AI

MDPI

Review

Sustainable Machine Vision for Industry 4.0: A Comprehensive Review of Convolutional Neural Networks and Hardware Accelerators in Computer Vision

Muhammad Hussain

Department of Computer Science, University of Huddersfield, Huddersfield HD1 3DH, UK; m.hussain@hud.ac.uk

Abstract: As manifestations of Industry 4.0. become visible across various applications, one key and opportune area of development are quality inspection processes and defect detection. Over the last decade, computer vision architectures, in particular, object detectors have received increasing attention from the research community, due to their localisation advantage over image classification. However, for these architectural advancements to provide tangible solutions, they must be optimised with respect to the target hardware along with the deployment environment. To this effect, this survey provides an in-depth review of the architectural progression of image classification and object detection architectures with a focus on advancements within Artificially Intelligent accelerator hardware. This will provide readers with an understanding of the present state of architecture–hardware integration within the computer vision discipline. The review also provides examples of the industrial implementation of computer vision architectures across various domains, from the detection of fabric defects to pallet racking inspection. The survey highlights the need for representative *hardware-benchmarked* datasets for providing better performance comparisons along with envisioning object detection as the primary domain where more research efforts would be focused over the next decade.

Keywords: artificial intelligence; computer vision; hardware advancements; image classification; object detection

Citation: Hussain, M. Sustainable Machine Vision for Industry 4.0: A Comprehensive Review of Convolutional Neural Networks and Hardware Accelerators in Computer Vision. *AI* **2024**, *5*, 1324–1356. https://doi.org/10.3390/ai5030064

Academic Editor: Arslan Munir and Miguel Angel Cazorla

Received: 16 May 2024
Revised: 22 July 2024
Accepted: 25 July 2024
Published: 1 August 2024

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

Artificial Intelligence (AI) has experienced significant advancements over the past decade, and one notable advance is in deep learning [1]. The manifestation of AI-integrated applications based on machine learning (ML) and deep learning (DL) approaches can be observed across various domains from manufacturing [2] and renewable energy [3] to security [4] and healthcare [5,6]. Although conventional ML/DL approaches are well suited for numerical and textual data due to their structured nature and established preprocessing techniques, they are not the standard choice for image-based applications because images are high-dimensional and contain complex spatial hierarchies that require specialised handling. Image data can consist of millions of pixels, making the feature space vastly larger, and capturing meaningful patterns necessitates understanding spatial relationships and local features, which conventional algorithms struggle to achieve. Amongst the various subdomains attributed to AI, Computer Vision (CV) is one of the front-runners when it comes to a practical demonstration of AI within the corporate world. Simulating the visual perception ability of humans, CV provides computers with artificial visual perception, allowing them to perceive objects within the physical environment and suggest actions accordingly.

Modelling CV systems require the architectural design and the development of algorithms that contain the ability to perceive, analyse, comprehend and classify, according to the requirements of the application [7]. Researchers have segmented CV into three distinct

inter-domains: (1) image classification [8], for determining the presence of a particular object in an image; (2) object detection [9], for the determination of multiple objects along with the location details for each object; and (3) object segmentation [10], for the pixel-level segmenting of the object(s) of interest within a single image or a video stream. Prior to the advent of deep learning, for several decades, researchers focused their efforts on common object perception, via conventional image processing techniques, namely hand-crafted feature descriptors, albeit with limited success given the complex nature and high variations found even with the most common objects [11]. The success of artificial neural networks (ANN), when trained on large datasets, without the need for manual feature engineering, demonstrated promise in the CV domain [12]. It was thought that by pursuing the adoption of ANNs, large datasets containing millions of images could be used in the training of convolutional neural networks (CNNs). These are a variant of ANN, capable of enhanced perception, enabling better visual performance, robustness to variations and without the need for manual feature descriptors [13]. In further advancements, IMAGENET [14] was brought into existence by Feifei in 2009, the largest image classification database, containing 15 million images spanning 22,000 classes. Following on, CNN-based architectures were developed and benchmarked on the IMAGENET database to gain an understanding of their capabilities.

CNNs are unrivalled in their claim as the de facto architecture for vision-based applications. Assisted by large training datasets, CNNs focus on the transformation of high-dimensional input data into low-dimensional but highly abstracted, semantic outputs for accurate classification. The advancements in CV are self-evident with regard to their potential in enhancing operations across various sectors. However, as researchers aim to transfer CV architectures from academic laboratories to manufacturing facilities and production lines, incompatibilities with existing application requirements are becoming self-evident with rising severity. This includes aspects such as high computational cost, increased power consumption and high processing requirements from central processing units (CPUs) and graphical processing units (GPUs) [15]. These incompatibilities have rerouted the research direction from a one-dimensional criterion, i.e., high accuracy, to multi-dimensional design considerations including computational complexities, architectural footprint and energy efficiency.

1.1. Survey Objective

This paper presents a comprehensive survey of key advancements in the field of computer vision (CV). This survey focuses on two main aspects: (1) breakthroughs in proposed CV architectures, particularly internal architectural advancements that enable high-performing networks, and (2) hardware advancements aimed at addressing deployment issues faced by CV architectures, such as inference speeds, processing time, intra-connectivity, and power consumption. In terms of architectural advancements, this survey provides an in-depth analysis of the image classification and object detection domains. It highlights significant breakthroughs in these areas, emphasising the key contributions that have enabled advancements in CV algorithms.

Regarding hardware deployment options, this survey explores notable options including GPUs, Field-Programmable Gate Arrays (FPGAs) [16], and Application-Specific Integrated Circuits (ASICs) [17]. These hardware solutions aim to facilitate the smoother integration of CV frameworks into existing business processes.

It is worth noting that while there are a significant number of works in the literature available on the development of Convolutional Neural Networks (CNNs), research specifically focusing on CV architecture design and deployment strategies is relatively limited and often task-specific [6]. This survey addresses this gap by presenting a comprehensive review that encompasses both architectural and hardware options. By offering insights into the latest advancements and options in deployable CV development frameworks, this survey equips researchers with the necessary knowledge to stay updated and make informed decisions in their own CV-related research.

1.2. Existing Surveys

Several significant surveys have delved into the adoption of CNNs in the domain of industrial surface defect detection. Shengxiang et al. [18] provide an in-depth review starting with the delineation of computer vision domains before transitioning to implementation scenarios of these domains for industrial surface defect detection, while authors in [19] present a lightweight overview on CNNs before focusing their efforts on the segmentation of industrial surface level defects. Despite shedding light on the potential of CNNs for industrial surface-level defect detection, these surveys overlook the critical relationship between the utilised architectures and the hardware accelerators essential for successful real-world deployment.

Addressing this gap, authors in [20] investigated efficient CNN implementation across various hardware platforms, albeit from a generic perspective without focusing on the industrial requirements of the domains in question. A broader survey conducted by Capra et al. [21] explored the integration of computer vision architectures with hardware accelerators via object detection, and whilst this provided a detailed analysis of the connection between CNNs and hardware accelerators, it lacked the specific connection to industrial surface-level defect detection. This review stands out by presenting a comprehensive and up-to-date survey that specifically links object detection, hardware accelerators, and industrial defect detection within the overall CNN ecosystem focusing on industrial requirements such as low latency, high inference rates and optimal accuracy. It serves as a valuable resource for researchers, practitioners and industry professionals aiming to harness its collective potential for robust and efficient industrial defect detection systems.

1.3. Establishing the Link between CNNs and Hardware

The rapid advancement of CNN architectures has transformed the field of CV, facilitating significant breakthroughs in automated visual applications. However, the successful deployment of CNN models transcends the design of sophisticated architectures. Hardware considerations play a decisive role in determining the feasibility, efficiency and real-world applicability of CNNs. Hence, it is important to establish the connection between the development of CNN architectures and the corresponding hardware considerations before delving into the details of each.

Computational Requirements: CNN networks are computationally demanding, requiring substantial processing power to perform complex operations. As CNNs become larger in terms of depth and breadth to capture more complicated features, the computational requirements grow exponentially. Hence, hardware considerations, such as the selection of processors/accelerators, must be carefully evaluated to guarantee the efficient execution and real-time performance of CNN models. Specialised hardware, including GPUs, FPGAs or Application-Specific Integrated Circuits (ASICs), can provide the necessary computational power to accelerate CNN training and inferencing processes.

Memory Requirements: CNNs usually require substantial memory resources for storing model intermediate feature maps and inferencing weights. The size of these models, measured in terms of the number of parameters, has seen exponential growth with the advancement of deeper architectures. Hardware accelerators with sufficient memory capacity and bandwidth are critical to facilitate the memory requirements of CNNs.

Energy Efficiency: The energy efficiency of CNNs is a critical consideration, particularly for applications deployed on embedded systems and resource-constrained edge devices. Hardware accelerators, designed explicitly for CNN computations, aim to optimise power consumption while delivering real-time inference speeds [22]. Compression techniques like quantisation, pruning, and efficient memory access patterns can be deployed to reduce the energy footprint of CNN architectures. Hardware platforms with low power consumption, such as dedicated neural network accelerators or low-power edge devices, enable the energy-efficient deployment of CNN architectures [23].

Scalability and Parallelism: CNN architectural scalability refers to efficiently utilising multiple hardware resources for processing large amounts of data or performing parallel

computations. Hardware platforms offering parallel processing capabilities, such as GPUs, allow for faster training and inference speeds by exploiting parallelism within the internal architecture of CNNs. Additionally, developments in hardware architecture, such as systolic arrays and tensor processing units (TPUs), facilitate the effective parallel execution of CNN computations, further improving scalability.

Deployment Flexibility: Hardware considerations can impact the flexibility in the deployment of CNN architectures. Different application domains may require distinct hardware platforms depending on factors like weight, power, size and cost constraints. For instance, GPU-based systems are appropriate for high-performance applications but necessarily feasible when it comes to resource-constrained environments, whilst FPGA-based solutions provide a high degree of reconfigurability, enabling custom hardware acceleration for specific CNN architectures. ASICs, while presenting the highest performance and energy efficiency, induce higher development costs and limited flexibility.

The designing of CNN architectures and hardware considerations are closely intertwined, with each influencing the other. Hardware advancements stimulate the design of more sophisticated CNN architectures by granting the necessary computational capabilities and memory resources. Conversely, novel CNN architectures inspire the development of specialised hardware-enhanced architectures for deep learning tasks [24]. This iterative process promotes advancements in both the algorithmic and hardware spheres, fostering innovation and progress in computer vision applications.

1.4. Linking Industrial IoT, CNNs and Hardware Accelerators

The connection between Industrial IoT (IIoT), Convolutional Neural Networks (CNNs) and Hardware Accelerators (HAs) is of paramount importance in transforming various aspects of industrial operations and manufacturing processes.

Industrial IoT and CNNs: IIoT involves the amalgamation of sensors, devices and machinery in industrial settings to collect and exchange data, creating a connected ecosystem. These data can include information about machine performance, environmental conditions, production metrics and more. CNNs, with their proficiency in visual data administering, play a focal role in processing images and videos captured by IIoT devices.

For instance, CNNs can be exploited for quality control in industrial manufacturing by inspecting for products with defects in real time [25]. Additionally, they can be deployed for predictive maintenance, where they analyse visual image data to detect potential asset failures before they occur, thus reducing downtime and optimising productivity.

Hardware Accelerators for CNNs in IIoT: Given the computationally demanding nature of CNNs, notably for complex industrial applications, hardware accelerators become vital for ensuring the efficient and real-time inferencing of visual data. In industrial settings, where power consumption, low latency and real-time response is critical, hardware accelerators play a crucial role in handling the computational load of CNNs. By integrating hardware accelerators like GPUs, FPGAs or ASICs (Application-Specific Integrated Circuits), IIoT devices can execute advanced CNN tasks locally, without relying solely on cloud-based processing, thus reducing dependence on internet connectivity and minimising the risk of high latency.

In brief, the incorporation of CNNs with hardware accelerators in the Industrial IoT domain empowers smart manufacturing and real-time analysis, leading to enhanced productivity, reduced costs and increased safety in industrial settings, thus representing a powerful convergence of leading edge technology for facilitating the next wave of visual-based industrial automation and efficiency.

1.5. Structure

The structure of this review is presented in Figure 1. Section 2 provides an overview of image classification, beginning with conventional image processing techniques and then transitioning to convolutional neural networks. In Section 3, we delve into object detection, highlighting significant advancements in this domain, particularly the architectural

contributions of widely used object detectors. We focus on exploring the various YOLO variants, known for their efficient nature and real-time inference capabilities. Moving on to Section 4, we delve into the realm of industrial IoT acceleration for visual inspection, primarily examining three leading technologies: graphical processing units (GPUs), Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). Section 5 showcases industrial quality inspection applications based on the software and hardware mechanisms discussed in the previous sections, focusing on defect detection in fabric and photovoltaic manufacturing, as well as the emerging field of automated pallet racking inspection. Lastly, in Section 6, we discuss the challenges and future scope for integrating CNN with hardware solutions. Section 7 serves as the conclusion of this article.

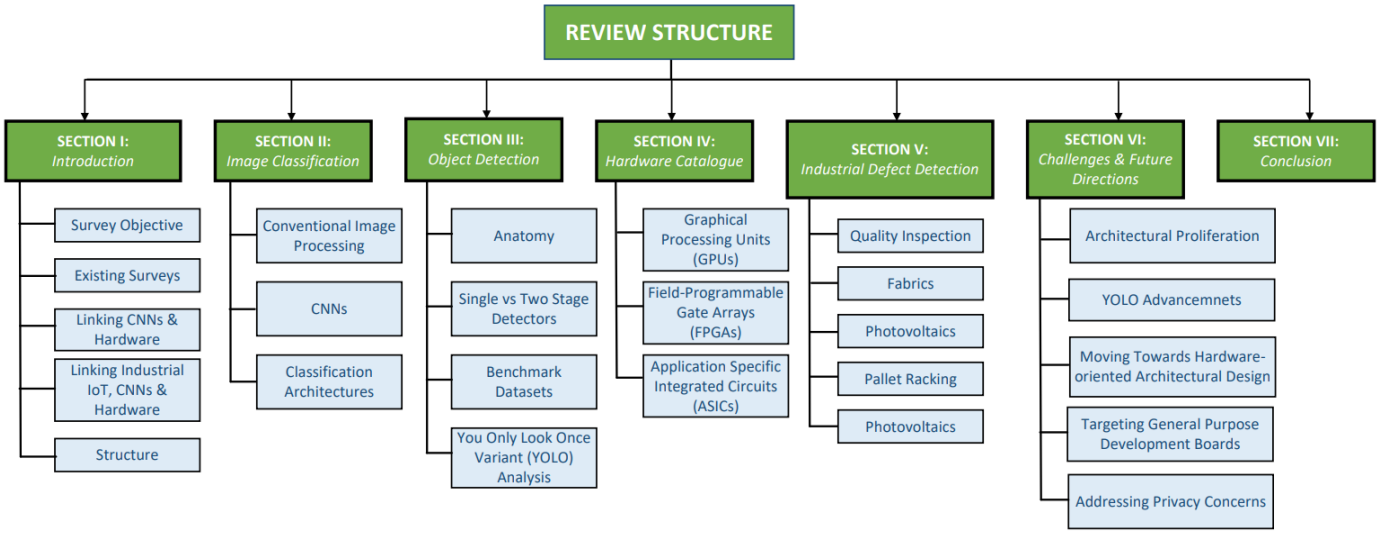

Figure 1. Visual structure of this review.

2. Image Classification

2.1. Conventional Image Processing

In the field of image processing, conventional techniques have long been employed to tackle various challenges related to image analysis, feature extraction and pixel accentuation. These traditional methods, based on using algorithms and handcrafted features, have shown considerable success in a wide range of applications. However, there are clear limitations attributed to conventional image processing that have been addressed through deep learning. Some of the key limitations of conventional image processing are presented as follows:

Manual Feature Engineering: Conventional image processing techniques frequently require manual feature engineering, where domain experts design task-specific feature descriptors. This procedure can be labour-intensive, time-consuming, costly and heavily dependent on expert knowledge. In contrast, CNNs eliminate the need for explicit feature engineering by automatically ascertaining features directly from the images, reducing reliance on human expertise.

Limited Generalisation Capacity: Conventional image processing algorithms can find it difficult to generalise on new and unseen data, as they rely on assumptions about image statistics. This is not always the case in complex and diverse real-world scenarios. Conversely, CNNs are able to learn from large-scale datasets, capturing intricate patterns that lead to improved generalisation on unseen data.

Handling Complex Variations: Conventional image processing can face challenges when it comes to handling complex variations in images, such as fluctuations in lighting conditions, orientations and occlusions. These variations may warrant the designing of specific feature descriptors tailored to each scenario. Meanwhile, CNNs, with their hierarchical

composition and data-driven learning, have demonstrated greater robustness and adaptability when it comes to handling such complex variations without explicit modifications.

Limited Representation Learning: Traditional image processing techniques often rely on handcrafted feature extractors that may not entirely capture the complexity and richness of visual information. These manual extractors may be limited in their ability to represent intricate patterns and semantic information embedded within the images. CNNs, by contrast, excel in representation learning by automatically mining hierarchical features from images, enabling them to acquire high-level semantics and intricate details simultaneously.

Scalability: Conventional image processing mechanisms may struggle to scale effectively as the size and variance of datasets increase or where more challenging computational requirements need to be followed. As datasets increase in size, computational demands also grow, and CNNs are able to leverage parallel processing on specialised hardware such as GPUs to efficiently process vast amounts of data, leading to significant improvements in scalability and performance.

2.2. Convolutional Neural Networks

Image classification in the context of CV refers to the ability of the computer/system to perceive the presence of a particular object in a given image. Before the advent of CNN-inspired architectures, research on image classification orbited around the development of scale-invariant feature descriptors (SIFT [26], GIST [27] and HOG [28]), feature representations (Fisher Kernel [29], bag-of-features [30]) and classifiers (SVM [31]). However, manual feature engineering resulted in feature descriptors and classifiers that were unable to generalise to natural variations such as complex backgrounds, light intensity, orientations, varying colours and occlusions. A breakthrough was achieved during the IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) in 2012 when AlexNet [32] secured first position by a significant margin over runners-up architectures based on SIFT and Fisher Vectors. This remarkable milestone, demonstrated the superiority and robustness of CNNs in feature engineering and classification over the long reining conventional approaches, ending the trough period for neural networks. A general CNN, denoted by C, consisting of n architectural parameters $\lambda_1, \ldots, \lambda_n$, with decision spaces $\lambda_1, \ldots, \lambda_n$, respectively, focuses on optimising the problem formulated as

$$\{\arg\min_{\lambda} L(C_\lambda, D_{\text{train}}, D_{\text{valid}}) \quad \text{s.t.} \quad \lambda \in \Lambda\} \tag{1}$$

where $C_{(\lambda,\cdot)}$ denotes the CNN, C adopting the architectural parameter setting λ, $\lambda = \{\lambda_1, \ldots, \lambda_n\}$, $\Lambda = \Lambda_1 \times \ldots \times \Lambda_n$, and $\mathcal{L}(\cdot)$ computes the performance of $C_{(\lambda,\cdot)}$ on the validation dataset D_{valid} post-training on D_{train} (training dataset). When dealing with classification, $\mathcal{L}(\cdot)$ focuses on the computation of the classification error with respect to the tasks for which C is applied.

The fundamental structure of a typical CNN architecture consists of several components. First, a convolutional layer receives the input image and generates a set of resultant feature maps, i.e., a 2D matrix of neurons for a single feature map, and a 3D volume of resultant feature maps for several input feature maps [33]. Activation functions, the most popular being Rectified Linear Unit (ReLu), provide non-linearity, followed by pooling layers for the removal of positional dependencies and finally connected layers play the intermediary role between the sparse convolutions and the output nodes [34]. The stacking of various layers coupled with performing multi-scale convolutions and regularisation strategies facilitate the discovery of highly abstract, semantically rich and discriminative features.

AlexNet [32] was one of the first techniques and, as mentioned earlier, initiated the way for the deeper architectural network. AlexNet consists of eight convolutional layers, followed by three pooling and three fully connected layers. It featured the successful implementation of the ReLu activation function, a mathematically simpler function compared to its predecessors Sigmoid [35] and TanH [36]. AlexNet contained 60 Million parameters. The race of claiming the superior CNN architecture had begun, and more breakthroughs

appeared at the 2014 ILSVRC. VGG-Net [37] and GoogleNet [38], based on the principle of deeper architectural depth for higher accuracy, secured second and first position, respectively, at the 2014 competition. For enhanced performance, VGGNet deployed a stacking strategy, whereby 3 × 3-based convolutional filters and 2 × 2 max-pooling layers were repeatedly stacked. GoogleNet, took a different approach, by completely eliminating the fully connected layers and focusing on the optimisation of the sparse matrices. Although GoogleNet contains 22 layers, due to its unwillingness to implement fully connected layers, its floating-point operations and parameter count were lighter than both AlexNet and VGG-Net. Simonyan et al. [37], focused the VGG-Net architecture on the stacking approach, essentially initialising and stacking several convents, as presented in Table 1, which is evaluating the layer stacking performance. As more researchers focused their attention, it was quickly discovered that simply increasing the depth of architecture via continuous stacking cannot guarantee higher accuracy. This is due to the loss of gradient information, also known as vanishing and exploding gradients. This issue was addressed by the winner of ILSVRC 2015, ResNet [39], introducing the concept of skip connections between residual blocks, allowing the conservation of gradient information from previous layers, during the training process. The concept of skip connections between residual blocks facilitated the development of very deep architectures reaching 152 layers i.e., ResNet-152. Guided by the underlying principle of information preservation presented by ResNet, DenseNet [40], progressed further, by sanctioning links between all current and preceding layers. Through concatenation, DenseNet promoted the re-usability of features throughout the architectural layers, providing high performance.

Table 1. VGG-Net Configuration.

Layer	Output Size	Filter Size/Stride
Input	224 × 224 × 3	-
Conv1-64	224 × 224 × 64	3 × 3/1
Conv2-64	224 × 224 × 64	3 × 3/1
MaxPool1	112 × 112 × 64	2 × 2/2
Conv3-128	112 × 112 × 128	3 × 3/1
Conv4-128	112 × 112 × 128	3 × 3/1
MaxPool2	56 × 56 × 128	2 × 2/2
Conv5-256	56 × 56 × 256	3 × 3/1
Conv6-256	56 × 56 × 256	3 × 3/1
Conv7-256	56 × 56 × 256	3 × 3/1
MaxPool3	28 × 28 × 256	2 × 2/2
Conv8-512	28 × 28 × 512	3 × 3/1
Conv9-512	28 × 28 × 512	3 × 3/1
Conv10-512	28 × 28 × 512	3 × 3/1
MaxPool4	14 × 14 × 512	2 × 2/2
Conv11-512	14 × 14 × 512	3 × 3/1
Conv12-512	14 × 14 × 512	3 × 3/1
Conv13-512	14 × 14 × 512	3 × 3/1
MaxPool5	7 × 7 × 512	2 × 2/2
FC1-4096	4096	-
FC2-4096	4096	-
FC3-1000	1000	-
Softmax	1000	-

In the subsequent years, researchers shifted their focus from developing deeper networks to improving internal block-wise functionality by utilising ResNet/DenseNet as the backbone structure for feature extraction. A manifestation of the active and rapid advancements was observed at the ILSVRC 2017, with SENet [41], securing first place.

SENet unveiled a 'squeeze and excitation' module aimed at re-calibration of channel-wise feature maps through explicit modelling of inter-dependencies among channels, consequentially leading to the accentuation of important and suppression of non-informative channels.

Notwithstanding the high classification ability of the aforementioned architectures, appropriate design and robust development of a CNN architecture to meet stringent predefined criteria can become a significant and complex design task. To address this, researchers have developed neural search architectures such as NAS-Net [42], deduced from reinforcement learning [43] for determining the optimal CNN architecture concerning the training data.

Additionally, NASNet facilitates proxy dataset mapping, i.e., CIFAR-10 to IMAGENET, along with a regularisation strategy for improved generalisation capacity. As is evident from Table 2, NAS-Net is computationally lightweight compared to SENet whilst achieving the same accuracy (82.7%) on the IMAGENET dataset.

Table 2. YOLO variants benchmarked on COCO 2017 dataset.

Variant	AP-Val (%)	Fps (b = 32)	Latency (ms)	Param (M)	Flops (G)
v5-L	67.3	126	8.8	46.5	109.1
X-Tiny	50.3	1143	1.4	5.1	6.5
X-L	68.0	103	10.6	54.2	155.6
PPE-L	68.6	127	10.1	52.2	110.1
v6-N	51.2	1234	4.3	11.1	26.3
v6-L-ReLU	69.2	149	58.5	144.0	354.2
v6-L	70.0	121	58.5	144.0	354.2
v7-Tiny	49.9	1196	6.2	5.8	8.8

The preceding architectures all present respectable performance concerning classification accuracy. However, cracks appear when the deployment criterion, explicitly sanctions limited computational resources about the deployment infrastructure, i.e., edge deployment. Hence, an additional variable is introduced into the design phase, mandating a stringent computational budget (i.e., processing memory, FLOPs) in addition to high classification accuracy. The practicality element renewed the research direction, positioning it on the path to lightweight architecture development, with noteworthy advancements manifested in the form of SqueezeNet [44], MobileNet [45], ShuffleNet [46] and more.

To achieve computational efficacy, SqueezeNet [44] substitutes the majority of 3×3 kernels with 1×1 kernels, in addition to reducing the input channels for the remaining 3×3 kernels. To ensure this does not have a detrimental effect on the accuracy, downsampling is postponed to later layers, avoiding information loss within earlier layers.

The effectiveness of this strategy in achieving a lightweight architecture, is demonstrated by the fact it is smaller than AlexNet [32] by greater than 50 times. In addition, the implementation of deep compression [47] can further reduce the computational load concerning AlexNet [32] by 510 times.

Tasked with computational efficiency, MobileNet [45] proposes a depth-wise separable approach for the decomposition of standard convolutions into depth-wise convolutions followed by point-wise convolutions. The proposed mechanism was based on the depth-wise module performing the convolution process via a single filter on each input channel, whilst point-wise convolution combines the single channels via 1×1 convolution. The proposed mechanism facilitated a notable reduction in the number of parameters and overall computational complexities, in some cases by nine-fold [48]. The application of depth-wise convolution with a single filter per input channel, i.e., input depth can be expressed as

$$\widehat{G}_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} \cdot F_{K+i-1,l+j-1,m} \tag{2}$$

where $\widehat{k}$ signifies depth-wise convolutional kernel with dimensions $D(K\times)D(K\times)M$, where the m_{th} filter for $\widehat{k}$ is applied to m_{th} channel in F for producing the m_{th} channel of the filtered output feature map $\widehat{G}$.

Similarly, ShuffleNet [46] implements grouped point-wise convolution, segmenting the input feature maps into groups prior to implementing the convolutional process separately

in each group, aimed at reducing the computational baggage. As shown in Figure 2, due to point-wise group convolutions coupled with channel shuffle, all components within the ShuffleNet block can be efficiently computed.

Figure 2a presents a bottle-neck block with depth-wise convolution. Figure 2b presents the ShuffleNet block with point-wise convolution and channel shuffle. Figure 2c presents a ShuffleNet block based on a stride of 2. The grouping strategy does, however, limit the dissemination of information between the various channels. To compensate, channels are further shuffled before each group in the subsequent layer is fed with multiple channels attributed to different groups, ensuring sufficient information distribution across channels.

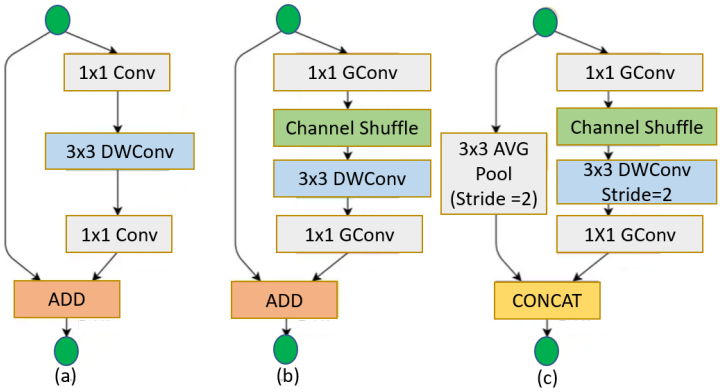

Figure 2. ShuffleNet block (**a**) depth-wise bottleneck, (**b**) ShuffleNet block and (**c**) ShuffleNet block (stride = 2).

To reduce computational costs attributed to spatial convolutions, Shift-Net [49] introduces the concept of information communicated through the internal network via feature map shifting, enabling the aggregation of spatial data through subsequent point-wise convolution layers.

In addition, FE-Net [50] demonstrates the requirement of only a small number of shift procedures for ascertaining spatial information. Hence, FE-Net [50] features a sparse-shift layer (SSL) focused on conducting shift operations on a reduced number of feature maps.

3. Object Detection

3.1. Anatomy for Object Detection

Object detection can be designated as an extension of image classification, where the focus is not only on detecting an object but rather multiple objects along with the respective locations for each object. Primarily, object detection is based on feature extraction via a module known as backbone; various architectures can be deployed to serve this purpose, such as ResNet [39], which is followed by bounding box prediction along with class assertion. Figure 3 presents the anatomy of an object detector. Before CNN-based object detection, researchers exerted their efforts on selective category detection such as faces [51] and humans [52] via manual feature descriptors, i.e., HOG [53], LBP [54] and Harr like. The historical approach is anchored around the concept of feature template matching concerning the location.

Similar to IMAGENET for image classification, researchers within the object detection domain have prioritised the development of large, multi-scale datasets for streamlining the process of benchmarking newly introduced architectures. The two iconic datasets in this regard are the PASCAL-VOC 2007 [55] containing 20 classes and MS-COCO [56] consisting of 80 categories. Performance evaluation is based on two metrics: (1) Average Precision (AP) for computing true bounding box detections with an overlap ratio of 0.5 or greater concerning the ground truth; and (2) Mean Average Precision (MAP), taking the average AP values attributed to various overlap ratio thresholds. The popularity of the two respective datasets and wide acceptance amongst researchers can be gauged via Figure 4, presenting the detection accuracies for different architectures based on the two benchmark datasets.

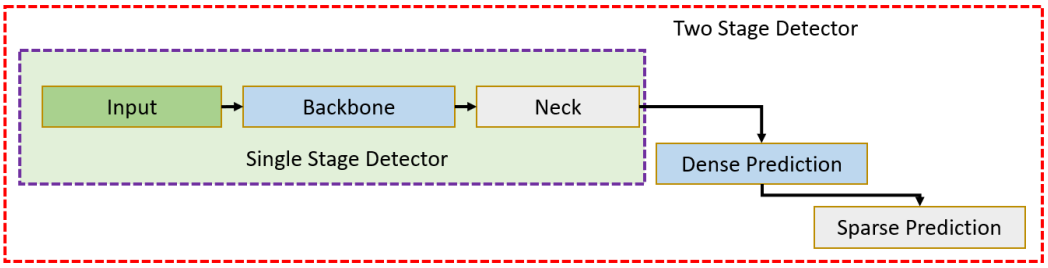

Figure 3. Object detector anatomy.

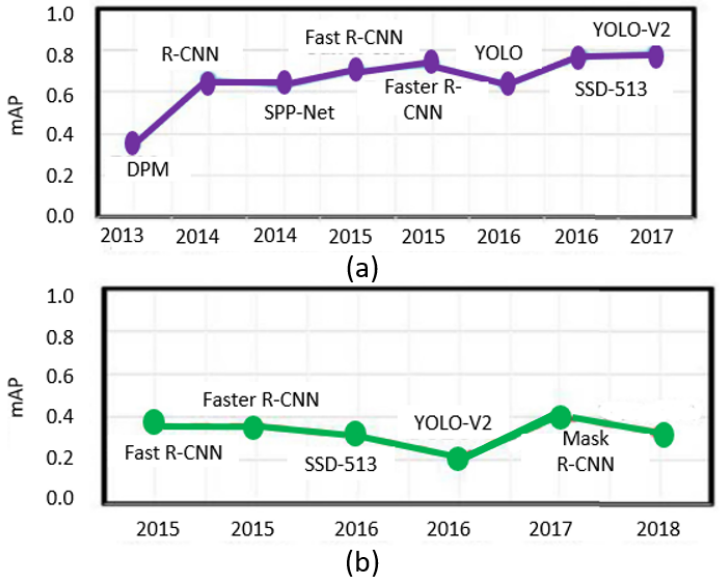

Figure 4. Accuracy for several object detectors on (**a**) Pascal VOC 2007 and (**b**) MS COCO.

R-CNN is considered among the early CNN-based object detection architectures. It subscribes to the two-stage detection framework, by focusing first on the generation of a sparse set of candidate bounding boxes from the input image, segueing into the second phase focused on suppressing and selecting the most relevant candidate proposals, for output determination. R-CNN embraces the AlexNet framework for the extraction of fixed-length feature vectors from regional candidate proposals, achieved by the selective search algorithm [57]. Each proposal is classified via a batch of class-specific linear SVMs. As shown in Figure 4, R-CNN demonstrated significant improvement utilisation compared to the prior state-of-the-art DPM network [52]. However, its cumbersome, multistage pipeline resulted in inefficiencies due to high redundancy, attributed to a large number of irrelevant regional proposals.

Its successor, Fast R-CNN [58], introduced a region of interest (RoI) pooling mechanism before the fully connected block. Hence, the acquiring of the fixed-length feature vector for every region proposal was limited to a single convolutional operation. Although Fast R-CNN significantly improved the MAP, high computational baggage remained a hindrance due to external region proposals. Focusing on reducing the computational complexities, Faster R-CNN [59] proposed the integration of the region pooling network (RPN) with the Fast R-CNN architecture, as presented in Figure 5, replacing the selective search algorithm with the region proposal network. For each location predefined set of anchor boxes consisting of different sizes and aspect ratios is applied for acquiring candidate regions. Next, each region candidate is dimensionally suppressed by a factor of 32 before the application of max-pooling. Finally, the joint regression and classification of the candidate regions facilitate the determination of regions of interest (ROI), asserting normalisation concerning the bounding boxes and class probability. This integration led to shared convolutional layers, resulting in an inference speed of 6 fps via a GPU on the PASCAL-VOC 2007 dataset.

Lin Dollar et al. [60] proposed a feature pyramid network (FPN) to leverage the pyramidal structure of a CNN for generating semantically optimised feature maps at various layers of the network.

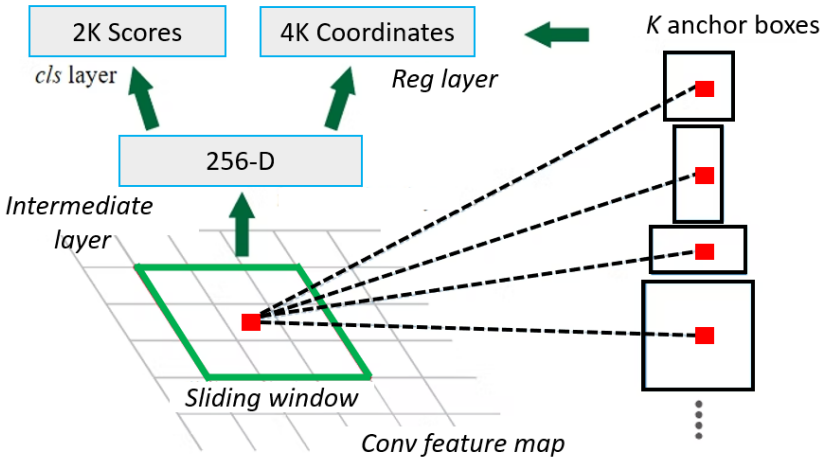

Figure 5. Region proposal network mechanism.

Two-stage architecture has demonstrated impressive performance concerning MAP; however, this comes at the expense of high computation and low FPS, making them infeasible for many real-world applications. To address these issues, one-stage detectors focus on the direct regression and classification of the object without pre-generated regional proposals.

To break the ice, YOLO [61] was introduced in 2016, framing object detection as a regression task as opposed to classification. The proposed methodology was hinged on the segmenting of the input image into several $M \times M$ grids, followed by several B bounding boxes for every grid. Next, through the global utilisation of the CNN features for the given input, the network directly predicted class probabilities, bounding box dimensions and confidence scores for the bounding boxes normalisation of the network from the proposal generation stage, enabling it to achieve a 45 fps inference speed. However, the reduced proposal candidates resulted in less accuracy compared to two-stage detectors. The same year, Liu et al. [62] introduced another single-stage architecture, famously known as the single-shot detector (SSD). The methodological approach was based on eliminating the process of proposal candidate generation, by processing a predefined number of multiscale feature maps, before adopting a default set of anchor boxes, facilitating the better handling of different resolution feature maps due to varying object dimensions of objects. The result was an increased detection speed reaching 59 fps based on a 300 $\times$ 300-pixel input image. By increasing the input size to 512 $\times$ 512, an MAP of 76.5% was achieved on the PASCAL-VOC 2007 dataset, outperforming the state-of-the-art Faster R-CNN.

The following year, YOLO made a comeback, introducing YOLO9000, famously known as YOLOv2 [63]. The proposed framework adopted the anchor approach from SSD, also proposing the use of batch normalisation, dimension clustering and anchor-based convolution, i.e., focusing on offset prediction as opposed to bounding box coordinates. YOLOv2 was able to reach 40 fps on the PASCAL-VOC 2007 dataset, with a MAP of 78.6%, outperforming various state-of-the-art architectures, including two-stage detectors.

YOLOv3, 2018 [64], focused on improving the performance of the network on small-size defects by presenting multiple anchor box approaches assigned at three different feature map scales. This increased the number of proposals, facilitating better classification ability. The backbone of the architecture, YOLOv3, evolved from Darknet-19 for YOLOv2 to Darknet-53, increasing the depth of the architectural layers. DarkNet is known as a flexible framework for research purposes, written in low-level languages. Although this decreased the inference speed compared to that of Darknet-19, it maintained similar accuracy to Faster R-CNN, with respectable real-time inference. The original author, Redmon, citing

the manipulation of computer vision for use in questionable applications, discontinued his work post-YOLOv3 [64]. The YOLO variants in the post-Redmon period, to the present date, have been released by various authors, both individuals and reparameterised research groups, such as Baidu.

YOLOv4 [65], released in 2018, enables the quantisation of various concepts, including cross mini-batch normalisation (CMBN), cross-stage partial connections and mish activation. Figure 6 presents the conceptual pipeline for the cross-stage partial connection, separating the input feature map reparameterised into two parts. Part one avoids passing through the dense block, directly composing the input for the following transition layer. Whilst part two feeds through the dense block before joining part one at the input of the transition layer, parameterisation increases complexity. The objective behind CMBN was to enable processing on any GPU as opposed to the requirement of multiple GPUs operating in tandem. Additionally, bag-of-freebies was proposed, with the primary contribution known as the Mosaic augmentation, based on tiling images together with a focus on smaller object accentuation and the suppression of background pixels. YOLOV4 achieved 65.7% AP on the MS-COCO [56] dataset at approximately 65 fps, benchmarked on a Tesla V100.

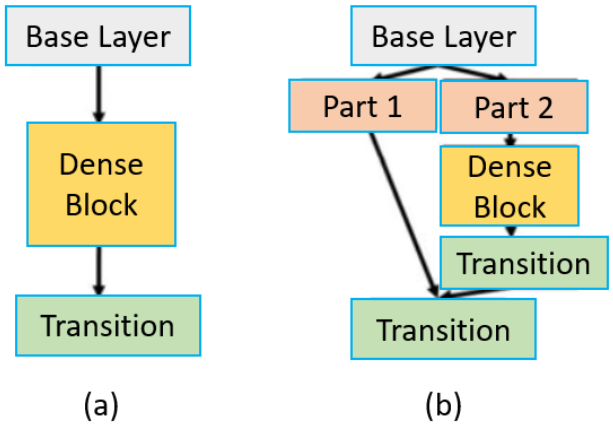

Figure 6. Cross-stage partial connection mechanism (**a**) DenseNet and (**b**) CSP-DenseNet.

YOLOv5 [66] was released via a Github repository as opposed to the conventional approach (academic paper), by Glen Jocher, around one month after YOLOv4. From the perspective of accessibility, a significant contribution was the migration from darknet, written primarily in C, to PyTorch, written in the popular Python language. The technical contribution included the integration of the anchor box selection mechanism via k-Means into the training process as opposed to a separate clustering procedure for default anchor box selection, presented in YOLOv2 [63], requiring manual configuration into the architecture.

YOLOv6, the initial codebase, was introduced in June 2022, followed by an updated version and then being published in a paper in September 2022 [67] by researchers at Meituan [68]. The researchers aimed to propagate its implementation within industrial environments, hence following stringent design considerations such as speed and accuracy. Various variants of YOLOv6 have been introduced, aimed at a diverse set of industrial applications, from YOLOv6-Nano (addressing speed) to YOLOv6-Large (addressing accuracy) and in between. Unravelling the technicalities of the latest variant presented rich contributions on various fronts, including reparameterised backbones, various augmentations, network quantisation and more. The first significant contribution was the adoption of an anchor-free approach, providing better generalisability and making it 51% faster when compared to the majority of anchor-based object detection architectures. Next, reparameterised backbones were implemented for the feature extraction stage. Backbones are one of the more computationally expensive components within a network, hence increasing the computational baggage and adversely affecting the inference speed. Hence, reparameterisation

enabled switching between backbone structures during the training and inference phase. As shown in Figure 7, RepVGG blocks with skip connections were implemented during the training for YOLOv6, whilst RepConv, i.e., simple 3 × 3 convolutional blocks, were deployed during inference. A couple of weeks after the introduction of YOLOv6, YOLOv7 [69] was introduced to the computer vision industry. The authors aimed to improve speed and accuracy via the implementation of several reforms at the architectural level. YOLOv7, similar to scaled YOLOv4 backbones, avoids pretraining the backbone via IMAGENET rather than opting for the COCO dataset. A significant contribution is manifested in the form of an Extended Efficient Layer Aggregation Network (E-ELAN) as the computational backbone for the architecture. To cater for various applications, networks need to be tuned to meet specific requirements, i.e., accuracy and speed. For this, the authors introduced compound model scaling for facilitating the scaling of the depth and width in coherence for concatenation-based networks. YOLOv7 is present in several variants, with all variants able to achieve greater than 30 fps on the Tesla V100 GPU; however, the authors explicitly state that no variant is designed for mobile device CPUs.

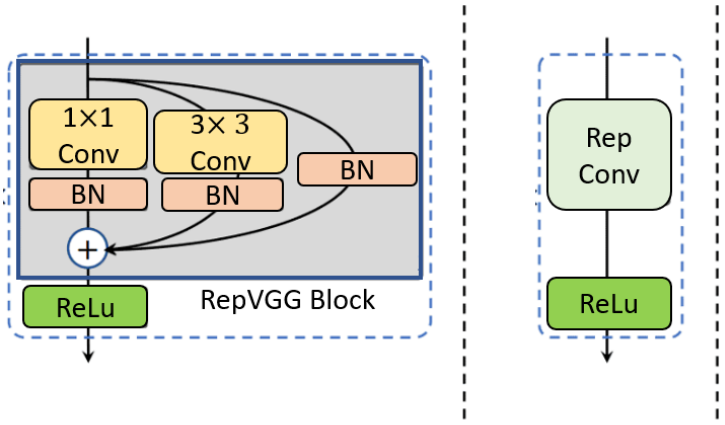

Figure 7. YOLOV6—RepVGG blocks utilised for training and RepConv blocks for inferencing.

Table 2 presents the quantitative results of COCO 2017 validation benchmarks for various YOLO architectures with input dimensions of 640 × 640 pixels, except for YOLOX-Tiny and YOLOV7-Tiny at 416 × 416 pixels. Surprisingly, YOLOv6-ReLU, consisting of 58.5 million parameters, surpassed both PPYE-L and YOLOX-L in speed and accuracy. Additionally, it can be observed that the majority of YOLOv6 variants provide higher FPS concerning a batch size of 32.

3.2. YOLOv8

In January 2023, Ultralytics introduced YOLOv8, object detector for computer vision tasks [70]. Demonstrating impressive precision, YOLOv8 performance was benchmarked via the COCO and Roboflow 100 datasets [70]. YOLOv8 is seen to be more user firendly via its user-oriented features, including a user-friendly command-line interface and a well-structured Python package. YOLOv8 deviates from conventional anchor-based methods, employing an anchor-free approach that predicts the target center. This method addresses challenges posed by anchor boxes that may not accurately represent custom dataset distributions, reducing the number of box predictions and accelerating the post-processing step involving non-maximum suppression. The YOLOv8 training routine encompasses techniques such as online image augmentation and mosaic augmentation, enhancing its ability to detect objects in diverse conditions and spatial arrangements. YOLOv8 also introduces architectural changes from its predecessor, YOLOv5, including the direct concatenation of features in the neck segment without enforcing uniform channel dimensions, reducing the parameter count and overall tensor size. When tested on the MS COCO

dataset's test-dev 2017 subset, YOLOv8x delivered an average precision (AP) of 53.9% at an image size of 640 pixels, compared to YOLOv5's 50.7% AP at the same input size. Additionally, YOLOv8x achieved a processing speed of 280 fps using an NVIDIA A100 with TensorRT and is available in five variants, each tailored to specific accuracy and computational requirements.

3.3. YOLOv9

In February 2024, Wang et al. [71] introduced YOLOv9, featuring two key innovations: the Programmable Gradient Information (PGI) framework and the Generalized Efficient Layer Aggregation Network (GELAN). The PGI framework focuses on the information bottleneck problem in deep neural networks, facilitating compatibility with lightweight architectures and enhancing performance accuracy. PGI warrants reliable gradient information propagation during training, improving learning capacity and prediction accuracy. GELAN builds on the gradient path optimisation principles of CSPNet [72] and ELAN [73], balancing model lightweightness, inference speed, and accuracy. This design enables GELAN to perform consistently across various computational blocks and depth configurations, making it suitable for deployment on resource-constrained edge devices. With the strengths of PGI and GELAN, YOLOv9 marks a significant advancement in lightweight object detection, surpassing YOLOv8 in parameter reduction and computational efficiency while achieving a 0.6% improvement in AP on the MS COCO dataset. The performance metrics of different YOLOv9 models are detailed in Table 3 [74].

Table 3. Performance metrics of YOLOv9 models [71].

Model	Size (Pixels)	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	Param.	FLOPs
YOLOv9-S	640	46.8%	63.4%	50.7%	7.2 M	26.7 G
YOLOv9-M	640	51.4%	68.1%	56.1%	20.1 M	76.8 G
YOLOv9-C	640	53.0%	70.2%	57.8%	25.5 M	102.8 G
YOLOv9-E	640	55.6%	72.8%	60.6%	58.1 M	192.5 G

3.4. YOLOv10

Released in May 2024 by researchers at Tsinghua University, YOLOv10 represents a substantial advancement in real-time object detection (OD) [75]. This architecture addresses the challenge of balancing accuracy with computational efficiency through innovative training strategies and architectural modifications. The core concept involves "Consistent Dual Assignments" during training, allowing the model to learn from rich supervision while eliminating the need for computationally expensive non-maximum suppression (NMS) during inference, significantly reducing processing time. YOLOv10 enhances efficiency with the Parallel Split-Attention (PSA) module and the Compact Inverted Bottleneck (CIB) block, enabling efficient multi-scale feature processing and effective attention mechanisms. To boost accuracy, the Scaled Residual Connection and Scaled Weight Shortcut techniques improve information flow within the network. Extensive evaluations show that YOLOv10 surpasses previous YOLO versions and other state-of-the-art models in the accuracy–efficiency trade-off. For instance, as shown in Table 4, YOLOv10-B reduces latency and parameter count compared to YOLOv9-C with equivalent performance. Additionally, YOLOv10-L and YOLOv10-X variants outperform their YOLOv8 counterparts in accuracy while requiring fewer parameters.

Table 4. Performance metrics of YOLOv10 models [76].

Model	Size (Pixels)	AP^{val} (%)	FLOPs (G)	Latency (ms)
YOLOv10-N	640	38.5	6.7	1.84
YOLOv10-S	640	46.3	21.6	2.49
YOLOv10-M	640	51.1	59.1	4.74
YOLOv10-B	640	52.5	92.0	5.74
YOLOv10-L	640	53.2	120.3	7.28
YOLOv10-X	640	54.4	160.4	10.70

4. Hardware Configurations

Architectural advancements within the software realm, as presented in the preceding section, have significantly contributed to the success of CNN. However, architectural breakthroughs are not the sole driving factor for the success of computer vision. Advancements in hardware, over the past decades, have equally contributed to the proliferation of computer vision applications, in particular when referring to CNN deployment [77]. Major breakthroughs in hardware acceleration have resulted in robust parallel computing architectures, facilitating the enhanced training and inferencing of complex, multi-layered and deep CNN-inspired architectures.

Hardware acceleration constructively manipulates computer hardware with the objective of executing computational tasks at reduced latency and increased throughput, compared to customary software execution on CPUs. Historically, Princeton computing architectures are primarily focused on serial computations coupled with complex task planning [78], hence suffering from low memory bandwidth and high-power consumption when dealing with CNN architectures requiring dense parallel computation, increased memory bandwidth and high data reusability [79]. To address these limitations, researchers and hardware vendors are actively engaged in developing strategies for enhanced processing capabilities leading to higher parallelisation, improved inferencing and efficient power consumption. The section focuses on evaluating significant hardware acceleration implementations, their contribution, limitations and implications with respect to computer vision applications.

4.1. Graphical Processing Units (GPUs)

A graphical processing unit (GPU) can be defined as a specialised processor, originally focused on accelerating real-time 3D graphics applications, rendering and games [80,81]. As the dawn broke for the 21st century, scientists and researchers started investigating and swiftly realised the potential of GPU integration with computing systems for addressing a wide range of computing problems. This was primarily due to the underlying nature of a GPU, incorporating extraordinary levels of computational capabilities and administering incredible acceleration to computing workloads that exploit the highly parallel nature of GPUs, such as CNNs [82]. Hence, a GPU in the modern era is not only seen as a compelling graphics engine but also a highly parallelised processor for computing purposes, featuring high throughput and memory bandwidth for parallel architectures.

Multicore CPUs are in general multi-instructional, and out-of-order, utilising large caches for suppressing the latency of a single thread and running a high frequency. On the contrary, GPUs contain thousands of cores that are in order, are reliant on small-size caches and operate on lower frequencies [83]. To address barriers around the implementation and integration of GPU-based applications, several development platforms have emerged such as Open Computing Language (OpenCL) [84] and the well-established, widely used Compute Unified Device Architecture (CUDA) by NVIDIA [85].

Along with other domains, deep learning has also been a primary beneficiary of GPU-based acceleration. Unravelling the CNN structure, one can observe the perfect match between the two applications in regard to the parallelisation of the convolutional operations, different sub-sampling strategies and neuron activations within the fully connected layers via a binary-tree multiplier [86]. Realising the potential of GPUs in CNN structures,

several libraries have emerged for GPU-CNN integration, such as cuDNN [87], Cuda-convert [88] in addition to libraries built upon deep learning frameworks such as Caffe [89], Tensorflow [90] and Torch [91].

Benchmarking the efficacy of GPUs with respect to Deep Learning is primarily based upon three performance indicators, namely memory efficiency, computational throughput and power consumption. NVIDIA is accepted as the leading vendor when it comes to deep learning on GPUs. Considering the vast variety of applications, stringent deployment environments and financial allotments, NVIDIA has progressively introduced various GPUs over the last two decades.

Recognising the incompatibility of many GPU variants, when it comes to constrained environments requiring edge deployment, small-size form factor and reduced cost, NVIDIA introduced Jetson, featuring a heterogeneous architecture, where the CPU was tasked with booting the operating system (OS), and CUDA-based GPU with accelerating deep learning tasks. Several variants of NVIDIA Jetson have been released focused on providing a low-power embedded device, facilitating server-grade computing performance at an affordable cost. NVIDIA accelerator kits have been widely utilised for a wide range of machine learning and deep learning experiments and applications. Authors in [92] presented CNN performance evaluation on CNN architectures, concluding the Jetson TX2 variant as outputting high efficiency compared to other Jetson variants. Rui Jin [93] proposed a teacher/student architecture for real-time fabric defect detection during the production phase. The architecture consisted of a YOLOv5 backbone and deployed onto a Jetson TX2 for edge inferencing rather than the Raspberry Pi [94]. The student network achieved an impressive 96.5% AUC with an inference time of 16 ms, hence guaranteeing real-time performance.

4.2. Field-Programmable Gate Arrays (FPGAs)

GPUs have proven to be efficient in providing high parallelism and throughput, making them a strong contender for hardware acceleration. However, the progression of the Internet of Things (IoT) [95] and its delineation have resulted in accessibility to a wider enterprise base. In particular, for Industry 4.0 [96], businesses are censoring for close-to-the-source, edge device deployment solutions. One of the key requirements, in addition to high accuracy and inference speeds, is power efficiency, an area in which Field-Programmable Gate Arrays (FPGAs) [97] have an advantage over GPUs. Taking into account recent architectural advancements, leading to increasingly sparse yet compact architectural structures, FPGAs facilitate irregular parallelism, customised data types and specific hardware designs. Additionally, post manufacturing, FPGAs allow reprogramming in order to stay in tune with the dynamic application/environmental requirements. Due to the flexibility and tailor design feature, FPGA accelerators have found their way into the embedded application domain [98].

FPGA modules consist of a set of programmable logic blocks coupled through a hierarchy of interconnects that are reconfigurable. A typical FPGA is a host to various subcomponents such as digital signal processing (DSP) units catering for multiply–add–accumulate operations (MAC), look-up tables (LUTs), addressing combinatorial logic operations and the facilitation of on-chip data storage via block RAMs [99]. With respect to the implementation of CNNs, Figure 8 presents a characteristic FPGA architecture. The internal blocks consist of several sub-modules, including a memory-data-management unit (MDM), on-chip-data-management unit (ODM), general-purpose-matrix-multiply unit (GEMM), implemented via a set of processing elements (PEs) for computing MAC operations and a Msic-layers unit (MLU) for computing batch normalisation, ReLU and pooling [100].

The execution of a CNN via an FPGA consists of several steps, starting with the transfer of CNN weights along with the input feature maps from the MDM into an on-chip buffer, i.e., ODM. Next, the GEMM unit is tasked with computing the matrix operations and transferring the results to MLU for batch normalisation, ReLU and pooling. MLU resultants

are transferred to another ODM unit to be accessed by subsequent convolutional/fully connected layers. In the case of memory shortage in the on-chip buffer, intermediary results are temporarily hosted in the on-chip/off-chip memory.

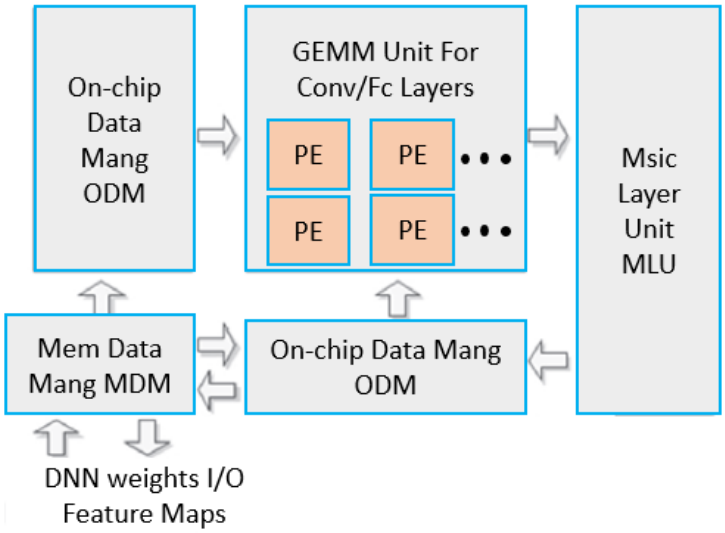

Figure 8. Typical FPGA architecture for DNN implementation.

Conventionally, FPGAs are specified at a register transfer level (RTL) through hardware description language (HDL), i.e., VHDL [101] and Verilog [102]. However, the less abstract nature of these languages demands specific hardware design expertise, as well as time and effort for the theoretical and practical implementation of the required architecture, in addition to factoring in high concurrency requirements between varying hardware modules. To address these limitations, high-level-synthesis (HLS) methods have been successfully proposed, facilitating FPGA hardware design through high-level languages such as C [103]. In addition to the automated compilation of high-level descriptions to low-level specifications, hence widening the accessibility to a wider research audience [104].

In general, CNN architectures feature high computational requirements due to increased architectural parameters (FLOPs) and high memory storage requirements; however, FPGAs typically feature a memory bandwidth often 10% or less than that of GPUs. Classifying this as a threat to the compatibility of CNNs on FPGAs, researchers have developed several algorithmic optimisations for reducing the architectural computational load of CNNs, making them more lightweight for deployment on computationally constrained FPGA devices.

One of the optimisation strategies is known as algorithmic operation, based on the implementation of computational transforms such as Fast Fourier Transform (FFT) [105], GEMM and Winograd [106] on convolutional kernels/feature maps, aimed at suppressing the number of arithmetic operations post-deployment i.e., during inference. FFT reduces arithmetic complexity, by casting a 2D convolution to an element-wise matrix multiplication [107]. This can be particularly useful when dealing with large kernel sizes, due to the increased number of computational operations required between the kernels and feature maps.

GEMM is a widely applied technique for deep neural network (DNN) processing in CPUs and GPUs, as it vectorises both convolutional and fully connected layer computations [108]. When dealing with small kernels, Winograd provides a more efficient approach for arithmetic reduction compared to FFT, by reutilising the intermediate results [109]. The manifestation of its efficiency can be gauged by the fact that Winograd transforms can achieve 7.28x runtime speed-up, on a VGG-Net, compared to GEMM, when running on a Titan-X GPU [107], whilst delivering 46 GOPs throughput for AlexNet on an FPGA [110].

Data-path optimisation is another strategy aimed at more efficient architectures with respect to computation. Traditionally, FPGAs designed, and deployed processing elements

as 2D systolic arrays [111–113]. However, due to the resultant kernel size of the CNN, data caching was not possible, limiting architectural efficacy.

The loop optimisation mechanism looks to alleviate this issue, consisting of several sub-components; loop reordering for blocking redundant memory access between loops enhancing cache capacity [114], loop unrolling, and pipe-lining improve FPGA resource utilisation [115,116], whilst loop tiling, partitions weights/feature maps for each layer arriving from the memory into 'tiles', for efficient hosting onto the on-chip buffers [117].

CNNs have the potential to be deployed across the spectrum with respect to applications. Many applications are able to accommodate a certain degree of error, i.e., error-tolerance, in particular quality inspection applications within domains like manufacturing. This has provided further leverage and an initiative for researchers to focus on model compression strategies for further reducing architectural and hardware complexities. Model compression techniques can be categorised into three distinct types: pruning [118], low-rank approximation [98] and quantisation [119].

Pruning [120–122] focuses on reducing architectural complexity through the elimination of redundant parameters/weights within a network. For example, CNNs, contain a large number of weights; however, not all of the weights have a substantial or in some cases any contribution to the performance; hence, by alleviating the network of redundant weights, a more lightweight and energy-efficient footprint can be ascertained allowing more fruitful deployment on constrained devices such as FPGAs. Low-rank approximation [123], when utilised for CNN compression, decomposes the convolutional weight matrix or fully connected layers into a set of low-rank filters, that can be evaluated at reduced computational cost, again, this can be particularly useful in the case of the target deployment hardware being constrained in terms of the computational capacity.

Quantisation [124,125], is based on the fact that fixed-point arithmetic operations demand less computational resources compared to float-point operations; hence, quantising CNN feature maps and weight matrices based on a fixed-point representation can further reduce computational cost. In cases where deployment infrastructure is extremely constrained, quantisation via binary transformation of weights can be deployed, known as binary neural networks (BNN); however, in some cases, this can have a detrimental impact on the accuracy [126].

Jin Rui et al. [127] demonstrate the effectiveness of pruning by applying it to a CNN architecture aimed at textile defect detection within the production environment. Authors achieved pruning via tensorRT, prior to deployment on an NVIDIA Jetson TX2. Evaluating the impact of pruning, authors state processing defects before implementing pruning required 80 ms, whilst post-pruning, this was reduced to 36 ms.

4.3. Application-Specific Integrated Circuits

In the context of deep learning, Application-Specific Integrated Circuits (ASICs) are custom-designed hardware accelerators, with a reduced focus on achieving the requirements of a specific application, i.e., accuracy, and inference speed [128]. The tailored design of these accelerators enables them to outperform GPUs and FPGAs when evaluated on the particular application for which the ASIC was designed. However, due to their custom design, the design, development and implementation requires substantial time cycles. Over the last decade, various ASIC accelerators have been introduced to the AI market such as the HiSilicon Kirin-970, developed by Huawei [129], featuring a heterogeneous architecture with a dedicated neural processing unit, improving throughput by 25 times and energy efficiency by 50 times, based on a Cortex-A73 (quadcore) CPU cluster. Google boasts its own customised ASIC known as Tensor Processing Unit (TPU) [130], tailored for deep neural networks via the TensorFlow platform [91]. Pencilling its own contribution, Apple introduced the neural engine, a set of processor cores aimed at certain deep learning networks for applications such as face identification.

Table 5 presents a comparison between GPUs, FPGAs and ASICs on a wider metric base. Additionally, once manufactured, the design footprint of ASICs is not reconfig-

urable. This is a significant bottleneck for ASICs, as, due to the changing nature of various deployment environments, reconfigurability is required for making relevant adjustments.

Table 5. Hardware metric-based comparison with increased row height.

Metric	GPUs	FPGAs	ASICs
Energy Efficiency	Low	Medium	High
Ability to Reconfigure	Low	High	Low
Area	Large	Large	Small
Digital Signal Processing Blocks	-	Fixed Precision	Custom
Power	High	Medium	Low
Time to Market	Low	Medium	High

5. Industrial Defect Detection

Machine vision is the coined term when referring to the application of computer vision within the industrial domain. A decade back, both hardware and architectural capacity had not developed to the level of maturity to warrant its integration into existing industrial processes. Hence, quality inspection in the majority of cases was administered via human-based inferencing.

However, as automated process integration found its way into manufacturing facilities and the types of defects became harder to differentiate, the integration of machine vision-based inspection has been on the rise, in particular within the surface defect detection domain [131] . Quality inspection is an indispensable component within the manufacturing domain. Hence, by upgrading its efficacy via machine vision, additional benefits can be reaped such as reduced labour costs, the elimination or suppression of human bias, reduced inferencing time, elimination of human fatigue, etc. The implementation of machine vision across several industries for surface defect detection is presented.

5.1. Textiles

Rui Jin et al. [93] initiate their research by mentioning the potential benefits of automated quality inspection within the fabric manufacturing industry, referring to reduced labour cost and higher detection speed. The authors propose a customised YOLOV5 architecture, integrating a spatial attention mechanism for the identification of smaller defects. The process consisted of a trained teacher network on the fabric dataset. Knowledge distillation was adopted to transfer the learning to a student (computationally lighter network), before being deployed to a Jetson TX2 via TensorRT. The authors state that although the teacher architecture provided higher performance 98.8% (AUC) compared to student architecture (96.5% AUC), the inference speed for the latter was significantly less (16 ms) compared to 35 ms for the former on the Jetson TX2. The authors argue the proposed architecture coupled with the hardware selection guarantees real-time performance with respect to the required identification time.

Jiaqi Zhang et al. [132] propose a modified Mo-bileNetV2-SSD-Lite architecture for automated fabric surface defect detection. The modification is manifested in the form of introducing a channel-based attention mechanism, aimed at the accentuation of the defective regions relative to the background surface. Additionally, the default loss function is replaced via focal loss and K-means clustering for candidate box parameter optimisation. The results demonstrated a respectable performance, with the proposed architecture achieving greater than 90% MAP (mean average precision) whilst reaching an inference speed of 14.19 fps on the camouflage dataset. With respect to the channel-based attention mechanism, the authors reveal that the computational implications of this integration were negligible, with the parameters experiencing an increase of incorporation of 0.001 million.

Feng Li et al. [133] propose fabric defect detection through the implementation of a cascaded-RCNN, similar. Figure 9 represents the concept of cascading, focused on accuracy enhancement through a multi-headed approach. Multiscale training was utilised followed by dimensional clustering for characterising the prior anchors with ResNet-50 selected as

the feature extractor. Additionally, a feature pyramid network (FPN) was utilised based on a bottom-up–top-down path corresponding to the lateral connections. The authors claim to quantify the proposed implementation result in improving the overall precision of the architecture by 8.9%; however, it is worth acknowledging the reduced complexity of the testing dataset due to the lack of patterned fabric images, which would make accurate classification more difficult.

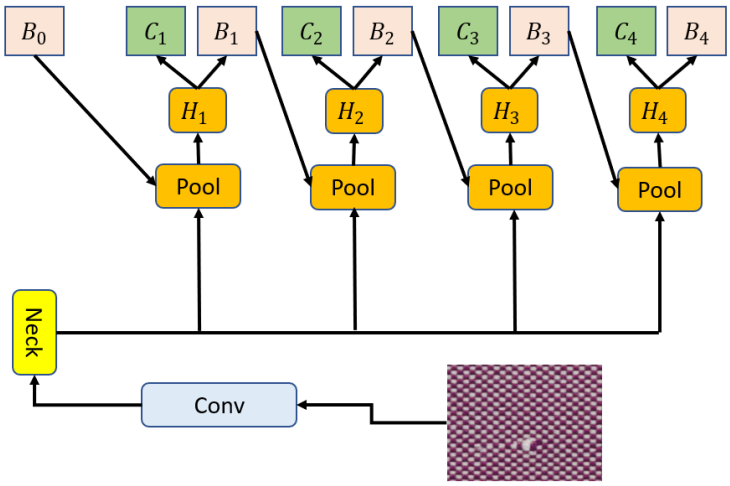

Figure 9. Cascaded R-CNNconcept.

Shaojun Song et al. [134] aim to address industrial-based fabric defect detection via an efficient architecture. To cater for the constrained deployment environment, the authors exert efforts in developing a computationally lightweight footprint providing low latency and power consumption whilst reaching high accuracies. Additionally to the application of representative data augmentation, architectural optimisation is sought through TensorRT, facilitating layer and tensor fusion along with weight and activation precision calibration. The authors justify the implementation of CBR (convolution, bias and relu merging into a single layer) aimed at acquiring a computationally lighter architecture. Their results reinforced the high efficacy of the proposed solution, with the edge deployment being 2.5x faster compared to a cloud-based configuration in addition to the detection accuracy achieving 98%. Furthermore, post tensorRT, results improved to 22.78 fps and an inference time of 43.9 ms compared to pre-optimisation (13.74 fps, inference time 72.8 ms).

5.2. Photovoltaics

Photovoltaic systems are seen as a major contributor when it comes to renewable energy, focused on the reduction of global emission-based alleviating the dependence on conventional energy generation systems [135]. Solar cells are the fundamental component within the PV setup, and during PV module manufacturing, cells are exposed to high temperature differentials, in addition to external pressure sources, resulting in the inception of micro-cracks [136].

The quality inspection led by human inspectors carries similar bottlenecks to those found within the fabric manufacturing industry. As per a recent case study (2018), based on the inspection of 180,000 PV panels, post-deployment, over 4000 faulty modules were shipped, incurring six-figure losses for the client. The non-uniformity of the cell surface can make defect detection a cumbersome task; hence, researchers have focused attention on the implementation, manipulation and enhancement of CNN architectures for this task.

Hussain et al. [3] propose a tailored CNN architecture for industrial-based Micro-crack detection during PV manufacturing. Addressing the difficulty in industrial data procurement, the authors introduce the concept of internal feature map extraction for utilisation as representative augmented samples, improving the variance of the dataset without affecting the true characteristics of the dataset. Post-training, the proposed architecture was bench-

marked against SOTA architectures on a broad metric baser compromising of architectural complexities, computational complexities, accuracy, FPS and latency. The reported results endorsed the efficiency of the proposed methodology achieving an impressive F1 score of 98.8% whilst containing only 6.42 million learnable parameters.

Z. Luo et al. [137] focus on addressing data scarcity in the context of industrial PV manufacturing i.e., Electroluminescence Imaging (EL) of PV cells. The methodology is anchored on the utilisation of a Generative Adversarial Network (GAN) for the generation of representative data samples from within the original dataset. Three SOTA architectures are trained for methodology evaluation purposes; ResNet, AlexNet and SqueezeNet [44]. The former, i.e., ResNet, provided the optimal performance. However, critically evaluating their methodology, the authors indicated the high instability of the training process along with the increased computational baggage experienced during training, which would have a knock-on impact on the scalability of the proposed solution.

Binyi Su et al. [138] argue that automated EL-based PV cell surface defect detection is a challenging process due to the blurring of boundaries with respect to defective and normal pixel regions. Hence, the authors propose a Complementary Attention Network (CAN) presented in Figure 10, based on the coupling of channel and spatial-wise attention modules. The proposed mechanism is incorporated into the Faster-RCNN architecture facilitating background pixel suppression, resulting in the accentuation of defective regions. VGG-16 was selected as the feature extractor integrated into the Faster-RCNN, with architecture pretraining conducted on the IMAGENET dataset. Focusing on the computational complexities of the proposed architecture, it can be observed that a significant computational space was required, with the architecture containing 261.26 Million parameters. However, the CAN mechanism had a minimal contribution to the computational complexity, with Faster-RCNN contributing 260.50 Million parameters.

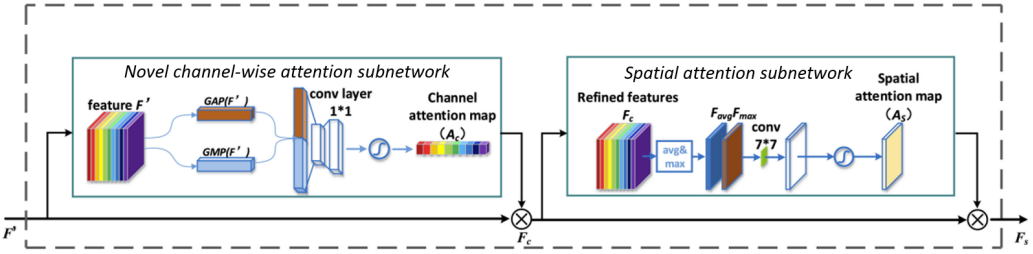

Figure 10. Complementary attention network [128].

Ashfaq Ahmad et al. [139] highlight key challenges in PV surface defect detection as inhomogeneous surface intensities and varying background complexity. The proposed solution consists of a custom CNN architecture compromising four convolutional blocks containing 32 filters each, followed by dual convolutional blocks containing 64 filters each and ending with two convolutional blocks with 128 filters each, feeding into a single fully connected layer. Additionally, the authors apply various augmentations for improving generalisation, reporting respectable performance in terms of accuracy at 91.58%.

5.3. Warehousing

A new potential application for automated defect detection via machine vision within the warehousing/manufacturing industry is pallet-rack inspection. Warehousing, distribution and storage centres rely on critical infrastructure for stock storage and preservation known as pallet racking. Unnoticed racking damage can lead to multiple complications in the case of the racking collapsing such as ruined stock leading to financial loss, operational downtime, injured employees and, in severe cases, the costing of lives. Although various mechanical solutions exist, in the form of rackguards [140], these are only limited to impact absorption, lacking any intelligence for damage perception.

Hussain et al. [141] initiate their research within this field focusing on the adoption of deep learning architectures for automated defect detection. Due to the constrained deployment requirements, the authors propose a MobileNetV2-based solution trained on the first pallet-racking dataset acquired from several industrial partners. The authors focus on enhancing the dataset via representative sample scaling, achieving mean average precision of 92.7%. An additional novelty, compared to previous mechanical and sensor-based solutions, was the device placement proposal. Rather than attaching any hardware to the racking itself, the authors proposed the placement of the hardware device hosting the inferencing mechanism, onto the forklift adjustable brackets. This strategic placement resulted in a significant cost reduction, i.e., reducing hardware in some cases by 95% @ IoU of 50% whilst expanding the coverage area relative to the operating forklift. With the aim to further improve the performance along with enhancement of the solution with respect to real-time operational requirements, Hussain et al. provided a recent paper [142] proposing the training of the YOLOV7 architecture based on the concept of domain variance modelling (DVM). Additionally, the defect classes were expanded from only vertical defects to including vertical, horizontal and rack support damage. The presented results were impressive, demonstrating an overall performance at an IoU of 50% of 91.1% running at 19 fps.

Fahimeh Farahnakian et al. [143] also focus on automated racking inspection, subscribing to the segmentation domain by proposing Mask-RCNN as the inferencing architecture. Although the reported performance is slightly higher than [141], when observing the dataset, it was evident the dataset is not representative of the production floor, as the captured images are based on disconnected racking without any contextual details, i.e., warehouse environment, loaded stock onto racking for training a representative architecture.

Additionally, Table 6 presents a comparison of the four research papers within this field. It is believed that the introduction of the first racking dataset by Hussain et al. [141] will prove to be a spark, enabling other researchers to develop CNN architectures for addressing pallet-racking inspection given the tremendous benefits of this application for the warehousing industry.

Table 6. Pallet racking recent work comparison with increased row height.

	[141]	[142]	[143]	[144]
Domain	Detection	Detection	Segment	Detection
Dataset Size	19,717	2094	75	2034
Classes	2	5	1	2
Detector	Single stage	Two stage	Two stage	Single stage
MAP@0.5 (IoU)	92.7%	91.1%	93.45%	96.8%

5.4. Diverse Set of Applications

Table 7 highlights various studies and their contributions across different domains of surface defect detection. This comprehensive overview illustrates the substantial breadth of Deep Learning (DL)-based machine vision (MV) technology applications, showcasing its versatility and potential in enhancing industrial quality inspection processes.

By leveraging advanced DL techniques, these studies have addressed a wide range of industrial challenges, from detecting defects in metals and electronic components to inspecting the surfaces of wheels, diodes, and ceramic materials. The table underscores the adaptability of DL methods to various inspection tasks, reflecting their ability to improve accuracy, efficiency, and overall reliability in quality control across multiple industries.

Each entry in the table represents a unique application, highlighting the innovations and improvements brought about by DL in specific contexts. This diversity not only demonstrates the technology's broad applicability but also its capacity to be tailored to meet the specific needs of different inspection scenarios, ultimately contributing to more robust and automated industrial processes.

Table 7. DL-based machine vision applications in industrial surface defect detection.

Application	References
Metal Surface Defect Detection	Tao et al. (2018) [145], Xu et al. (2021) [146], Lin et al. (2021) [147]
Electronic Component Defect Detection	Xin et al. (2021) [148], Jeon et al. (2022) [149], Santoso et al. (2022) [150]
Optical Fiber Defect Detection	Wang et al. (2019) [151], Mei et al. (2021) [152]
Wheel Hub Surface Defect Detection	Han et al. (2017) [153], Sun et al. (2019) [154], Cheng et al. (2023) [155]
Diode Chip Defect Detection	Lin et al. (2019) [156], Stern et al. (2021) [157], Zheng et al. (2023) [158]
Bottle Mouth Defect Detection	Koodtalang et al. (2019) [159], Zhang et al. (2021) [160], Gizaw et al. (2022) [161]
Precision Parts Defect Detection	Qu et al. (2018) [162]
Varistor Defect Detection	Yang et al. (2019) [163], Yang et al. (2020) [164]
Ceramic Defect Detection	Stephen et al. (2021) [165], Lu et al. (2022) [166], Wan et al. (2022) [167]
Wood Defect Detection	Shi et al. (2020) [168], Chen et al. (2022) [169], Lim et al. (2023) [170]
LCD and Touch Display Defect Detection	Qi et al. (2020) [18]
Magnetic Tile Surface Defect Detection	Huang et al. (2020) [171]
Rail Surface Defect Detection	Soukup et al. (2014) [172]
Concrete and Steel Surface Crack Detection	Cha and Choi (2017) [173]
Structural Visual Inspection	Cha et al. (2018) [174]

5.5. Robotic Vision

A major benefit of machine vision providers like Cognex [175] and Keyence [176] is their end-to-end development and deployment platforms, which enables consumers to develop complete solutions within a single integrated ecosystem. This approach offers significant benefits over conventional methods that require training architectures on separate platforms and sourcing compatible hardware configurations for deployment.

Cognex's In-Sight D900 [177] vision system exemplifies this integrated approach. It combines a high-resolution camera, powerful processing capabilities and deep learning software tools in a single package. The system allows users to develop, train and deploy machine vision applications using the In-Sight ViDi software (D900) and In-Sight spreadsheet interface, all without requiring a separate PC or extensive programming knowledge. Key advantages of end-to-end platform include

Simplified workflow: Users can develop, train and deploy applications within a single environment, streamlining the entire process.

Reduced integration challenges: The hardware and software are designed to work seamlessly together, eliminating compatibility issues.

Faster deployment: With pre-integrated components and user-friendly interfaces, solutions can be implemented more quickly.

Accessibility: The intuitive tools make advanced machine vision capabilities accessible to users without extensive programming expertise.

Optimised performance: The tightly integrated hardware and software ensure optimal performance and efficiency.

Scalability: These platforms often offer a range of compatible products, allowing users to scale their solutions as needed.

By providing this comprehensive ecosystem, companies like Cognex enable their customers to focus on solving their specific machine vision challenges rather than dealing with the complexities of integrating disparate components and platforms. This approach significantly reduces the time, cost and technical expertise required to implement advanced machine vision solutions in industrial settings.

Mujadded et al. [178] look into the integration of a lightweight convolutional neural network (CNN) architecture with an industrial KUKA robotic arm for automated crit-

ical component detection, as shown in Figure 11. This integration demonstrates how lightweight CNNs can be combined with robotic hardware to create intelligent systems capable of performing complex visual inspection tasks, thereby enhancing operational efficiency, improving product quality and automating labor-intensive processes for critical component detection. The authors integrate an attention mechanism into the CNN architecture to guide the network towards the region of interest without significantly adding to the computational depth of the CNN, reporting an impressive F1-score of 96%.

Mateusz et al. [179] look into the automated monitoring of cutting tool wear in the manufacturing industry, to improve production efficiency. This research explores the application of ViDiDetect, a deep learning-based defect detection solution, in inspecting cutting tool wear using a Cognex D900 series monochrome camera. The camera was selected due to its high resolution, adaptability and communication capabilities. The vision system, equipped with an illuminator and other necessary components, captures high-resolution images of a car suspension knuckle's machined surface, focusing on detecting burrs, chips and tool wear. The results demonstrated the accurate classification of surface defects indicating towards better optimisation for automated tool replacement process.

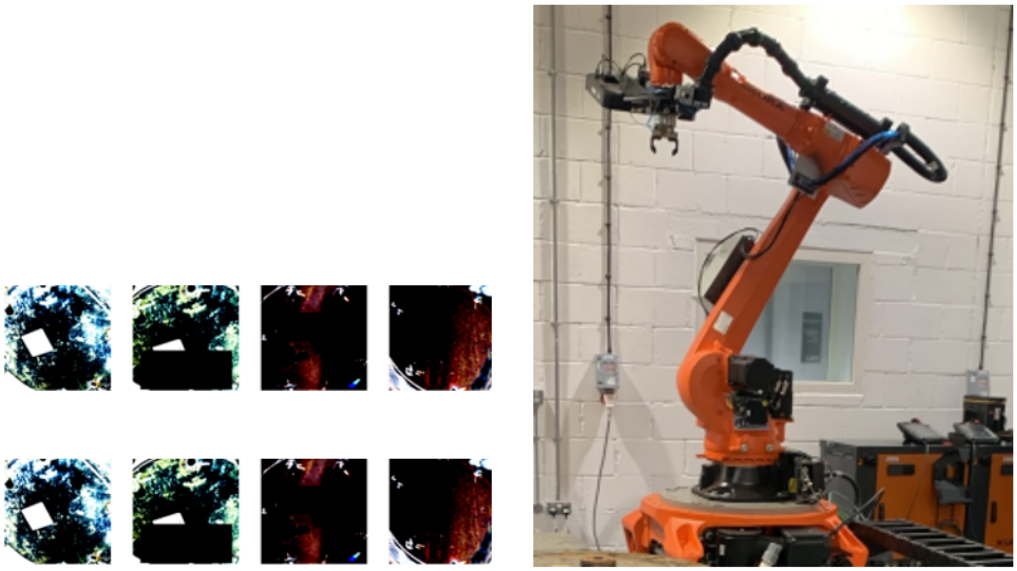

Figure 11. Lightweight CNN via KUKA industrial robotic arm.

6. Challenges and the Future Path

This work focused on presenting a holistic review of the existing state of computer vision from an algorithmic and deployment point of view. Based on the analysis presented, research trends, potential focus areas and future directions are summarised.

Proliferation of object detection-based architectures: It is clear from the analysis: the vast computer vision research industry is focused on the implementation and advancement of object detector algorithms. There are multiple reasons for this. Firstly, it is more effective in practical scenarios as compared to image classification. This is due to the ability of object detectors to not only determine if the object of interest is residing within the image frame but also extract the spatial dimensions of the particular object. This unlocks multiple options for various applications such as industrial manufacturing, where object detection and localisation may trigger external actuation.

Evolving variants of YOLO: Since its inception in 2015, without a doubt the YOLO family of architectures have been the most popular, reaching YOLOV8 by January 2023. The success of YOLO can be attributed to the fact that its authors have continuously focused on the optimisation of two key metrics required for deployable solutions, that is, accuracy and lightweight computation, leading to higher inference speed. Also, what seemed to be a setback for the YOLO family after the original author halted further development,

citing privacy concerns, has been a blessing in disguise with various researchers and well-renowned research groups actively and rigorously engaged in architectural optimisations. This competition can be equated to an arms race, where various research organisations are competing for superiority within the object detection arena, knowing full well the large scope of potential applications that can benefit from lightweight and real-time detection.

Dataset Quality: The quality of datasets is critical for the performance of image classification and object detection models. In industrial research, the procurement of large volumes of representative data can be difficult, expensive, and time-consuming. Data scarcity, particularly in domains like product defect detection, presents a significant challenge. Generating sufficient defect samples often necessitates the production of potentially flawed products, which incurs financial and ethical concerns. To mitigate these issues, future research should explore advanced data augmentation techniques, synthetic data generation, and the use of few- or zero-shot learning and generative AI scale their respective data samples.

Architectural Efficiency: Deploying CNNs in industrial applications, such as quality inspection and defect detection, promises significant advancements. However, these environments often face constraints on processing resources, making the deployment of computationally expensive models a challenge. Lightweight networks via strategies such as attention mechanisms and model compression techniques, such as pruning and quantisation, offer a compelling solution. These approaches can significantly reduce the computational load without sacrificing accuracy, enabling the effective and efficient implementation of CNNs in resource-constrained industrial settings.

Foundation Models: The development of generalised 'Foundation Models' with an industrial theme can expedite deployment across a broader spectrum of industrial environments, resulting in notable reductions in development costs and shorter development cycles. However, achieving generalisation remains a significant challenge. CNNs must be able to perform well on unseen data and adapt to varying conditions across different industrial settings. This requires robust training protocols, such as domain adaptation, to enhance the generalisation capabilities of the networks. Additionally, continuous learning and model updating mechanisms should be implemented to ensure that models remain accurate and reliable as new data becomes available.

Scalability: As CV architectures advance, scalability becomes a crucial factor for successful deployment in large-scale industrial settings. Achieving scalability involves several key considerations: managing substantial data volumes efficiently, maintaining high performance across diverse operational environments, and adapting architectures to various modes of operation. This includes designing modular and flexible systems that can be scaled up or down as required, optimising models to handle varying data loads with minimal performance degradation, and ensuring that infrastructure can support dynamic, real-time scaling. Strategies such as distributed computing and cloud-based resources can facilitate efficient data processing, while techniques like model compression and quantisation can reduce computational demands without compromising accuracy. Additionally, incorporating edge computing can help manage latency and improve responsiveness by processing data closer to its source. Future research should prioritise developing scalable architectures and deployment strategies that incorporate these approaches, focusing on robust resource management, adaptive algorithms and resilient infrastructure to maintain consistent performance and efficiency as applications grow and evolve.

Future Directions

Need for hardware benchmarked architectural design: Architecture design over much of the past decade has focused on acquiring high accuracy on benchmark datasets. These datasets consist of, in some cases, 1000 classes with over a million images, hence resulting in computationally demanding architectures. With the focus over the past couple of years shifting from the theoretically constrained performance optimisation towards real-world practical implementation, hardware-based benchmarks need to be developed,

expanding the competition metrics beyond accuracy to be inclusive of wider performance indicators such as computational efficacy, resource allocation and inference speed on various constrained devices such as FPGAs.

Targeting general-purpose development boards: Although FPGAs can be used for deployment purposes, they are not as widespread, user-friendly and accessible as general-purpose boards such as Raspberry PI and Arduino. These general-purpose development boards, although significantly constrained in terms of computational accommodation, can still host computationally lightweight architectures as recently shown by works such as [2]. To achieve deployment requirements, researchers during the algorithmic design phase need to focus more on parameter/block/layer compression strategies such as pruning, quantisation and model compression.

Privacy Concerns: As computer vision research makes its way from academic circles into real-world applications, the architectural development stage, in addition to accuracy and lightweight computation, must cater for privacy concerns along with explainable models. This will be critical if the computer vision solutions are to penetrate and integrate themselves into critical applications within industries such as healthcare, security, manufacturing and renewable energy. Hence, developers/researchers need to account for the security factor when developing their design footprints, providing confidence to potential clients, in particular within sensitive domains such as healthcare.

7. Conclusions

In conclusion, this review has provided a comprehensive examination of the architectural and hardware advancements in the field of computer vision, with a specific focus on object detection. Through our analysis, we have illustrated the remarkable pace of evolution seen in CNN architectures and AI accelerators, exemplified by architectural variants such as the YOLO family, which has experienced significant refinements in just seven years. Furthermore, we have showcased the wide-ranging industrial applications of these architectures, spanning from the detection of fabric defects in production lines to the automated inspection of industrial pallet racking in modern warehouses.

Our findings emphasise the growing tendency towards designing architectures that prioritise resource efficiency and smooth integration with deployment requirements, right from the initial architectural design phase. This shift in focus highlights the need for hardware-benchmarked architectural designs and the development of computationally lightweight models suitable for deployment on general-purpose development boards. Additionally, we have highlighted the obligation of addressing privacy concerns and promoting explainable architectures alongside considerations of accuracy and computational efficiency, particularly in sensitive and precision-demanding domains.

Future Anticipations

Looking ahead, several prominent themes of research emerge from our survey. Firstly, there is a urgent need for the further exploration and optimisation of object detection algorithms, particularly in the context of lightweight footprints for emerging applications such as autonomous vehicles, smart surveillance systems, and precision agriculture. Moreover, the amalgamation of object detection techniques with complementary domains such as natural language processing (NLP), robotics and augmented reality (AR) presents exciting avenues for interdisciplinary research and innovation. For instance, the blend of object detection with NLP can facilitate more intuitive human–computer interactions, while its incorporation with robotics holds promise for improving automation and efficiency in various industrial settings.

Furthermore, the application of object detection transcends conventional visual domains, with potential use cases in medical imaging, environmental monitoring and remote sensing. By leveraging object detection strategies, researchers can contribute to advancements in disease diagnosis, disaster response and environmental conservation efforts. In addition, the advent of edge computing and IoT presents new opportunities for deploy-

ing object detection architectures in resource-constrained environments, enabling real-time decision making and intelligent automation at the network edge. In essence, the future of object detection research lies in utilising its potential to address real-world challenges across a diverse set of domains, while concurrently advancing the fundamental principles of computer vision. By adopting interdisciplinary collaboration and innovation, researchers can unlock new frontiers in object detection, paving the way for transformative applications that improve our lives and shape the future of visual automation.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No data is generated as this is a review article evaluating existing data.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Stipic, A.; Bronzin, T.; Prole, B.; Pap, K. Deep Learning Advancements: Closing the Gap. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2019; pp. 1087–1092.
2. Hussain, M.; Al-Aqrabi, H.; Hill, R. Statistical Analysis and Development of an Ensemble-Based Machine Learning Model for Photovoltaic Fault Detection. *Energies* **2022**, *15*, 5492. [CrossRef]
3. Hussain, M.; Chen, T.; Titrenko, S.; Su, P.; Mahmud, M. A Gradient Guided Architecture Coupled With Filter Fused Representations for Micro-Crack Detection in Photovoltaic Cell Surfaces. *IEEE Access* **2022**, *10*, 58950–58964. [CrossRef]
4. Tariq, M.I.; Memon, N.A.; Ahmed, S.; Tayyaba, S.; Mushtaq, M.T.; Mian, N.A.; Imran, M.; Ashraf, M.W. A Review of Deep Learning Security and Privacy Defensive Techniques. *Mob. Inf. Syst.* **2020**, *2020*, 1–18. [CrossRef]
5. Abdullah, M.; Fraz, M.M.; Barman, S.A. Localization and segmentation of optic disc in retinal images using circular Hough transform and grow-cut algorithm. *PeerJ* **2016**, *4*, e2003. [CrossRef] [PubMed]
6. Hussain, M. Exudate Detection: Integrating Retinal-Based Affine Mapping and Design Flow Mechanism to Develop Lightweight Architectures. *IEEE Access* **2023**, *11*, 125185–125203. [CrossRef]
7. Chai, J.; Zeng, H.; Li, A.; Ngai, E.W.T. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **2021**, *6*, 100134. [CrossRef]
8. Hussain, M.; Bird, J.J.; Faria, D.R. A Study on CNN Transfer Learning for Image Classification. In *Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, Nottingham, UK, 5–7 September 2018*; Springer: Cham, Switzerland, 2018.
9. Du, J. Understanding of Object Detection Based on CNN Family and YOLO. *J. Physics Conf. Ser.* **2018**, *1004*, 012029. [CrossRef]
10. Yang, R.; Yu, Y. Artificial Convolutional Neural Network in Object Detection and Semantic Segmentation for Medical Imaging Analysis. *Front. Oncol.* **2021**, *11*, 638182. [CrossRef] [PubMed]
11. Haupt, J.; Nowak, R. Compressive Sampling Vs. Conventional Imaging. In Proceedings of the 2006 International Conference on Image Processing, Atlanta, GA, USA, 8–11 October 2006; pp. 1269–1272. [CrossRef]
12. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef]
13. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; pp. 1–6. [CrossRef]
14. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
15. Strigl, D.; Kofler, K.; Podlipnig, S. Performance and Scalability of GPU-Based Convolutional Neural Networks. In Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, Pisa, Italy, 17–19 February 2010; pp. 317–324. [CrossRef]
16. Mittal, S. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput. Appl.* **2018**, *29*, 1–31. [CrossRef]
17. Lee, S.S.; Nguyen, T.D.; Meher, P.K.; Park, S.Y. Energy-Efficient High-Speed ASIC Implementation of Convolutional Neural Network Using Novel Reduced Critical-Path Design. *IEEE Access* **2022**, *10*, 34032–34045. [CrossRef]
18. Qi, S.; Yang, J.; Zhong, Z. A review on industrial surface defect detection based on deep learning technology. In Proceedings of the 2020 3rd International Conference on Machine Learning and Machine Intelligence, Hangzhou, China, 18–20 September 2020; pp. 24–30.
19. Cumbajin, E.; Rodrigues, N.; Costa, P.; Miragaia, R.; Frazão, L.; Costa, N.; Fernández-Caballero, A.; Carneiro, J.; Buruberri, L.H.; Pereira, A. A Systematic Review on Deep Learning with CNNs Applied to Surface Defect Detection. *J. Imaging* **2023**, *9*, 193. [CrossRef] [PubMed]

20. Ghimire, D.; Kil, D.; Kim, S.H. A survey on efficient convolutional neural networks and hardware acceleration. *Electronics* **2022**, *11*, 945. [CrossRef]
21. Capra, M.; Bussolino, B.; Marchisio, A.; Shafique, M.; Masera, G.; Martina, M. An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks. *Future Internet* **2020**, *12*, 113. [CrossRef]
22. Zahid, A.; Hussain, M.; Hill, R.; Al-Aqrabi, H. Lightweight Convolutional Network For Automated Photovoltaic Defect Detection. In Proceedings of the 2023 9th International Conference on Information Technology Trends (ITT), Dubai, United Arab Emirates, 24–25 May 2023; pp. 133–138.
23. Aydin, B.A.; Hussain, M.; Hill, R.; Al-Aqrabi, H. Domain Modelling For A Lightweight Convolutional Network Focused On Automated Exudate Detection in Retinal Fundus Images. In Proceedings of the 2023 9th International Conference on Information Technology Trends (ITT), Dubai, United Arab Emirates, 24–25 May 2023; pp. 145–150.
24. Hussain, M.; Hill, R. Custom Lightweight Convolutional Neural Network Architecture for Automated Detection of Damaged Pallet Racking in Warehousing & Distribution Centers. *IEEE Access* **2023**, *11*, 58879–58889. [CrossRef]
25. Hussain, M.; Al-Aqrabi, H.; Munawar, M.; Hill, R. Feature Mapping for Rice Leaf Defect Detection Based on a Custom Convolutional Architecture. *Foods* **2022**, *11*, 3914. [CrossRef] [PubMed]
26. Lowe, D. Object recognition from local scale-invariant features. In Proceedings of the International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2.
27. Oliva, A.; Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **2001**, *42*, 145–175. [CrossRef]
28. Rakotomamonjy, A.; Gasso, G. Histogram of gradients of Time-Frequency Representations for Audio Scene Detection. *IEEE/ACM Trans. Audio, Speech, Lang. Process.* **2014**, *23*, 142–153. [CrossRef]
29. Daniilidis, K.; Maragos, P.; Paragios, N. Improving the Fisher kernel for large-scale image classification. In Proceedings of the European Conference on Computer Vision, Crete, Greece, 5–11 September 2010; pp. 143–156.
30. Li, F.F.; Perona, P. A Bayesian hierarchical model for learning natural scene categories. In Proceedings of the Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005.
31. Cortes, C.; Vapnik, V. Support-vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
32. Krizhevsky, A.; Sutskever, I.; Hinton, G. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
33. Perez, H.; Tah, J.H.M.; Mosavi, A. Deep Learning for Detecting Building Defects Using Convolutional Neural Networks. *Sensors* **2019**, *19*, 3556. [CrossRef]
34. Eckle, K.; Schmidt-Hieber, J. A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Netw.* **2019**, *110*, 232–242. [CrossRef] [PubMed]
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
36. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
37. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
38. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
40. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
41. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
42. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA,18–23 June 2018.
43. Zoph, B.; Le, Q. Neural architecture search with reinforcement learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
44. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: Alexnet-Level Accuracy with 50x Fewer Parameters and <0.5 Mb Model Size. *arXiv* **2016**, arXiv:1602.07360.
45. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Wey, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
46. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA,18–23 June 2018.
47. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
48. Khan, Z.Y.; Niu, Z. CNN with depthwise separable convolutions and combined kernels for rating prediction. *Expert Syst. Appl.* **2021**, *170*, 114528. [CrossRef]

49. Wu, B.; Wan, A.; Yue, X.; Jin, P.; Zhao, S.; Golmant, N.; Gholaminejad, A.; Gonzalez, J.; Keutzer, K. Shift: A Zero Flop, Zero Parameter Alternative to Spatial Convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA,18–23 June 2018.
50. Chen, W.; Xie, D.; Zhang, Y.; Pu, S. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
51. Viola, P.; Jones, M.J. Robust real-time face detection. *Int. J. Comput. Vis.* **2004**, *57*, 137–154. [CrossRef]
52. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef]
53. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005; Volume 1.
54. Ahonen, T.; Hadid, A.; Pietikainen, M. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 2037–2041. [CrossRef]
55. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
56. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014.
57. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
58. Girshick, R. Fast R-CNN. In Proceedings of the International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1137–1149.
59. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
60. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
61. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
62. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016.
63. Redmon, J.; Ali, A. YOLO9000: Better, faster, stronger. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
64. Redmon, J.; Ali, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
65. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
66. ultralytics/yolov5: V3.0. 2020. Available online: https://zenodo.org/records/3983579 (accessed on 11 Feburary 2024).
67. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976.
68. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y. YOLOv6. GitHub, 2022. Available online: https://github.com/meituan/YOLOv6 (accessed on 3 June 2024).
69. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
70. Sohan, M.; Sai Ram, T.; Reddy, R.; Venkata, C. A Review on YOLOv8 and Its Advancements. In Proceedings of the International Conference on Data Intelligence and Cognitive Informatics, Tirunelveli, India, 27–28 June 2023; pp. 529–545.
71. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv* **2024**, arXiv:2402.13616.
72. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 390–391.
73. Wang, C.Y.; Liao, H.Y.M.; Yeh, I.H. Designing network design strategies through gradient path analysis. *arXiv* **2022**, arXiv:2211.04800.
74. Wong, K.Y. YOLOv9 GitHub Repository. 2024. Available online: https://github.com/WongKinYiu/yolov9 (accessed on 3 June 2024).
75. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. *arXiv* **2024**, arXiv:2405.14458.
76. Ultralytics. YOLOv10 Documentation: Model Variants. Available online: https://docs.ultralytics.com/models/yolov10/#model-variants (accessed on 3 June 2024).
77. Neshatpour, K.; Malik, M.; Ghodrat, M.A.; Sasan, A.; Homayoun, H. Energy-efficient acceleration of big data analytics applications using FPGAs. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 115–123. [CrossRef]

78. Kontorinis, V.; Zhang, L.; Aksanli, B.; Sampson, J.; Homayoun, H.; Pettis, E. Managing distributed ups energy for effective power capping in data centers. *ACM SIGARCH Comput. Archit. News* **2012**, *40*, 488–499. [CrossRef]
79. Hardavellas, N.; Ferdman, M.; Falsafi, B.; Ailamaki, A. Toward dark silicon in servers. *IEEE Micro* **2011**, *31*, 6–15. [CrossRef]
80. Yan, C.; Yue, T. A Novel Method for Dynamic Modelling and Real-time Rendering Based on GPU. *Geo-Inf. Sci.* **2012**, *14*, 149–157. [CrossRef]
81. Brodtkorb, A.R.; Hagen, T.R.; Sætra, M.L. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *J. Parallel Distrib. Comput.* **2013**, *73*, 4–13. [CrossRef]
82. Barrett, R.; Chakraborty, M.; Amirkulova, D.; Gandhi, H.; Wellawatte, G.; White, A. HOOMD-TF: GPU-Accelerated, Online Machine Learning in the HOOMD-blue Molecular Dynamics Engine. *J. Open Source Softw.* **2020**, *5*, 2367. [CrossRef]
83. Ma, H. Development of a CPU-GPU heterogeneous platform based on a nonlinear parallel algorithm. *Nonlinear Eng.* **2022**, *11*, 215–222. [CrossRef]
84. Stone, J.E.; Gohara, D.; Shi, G. OpenCL: A parallel programming standard for heterogeneous computing systems. *Comput. Sci. Eng.* **2010**, *12*, 66–73. [CrossRef] [PubMed]
85. Garland, M.; Le Grand, S.; Nickolls, J.; Anderson, J.; Hardwick, J.; Morton, S.; Phillips, E.; Zhang, Y.; Volkov, V. Parallel computing experiences with CUDA. *IEEE Micro* **2008**, *28*, 13–27. [CrossRef]
86. Halvorsen, M. Hardware Acceleration of Convolutional Neural Networks. Master's Thesis, Norwegian University of Science Technology, Trondheim, Norway, 2015.
87. Chetlur, S.; Woolley, C.; Vandermersch, P.; Cohen, J.; Tran, J.; Catanzaro, B.; Shelhamer, E. CUDNN: Efficient Primitives for Deep Learning. *arXiv* **2014**, arXiv:1410.0759.
88. Cudaconvnet2. Available online: https://code.google.com/archive/p/cuda-convnet2/ (accessed on 3 June 2024).
89. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014.
90. TensorFlow. Available online: https://www.tensorflow.org/ (accessed on 3 June 2024).
91. Collobert, R.; Kavukcuoglu, K.; Farabet, C. Torch7: A MATLAB-like environment for machine learning. In Proceedings of the Conference on Neural Information Processing System Workshop, Granada, Spain, 12–15 December 2011.
92. Mittal, S. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. *J. Syst. Archit.* **2019**, *97*, 428–442. [CrossRef]
93. Jin, R.; Niu, Q. Automatic Fabric Defect Detection Based on an Improved YOLOv5. *Math. Probl. Eng.* **2021**, *2021*, 7321394. [CrossRef]
94. Raspberry Pi 4 Model B. Available online: https://thepihut.com/collections/raspberry-pi/products/raspberry-pi-4-model-b (accessed on 25 May 2022).
95. Mohamad Noor, M.b.; Hassan, W.H. Current research on Internet of Things (IoT) security: A survey. *Comput. Netw.* **2019**, *148*, 283–294. [CrossRef]
96. Frank, A.G.; Dalenogare, L.S.; Ayala, N.F. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *Int. J. Prod. Econ.* **2019**, *210*, 15–26. [CrossRef]
97. Farooq, U.; Marrakchi, Z.; Mehrez, H. FPGA Architectures: An Overview. In *Tree-based Heterogeneous FPGA Architectures*; Springer: New York, NY, USA, 2012. [CrossRef]
98. Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; et al. Going deeper with embedded FPGA platform for convolutional neural network. In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016.
99. Nurvitadhi, E.; Venkatesh, G.; Sim, J.; Marr, D.; Huang, R.; Ong Gee Hock, J.; Liew, Y.T.; Srivatsan, K.; Moss, D.; Subhaschandra, S.; et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017.
100. Liu, Y.; Liu, P.; Jiang, Y.; Yang, M.; Wu, K.; Wang, W.; Yao, Q. Building a multi-fpga-based emulation framework to support noc design and verification. *Int. J. Electron.* **2010**, *97*, 1241–1262. [CrossRef]
101. Dondon, P.; Carvalho, J.; Gardere, R.; Lahalle, P.; Tsenov, G.; Mladenov, V. Implementation of a feed-forward Artificial Neural Network in VHDL on FPGA. In Proceedings of the 12th Symposium on Neural Network Applications in Electrical Engineering (NEUREL), Belgrade, Serbia, 25–27 November 2014. [CrossRef]
102. Ünsalan, C.; Tar, B. *Digital System Design with FPGA: Implementation Using Verilog and VHDL*; McGraw-Hill Education: New York, NY, USA, 2017.
103. Zhao, R.; Song, W.; Zhang, W.; Xing, T.; Lin, J.H.; Srivastava, M.; Gupta, R.; Zhang, Z. Accelerating binarized convolutional neural networks with software-programmable FPGAs. In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017.
104. Wei, X.; Liang, Y.; Cong, J. Overcoming Data Transfer Bottlenecks in FPGA-based DNN Accelerators via Layer Conscious Memory Management. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019.
105. Abtahi, T.; Shea, C.; Kulkarni, A.; Mohsenin, T. Accelerating Convolutional Neural Network With FFT on Embedded Hardware. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1737–1749. [CrossRef]

106. Kala, S.; Jose, B.R.; Mathew, J.; Nalesh, S. High-Performance CNN Accelerator on FPGA Using Unified Winograd-GEMM Architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 2816–2828. [CrossRef]
107. Lavin, A.; Gray, S. Fast algorithms for convolutional neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
108. Bottleson, J.; Kim, S.; Andrews, J.; Bindu, P.; Murthy, D.N.; Jin, J. CLCAFFE: OpenCL accelerated CAFFE for convolutional neural networks. In Proceedings of the International Parallel and Distributed Processing Symposium Workshops, Chicago, IL, USA, 23–27 May 2016.
109. Winograd, S. *Arithmetic Complexity of Computations*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1980.
110. DiCecco, R.; Lacey, G.; Vasiljevic, J.; Chow, P.; Taylor, G.; Areibi, S. Caffeinated FPGAs: FPGA Framework for Convolutional Neural Networks. In Proceedings of the Field-Programmable Technology, Xi'an, China, 7–9 December 2016.
111. Sankaradas, M.; Jakkula, V.; Cadambi, S.; Chakradhar, S.; Durdanovic, I.; Cosatto, E.; Graf, H.P. A Massively Parallel Coprocessor for Convolutional Neural Networks. In Proceedings of the Application-Specific Systems, Architectures and Processors, Boston, MA, USA, 7–9 July 2009.
112. Chakradhar, S.; Sankaradas, M.; Jakkula, V.; Cadambi, S. A dynamically configurable coprocessor for convolutional neural networks. In Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 19–23 June 2010; pp. 247–257.
113. Farabet, C.; Martini, B.; Corda, B.; Akselrod, P.; Culurciello, E.; LeCun, Y. Neuflow: A runtime reconfigurable dataflow processor for vision. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011.
114. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015.
115. Rahman, A.; Oh, S.; Lee, J.; Choi, K. Design Space Exploration of FPGA Accelerators for Convolutional Neural Networks. In Proceedings of the Design, Automation & Test in Europe, Lausanne, Switzerland, 27–31 March 2017.
116. Li, Y.; Liu, Z.; Xu, K.; Yu, H.; Ren, F. A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks. *J. Emerg. Technol. Comput. Syst.* **2018**, *14*, 18. [CrossRef]
117. Derrien, S.; Rajopadhye, S. Loop tiling for reconfigurable accelerators. In Proceedings of the Conference on Field Programmable Logic and Applications, Belfast, UK, 27–29 August 2001.
118. Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; Pensky, M. Sparse convolutional neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
119. Courbariaux, M.; Bengio, Y.; David, J.P. Training Deep Neural Networks with Low Precision Multiplications. *arXiv* **2014**, arXiv:1412.7024.
120. Zhang, X.; Liu, X.; Ramachandran, A.; Zhuge, C.; Tang, S.; Ouyang, P.; Cheng, Z.; Rupnow, K.; Chen, D. High-performance video content recognition with long-term recurrent convolutional network for FPGA. In Proceedings of the Conference on Field Programmable Logic and Applications, Ghent, Belgium, 4–8 September 2017.
121. Yang, T.J.; Chen, Y.H.; Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
122. Page, A.; Jafari, A.; Shea, C.; Mohsenin, T. SPARCNet: A hardware accelerator for efficient deployment of sparse convolutional networks. *J. Emerg. Technol. Comput. Syst.* **2017**, *13*, 31. [CrossRef]
123. Rigamonti, R.; Sironi, A.; Lepetit, V.; Fua, P. Learning separable filters. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013.
124. Ma, Y.; Cao, Y.; Vrudhula, S.; Seo, J.S. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017.
125. Suda, N.; Chandra, V.; Dasika, G.; Mohanty, A.; Ma, Y.; Vrudhula, S.; Seo, J.S.; Cao, Y. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. In Proceedings of the International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016.
126. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained To±1. *arXiv* **2016**, arXiv:1602.02830.
127. Rui, J.; Niu, Q. Research on textile defects detection based on improved generative adversarial network. *J. Eng. Fibers Fabr.* **2022**, *17*, 15589250221101382. [CrossRef]
128. Qin, Y.; Purdy, R.; Probst, A.; Lin, C.Y.; Zhu, J.G. ASIC Implementation of Non-linear CNN-based Data Detector for TDMR System in 28nm CMOS at 200Mbits/s Throughput. *IEEE Trans. Magn.* **2022**, *59*, 1–8. [CrossRef]
129. HUAWEI. *HUAWEI Reveals the Future of Mobile AI and IFA 2017*; HUAWEI: Shenzhen, China, 2017.
130. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017.
131. Malamas, E.N.; Petrakis, E.G.; Zervakis, M.; Petit, L.; Legat, J.D. A survey on industrial vision systems, applications and tools. *Image Vis. Comput.* **2003**, *21*, 171–188. [CrossRef]

132. Zhang, J.; Jing, J.; Lu, P.; Song, S. Improved MobileNetV2-SSDLite for automatic fabric defect detection system based on cloud-edge computing. *Measurement* **2022**, *201*, 111665. [CrossRef]
133. Li, F.; Li, F. Bag of tricks for fabric defect detection based on Cascade R-CNN. *Text. Res. J.* **2020**, *91*, 599–612. [CrossRef]
134. Song, S.; Jing, J.; Huang, Y.; Shi, M. EfficientDet for fabric defect detection based on edge computing. *J. Eng. Fibers Fabr.* **2021**, *16*, 155892502110083. [CrossRef]
135. Hussain, M.; Al-Aqrabi, H.; Hill, R. PV-CrackNet Architecture for Filter Induced Augmentation and Micro-Cracks Detection within a Photovoltaic Manufacturing Facility. *Energies* **2022**, *15*, 8667. [CrossRef]
136. Dhimsih, M.; Mather, P. Development of Novel Solar Cell Micro Crack Detection Technique. *IEEE Trans. Semicond. Manuf.* **2019**, *32*, 277–285. [CrossRef]
137. Luo, Z.; Cheng, S.Y.; Zheng, Q.Y. Corrigendum: GAN-Based Augmentation for Improving CNN Performance of Classification of Defective Photovoltaic Module Cells in Electroluminescence Images. *IOP Conf. Ser. Earth Environ. Sci.* **2019**, *354*, 012132. [CrossRef]
138. Su, B.; Chen, H.; Chen, P.; Bian, G.; Liu, K.; Liu, W. Deep Learning-Based Solar-Cell Manufacturing Defect Detection With Complementary Attention Network. *IEEE Trans. Ind. Inform.* **2021**, *17*, 4084–4095. [CrossRef]
139. Ahmad, A.; Jin, Y.; Zhu, C.; Javed, I.; Maqsood, A.; Akram, M.W. Photovoltaic cell defect classification using convolutional neural network and support vector machine. *IET Renew. Power Gener.* **2020**, *14*, 2693–2702. [CrossRef]
140. Langley, C.J.; Novack, R.A.; Gibson, B.J.; Coyle, J.J. *Supply Chain Management: A Logistics Perspective*, 11th ed.; Cengage Learning: Boston, MA, USA, 2020.
141. Hussain, M.; Chen, T.; Hill, R. Moving toward Smart Manufacturing with an Autonomous Pallet Racking Inspection System Based on MobileNetV2. *J. Manuf. Mater. Process.* **2022**, *6*, 75. [CrossRef]
142. Hussain, M.; Al-Aqrabi, H.; Munawar, M.; Hill, R.; Alsboui, T. Domain Feature Mapping with YOLOv7 for Automated Edge-Based Pallet Racking Inspections. *Sensors* **2022**, *22*, 6927. [CrossRef] [PubMed]
143. Farahnakian, F.; Koivunen, L.; Mäkilä, T.; Heikkonen, J. Towards Autonomous Industrial Warehouse Inspection. In Proceedings of the 2021 26th International Conference on Automation and Computing (ICAC), Portsmouth, UK, 2–4 September 2021. [CrossRef]
144. Hussain, M. YOLO-v5 Variant Selection Algorithm Coupled with Representative Augmentations for Modelling Production-Based Variance in Automated Lightweight Pallet Racking Inspection. *Big Data Cogn. Comput.* **2023**, *7*, 120. [CrossRef]
145. Tao, X.; Zhang, D.; Ma, W.; Liu, X.; Xu, D. Automatic metallic surface defect detection and recognition with convolutional neural networks. *Appl. Sci.* **2018**, *8*, 1575. [CrossRef]
146. Xu, Y.; Zhang, K.; Wang, L. Metal surface defect detection using modified YOLO. *Algorithms* **2021**, *14*, 257. [CrossRef]
147. Lin, H.I.; Wibowo, F.S. Image data assessment approach for deep learning-based metal surface defect-detection systems. *IEEE Access* **2021**, *9*, 47621–47638. [CrossRef]
148. Xin, H.; Chen, Z.; Wang, B. PCB electronic component defect detection method based on improved YOLOv4 algorithm. *J. Phys. Conf. Ser.* **2021**, *1827*, 012167. [CrossRef]
149. Jeon, M.; Yoo, S.; Kim, S.W. A contactless PCBA defect detection method: Convolutional neural networks with thermographic images. *IEEE Trans. Components, Packag. Manuf. Technol.* **2022**, *12*, 489–501. [CrossRef]
150. Santoso, A.D.; Cahyono, F.B.; Prahasta, B.; Sutrisno, I.; Khumaidi, A. Development of PCB Defect Detection System Using Image Processing With YOLO CNN Method. *Int. J. Artif. Intell. Res.* **2022**, *6*.
151. Wang, S.; Wu, L.; Wu, W.; Li, J.; He, X.; Song, F. Optical fiber defect detection method based on DSSD network. In Proceedings of the 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, 9–11 August 2019; pp. 422–426.
152. Mei, S.; Cai, Q.; Gao, Z.; Hu, H.; Wen, G. Deep learning based automated inspection of weak microscratches in optical fiber connector end-face. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–10. [CrossRef]
153. Han, K.; Sun, M.; Zhou, X.; Zhang, G.; Dang, H.; Liu, Z. A new method in wheel hub surface defect detection: Object detection algorithm based on deep learning. In Proceedings of the 2017 International Conference on Advanced Mechatronic Systems (ICAMechS), Xiamen, China, 6–9 December 2017; pp. 335–338.
154. Sun, X.; Gu, J.; Huang, R.; Zou, R.; Giron Palomares, B. Surface defects recognition of wheel hub based on improved faster R-CNN. *Electronics* **2019**, *8*, 481. [CrossRef]
155. Cheng, S.; Lu, J.; Yang, M.; Zhang, S.; Xu, Y.; Zhang, D.; Wang, H. Wheel hub defect detection based on the DS-Cascade RCNN. *Measurement* **2023**, *206*, 112208. [CrossRef]
156. Lin, H.; Li, B.; Wang, X.; Shu, Y.; Niu, S. Automated defect inspection of LED chip using deep convolutional neural network. *J. Intell. Manuf.* **2019**, *30*, 2525–2534. [CrossRef]
157. Stern, M.L.; Schellenberger, M. Fully convolutional networks for chip-wise defect detection employing photoluminescence images: Efficient quality control in LED manufacturing. *J. Intell. Manuf.* **2021**, *32*, 113–126. [CrossRef]
158. Zheng, P.; Lou, J.; Wan, X.; Luo, Q.; Li, Y.; Xie, L.; Zhu, Z. LED Chip Defect Detection Method Based on a Hybrid Algorithm. *Int. J. Intell. Syst.* **2023**, *2023*. [CrossRef]
159. Koodtalang, W.; Sangsuwan, T.; Sukanna, S. Glass bottle bottom inspection based on image processing and deep learning. In Proceedings of the 2019 Research, Invention, and Innovation Congress (RI2C), Bangkok, Thailand, 11–13 December 2019; pp. 1–5.
160. Zhang, X.; Yan, L.; Yan, H. Defect detection of bottled liquor based on deep learning. In Proceedings of the CSAA/IET International Conference on Aircraft Utility Systems, Online, 18–21 September 2020; IET: Stevenage, UK, 2021. [CrossRef]

161. Gizaw, A.; Kebebaw, T. Water Bottle Defect Detection System Using Convolutional Neural Network. In Proceedings of the 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 28–30 November 2022; pp. 19–24.
162. Qu, Z.; Shen, J.; Li, R.; Liu, J.; Guan, Q. Partsnet: A unified deep network for automotive engine precision parts defect detection. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, 2018; pp. 594–599.
163. Yang, T.; Xiao, L.; Gong, B.; Huang, L. Surface defect recognition of varistor based on deep convolutional neural networks. In Proceedings of the Optoelectronic Imaging and Multimedia Technology VI, Hangzhou, China, 20–23 October 2019; Volume 11187; pp. 267–274.
164. Yang, T.; Peng, S.; Huang, L. Surface defect detection of voltage-dependent resistors using convolutional neural networks. *Multimed. Tools Appl.* **2020**, *79*, 6531–6546. [CrossRef]
165. Stephen, O.; Maduh, U.J.; Sain, M. A machine learning method for detection of surface defects on ceramic tiles using convolutional neural networks. *Electronics* **2021**, *11*, 55. [CrossRef]
166. Lu, F.; Zhang, Z.; Guo, L.; Chen, J.; Zhu, Y.; Yan, K.; Zhou, X. HFENet: A lightweight hand-crafted feature enhanced CNN for ceramic tile surface defect detection. *Int. J. Intell. Syst.* **2022**, *37*, 10670–10693. [CrossRef]
167. Wan, G.; Fang, H.; Wang, D.; Yan, J.; Xie, B. Ceramic tile surface defect detection based on deep learning. *Ceram. Int.* **2022**, *48*, 11085–11093. [CrossRef]
168. Shi, J.; Li, Z.; Zhu, T.; Wang, D.; Ni, C. Defect detection of industry wood veneer based on NAS and multi-channel mask R-CNN. *Sensors* **2020**, *20*, 4398. [CrossRef]
169. Chen, L.C.; Pardeshi, M.S.; Lo, W.T.; Sheu, R.K.; Pai, K.C.; Chen, C.Y.; Tsai, P.Y.; Tsai, Y.T. Edge-glued wooden panel defect detection using deep learning. *Wood Sci. Technol.* **2022**, *56*, 477–507. [CrossRef]
170. Lim, W.H.; Bonab, M.B.; Chua, K.H. An Aggressively Pruned CNN Model With Visual Attention for Near Real-Time Wood Defects Detection on Embedded Processors. *IEEE Access* **2023**, *11*, 36834–36848. [CrossRef]
171. Huang, Y.; Qiu, C.; Yuan, K. Surface defect saliency of magnetic tile. *Vis. Comput.* **2020**, *36*, 85–96. [CrossRef]
172. Soukup, D.; Huber-Mörk, R. Convolutional neural networks for steel surface defect detection from photometric stereo images. In Proceedings of the International Symposium on Visual Computing, Las Vegas, NV, USA, 8–10 December 2014; pp. 668–677.
173. Cha, Y.J.; Choi, W.; Büyüköztürk, O. Deep learning-based crack damage detection using convolutional neural networks. *Comput.-Aided Civ. Infrastruct. Eng.* **2017**, *32*, 361–378. [CrossRef]
174. Cha, Y.J.; Choi, W.; Suh, G.; Mahmoudkhani, S.; Büyüköztürk, O. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput.-Aided Civ. Infrastruct. Eng.* **2018**, *33*, 731–747. [CrossRef]
175. Cognex Corporation. Cognex Corporation—Machine Vision and Industrial Barcode Reading Products. 2024. Available online: https://www.cognex.com/en-gb/products.aspx?langtype= (accessed on 11 June 2024).
176. Keyence Corporation. Keyence Corporation—Sensors and Machine Vision Systems. 2024. Available online: https://www.keyence.co.uk/products/vision/vision-sys/ (accessed on 12 June 2024).
177. Cognex Corporation. In-Sight D900—Deep Learning Vision System. 2024. Available online: https://www.cognex.com/en-gb/products/machine-vision/2d-machine-vision-systems/in-sight-9000-series (accessed on 11 June 2024).
178. Alif, M.A.R.; Hussain, M. Lightweight Convolutional Network with Integrated Attention Mechanism for Missing Bolt Detection in Railways. *Metrology* **2024**, *4*, 254–278. [CrossRef]
179. Dziubek, M.; Rysiński, J.; Jancarczyk, D. Exploring the ViDiDetect Tool for Automated Defect Detection in Manufacturing with Machine Vision. *Appl. Sci.* **2023**, *13*, 11098. [CrossRef]