech and signal processing (ICASSP), 4–8 May 2020, pp 2243–2247. https://doi.org/10.1109/ICASSP40776.2020.9054101
95. Li Y, Li JJ, Lin W, Li JJ (2018) Tiny-DSOD: lightweight object detection for resource-restricted usages. In: 29th British machine vision conference (BMVC), 2018
96. Wong A, Shafiee MJ, Li F, Chwyl B (2018) Tiny SSD: a tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In: 2018 15th Conference on computer and robot vision (CRV), 8–10 May 2018, pp 95–101. https://doi.org/10.1109/CRV.2018.00023
97. Azimi SM (2019) ShuffleDet: real-time vehicle detection network in on-board embedded UAV imagery. In: Leal-Taixé L, Roth S (eds) Computer vision—ECCV 2018 workshops, 2019. Springer International Publishing, Cham, pp 88–99. https://doi.org/10.1007/978-3-030-11012-3_7
98. Mao QC, Sun HM, Liu YB, Jia RS (2019) Mini-YOLOv3: real-time object detector for embedded applications. IEEE Access 7:133529–133538. https://doi.org/10.1109/ACCESS.2019.2941547
99. Chiu YC, Tsai CY, Ruan MD, Shen GY, Lee TT (2020) Mobilenet-SSDv2: an improved object detection model for embedded systems. In: 2020 International conference on system science and engineering (ICSSE), 31 Aug–3 Sept 2020, pp 1–5. https://doi.org/10.1109/ICSSE50014.2020.9219319
100. Oh S, You J-H, Kim Y-K (2020) FRDet: balanced and lightweight object detector based on fire-residual modules for embedded processor of autonomous driving. arXiv preprint https://arxiv.org/abs/2011.08061
101. Chen C, Liu M, Meng X, Xiao W, Ju Q (2020) RefineDetLite: a lightweight one-stage object detection framework for CPU-only devices. In: 2020 IEEE/CVF Conference on computer vision and pattern recognition workshops (CVPRW), 14–19 June 2020, pp 2997–3007. https://doi.org/10.1109/CVPRW50498.2020.00358
102. Ling H, Zhang L, Qin Y, Shi Y, Wu L, Chen J, Zhang B (2020) BMNet: a reconstructed network for lightweight object detection via branch merging. In: 2019 30th British machine vision conference (BMVC), 2019, pp 1–12
103. Fang W, Wang L, Ren P (2020) Tinier-YOLO: a real-time object detection method for constrained environments. IEEE Access 8:1935–1944. https://doi.org/10.1109/ACCESS.2019.2961959
104. Han J, Yang Y (2021) L-Net: lightweight and fast object detector-based ShuffleNetV2. J Real-Time Image Proc. https://doi.org/10.1007/s11554-021-01145-4
105. Zhou Q, Wang J, Liu J, Li S, Ou W, Jin X (2021) RSANet: towards real-time object detection with residual semantic-guided attention feature pyramid network. Mobile Netw Appl 26(1):77–87. https://doi.org/10.1007/s11036-020-01723-z
106. Wang C-Y, Bochkovskiy A, Liao H-YM (2021) Scaled-yolov4: scaling cross stage partial network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp 13029–13038
107. Wang RJ, Li X, Ao S, Ling CX (2018) Pelee: a real-time object detection system on mobile devices. In: 6th International conference on learning representations, ICLR 2018—workshop track proceedings, Montréal, Canada, 2018. Curran Associates Inc., pp 1963–1972
108. Liau HF, Yamini N, Wong YL (2018) Fire SSD: wide fire modules based single shot detector on edge device. arXiv preprint https://arxiv.org/abs/1806.05363
109. Gong H, Li H, Xu K, Zhang Y (2019) Object detection based on improved YOLOv3-tiny. In: 2019 Chinese automation congress (CAC), 22–24 Nov 2019, pp 3240–3245. https://doi.org/10.1109/CAC48633.2019.8996750
110. Jiun-In G, Chi-Chi T, Ching-Kan T (2019) Pvalite CLN: lightweight object detection with classfication and localization network. In: 2019 32nd IEEE International system-on-chip conference (SOCC), 3–6 Sept 2019, pp 118–121. https://doi.org/10.1109/SOCC46988.2019.1570561207
111. Ghiasi G, Lin TY, Le QV (2019) NAS-FPN: learning scalable feature pyramid architecture for object detection. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 15–20 June 2019, pp 7029–7038. https://doi.org/10.1109/CVPR.2019.00720
112. Howard A, Sandler M, Chen B, Wang W, Chen L, Tan M, Chu G, Vasudevan V, Zhu Y, Pang R, Adam H, Le Q (2019) Searching for MobileNetV3. In: 2019 IEEE/CVF International conference on computer vision (ICCV), 27 Oct–2 Nov 2019, pp 1314–1324. https://doi.org/10.1109/ICCV.2019.00140
113. Sun Y, Wang C, Qu L (2019) An object detection network for embedded system. In: 2019 IEEE International conferences on ubiquitous computing & communications (IUCC) and data science and computational intelligence (DSCI) and smart computing, networking and services (SmartCNS), 21–23 Oct 2019, pp 506–512. https://doi.org/10.1109/IUCC/DSCI/SmartCNS.2019.00110
114. Zhao H, Zhou Y, Zhang L, Peng Y, Hu X, Peng H, Cai X (2020) Mixed YOLOv3-LITE: a lightweight real-time object detection method. Sensors (Switzerland) 20:1861. https://doi.org/10.3390/s20071861
115. Fan B, Chen Y, Qu J, Chai Y, Xiao C, Huang P (2019) FFBNet: lightweight backbone for object detection based feature fusion block. In: 2019 IEEE International conference on image processing (ICIP), 22–25 Sept 2019, pp 3920–3924. https://doi.org/10.1109/ICIP.2019.8803683
116. Hu L, Li Y (2021) Micro-YOLO: exploring efficient methods to compress CNN based object detection model. In: Proceedings of the 13th International conference on agents and artificial intelligence (ICAART), 2021. SciTePress, pp 151–158. https://doi.org/10.5220/0010234401510158
117. Guo S, Liu Y, Ni Y, Ni W (2021) Lightweight SSD: real-time lightweight single shot detector for mobile devices. In: Proceedings of the 16th international joint conference on computer vision, imaging and computer graphics theory and applications (VISIGRAPP), 2021, pp 25–35. https://doi.org/10.5220/0010188000250035
118. Wu B, Wan A, Iandola F, Jin PH, Keutzer K (2017) SqueezeDet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: 2017 IEEE Conference on computer vision and pattern recognition workshops (CVPRW), 21–26 July 2017, pp 446–454. https://doi.org/10.1109/CVPRW.2017.60
119. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-ResNet and the impact of residual connections on learning. Paper presented at the Proceedings of the thirty-first

AAAI conference on artificial intelligence, San Francisco, California, USA
120. YOLO: Real-time object detection. https://pjreddie.com/darknet/yolo/. Accessed 2021-03-09
121. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 7132–7141. https://doi.org/10.1109/CVPR.2018.00745
122. Hong S, Roh B, Kim K-H, Cheon Y, Park M (2016) PVANet: lightweight deep neural networks for real-time object detection. arXiv preprint https://arxiv.org/abs/1611.08588v2
123. Woo S, Park J, Lee J-Y, Kweon IS (2018) CBAM: convolutional block attention module. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer vision—ECCV 2018. Springer International Publishing, Cham, pp 3–19. https://doi.org/10.1007/978-3-030-01234-2_1
124. Gao S-H, Cheng M-M, Zhao K, Zhang X-Y, Yang M-H, Torr P (2021) Res2Net: a new multi-scale backbone architecture. IEEE Trans Pattern Anal Mach Intell 43(2):652–662. https://doi.org/10.1109/tpami.2019.2938758
125. Wang C, Liao HM, Wu Y, Chen P, Hsieh J, Yeh I (2020) CSPNet: a new backbone that can enhance learning capability of CNN. In: 2020 IEEE/CVF Conference on computer vision and pattern recognition workshops (CVPRW), 14–19 June 2020, pp 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203
126. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, Liu D, Mu Y, Tan M, Wang X, Liu W, Xiao B (2020) Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell 43:3349–3364. https://doi.org/10.1109/TPAMI.2020.2983686
127. Tian Z, Shen C, Chen H, He T (2019) FCOS: fully convolutional one-stage object detection. In: 2019 IEEE/CVF International conference on computer vision (ICCV), 27 Oct–2 Nov 2019, pp 9626–9635. https://doi.org/10.1109/ICCV.2019.00972
128. Wu B, Wan A, Yue X, Jin P, Zhao S, Golmant N, Gholaminejad A, Gonzalez J, Keutzer K (2018) Shift: a zero FLOP, zero parameter alternative to spatial convolutions. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018. IEEE Computer Society, pp 9127–9135. https://doi.org/10.1109/CVPR.2018.00951
129. Lee Y, Hwang J, Lee S, Bae Y, Park J (2019) An energy and GPU-computation efficient backbone network for real-time object detection. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition workshops (CVPRW), 16–17 June 2019, pp 752–760. https://doi.org/10.1109/CVPRW.2019.00103
130. Zhang D (2018) clcNet: improving the efficiency of convolutional neural network using channel local convolutions. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 7912–7919. https://doi.org/10.1109/CVPR.2018.00825
131. Gao H, Wang Z, Cai L, Ji S (2021) ChannelNets: compact and efficient convolutional neural networks via channel-wise convolutions. IEEE Trans Pattern Anal Mach Intell 43(8):2570–2581. https://doi.org/10.1109/TPAMI.2020.2975796
132. Xiong Y, Kim HJ, Hedau V (2019) ANTNets: mobile convolutional neural networks for resource efficient image classification. arXiv preprint https://arxiv.org/abs/1904.03775
133. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020) GhostNet: more features from cheap operations. In: 2020 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 13–19 June 2020, pp 1577–1586. https://doi.org/10.1109/CVPR42600.2020.00165
134. Wang W, Hu Y, Zou T, Liu H, Wang J, Wang X (2020) A new image classification approach via improved MobileNet models with local receptive field expansion in shallow layers. Comput Intell Neurosci. https://doi.org/10.1155/2020/8817849
135. Mehta S, Hajishirzi H, Rastegari M (2020) DiCENet: dimension-wise convolutions for efficient networks. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2020.3041871
136. Gholami A, Kwon K, Wu B, Tai Z, Yue X, Jin P, Zhao S, Keutzer K (2018) SqueezeNext: hardware-aware neural network design. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition workshops (CVPRW), 18–22 June 2018, pp 1719–1728. https://doi.org/10.1109/CVPRW.2018.00215
137. Huang G, Liu Z, Maaten LVD, Weinberger KQ (2017) Densely connected convolutional networks. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), 21–26 July 2017, pp 2261–2269. https://doi.org/10.1109/CVPR.2017.243
138. Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, 18–23 June 2018, pp 8697–8710. https://doi.org/10.1109/CVPR.2018.00907
139. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV (2019) MnasNet: platform-aware neural architecture search for mobile. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 15–20 June 2019, pp 2815–2823. https://doi.org/10.1109/CVPR.2019.00293
140. Stamoulis D, Ding R, Wang D, Lymberopoulos D, Priyantha B, Liu J, Marculescu D (2020) Single-path NAS: designing hardware-efficient ConvNets in less than 4 h. In: Brefeld U, Fromont E, Hotho A, Knobbe A, Maathuis M, Robardet C (eds) Machine learning and knowledge discovery in databases. Springer International Publishing, Cham, pp 481–497. https://doi.org/10.1007/978-3-030-46147-8_29
141. Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K (2019) FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 15–20 June 2019, pp 10726–10734. https://doi.org/10.1109/CVPR.2019.01099
142. Guo Z, Zhang X, Mu H, Heng W, Liu Z, Wei Y, Sun J (2020) Single path one-shot neural architecture search with uniform sampling. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer vision—ECCV 2020. Springer International Publishing, Cham, pp 544–560. https://doi.org/10.1007/978-3-030-58517-4_32
143. Cai H, Wang T, Wu Z, Wang K, Lin J, Han S (2019) On-device image classification with proxyless neural architecture search and quantization-aware fine-tuning. In: 2019 IEEE/CVF International conference on computer vision workshop (ICCVW), 27–28 Oct 2019, pp 2509–2513. https://doi.org/10.1109/ICCVW.2019.00307
144. Wan A, Dai X, Zhang P, He Z, Tian Y, Xie S, Wu B, Yu M, Xu T, Chen K, Vajda P, Gonzalez JE (2020) FBNetV2: differentiable neural architecture search for spatial and channel dimensions. In: 2020 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), 13–19 June 2020, pp 12962–12971. https://doi.org/10.1109/CVPR42600.2020.01298