IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received December 27, 2021, accepted January 25, 2022, date of publication January 28, 2022, date of current version February 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3147495

Toward Performing Image Classification and Object Detection With Convolutional Neural Networks in Autonomous Driving Systems: A Survey

TOLGA TURAY AND TANYA VLADIMIROVA, (Senior Member, IEEE)
School of Engineering, University of Leicester, Leicester LE1 7RH, U.K.

Corresponding author: Tanya Vladimirova (tv29@leicester.ac.uk)

This work was supported in part by the Ministry of National Education, Republic of Turkey; and in part by the University of Leicester, U.K.

ABSTRACT Nowadays Convolutional Neural Networks (CNNs) are being employed in a wide range of industrial technologies for a variety of sectors, such as medical, automotive, aviation, agriculture, space, etc. This paper reviews the state-of-the-art in both the field of CNNs for image classification and object detection and Autonomous Driving Systems (ADSs) in a synergetic way. Layer-based details of CNNs along with parameter and floating-point operation number calculations are outlined. Using an evolutionary approach, the majority of the outstanding image classification CNNs, published in the open literature, is introduced with a focus on their accuracy performance, parameter number, model size, and inference speed, highlighting the progressive developments in convolutional operations. Results of a novel investigation of the convolution types and operations commonly used in CNNs are presented, including a timing analysis aimed at assessing their impact on CNN performance. This extensive experimental study provides new insight into the behaviour of each convolution type in terms of training time, inference time, and layer level decomposition. Building blocks for CNN-based object detection are also discussed, such as backbone networks and baseline types, and then representative state-of-the-art designs are outlined. Experimental results from the literature are summarised for each of the reviewed models. This is followed by an overview of recent ADSs related works and current industry activities, aiming to bridge academic research and industry practice on CNNs and ADSs. Design approaches targeted at solving problems of automakers in achieving real-time implementations are also proposed based on a discussion of design constraints, human vs. machine evaluations and trade-off analysis of accuracy vs. size. Current technologies, promising directions, and expectations from the literature on ADSs are introduced including a comprehensive trade-off analysis from a human-machine perspective.

INDEX TERMS Autonomous vehicles, computer vision, convolution, convolutional neural networks, embedded systems, object detection, image classification.

I. INTRODUCTION

In recent years, Deep Learning (DL) techniques have been exhaustively utilised in a large variety of fields, and Convolutional Neural Networks (CNNs) are one of the most frequently used of them in solving real-time problems of computer vision tasks, as it enables most accurate acquisitions. The concept behind CNNs drew inspiration from the structure of biotic visual systems. In the 1960s, it was theorized that the cats' visual cortex is based on sensitively to smaller sub-regions in the brain, called receptive fields [1]. Two decades later, Fukushima [2] introduced the first CNN model, based upon [1] and referred to as Neocognitron, which was simulated and implemented on a computer. This network, designed by using multi-layer artificial neuron connections for the transformation of images, is still considered as the source of inspiration for CNNs. Then, LeCun *et al.* [3], [4] created a CNN model, referred to as LeNet-5, which specifically classifies handwritten digits and can be trained to recognize patterns from raw pixels through using the back-propagation algorithm [5].

Despite all the innovative approaches, LeNet-5 [4] was ineffective in the implementation of complex problems, such as image classification, requiring large training data

The associate editor coordinating the review of this manuscript and approving it for publication was Yonggang Liu.

and powerful processors for computation. Advancements in hardware accelerators, such as Graphical Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), etc., made these devices a suitable implementation choice for Machine Learning (ML) techniques [6]. Becoming widespread in the early 2010s, CNN models along with efficient training methods were released and implemented on GPUs [7]–[9]. Following that revolution, many CNN algorithms have been proposed addressing the needs of different areas, with computer vision and natural language processing seen as major application fields. Computer vision tasks to which CNNs have mostly been applied are image classification, object detection, face recognition, scene labelling, action recognition, human pose estimation, and document analysis. As for natural language processing tasks, the two main prevalent fields are speech recognition and text classification. Nowadays CNNs are being employed in a wide range of industrial technologies for a variety of sectors, such as medical, automotive, aviation, agriculture, space etc.

A. CNN DEPLOYMENT IN AUTOMOTIVE

Along with recent technological advancements, consumer expectations from vehicles have evolved to extending their digital lifestyles into vehicle cabins and connecting with outside worlds. Concurrently, growing traffic congestion, rise in numbers of new drivers, and changing priorities have directed the focus on safety-based systems as well. As a result, automakers have started to transition from horsepower to digital technology as a standard feature, enabling self-driving capabilities in the production of automobiles, which will be referred to as Autonomous Driving Systems (ADSs) in the rest of the paper.

In the near future, the next generation of ADSs promises an advanced level of self-driving experience. To enable this advance, the industry needs to combine multiple technologies into a single system, which has the capability of communicating, collaborating, and eventually performing human functions for almost all driving scenarios. According to Intel [10], all these capabilities need numerous sensors, such as radar, lidar, GPS, and advanced camera technologies, capable of collecting the necessary information of $\sim$1 GB/second. This data should be processed in real-time in order to make safe decisions for the vehicle on the road. Despite the advantages of lidars, such as high precision and 3D mapping, their extreme cost prevents their commercial deployments in the industry. For example, existing lidars in the market are as expensive as \$75k [11]. Thereby, some companies like Tesla [12] and Mobileye [13], [14] have announced that optical cameras and radars are their preferred choice of sensing devices in vision-based parts of ADSs. From this point of view, CNNs as an image-based technique have been effectively and widely utilised by automakers in object detector modules of automobiles.

Although there has been lots of advancements in object detection with CNNs for various applications, because of the stringent real-time limitations on both safe operational decision-making and latency of data processing, ADSs are notably challenging and still under experimentation. In fact, designing an end-to-end ADS remains an unsolved research topic [15]. Furthermore, the computational pipeline system of ADSs includes various modules executing multiple tasks, and every module needs improvements on their performances [16].

B. PAPER CONTRIBUTIONS

This paper reviews the state-of-the-art on the field of CNNs for image classification and object detection, and the field of ADSs in a synergetic way. The main contributions of the paper are as follows:

The first main contribution of this paper is a detailed review of the literature on CNNs for both image classification and object detection, comparing them regarding their performance and model size. In recent years, the mainstream CNN research on image classification was focused on the design of two types of algorithms: heavyweight and lightweight algorithms. The former prioritizes accuracy performance regardless of other factors, whereas the latter aim to reduce the model size and the computational load achieving a satisfying accuracy. Therefore, these types of CNNs are reviewed as separate classes in the order of their historical evolution. The CNNs on object detection are also divided into two classes: two-stage detectors and one-stage detectors, which are discussed in a historical order too.

The second main contribution of the paper is a novel investigation of the convolution types and operations commonly used in CNNs for image classification that distinguish one model from another. The purpose is to establish the effect of these on the CNN performance. For that, their training and inference performances were evaluated using a CPU as well as two identical hardware accelerators (2xNVIDIA Pascal P100s GPUs). The evaluation was based on a reference CNN model, ResNet [17], for a fair assessment. The pursued strategy was to place and test different types of filters on the ResNet's modules. By means of experiments with two well-known datasets CIFAR-10 and CIFAR-100 [18], a runtime timing analysis was carried out by decomposing the architectures into basic components.

The third main contribution of this paper is an overview of recent ADS related works and current industry activities, in which an attempt is made to bridge academic research and industry practice on CNNs and ADSs. Design approaches aimed at solving problems of automakers in achieving real-time implementations are also proposed based on discussion of design constraints, human vs. machine evaluations and a trade-off analysis of accuracy vs. size.

In other words, this paper is an up-to-date application-based review of CNNs. Unlike other published reviews, ours involves a full investigation end-to-end. The difference between our review and the previously published ones is shown in Table 1, where it can be seen that our paper is the

TABLE 1. Comparison of existing studies in the scope of survey title.

Topics	[15]	[16]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[ours]
Layer-based overview on CNNs	-	-	✓	-	-	-	-	-	-	-	-	✓
Parameter and FLOPs [a] calculation	-	-	-	-	-	-	-	-	-	-	-	✓
Compression techniques	-	-	-	✓	-	✓	-	-	-	-	-	✓
Convolutional kernels	-	-	-	-	-	-	-	-	-	-	-	✓
Optimization	-	-	-	-	-	-	-	✓	-	-	-	✓
Learning types	-	-	✓	-	-	✓	✓	-	-	-	✓	✓
Classification & detection	✓	✓	✓	-	-	✓	✓	-	-	-	✓	✓
ADS pipeline	✓	✓	-	-	-	✓	✓	-	✓	✓	-	✓
Control	-	-	-	-	-	-	-	-	-	✓	-	-
Libraries and datasets	✓	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓
Common practices of automakers	✓	✓	-	-	-	✓	✓	-	✓	-	-	✓
Constraints in implementation	✓	-	-	-	-	✓	✓	-	✓	-	-	✓
Real-time implementation	-	-	-	-	-	-	✓	-	✓	✓	-	✓
Embedded system selection	✓	-	-	-	-	✓	✓	-	✓	✓		-
Human driver vs ADSs (real-time)	✓	-	-	-	✓	-	-	-	-	-	-	✓
Trade-off analysis	✓	-	-	-	-	-	✓	-	-	-	-	✓
Over-the-air update	-	-	-	-	-	-	-	-	✓	-	-	-
Trends and open issues	✓	✓	✓	-	✓	✓	✓	-	✓	-	✓	✓
Up to date	-	-	-	-	-	-	-	-	-	-	-	✓

[a] Floating-Point Operations

most comprehensive study in terms of the scope of the work covered.

The rest of the paper is structured as follows. Section II introduces the main components of CNNs in terms of their mathematical and architectural properties as well as frequently used datasets and libraries in the scope of the survey. Section III provides a review of CNN models for image classification and examines the different convolution and operation types in an evolutionary manner using experimentally derived execution time, while Section IV reviews CNN models for object detection. Most recent ADSs related works concerned with CNN-based classification and detection, industry status, and ADSs system architectures are outlined in Section V, which also includes a discussion on design constraints and promising new directions. Finally, Section VI concludes the paper.

II. OVERVIEW OF CNNs

CNNs are a type of layered Deep Neural Networks (DNNs), which are composed of artificial neurons. They utilise several distinctive properties compared to other neural network types, such as local receptive fields, spatial subsampling, shared weights, etc. CNNs enable cooperation throughout multiple sequential stages, which consist of a convolution, padding, and pooling operation, adding nonlinearities via activation functions, and Fully Connected Layers (FCLs).

The input and output of each convolutional layer are known as feature maps. If the input is an image, each feature map is a 2D array containing colour channels and if a video, the feature map has to be a 3D array and 1D arrays for audio inputs. In every output stage along with the network, there are new features extracted from the pixels of its input, and by the convolutional operations, more distinguishing features are detected in the later layers.

A. CNN ARCHITECTURE

As an example of a typical CNN, the entire architecture of the popular CNN AlexNet [28] is illustrated in Fig. 1, which includes five convolutional layers with padding operations, three max-pooling layers, and three FCLs. In Fig. 1, I_D, O_D, K_D denote the spatial height and width dimension of the inputs, outputs, and kernels, while I_C, O_C, K_C are the channel numbers of those, respectively. In addition, F is the filter number, which is equal to the outputs O_C, and the kernel dimension in the padding operation is symbolized by K_P. A notable term that will be used in the layer-based definitions is tensor, which in the simplest form is a generalization of matrices to n-dimensional space.

The following sub-sections define all the architectural components of a CNN, based on the reference architecture AlexNet in Fig. 1. Weight number calculation and areas of usage are also discussed.

1) INPUT LAYER

Input layers introduce input data in the network, which normally represents an image structured as a data array of pixel values. Before feeding the image into a designed model, the spatial and channel dimensions of the input have to be reshaped according to the model or the used deep learning library specifications.

2) CONVOLUTIONAL LAYER

The core of a CNN is the convolutional layers. Learnable parameters in these layers form the kernels and the collection

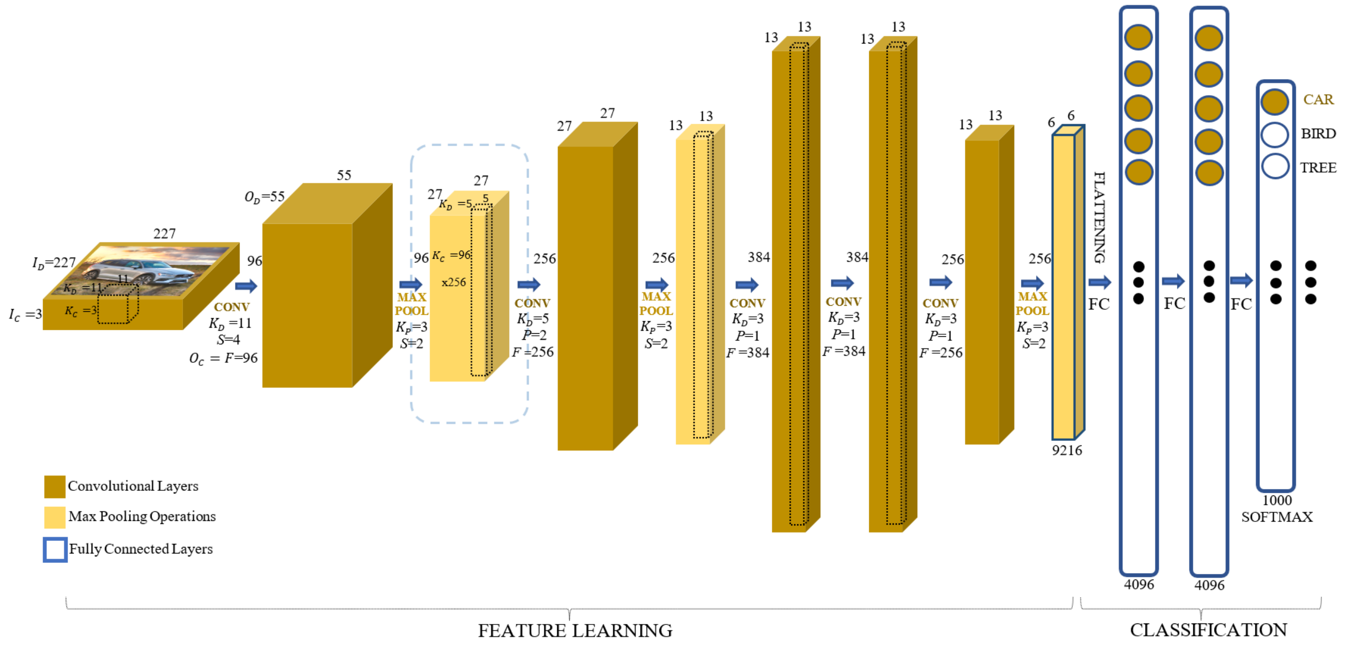


FIGURE 1. **A comprehensive view of the AlexNet [28] architecture.**

of the kernels in a layer is called a filter, which is subjected to a convolution multiplication through the full spatial depth of the input. A kernel defines the field of view on the convolution operation, whereas, as a tensor sample, a filter is the total number of kernels in a layer channel-wise. For instance, a filter with a size of "$m \times n \times c$" includes c kernels with "$m \times n$" kernel dimension.

Via the convolution operation, a feature map is obtained with a new spatial dimension and channel number based on the input and filter dimensions. Based on the used kernel's parameter values, outputs are extracted with different features. In Fig. 1, all the convolutional layers are composed of standard spatial 2D convolutions. Consider a spatial convolution, which takes a tensor $I \in \mathbb{R}^{I_D \times I_D \times I_C}$ as the input, and the filter of the convolution is the tensor $K \in \mathbb{R}^{K_D \times K_D \times I_C \times F}$. For simplicity, we assume the spatial dimensions of the input/output to be identical, which means that $O_D = I_D$, and the stride (i.e., the convolution kernel's step size) is 1. Then, the spatial 2D convolution outputs a tensor $O \in \mathbb{R}^{I_D \times I_D \times F}$, computed as

$$O_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} I_{k+\hat{i},l+\hat{j},m}, \tag{1}$$

where $\hat{i} = i - \lfloor I_D/2 \rfloor$ and $\hat{j} = j - \lfloor I_D/2 \rfloor$ denote the re-centred spatial indices; k, l and i, j are the indices over the spatial dimensions; whereas m, n provide indexation to the channels and filters.

3) PADDING OPERATION

Padding is the placement of several extra pixel grids to the input's spatial plane to handle the output's spatial dimensions. In case of a demand for an equal dimension of input and output tensors, the padding operation could enable it. As seen in Fig. 1, the output tensor's spatial dimensions are in the control of the padding operation. The effect of the padding operation in the output tensor's dimension ($O_D \times O_D$) is defined by

$$O_D = I_D + 2P - K_D + 1. \tag{2}$$

4) STRIDE OPERATION

The stride value indicates that the filter, which is the weight tensor for the convolution process, slides on the input tensor in increments of one or more-pixel steps. This is another parameter that directly affects the output size. The dimension of the output tensor ($O_D \times O_D$) is defined by

$$O_D = \frac{I_D + 2P - K_D}{S} + 1, \tag{3}$$

where S is the stride step parameter.

5) ACTIVATION FUNCTIONS

CNNs use various activation functions, such as Rectified Linear Units (ReLU), Sigmoid, TanH, etc., to build the feature map, obtained through the convolution operation. These functions enable the introduction of nonlinearities to the layers, which increases learnability. In particular, ReLU has been frequently preferred in CNNs on the score of enabling several times faster training in comparison to the other activation functions.

6) POOLING OPERATION

Pooling layers take small-sized rectangular blocks, defined by K_P, from the output feature map, and form subsamples from it to produce a single maximum, minimum, or average output from every block. After that, a new sampled feature map is formed, which is then used as an input for the next convolutional layer. By reducing the parameters of the feature map, the pooling layers allow to reduce the spatial size as well as the control of overfitting. As instance, in Fig. 1, following the convolutional layers and the application of the activation functions, there are three maximum pooling layers, represented with light amber layers. Additionally, global average pooling is another pooling method, used in [29] as well. It does a pooling to the whole image at a time without any sliding.

7) FULLY CONNECTED LAYER

The CNN architecture performs feature learning via the operations introduced above until the last convolutional and pooling layer output is derived. The final CNN stage consists of FCLs and performs a high-level classification. It is placed after the last convolution or pooling operation has occurred. The flattening operation in Fig. 1 provides the transition from convolutional later to FCL by converting the data to a 1-dimensional array. Essentially, in this stage of the architecture, a usual neural network process for the operation of classification takes place.

8) OUTPUT LAYER

This is the final stage, and the predicted result for the input image is obtained via different cost functions. For instance, in Fig. 1, Softmax multi-class classifier [30] is deployed. Thus, a cost is produced for a prediction by comparing the predicted result and the real data from the training set.

B. FULLY CONVOLUTIONAL NETWORKS

Fully Convolutional Networks (FCNs) are neural networks that are composed of only convolutional layers without adding any FCLs. The difference from CNNs is that FCNs are fully convolutional networks, which do all the learning with convolution-based filters even for decision making in the last layer, whereas CNNs include FCLs at the end of its architecture like Fig. 1. Another difference is that FCNs learns everything by using global information, whereas FCLs try to learn and make decisions only based on local spatial inputs. A few examples of this type of networks are included in Section III and IV.

C. NUMBER OF CNN PARAMETERS

Definition of the output tensors' dimension and calculation of the number of parameters, memory accesses, and FLOPs in CNNs are crucial for understanding superior aspects of the different types of convolution filters discussed in the rest of the paper. This section, therefore, introduces the widely used method to calculate the number of parameters for frequently used 2D convolutions, based on the AlexNet architecture [28] in Fig. 1. The number of parameters, memory accesses, and FLOPs for different types of convolution kernels are also defined in Section III. Initially, output dimensions have to be examined regarding different layer types. Equation (3) is employable in convolutional and pooling layers with the difference that K_D needs to be replaced by the kernel size of the pooling operation, K_P. As for the FCLs, it is a length vector and thus, the output size is equal to the number of neurons in the particular layer.

TABLE 2. Layer by layer parameters of AlexNet.

Layer Name	Tensor Size	Weights	Biases	Parameters
Input Image	227x227x3	0	0	0
Conv-1	55x55x96	34,848	96	34,944
MaxPool-1	27x27x96	0	0	0
Conv-2	27x27x256	614,400	256	614,656
MaxPool-2	13x13x256	0	0	0
Conv-3	13x13x384	884,736	384	885,120
Conv-4	13x13x384	1,327,104	384	1,327,488
Conv-5	13x13x256	884,736	256	884,992
MaxPool-3	6x6x256	0	0	0
FC-1	4096×1	37,748,736	4,096	37,752,832
FC-2	4096×1	16,777,216	4,096	16,781,312
FC-3	1000×1	4,096,000	1,000	4,097,000
Output	1000×1	0	0	0
Total				62,378,344

To calculate the number of CNN parameters, we recognize that it can be represented by the sum of the number of parameters in three different CNN operations: the convolutional layers, the transition from convolutional layer to FCL, and the transition from FCL to FCL. Pooling layers do not include any parameters, as they are aimed at size reduction of the outputs.

The number of parameters in the convolutional layers can be computed by

$$F \times K_C \times K_D^2 + B_C, \tag{4}$$

where B_C denotes the bias number in the convolutional layer.

In the transition from convolutional layer to FCL, the number of parameters is

$$N \times I_C \times I_D^2 + B_{CF}, \tag{5}$$

where N is the neuron number in the first FCL; I_C an I_D are the channel number and spatial dimension of the tensor previous to the first FCL; and B_{CF} represents the bias number in the layer.

In the transition from FCL to FCL, the number of parameters is

$$F_P \times F + B_{FF}, \tag{6}$$

where F_P and B_{FF} are the neuron number in the previous layer and biases respectively.

Consequently, the total number of parameters in AlexNet, calculated by means of (4)-(6) defined above, is 62,378,344, as detailed in Table 2.

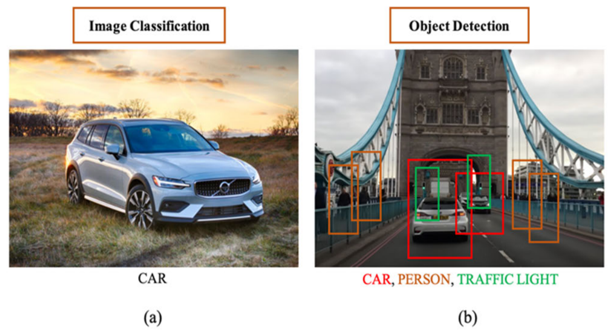


FIGURE 2. Difference between image classification (a) and object detection (b) tasks.

D. CNNs FOR IMAGE CLASSIFICATION AND OBJECT DETECTION

Before delving into the recent studies of CNNs for classification and object detection, it is essential to explain these two imaging tasks. In general, classifying an object in an image aims at assigning it to a certain category, while object detection involves localization and classification of all present objects. Fig. 2 illustrates the outcome of the two operations, where as a result of the image classification task, the image in Fig. 2 (a) is identified as a car; and as a result of object detection in Fig. 2 (b), cars, persons, and traffic lights are detected, and their positions and sizes are established by drawing a bounding box around the objects of interest in the image.

There is an obvious overlap between the two tasks, which requires that designs of high-performance object detection CNNs incorporate high-quality image classification properties in their architectures. In other words, an effective CNN architecture for object detection depends directly on the image classification quality of that CNN architecture.

In the rest of this section, frequently used datasets and libraries for the implementation of CNNs for image classification and object detection are introduced.

Representative CNN architectures for image classification and object detection are discussed in Sections III and IV respectively.

1) DATASETS

The learning capabilities of CNNs are obtained during training and heavily rely on the suitability and comprehensiveness of the available training datasets. In fact, features of datasets, such as collection scenarios, class numbers, resolution, etc., are crucial for CNN performance. In this section, we introduce the most widely cited datasets that are related to the scope of the paper along with their distinctive characteristics. For clarity of presentation, we divide the existing datasets into two classes: domain-general and domain-specific. Domain-general datasets here represent datasets that can be used for the training of CNNs for image classification or object detection, which then could be utilised for a particular application domain. For instance, ResNet-50 [17] has been trained on a domain-general dataset, ImageNet [31], and deployed for image classification tasks in various domains, such as medical [32], agriculture [33], autonomous driving [34], and so on. Contrary to domain-general, domain-specific datasets are originally created for a particular domain with the aim to facilitate learning of specialised domain-related features. Main features of renowned examples of both classes are summarised in Table 3, where the names in the left-most column are shaded light grey for domain-general datasets and dark grey for domain-specific datasets.

There exist two prominent domain-general datasets for image classification, which are CIFAR-10 & 100 [18] and ImageNet [31]. As an intermediate level dataset for image classification, CIFAR-10 & 100, have been introduced by one of the creators of AlexNet [28], Alex Krizhevsky. Both consist of a 50k training set and 10k test set fixed size 32×32 colour images. In CIFAR-10, each class is represented by 6k images, whereas in CIFAR-100 there are 600 images per class. As for ImageNet [31], it is regarded as an advanced level dataset, which has been developed with around 14M labelled high-resolution images according to the WordNet hierarchy with 22k subcategories. There has been also held a CNN competition since 2010 based on this dataset, referred to as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Since the 2012 ILSVRC competition, the dataset has been kept unchanged and involves 1k subcategories, $\sim$1.3M training images, 50k images for validation, and 100k images for testing. In addition, the number of images for each class, termed 'synset' [31], ranges between 732 and 1300. The newly proposed CNNs for domain-general image classification in the academic community have been mostly trained on these two datasets. The main features of these datasets are summarised in Table 3 and further details are provided in Section III.

Frequently used domain-general object detection datasets according to the open literature are Pascal VOC (Visual Object Classes) [35], MS COCO (Microsoft Common Objects in Context) [36], and ILSVRC 2014 [31]. Pascal VOC is a standardised dataset for object recognition, and popular challenge competitions based on it had been run from 2005 to 2012. Pascal VOC 2012 has 11,530 images with 27,450 annotations and 20 object categories. The currently most popular object detection dataset, MS COCO, is a large-scale dataset consisting of 328k images. It is suitable for multiple tasks, such as object detection, segmentation, keypoint detection, and captioning. Another widely used large-scale dataset in detector training is ILSVRC 2014, which is a part of the 2014 ILSVRC competition [31]. It comprises 450k training images with 200 object categories, 20k validation and 40k test images.

The main features of the datasets, outlined above, are summarised in Table 3 and further details are included in Section IV.

While there is no doubt of the usefulness of ImageNet [31] and MS COCO [36], their images often represent single-object scenes, which are not representative enough of what

TABLE 3. Main features of training datasets for image classification and object detection CNNs.

Domain-general & Domain-specific	~Data Size & Resolution [a]	~Number of Images & Classes [b]	Annotations [c]		Collection Sensors				Diversity on Collection Scenarios			Usage Scenarios		
			2D	3D	Camera	LIDAR	Radar	GPS (IMU/INS)	All Weather Conditions	Day / Night Time	Different Traffic Conditions	Image Classif. / Object Detection [d]	Semantic Segmentation	Localization / Trajectory Prediction/ Tracking [d]
CIFAR-10 & 100 [18]	162 MB 32x32	60k 10 & 100	-	-	✓	-	-	-	-	-	-	✓	-	-
ImageNet [31]	400 GB 224x224 [e]	1.4M 1000	-	-	✓	-	-	-	-	-	-	✓	-	-
Pascal VOC [35]	2 GB 469x387 [f]	12k 20	✓	-	✓	-	-	-	-	-	-	✓	✓	-
ILSVRC 2014 [31]	- 482x415	510k 200	✓	-	✓	-	-	-	-	-	-	✓	-	-
MS COCO [36]	37.57 GB 640x480	328k 80	✓	-	✓	-	-	-	-	-	-	✓	✓	-
KITTI [37]	180 GB 1392x512	12k 16	✓	✓	✓	✓	-	✓	-	-	-	✓	✓	✓
BDD100K [38]	1.8 TB 1280x720	100k 10	✓	-	✓	-	-	✓	✓	✓	✓	✓	✓	-
CityScapes [39]	63 GB 1024x2048	25k 30	✓	-	✓	-	-	✓	-	-	-	✓	✓	-
A2D2 [40]	2.3 TB 1920x1208	12k 14	✓	✓	✓	✓	-	✓	✓	-	-	✓	✓	-
nuScenes [41]	400 GB 1600x900	1.4M 23	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓
ApolloScape [42]	270 GB 3384x2710	140k 28	✓	✓	✓	✓	-	✓	✓	-	-	✓	✓	✓
Caltech [43]	11 GB 640x480	250k -	✓	-	✓	-	-	-	-	-	-	✓	-	-
EuroCity [44]	100 GB 1920x1024	47k 30	✓	-	✓	-	-	-	✓	✓	-	✓	-	-
Leddar PixSet [45]	- 1440x1080	29k 20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Oxford RobotCar [46]	23 TB 1024x1024	20M -	-	-	✓	✓	-	✓	✓	✓	✓	-	-	✓
Waymo [47]	336 GB 1280x1920	12M 3	✓	✓	✓	✓	-	-	✓	✓	✓	✓	-	✓
Udacity [48]	3 GB 1920x1200	15k 11	✓	-	✓	✓	-	✓	-	-	-	✓	-	-
TorontoCity [49]	- 1280x640	2k 8	-	✓	✓	✓	-	-	-	-	-	✓	✓	-

[a] Parameters related to the total dataset. [b] Parameters related to the image classification or object detection parts of the datasets, if exist. [c] Annotations can vary from bounding box to segmentation mask, please, see the sources for details. [d] At least one is provided. [e] The common cropped resolution in usage by state-of-the-art networks. [f] Average image resolution.

is encountered in real-time self-driving situations. Besides that, these domain-general datasets lack the necessary content to support the learning capabilities of ADSs with regard to specific self-driving scenarios. For example, adverse weather conditions could lead to poor CNN detection performance in ADSs, if weather-related information has not been present in the training data. In consequence, various domain-specific datasets have been created including different self-driving scenarios to facilitate CNN classification and detection learning capabilities for ADSs.

The most cited self-driving-specific datasets developed in recent years, main features of which can be found in Table 3, are briefly overviewed below:

(i) KITTI Vision Benchmark [37] is the most widely cited dataset [50] among researchers and developers since its publication year, 2012. It offers a relatively large amount of self-driving scenes with 2D, 3D, and bird's eye-view object detection datasets together with annotations.

(ii) UC Berkeley DeepDrive, BDD100K [38] is a recent dataset with 100k annotated videos and 10 different tasks for evaluation of self-driving image recognition algorithms. The dataset comprises more than 1k hours of driving experience and 1M frames. It also provides a large context diversity for self-driving by being strengthened with all-weather, geographic, and environmental conditions in the day/night time.

(iii) CityScapes [39] is a large-scale dataset and focuses on the semantic understanding of urban street scenes with 25k annotated images and 30 classes.

(iv) A2D2 [40] has been released by Audi and includes 41k labelled data with 38 features and 390k unlabelled data. Its size is around 2.3 TB and is split by annotation types, such as semantic segmentation, 3D bounding boxes, etc.

(v) nuScenes [41] has been created by a full sensor suite mounting six 360° cameras, five radars, a 36-beam LIDAR, and a Global Positioning System and Inertial Measurement Unit (GPS-IMU). It contains over 1.4M images with a diversity of driving manoeuvres, traffic situations and unexpected behaviours.

(vi) ApolloScape [42] is a part of the Apollo project, which is an evolutive and ever-growing research project across all aspects of autonomous driving. The project website offers various simulation tools over 80k lidar point cloud, 100k street views, and 1000km trajectories for city traffic.

(vii) Caltech [43] is a pedestrian detection dataset with annotations.

(viii) Leddar PixSet [45] was released in 2021. It consists of 29k frames in 97 sequences with more than 1.3M 3D bounding box annotations, collected by a full Autonomous Vehicle (AV) sensor suite (LIDARs, cameras, radars, and GPS-IMU). What presents this dataset novel is the deployment of a flash LIDAR with the inclusion of the full-waveform data.

(ix) Oxford RobotCar [46] contains over 100 records of a consistent route during a year in Oxford, UK, which brings a long-term diversity approach in data collection. Its sensor suite comprises six cameras, LIDAR, and Global Positioning System and Inertial Navigation System (GPS-INS).

(x) Waymo [47] was collected by Waymo self-driving cars. More details about it are presented in Section V.

The domain-specific datasets EuroCity [44], Udacity [48], and TorontoCity [49] in Table 3, feature person instances, mountain views and large-scale urban views, respectively.

Apart from the above, more sophisticated types of training datasets are also available, for example, naturalistic driving datasets, such as NUDrive [51], euroFOT [52], etc., which focus on the human element in autonomous driving: the driver.

2) LIBRARIES AND FRAMEWORKS

In this subsection, the most frequently used libraries and frameworks in the CNN field are reviewed. Compared to other programming languages, Python-based machine learning libraries have been the go-to choice for researchers and developers as they offer a variety of features and flexibility, which can increase productivity and quality of code writing, implementation and integration.

The CNN coding process can be divided into two main stages: 1) data pre-processing and 2) building of the CNN algorithm. Several general-purpose libraries can be employed at the first stage to insert and prepare datasets for the second stage, such as, the multi-dimensional array-processing library NumPy [53], scientific computing library SciPy [54], data frame processing library Pandas [55], etc. A few powerful drawing libraries, such as matplotlib [56], seaborn [57], plotly [58], are available as well.

As regards the second stage, there are a large number of frequently used libraries and frameworks. The most popular ones are summarised in Table 4, and some of them are outlined below:

(i) Google's Tensorflow [59] is an open-source ML library, available since 2015, which has been widely used. It enables the development of highly computational ML tasks by using data flow graphs in which edges signify tensors. Through this, a single Application Programming Interface (API) can distribute the load between multiple nodes, such as GPUs, and CPUs.

(ii) Theano [60] is compatible with numerical computations and simplifies code writing in deep learning. It also provides tight integration with NumPy [53] resulting in quite accurate calculations.

TABLE 4. Commonly used libraries with a Python interface.

LIBRARIES	Written in	Interface	Platforms	Features
Tensorflow [59]	Python, C++, CUDA	Python, C/C++, Java, R, Julia, Swift	Windows, macOS, Linux, Android	OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN [a], CNN, RBM/DBN [b], Dataflow Graphs
Theano [60]	Python	Python	Cross Platform	OpenMP, OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN, RBM/DBN
Chainer [61]	Python	Python	Linux, macOS	CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN
Keras [62]	Python	Python, R	Linux, macOS, Windows	OpenMP, OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN, Fast Experimentation
PyTorch [63]	Python, C, C++, CUDA	Python, C++, Julia	Linux, macOS, Windows	OpenMP, OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN, RBM/DBN
MXNet [64]	C++	Python, C++, Java, R, Go, Julia, Matlab, Perl, Clojure, JavaScript, Scala	Linux, macOS, Windows, AWS, Android, iOS, JavaScript	OpenMP, OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN, RBM/DBN, Portability
Caffe [65]	C++	Python, C++, Matlab	Linux, macOS, Windows	OpenMP, OpenCL, CUDA, Pretrained Models, Parallel Execution (Multi Node), RNN, CNN

[a] Recurrent Neural Network; [b] Restricted Boltzmann Machines/Deep Belief Network

(iii) Keras [62] is another widely used library that can be run on top of Theano or Tensorflow, as it is designed as a high-level API. As such, it offers a productive and user-friendly interface reducing developers' cognitive load.

(iv) PyTorch [63] has been a fast-growing library in recent years. It has been upgraded by the Facebook AI Research Lab (FAIR) with many outstanding novel features, such as scalability for distributed training [28] and cloud support, offering design simplicity from research to production. In addition, it smoothly integrates with the Python data science library NumPy. In fact, PyTorch was ranked as the most-used framework in ML implementations, reported in the open literature since March of 2019 [66], as shown in Fig. 3.

However, Keras has still a better real-time performance than PyTorch.

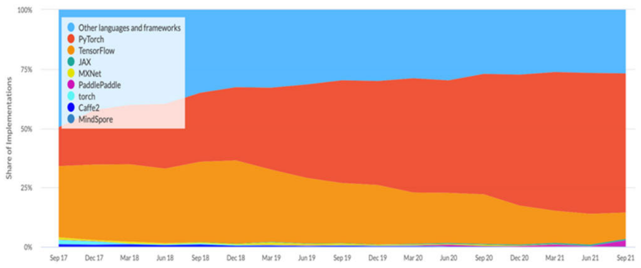


FIGURE 3. Trends in the selection of ML frameworks [66].

(v) Caffe [65] is a research-based library and has been written in C++ with Python interface by Berkeley AI Research (BAIR) and community contributors. Its remarkable features are speed and switching between CPU and GPU by a single flag. It also offers seamless integration with GPU training in image-based datasets.

Other powerful Python-based libraries and frameworks, such as MXNet [64], Chainer [61], ScikitLearn [67], have also been commonly used.

III. REVIEW OF CNNs FOR IMAGE CLASSIFICATION

One of the primary objectives of image classification algorithms is to be deployed in systems aimed at solving different computer vision tasks. For instance, they can act as a backbone network, also known as a base network, in various object detection modules. In this respect, a distinctive classification network can play a crucial role in enhancing the performance of detection systems.

This section looks comprehensively at outstanding CNN models for image classification by grouping them into two classes based on their model size: heavyweight and lightweight designs. It can be said that improving and designing a new CNN architecture is based on understanding the design and historical relations of previous architectures. Therefore, the CNN architectures discussed below are considered in chronological order taking into account the architectural influences inspired by previous designs.

There are certainly numerous qualities that affect the CNN design process. However, the types of convolutions and specially designed operations could be seen as one of the most significant design parameters in convolutional networks. In this section, we attempt to briefly introduce the architectures of the majority of the outstanding image classification CNNs, published in the open literature, with a focus on their accuracy performance, parameter number, model size, and inference speed. The aim is to provide evolutionary insight, highlighting the progressive developments in convolutional operations. The focal point is mainly on the most popular convolutional processes shown in Fig. 4, even though, numerous other types of convolutions and operations have been proposed, such as 3D [68], dilated [69], spatially separable [70], flattened [71], etc. The parameters $I_D, I_C, K_D, K_C, O_D, O_C, F$ used in Fig. 4 are introduced in Section II.A and the spatial 2D convolution in Fig. 4 (a) is discussed in Section II.A.2) above. The convolutional processes, shown in Fig. 4 (b) - (f) are described in Section A and B below, as follows: Fig. 4 (b) and (c) are explained in Section B.7); Fig. 4 (d) – in Section A.1); Fig. 4 (e) – in Section B.11); and Fig. 4 (f) – in Section B.9).Furthermore, multifaceted experimental results are reported revealing the individual training and inference performance of the convolutions and operations in Fig. 4 using ResNet [17] as the reference network to achieve a fair comparison. Layer-based timing performance is also provided through the decomposition of the network layers and modules, where the operations take place.

The performances of the reviewed heavyweight and lightweight CNN designs are summarised in Table 5, where the Top-1 and Top-5 accuracy metrics are used. The Top-1 accuracy means that the highest probability prediction of the CNN model for an image must be exactly the expected answer or otherwise it fails, while the Top-5 accuracy requires that the top five highest probability predictions of the model must include the expected answer or otherwise it fails.

A. HEAVYWEIGHT CNN DESIGNS

1) AlexNet

As introduced in Section I.A, the concept of CNNs was introduced for the first time in 1998 by LeCun *et al.* [3], [4] and named LeNet-5. The model was developed using a multi-layer artificial neural network including two convolutional, two pooling, and three fully connected layers, and had 60k parameters. As the ReLU activation function hadn't been improved in these years, the TanH activation function was mostly used in the learning process, whereas the sigmoid function was only used in the output layer. The average pooling method was employed rather than max pooling in the pooling layers. The training process was slow taking many days due to the type of processors at the time, however, nowadays this has improved significantly with the use of new-generation GPUs.

LeNet-5 was classifying handwritten digits [72], MNIST (Modified National Institute of Standards and Technology), which is referred to as the ancestor of basic level image classification with 60k training and 10k test fixed-size images. By means of using a backpropagation algorithm [5], it could be trained to detect the patterns from raw pixels and to eliminate a separate feature detection. The method had a 0.95% accuracy rate performance in the dataset. Despite all these advantages, LeNet-5 was ineffective when applied to complex problems such as video classification, which needs a large set of training data and powerful processors for the computation. However, the novelty of the LeNet-5 design is enormous, as it has provided a standard 'template' for almost all subsequent CNNs.

Following the improvements on both compute-bound and memory-bound computing platforms, Krizhevsky *et al.* [28] upgraded the AlexNet architecture shown in Fig. 1, and it achieved the best score in the ILSVRC competition based on ImageNet [31] dataset in 2012. Though it was only assembling three more convolutional layers into the LeNet design, there were four notable architectural contributions: (i) the first implementation of the ReLU activation function, enabling six times faster training than the TanH function; (ii) the first deployment of the overlapping pooling technique in the pooling layers; (iii) the application of a data augmentation technique [73]; and (iv) the introduction of the grouped convolution.

AlexNet was implemented on two Nvidia GPUs GTX 580 with 1.5GB virtual memory per node to allow efficient network training as shown in Fig. 5. The design features two separate spatial 2D convolution paths grouping the convolution operation with two filter groups, which represents model-parallelization across two GPU nodes, also known as a model-distributed training method. In addition, a compact and efficient version of AlexNet, named CaffeNet,

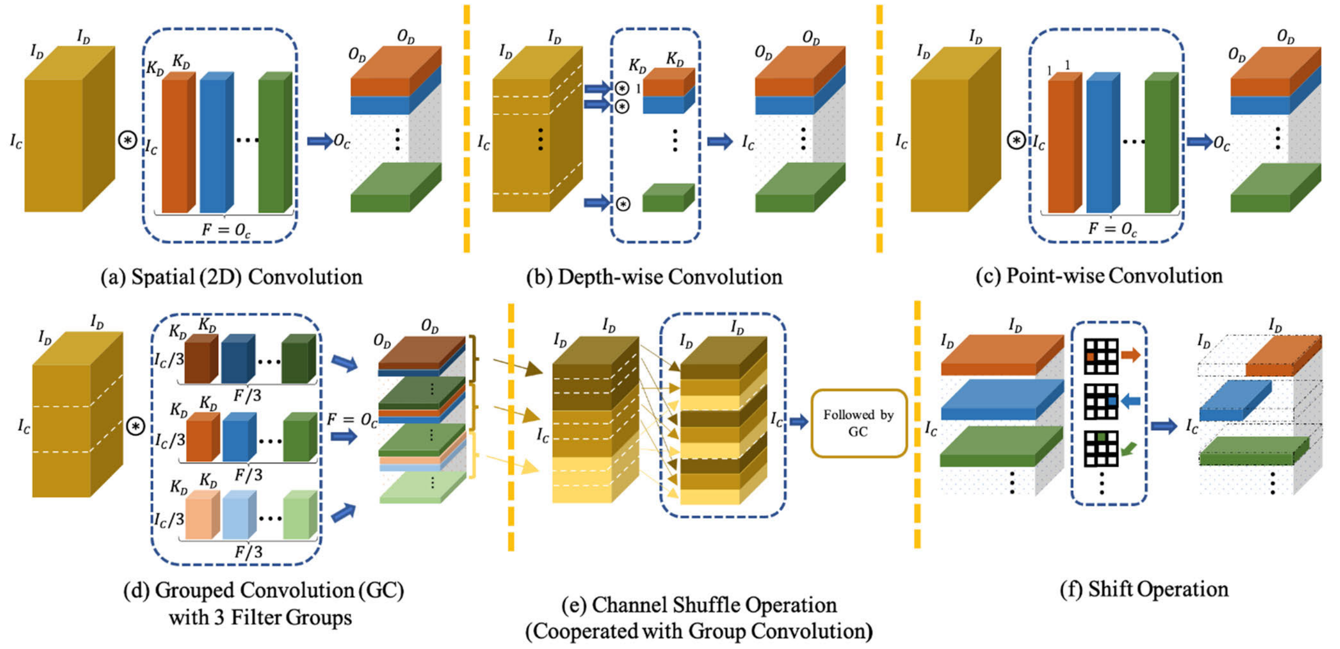

(a) Spatial (2D) Convolution
(b) Depth-wise Convolution
(c) Point-wise Convolution
(d) Grouped Convolution (GC) with 3 Filter Groups
(e) Channel Shuffle Operation (Cooperated with Group Convolution)
(f) Shift Operation
Followed by GC
I_D
I_C
K_D
O_D
O_C
$F = O_C$
$I_C/3$
$F/3$

FIGURE 4. Frequently used different types of convolutions and operations.

was executed on a single node NVIDIA GeForce graphics card GTX 1080Ti.

The 2D spatial and grouped convolutions are illustrated in Fig. 4 (a) and (d), respectively. In a grouped convolution, the number of kernel channels K_C is equal to the number of the input channels I_C, and the number of filters is separated into an equal number of sub-groups, which are responsible for a conventional spatial 2D convolution with the divided group depths. Unlike the AlexNet design with two filter groups in Fig. 5, we divide the kernel channels and filter number into three sub-groups as seen in Fig. 4 (d), whereby each filter group could be defined by the tensor $[K_D : K_D : K_C/3 : F/3]$.

As a result, each group creates $O_C/3 = F/3$ output channels. Overall, three groups create $3 \times O_C/3 = 3 \times F/3$ output channels, equal to $O_C = F$.

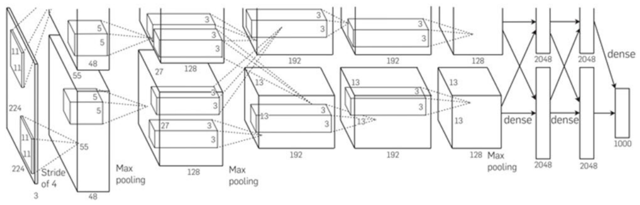


FIGURE 5. Implementation of AlexNet with grouped convolutions on two GPUs [28].

The use of grouped convolutions in networks brings three essential benefits. The first one is making it possible to parallelise the CNN model, which allows executing the convolutions over multiple GPUs. The second benefit is that the network could be fed with more images per step compared to a single GPU implementation. Furthermore, each filter group can learn a unique representation of the fed data achieving more structured learning. The third benefit is the improved efficiency since the number of parameters decreases when the number of filter groups increases. This can be easily shown analytically. In a spatial convolution, the number of parameters is calculated by the expression $F \times K_C \times K_D^2$, whereas in a grouped convolution it could be expressed as $(F/3 \times K_C/3 \times K_D^2) \times 3$. Thus, it can be seen that with three filters a decrease in the parameter number by two thirds or 67% is achieved.

It can be observed from the above that grouped convolutions bring substantial advantages compared to standard spatial convolutions, representing another notable contribution of the AlexNet model.

2) ZFNet

One year later, Zeiler and Fergus [74] proposed ZFNet, the winner of the 2013 ILSVRC competition [31], which significantly reduced the classification error of AlexNet by visualizing the convolutional networks layer by layer and by adjusting layer hyper-parameters, such as filter sizes and strides.

Although there had been some observation and improvement of shallow layer features, it was the ZFNet authors who investigated deeper features of the pixel domain. Using the deconvolution and un-pooling techniques, they were able to visualize the convolutional networks layer by layer, as shown in Fig. 6. By means of the deconvolution technique, they also analysed and rearranged several hyper-parameters of their algorithm, which led to reducing the error rate.

In deconvolution, two problems were determined in the first two layers. There was only high and low-frequency

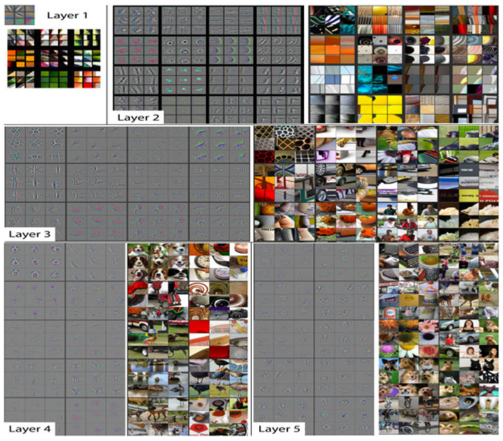


FIGURE 6. **Visualization of the layer-based features by deconvolution and un-pooling [74].**

information in the first layer, as well as the scarce number of mid-frequencies was causing a chain effect in the learning of some mid-frequency features. In the second layer, an aliasing problem, occurring in the event of a low level of the sampling frequency, was detected, which was due to the large stride in the convolution operation of the first layer. The researchers fixed the problems by reducing the filter size from to in the first layer and changing the stride step of the first layer to 2 instead of 4. As a result, a valuable performance increase was achieved.

3) VGGNet

VGGNet [75] is another heavyweight CNN, which was developed by VGG (Visual Geometry Group) from the University of Oxford. It was followed by many other improved models, such as VGG-11, VGG-11(LRN), VGG-13, VGG-16, VGG-16 (Conv1), and VGG-19, built on top of VGGNet. Unlike [28] and [74], which used 11×11 and 7×7 kernels respectively, it was based on the placing of 2×2 and 3×3 convolution kernels and is comprised of thirteen convolutional and three fully-connected layers by deploying the ReLU activation function. The aim was to go deeper into the layers in order to improve the accuracy. This work was highly important as it allowed to reduce the error rates in CNN image classification. However, that entailed a significant increase of the parameters and model size amounting to 138 million (M) and 575 megabytes (MB), respectively, where the model size specifies the required storage space for all parameters of the trained model. Nevertheless, it was the winner of [31] in localization tasks and the runner-up in general.

4) SENet

The introduced CNN designs so far have been focused on improving the spatial components to strengthen the representational effectiveness in each layer even though central blocks of CNNs might have been used to obtain informative features. From that point of view, Hu *et al.* [76] from Oxford University designed and proposed SENet as a novel architecture by enhancing channel-wise feature responses. Despite the fact that SENet had improved the accuracy level comparing to the former winners of the ImageNet competitions [31], it had also increased the computational costs slightly bringing the number of parameters to 146M. Its Top-5 error rate was tested as 3.7%, which secured its winning place in 2017 and ever since it has been the best performing classification CNN model.

5) NASNet

Zoph *et al.* [77] from Google Brain proposed NASNet. The followed method was initially tracing an architectural building block with a small dataset, CIFAR-10 [18]. That was then applied to a larger dataset, ImageNet [31]. A novel regularization technique, ScheduledDropPath, was also proposed that significantly improved the generalization of NASNet. Their architecture was composed of blocks and cells, called "normal" and "reduction" cell, whereby the former returned a feature map in the same dimension and the latter returned a feature map whose height and width were reduced by a factor of two. As a consequence, NASNet reached the same Top-1 accuracy 82.7% of SENet with a lower parameter number on the ImageNet dataset.

6) ResNeXt

Until the emergence of ResNeXt [78], the state-of-the-art image classification models had relied on supervised pre-training [79], and the ImageNet dataset had been widely used. In this study, Mahajan *et al.* presented a peerless pre-training scheme to predict hashtags for social media images. Through the application of their unique transfer learning technique, they demonstrated a remarkable performance of ResNeXt on ImageNet classification, and thus, it was listed as the winner of the 2018 competition [31].

7) EfficientNet-L2

In EfficientNet-L2 [80], a semi-supervised learning approach [81], named Noisy Student Training, was introduced under the assumption of an abundance of labelled data. The work draws on the machine learning method of self-training [82] and the data compression technique of distillation [83]. (More information about machine learning methods and compression techniques is provided in the Appendix.)

Firstly, the EfficientNet [84] model is trained with labelled images on ImageNet, and then, the model is used as a teacher in producing pseudo labels for 300M images. Next, a larger model of EfficientNet is trained as a student with a combination of labelled and pseudo images, produced in the first stage. Following that, the same process is applied iteratively by adding to the student different types of noise, such as dropout, stochastic depth, and data augmentation, to be able to learn better than the teacher. Thus, the performance of this model on the ImageNet [31] dataset proved to be explicitly outstanding in 2020 with a Top-1 accuracy of 85.5%.

8) FixEfficientNet-L2

FixEfficientNet-L2 [85] is an upgraded model of EfficientNet-L2 [80], reported by Touvron *et al.* from the Facebook AI Research group, and it utilises the augmentation technique proposed by the same researchers in [86], which is aimed at reducing the discrepancy between objects by using different resolutions at testing and training time. The method has resulted in notable performance enhancements in different existing algorithms, for example achieving Top-1 accuracy in ResNet-50 [17] - up to 79.8%, ResNetXt-101 32×48d [78] - up to 86.4%, and EfficientNet-L2 [80] - up to 88.5%. On the other hand, EfficientNet-L2 and FixEfficientNet-L2 have been one of the weightiest heavyweight algorithms in terms of parameter number, amounting to 480M.

9) VIT-H/14

Dosovitskiy *et al.* [87] from Google Research have brought to life the idea of the Transformer architecture [88], which is mainly used in natural language processing tasks, for image classification tasks. Their model, Vit-H/14, has shown a striking result with an 88.6% Top-1 accuracy on the ImageNet [31] dataset. The model's accuracy, thus, is ranked as the 2020's highest in Table 5.

10) LambdaResNet200

LambdaResNet200 [89] is a quite recent CNN architecture, which was reported at the 2021 ICLR conference. It introduces novel lambda layers, which enable a 4.5× faster training on modern ML accelerators compared to EfficientNet-L2 [80]. Along with the training performance, its Top-1 accuracy performance on [31] has been listed as 84.3%, which is excellent as it keeps a low model parameter number.

11) META PSEUDO LABELS

Google AI [90] introduced a novel semi-supervised learning method, named Meta Pseudo Labels, which has enabled the highest Top-1 accuracy on [31] ever. The idea improves on the student-teacher concept in [80], whereby the teacher network generates pseudo labels by using unlabelled data [91] to teach a student. The difference is that the teacher is not fixed, and it is constantly adapted by means of feedback that comes through the student network performance on a labelled dataset [31]. That produces better pseudo labels compared with the method in [80]. In this way, EfficientNet-L2 and EfficientNet-B6-Wide [80], when trained with the method, achieved the highest Top-1 scores of 90.0% and 90.2% accuracy performance, respectively. However, the parameter number is also correlative with the results reported in [80] and [87], which are the highest among the algorithms introduced above.

B. LIGHTWEIGHT CNN DESIGNS

1) INCEPTION-v1 (GoogLeNet)

Inception-v1, also known as GoogLeNet, proposed by Szegedy *et al.* [29], is a distinctive and lightweight architecture, which is based on [92], [93]. The authors introduced the inception modules, which comprise convolutional kernels of different sizes operating over the same input, and then stacking all the outputs from the different kernels. Using convolution kernels of different sizes, the architecture was able to capture all sorts of features, containing 22 layers with 6.8M parameters. It has been the first architecture ever using 1×1 convolution kernels at the middle of the network to reduce dimensionality, parameter numbers, and computational budget in layers, and at the same time increasing non-linearity. Global average pooling was only used at the end of the network instead of every layer. ReLU was the deployed activation function. By going deeper with these inception modules, it was ranked as the winner of the 2014 ImageNet competition [31].

Inception-v1 also introduced two auxiliary classifiers to improve the classification performance in the lower stages of the classification process and to rise the backpropagation gradient signal, as well as for additional regularization. The auxiliary networks are only employed for training and are not used in testing or at inference time.

2) INCEPTION-v3

Following the publication of Inception-v1 [29], Szegedy *et al.* [94] designed a new architecture called Inception-v3. The study first introduced the Inception-v2 model with a deep learning method, called batch normalization, which was created for normalizing the value distribution prior to the next layer, allowing to increase the accuracy performance and training speed. Keeping that novelty and introducing an additional factorization technique have led to the release of the third version, Inception-v3.

Inception-v2 had some flaws in the loss function, which was adding batch normalisation to the auxiliary classifiers in its layers, and in the optimization method. (Readers are referred to the Appendix for information on optimization algorithms.) Inception-v3 incorporated these weaknesses but avoided computational problems by using the factorization method, which factorises the $n \times n$ kernels to $1 \times n$ and $n \times 1$ asymmetric kernels. This led to a reduction in the parameter number of the architecture without compromising network efficiency. For instance, by using a 5×5 kernel in a layer, the number of parameters were equal to $5 \times 5 = 25$. However, by using two layers of 3×3 kernels, the number of parameters amounts to $3 \times 3 + 3 \times 3 = 18$. Thus, the number of parameters was reduced by 28%.

Other specific features in Inception-v3 were the use of only one auxiliary classifier, aimed for regularization and batch normalization, and employing efficient grid size-reduction techniques instead of using a pooling layer. Due to its new efficient properties, the Inception-v3 architecture was able to achieve increased accuracy levels on [31], as detailed in Table 5.

3) ResNet

As it could be observed from the architectures discussed until now, the increase of the number of network layers leads to achieving better accuracy results. Taking this on board, the Microsoft researchers He *et al.* designed the ResNet model [17], including five networks with different layer numbers from varying 18 to 152.

A ResNet architecture comprising 50 layers is illustrated in Fig. 7, where bottleneck modules comprise two 1 × 1 and one 3 × 3 spatial convolutions with varied channel numbers. There are residual (identity/projection) connections, allowing to connect sequential modules to increase feature reuse. The primary contribution of the residual connections was the prevention of the vanishing gradient problem in the first layers. By means of going deeper in the network, ResNet-50 was ranked as the winner of 2015 ILSVRC [31] with a 93.3% Top-5 accuracy performance.

4) INCEPTION-v4

The high performances of the Inception models [29], [94] and the high impact on the performance of the residual connections in ResNet [17], have raised the question as to whether there is an advantage of combining the Inception models and the residual connections. Inspired by that idea, Szegedy *et al.* [95] proposed the Inception-v4 architecture, which is an empirical study demonstrating that residual connections considerably accelerate the training performance of Inception networks. Moreover, it was proved that the training process might be facilitated by deploying an activation function scaling technique. As a result, Inception-v4 achieved a 3.8% Top-5 error rate in ImageNet ILSVRC competition [31]. Inception-ResNet-v2 was another model proposed as part of the same study, which even outperformed Inception-v4.

5) TRIMPS-SOUSHEN

The Trimps-Soushen model was the winner of ILSVRC 2016 [31], with a 2.99% Top-5 error rate. However, the creators of the Trimps-Soushen architecture did not publish any technical report or paper. They only released the testing results of their model on the ImageNet dataset in a presentation at the 2016 ECCV workshop. Nevertheless, the results have been approved and listed by ImageNet.

6) SqueezeNet

Iandola *et al.* [96] proposed a new compact model, called SqueezeNet, based on the AlexNet architecture [28]. They achieved a remarkably low model size of 0.5MB, keeping the AlexNet accuracy levels. In order to obtain that result, 3 steps were followed. Initially, the 3 × 3 spatial 2D filters were mostly changed with 1 × 1 filters providing lower parameter numbers. Following that, squeeze layers were used to reduce the number of input channels to 3 × 3 filters, and finally, for utilising large activation maps in the convolutional network, a late downsampling technique was applied.

As a result, the same level of accuracy as in [28] was achieved with ×50 fewer parameters, and thus, SqueezeNet has been one of the most striking lightweight algorithms.

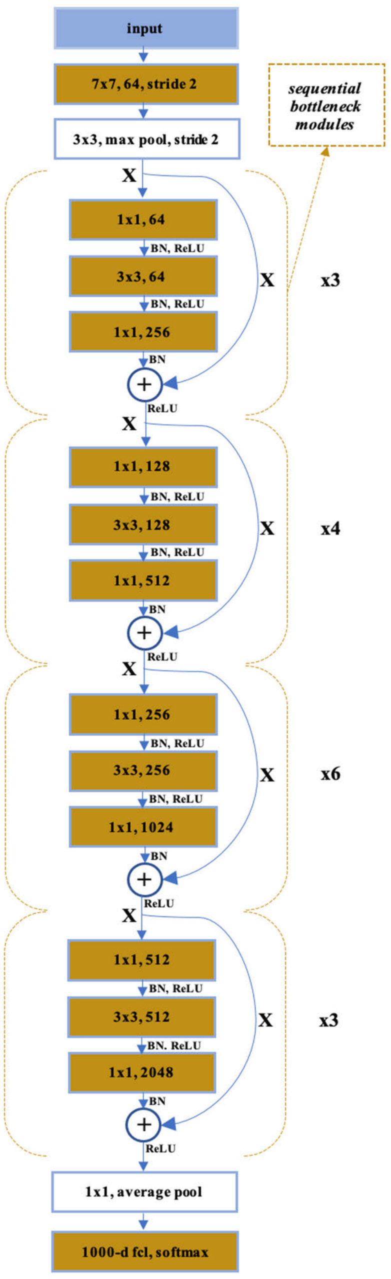


FIGURE 7. ResNet-50 architecture with bottleneck modules [17].

7) MobileNets

In 2017, Howard *et al.* [70] from Google Inc. introduced the MobileNets CNN, which was focused on reducing model size and complexity, targeting mobile and embedded vision applications. Depth-wise separable convolutions in Fig. 4 (b) and (c) form the basis of MobileNets. Compared to a same spatial convolution structure, the parameter numbers were reduced by means of depth-wise separable convolutions. But, neither the model size of SqueezeNet [96] nor its parameter number of around 1.2M were outperformed by MobileNets. However, in terms of the accuracy performance, MobileNets has shown a distinct enhancement with 13.1% higher Top-1 accuracy than SqueezeNet.

For a simple representation of a spatial convolution like the one expressed by (1), we suppose that the input/output spatial dimensions are identical, and the stride step is 1. Thus, we can compute the outputs of the depth-wise convolution, $\hat{O}_{k,l,m}$, in Fig. 4 (b) by

$$\hat{O}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} I_{k+\hat{i},l+\hat{j},m}, \tag{7}$$

where $\hat{K} \in R^{D_K \times D_K \times I_C}$ denotes a depth-wise convolution kernel.

Likewise, the point-wise convolution, $O_{k,l,n}$, in Fig. 4 (c) is defined by

$$O_{k,l,n} = \sum_{m} \hat{K}_{m,n} \hat{O}_{k,l,m}, \tag{8}$$

where $\hat{K} \in R^{1 \times 1 \times I_C \times F}$ is a point-wise convolution kernel.

Similar to grouped convolutions (Sec. III.A.1)), the reason why depth-wise convolutions are selected is related to their efficiency, as they reduce considerably the parameter number and computational cost. This is because, in the standard spatial convolutions, the parameter number is defined by $F \times K_C \times K_D^2$, whereas the parameter number in a depth-wise convolution followed by a point-wise convolution is calculated by $(K_C \times K_D^2) + (F \times 1 \times 1)$, which is quite less. As to the computational cost comparison, we define the FLOPs number as $F \times K_C \times K_D^2 \times O_D^2$ in spatial convolutions and as $(K_C \times K_D^2 \times O_D^2) + (F \times K_C \times 1^2 \times O_D^2)$ in depth-wise separable convolutions. These two expressions can be evaluated by using identical dimensions. They could also be compared by the ratio of the number of FLOPs between the depth-wise separable convolution and the 2D convolution, which is defined by

$$\frac{1}{F} + \frac{1}{K_D^2}. \tag{9}$$

It is evident from the above expression that it has a fractional value, which shows that the parameter number and computational budget of depth-wise separable convolutions are significantly more efficient compared to spatial convolutions.

8) XCEPTION

Xception is another model developed by Google [97], which was inspired by Inception-v3 [94], discussed in Section 2) above. It employs depth-wise separable convolutions that generally consist of depth-wise and point-wise convolutions, as illustrated in Fig. 4 (b) and (c). The Inception modules in the Inception-v3 were replaced with depth-wise separable convolutions instead of spatial 2D convolutions, which improved its performance. In spatial convolutions, whereas 1 × 1 convolution kernels provide for cross-feature map correlations, 3 × 3 and 5 × 5 convolution kernels support spatial correlations. In depth-wise separable convolutions, these correlations are provided without even using mid-level activations, and due to that Xception has outperformed Inception-v3.

9) ShiftNet

The popularity and extensive usage of depth-wise separable convolutions in CNNs led to several further research works, including a remarkable approach, proposed by Wu *et al.* [98]. In theory, depth-wise separable convolutions seem to improve the parameter number and computational cost of CNNs. This could be demonstrated by the ratio between computational cost and memory access, which is expressed in spatial 2D convolutions as follows:

$$\frac{F \times K_C \times K_D^2 \times O_D^2}{I_D^2 \times (K_C + F) + K_D^2 \times K_C \times F}, \tag{10}$$

whereas the ratio for depth-wise separable convolution is the following:

$$\frac{K_C \times K_D^2 \times O_D^2}{I_D^2 \times (2K_C) + K_D^2 \times K_C}, \tag{11}$$

where the input/output dimensions are identical.

A lower ratio indicates that memory accesses take more time, causing several orders of magnitude slower operation and higher energy consumption per floating-point operation. Thus, this drawback prevents I/O-bound devices to perform to their maximum computational ability. Based upon this viewpoint, a novel convolutional operation, named shift, was proposed.

The shift operation, illustrated in Fig. 4 (f), can be viewed as a special type of depth-wise convolutions, the output of which, $\tilde{O}_{k,l,m}$, could be expressed by

$$\tilde{O}_{k,l,m} = \sum_{i,j} \tilde{K}_{i,j,m} I_{k+\hat{i},l+\hat{j},m}, \tag{12}$$

where the kernel of the operation is a tensor $\tilde{K} \in \mathbb{R}^{K_D \times K_D \times I_C}$ and every kernel can be represented as follows:

$$\tilde{K}_{i,j,m} = \begin{cases} 1, & \text{if } i = i_m \text{ and } j = j_m, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

In (12), the i_m and j_m indices depend on channels and assign arbitrarily one of the values to 1 in $\tilde{K}_{:,:,m} \in \mathrm{R}^{K_D \times K_D}$ called shift matrix. Thus, K_D^2 possible shift matrices exist and any of them corresponds to a shift direction.

The shift operation could be seen as a bundle of memory operations shifting the input tensor channels into certain directions by convolving with shift kernels. In contrast to spatial 2D and depth-wise separable convolutions, the shift operation does not bring any overhead in terms of parameter or FLOPs cost due to fusing a point-wise convolution after it. This allows the point-wise convolution to directly fetch the shifted data from the cache. Thus, the shift operation has allowed a lightweight CNN model to achieve a remarkable performance on both parameter number and accuracy in [31], as seen from Table 5.

10) MobileNetV2

A year later, the MobileNet's creators, Howard *et al.* [99], proposed a new algorithm, MobileNetV2, improving the performance of MobileNets by revising the residual structure where skip connections were added between the bottleneck layers. Via those structural changes, MobileNet-V2 improved the Top-1 accuracy performance to 72.0 % and the parameter number to 3.4M in comparison with MobileNets. In addition, MobileNet-V2 was not only improved for image classification but was also designed for object detection tasks.

11) ShuffleNet

ShuffleNet [100] as a computation-efficient algorithm was designed for mobile devices. It is inspired by and linked to the grouped [28], and depth-wise [70], [97] types of convolutions. In a nutshell, shuffled grouped convolutions involve a grouped convolution combined with a channel shuffling.

As explained earlier in Section III.A.1) group convolutions significantly reduce the number of the total operations. Nevertheless, there is a drawback that each filter group could operate over a certain fixed portion of information from the previous layers, as shown in Fig. 4 (d). As such, filter groups are limited in the learning of few features, which weakens the representation and information flow throughout different channel groups. Channel shuffling overcomes that problem by mixing up the information between channels. In Fig. 4 (e), the feature map with three channels, $I \in \mathbb{R}^{I_D \times I_D \times I_C}$ obtained subsequently to the first grouped convolution via three filter groups in Fig. 4 (d), is first divided into several subgroups, and then these subgroups are mixed up. Following the shuffling, the usual second grouped convolution is performed with the difference of strengthening the representation and information flow between channel groups.

In ShuffleNet, the point-wise grouped 1×1 convolution is also considered instead of the 3×3 convolutions employed in [70], [97]. The idea behind that was the lower computational efficiency of the 1×1 point-wise convolutions. The operation is identical to the grouped convolutions with a minor alteration of the used kernel size.

As a result, the model utilized grouped, depth-wise, and point-wise convolutions enabling a computation-efficient and lightweight algorithm while maintaining the accuracy level of 73.6 % in [31] as seen from Table 5. Thus, it has gained popularity in CNNs for mobile devices.

12) SqueezeNext

After the SqueezeNet model [96] reduced dramatically the parameter number of AlexNet [28] by keeping similar accuracy levels, Gholami *et al.* [101] introduced a new family of CNN architecture, SqueezeNext, which enables the same accuracy with 112× fewer parameters. There are different types of SqueezeNext models with varied parameter numbers between 1.5 and 0.54M. One of them, the 1.0-SqNxt-44 model, has achieved a 5% better Top-1 and Top-5 performance compared to SqueezeNet with the same number of parameters.

Their design was based on the lower rank filters and compression of the redundant parameters of SqueezeNet, which do not affect the accuracy. Moreover, the architecture used a final bottleneck layer cooperating to reduce the input channel size of the last FCL in SqueezeNet. The application of these microarchitecture level strategies has allowed to achieve a considerable reduction of the parameter number. (Details about micro/macro architecture design parameters are introduced in the Appendix.)

13) ColorNet

Another interesting finding has been explored by Gowda and Yuan [102]. They have focused on the importance of colour spaces of RGB images in datasets and shown that colour spaces, especially transformations of RGB images, are able to significantly improve classification accuracy. By using that idea, their architecture, named ColorNet, takes RGB images as input and converts the images into 7 different colour spaces. Following that, every colour space is used as an input to individual DenseNet modules [103]. By applying this method, the 84.6% Top-1 accuracy performance of ColorNet on ImageNet [31] was listed as the winner in 2019 on Table 5.

C. DESIGN TRENDS ON IMAGENET DATASET

The performance of the most prominent CNNs for image classification is summarised in Table 5, in which the names of the lightweight models are shaded in light grey colour, while the names of the heavyweight models – in dark grey colour. For each model, the following data are provided: literature source, year of introduction, Top-1 accuracy, Top-5 accuracy, number of parameters, number of FLOPs, model size in MB, and number of layers.

It can be observed from Table 5 that the heavyweight models achieve better accuracy levels mostly by increasing the layer and parameter numbers without regard for the model size or energy consumption. For instance, ViT-H/14 [87], which has achieved the best ever Top-5 accuracy of 99.0%, has a high layer number of 132, and a very high parameter number of 632M as well. Similarly, MPL- EfficientNet-L2 [90], which is a deeper CNN model than ViT-H/14, has obtained the second-best score on Top-5 accuracy of 98.8% and it has reached the highest Top-1 accuracy ever of 90.2% by including 154 network layers and having 480M parameters. A major downside of the heavyweight CNN designs is that their large parameter numbers prevent them from being implemented on computational devices with limited off-chip memory, such as FPGAs, ASICs, SoCs, etc.

On the other hand, as it can be seen from Table 5, the lightweight CNNs, aim to reduce the model sizes by conceding their accuracy performance, for example SquuezeNet [96], MobileNets [70], ShiftNet [98], and SqueezeNext [101]. In SqueezeNet, the layer number has been reduced to 19, the parameter number to 1.2M, and the model size to 0.5MB by utilising 1×1.2D spatial convolutions. SqueezeNet was implemented in an FPGA with 10MB on-chip memory without using an off-chip memory [104]. By taking advantage of depth-wise and point-wise convolutions MobileNets obtained 4.2M parameter number. As the best performance lightweight algorithm in Table 5, ShiftNet has benefitted from a parameter-free shift operation in

TABLE 5. Performances of image classification algorithms on ImageNet.

Model Heavyweight Lightweight	Year	Top-1 %	Top-5 %	Parameters ($x10^6$)	FLOPs ($x10^6$)	~Model Size [a] (MB)	Layers
LeNet-5 [4] [b]	1998	95.5	98.5	0.06	-	-	5
AlexNet [28]	2012	57.2	80.3	62	720	240	8
ZFNet [74]	2013	73.9	88.3	60	630	170	8
VGGNet [75]	2014	68.5	88.5	138	15500	575	19
Inception-v1 [29]	2014	77.4	93.3	6.8	1550	53	22
Inception-v3 [94]	2015	82.8	94.5	23.6	-	-	159
ResNet [17]	2015	75.1	93.3	23.7	3800	-	50
Inception-v4 [95]	2016	82.3	96.2	43	1650	83	144
Trimps-Soushen [31]	2016	82.3	97.1	19	-	-	89
SqueezeNet [96]	2016	57.5	80.3	1.2	352	0.5	19
Xception [97]	2017	79.0	94.5	22	1200	88	126
ResNetXt-50 [105]	2017	77.7	93.8	22.2	4100	-	50
SENet [76]	2017	82.7	96.3	146	3870	210	201
MobileNets [70]	2017	70.6	89.9	4.2	569	16	29
ShiftNet [98]	2017	58.8	82.0	0.8	279	-	44
MobileNetV2 [99]	2018	72.0	90.1	3.4	300	300	114
NASNet Large [77]	2018	82.7	96.2	89	2380	343	32
ShuffleNet [100]	2018	73.6	89.7	143	140	75	50
SqueezeNext [101]	2018	60.3	83.5	0.9	310	-	112
ResNeXt [78]	2018	85.4	97.6	829	153000	458	101
ColorNet [102]	2019	84.6	96.1	19	250	26	-
EfficientNet-L2 [80]	2020	85.5	97.5	480	23500	-	154
FixEfficientNet-L2 [85]	2020	88.5	98.7	480	23500	-	154
ViT-H/14 [87]	2020	88.6	99.0	632	32120	534	132
LambdaResNet200 [89]	2021	84.3	96.3	42	34000	-	200
MPL-EfficientNet-B6-Wide [90]	2021	90.0	98.7	390	-	-	-
MPL-EfficientNet-L2 [90]	2021	90.2	98.8	480	-	-	154

[a] Model size means the required number of bytes to store all of the trained model's parameters; [b] Accuracy values of LeNet-5 for MNIST dataset [72].

its layers achieving AlexNet accuracy [28] with only 0.8M parameters. The aforementioned CNN developments reveal the importance and impact of enhancing the convolution operations and kernels, which have substantially contributed to these breakthroughs. However, it is beyond dispute, that the accuracy performances of the lightweight CNN designs are considerably lower compared to the heavyweight models. Nevertheless, both low and high parameter number/model sizes have pluses and minuses, as discussed in Section V.

D. EVALUATION OF COMMON CONVOLUTION PROCESSES

Due to the use of CNNs for image classification in a wide range of applications, many researchers have worked on techniques to reduce their high storage overhead and computational cost, resulting in a compact and accurate model design. As part of that, it has been recognised that carefully designed convolutional operations, which serve as a basic component of the network layers, as the ones are shown in Fig. 4, can bring significant benefits. Some of them are accuracy-oriented, whereas others improve efficiency. In this section, the training and runtime performances of each of the convolutional operations in Fig. 4 are examined and compared by employing them in the well-known CNN architecture ResNet [17].

As explained in Section II.A.2), the spatial 2D convolution in Fig. 4 (a) is frequently used especially in heavyweight models [28], [29], [75]. The rest of the convolution types in Fig. 4 are proposed to enhance the performance of the spatial convolutions. For example, the grouped convolution [28] in Fig. 4 (d) brings three essential benefits, which are: enabling of model-parallelization; more structured learning with unique representations of data; and computational efficiency by reducing multiply-add operation number. Furthermore, the shuffle operation [100] in Fig. 4 (e) allows improving the accuracy of the grouped convolution kernels. On the other hand, the depth-wise separable convolutions in Fig. 4 (b) and (c), which are used in designing lightweight architectures [70], [97] reduce the parameter and FLOPs numbers. However, despite their superiority in terms of lowering the computational cost, their fragmented memory footprints prevent efficient implementations in practice as shown by expressions (10) and (11). To overcome that constraint, the shift operation [98] in Fig. 4 (f) presents an alternative by cooperating with the point-wise convolutions in Fig. 4 (c) and aggregating the spatial information for free.

1) OUTLINE OF EXPERIMENT

The approach is to train separately different convolutions using the same reference architecture (ResNet-50 [17]). The six different convolution types in Fig. 4 are placed in the bottleneck module of the ResNet-50 architecture in Fig. 7 for a fair comparison. As the model in Fig. 7 is originally designed with spatial 2D convolutions, it forms the first experimental CNN with the required dimension adjustments arising from the used dataset, and four more designs are constructed using the remaining convolution types in Fig. 4. While the second is shaped with two-grouped convolutional kernels, the third design is created by three-grouped shuffle operations. As an exception in the fourth, 3×3 and 1×1 spatial convolutions in the original model are replaced by depth-wise and point-wise convolutional filters, respectively as they are usually deployed together [97], [70]. Regarding the final design, we remove the 3×3 spatial convolutions and place shift operations to form the fifth model. Then, we train the designed models, test them and analyse their accuracy and inference time on the CIFAR-10 and CIFAR-100 [18] datasets. Moreover, by decomposing the architectures into basic components they are analysed to find their individual execution times on compute-bound (CPU) and memory-bound (GPU) computation platforms during inference.

In the training, two NVIDIA GPUs Tesla P100s, part of the ALICE High-Performance Computing Facility at the University of Leicester, are employed according to five different model requirements with a mini-batch size of 128 and a base learning rate of 0.1. The weight decay and momentum are 0.0001 and 0.9, respectively. The training for each architecture stops after 32k iterations. In addition, the learning rate decays with a factor of 10 after 16k and 24k iterations. The inference time of the trained models is tested on both a CPU (Intel Core i5-8500) and a GPU (Tesla P100) by decomposing it into proportional slices to the time taken by the basic components of the architecture.

2) EVALUATION RESULTS

The individual performances of the evaluated convolution types are presented in Table 6 and Fig. 8.

As it could be seen from Table 6, different convolutions perform differently when deployed in the same CNN model. The grouped convolution and shuffle operation reduced the parameter and computational overheads compared to the spatial 2D kernels as well as enabled a better accuracy, which confirms the findings presented in the AlexNet and ShuffleNet sections above. Notably, as shown in Table 6, there is a sharp drop in the parameter and FLOPs number when the depth-wise separable and shift operations are being employed. Moreover, they achieve a similar accuracy performance compared to the others.

It should be mentioned that the accuracy levels of the evaluated models in Table 6 are lower than those reported in [17], [98], [100]. One reason for that is the smaller number of iterations in our experiment, as the aim here is to assess the relative performance of the different convolution types.

TABLE 6. Cifar-10 and 100 training of ResNet-50 with different convolution types.

Convolution Types	Parameters ($x10^6$) / FLOPs ($x10^6$)	CIFAR-10 Accuracy (%)	CIFAR-100 Accuracy (%)	Inference (ms)
Spatial	22.7/3800	79.32	62.12	79
Grouped	16.1/1970	81.40	61.02	73
Shuffle	10.3/1330	83.13	62.03	61
Depth-wise	4.5/878	79.42	60.70	53
Shift	1.4/318	78.13	59.87	37

It can also be seen from Table 6 that the accuracy rates on the CIFAR-10 dataset are higher compared to CIFAR-100 for all convolution types, which is due to the difference in the class number, as detailed in Section II.D.1).

Fig. 8 represents a summary of the timing analysis of the ResNet-50 [17] model during the inference stage for the evaluated convolutions and operations in Fig. 4. For a clear demonstration, element-wise operations, such as ReLU, batch normalization, tensor addition and concatenation, etc., are incorporated into a single word (Elwise.). Also, the pre-processing time for the data feeding is not represented here. Results are obtained under PyTorch and averaged from 30 runs. Fig. 8 (a) shows evaluation results for the ResNet-50 model with spatial filters; Fig. (b) and Fig. (c) - with grouped filters and shuffle operation, respectively; while (d) includes depth-wise and point-wise convolutions; and (e) uses shift operations. The convolution time (Conv.) is depicted by blue colour for the CPU-based executions, and by orange colour for the GPU-based executions.

First of all, it could be seen from Fig. 8 that the element-wise operation (Elwise.) in each ResNet-50 model takes more time in the GPU compared to the CPU during inference as they are more prone to compute-bound (CPU) computation platforms. The reason behind that is their memory access cost is heavy even if they have small FLOPs. As an expected outcome, the slices of the convolution (Conv.) operations in Fig. 8 (b) and (c) are lower compared to (a) as grouped convolutions reduce computational overhead. It can also be observed from Fig. 8 (c) that the slice of the shuffling operation (Shuffle) equals 12% of the runtime of the GPU, which

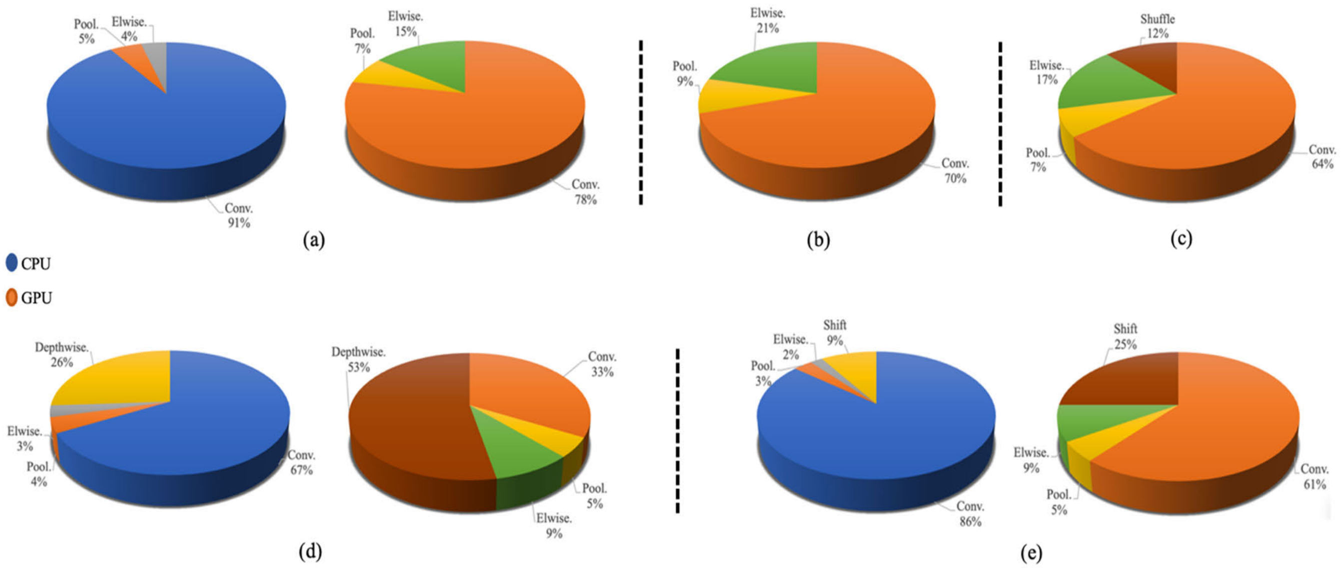


FIGURE 8. Graphical representation of the runtime timing analysis of the ResNet-50 model for the convolutions and operations in Fig. 4. (a) ResNet-50 model with spatial filters; (b) and (c) - ResNet-50 model with grouped filters and shuffle operation, respectively; (d) ResNet-50 model with depth-wise and point-wise convolutions; (e) ResNet-50 model with shift operations.

means that its parallel hardware is limited by the shuffling operation. Nevertheless, the accuracy level and efficiency of the shuffling operation outperform spatial convolutions as per the results in Table 6.

The pie charts in Fig. 8 (d), verify expressions (10) and (11). Even though depth-wise convolution filters require fewer parameters and lower computational overhead in theory, unlike spatial convolutions, their memory access demand dominates computations and limits the performance of memory-bound computation platforms such as GPUs. As it can be seen in Fig. 8 (d), the dept-wise convolution operations (Depthwise.) occupy a substantial part of the runtime in GPU amounting to 53%. This drawback also indicates higher energy consumption per floating-point operation, thereby leading to an inefficient computation. Lastly, Fig. 8 (e) proves that the shift operation takes a step further compared to the depth-wise separable convolution in terms of performance. Despite its considerable runtime, caused by the large number of memory accesses instigating bottlenecks, the shift operation requires a lower number of parameters and FLOPs and occupies a smaller slice of the GPU runtime compared to the dept-wise convolution in Fig. 8 (d), i.e. 25% versus 53%.

In conclusion, based on a fair comparison, Table 6 and Fig. 8 practically confirm the introduced theoretical findings for the popular convolution types and reveal invisible timing aspects related to their runtime execution.

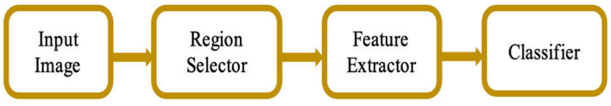


FIGURE 9. Data-flow diagram of traditional detectors.

IV. REVIEW OF CNNs FOR OBJECT DETECTION

Object detection has been a long-standing theme in computer vision research. With regards to self-driving vehicles, object detection is an essential part of modules in the ADSs pipeline, such as scene understanding and object tracking. The aim is to identify the locations and sizes of the objects present in images taken by cameras at the viewing angle of the car. These could be both static objects, e.g., traffic lights, road signs, etc., and dynamic objects, e.g., vehicles, pedestrians, etc.

Looking back over the last two decades of detection methods, until the early 2010s the limitations on computing resources, datasets, and the mostly theoretical nature of DNN development had led to employing traditional detectors, such as DPM [106], Selective Search [107], Oxford-MKL [108], HOG [109], NLPR-HOGLBP [110], SIFT [111], VJ Det [112], Bag of Words [113], etc. A modular data-flow diagram and a historical publication timeline of these popular detectors are depicted in Fig. 9 and Fig. 10, respectively.

The purpose of the Region Selector module in Fig. 9 is to prepare sliding windows of different sizes and to slide them over the image from left to right and top to bottom by keeping a certain step size. Then, cropped image blocks are produced by the sliding windows and converted to images with uniform dimensions. In the Feature Extractor module, features are extracted from the images by the way of deploying different algorithms such as HOG [109], SIFT [111], etc. The last module in Fig. 9 is a Classifier, which is used to identify the category of objects, extracted in the previous step via different algorithms like SVM [114] and Adaboost [115]. Despite their popularity, the traditional detectors were relatively mature and had multiple

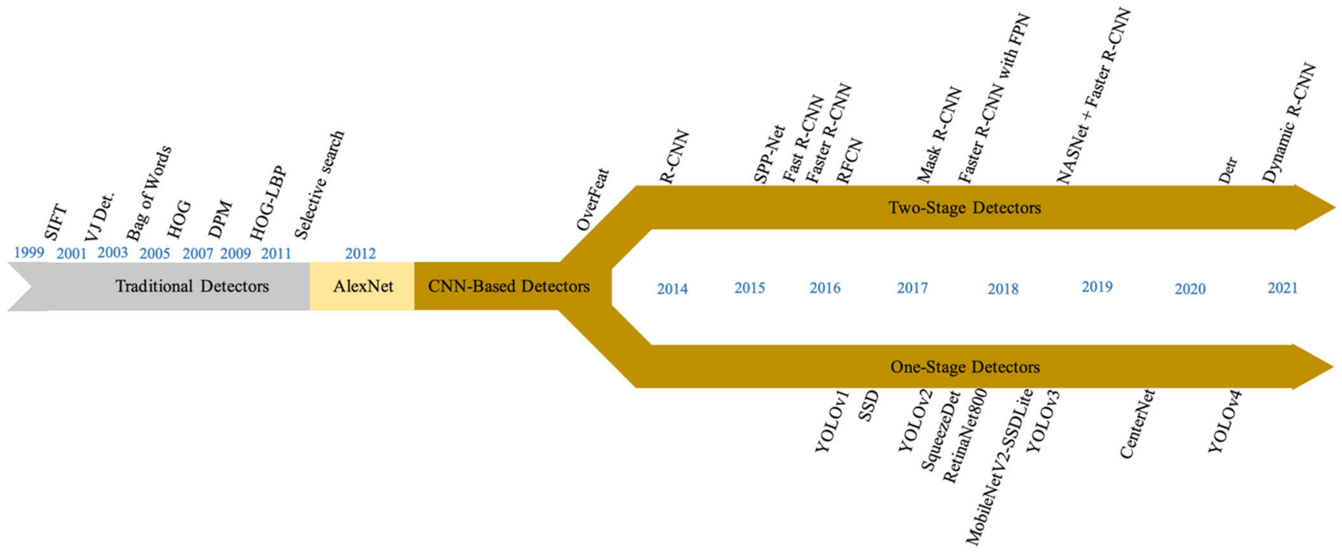


FIGURE 10. Timeline of traditional and CNN-based object detectors.

drawbacks, such as high window redundancy, computing complexity, etc.

Following the emergence of high-performance computing devices and inclusive datasets as well as the staggering performance of AlexNet [28], in the last decade, the attention in the field gradually turned to CNN-based solutions, as illustrated in Fig. 10.

In the rest of this section, first, building blocks for CNN-based object detection are discussed, such as backbone networks and baseline types, then representative state-of-the-art designs are outlined, dividing them into two categories: one-stage detectors and two-stage detectors. Table 7 presents experimental results from the literature for each of the reviewed models based on particular image datasets.

A. BACKBONE NETWORKS

Object detection models deploy a classification algorithm as a backbone or a base network that acts as a basic feature extractor. The CNN models for image classification, introduced in Section III above, such as ResNet [17], Xception [97], SqueezeNet [96], MobileNets [70], ShuffleNet [100], can be directly adopted or improved with new features to be used as a backbone for object detection tasks. In several publications [116]–[118], specific requirements have been stated for existing classification algorithms to perform better in a detection pipeline. Additionally, it is recognised that even if detection speed is a key factor, e.g., in real-time applications, high precision and accuracy hold at least an equal importance as well. That means that there is a need for a good trade-off between speed and accuracy [19] when selecting backbone networks for detectors. The recently published high-performance CNN architectures for classification may help to solve the problem, as quality feature extraction raises up the whole detection performance. For example, He *et al.* [119] achieved a remarkable performance following this approach. Further details are included in the descriptions of the individual detection algorithms in Sections C and D below.

B. TYPICAL BASELINES IN CNN-BASED DETECTOR DESIGN

In CNN-based detector design, there are two commonly used baseline schemes: one-stage and two-stage detection pipelines. Among the two-stage detectors, the Region-based Convolutional Neural Network (R-CNN) series [120]–[122] are the most prevalent ones, whereas YOLO [116] and SSD [123] are the CNNs of choice for the one-stage detectors.

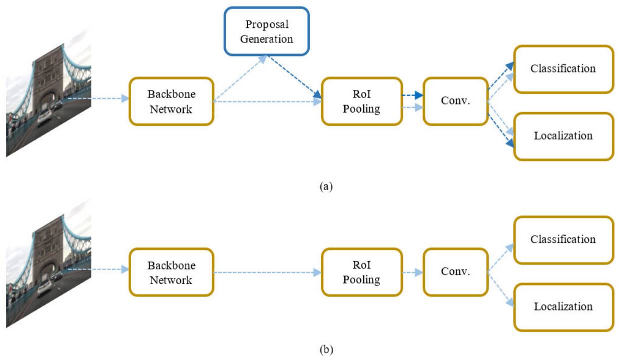


FIGURE 11. Functional diagrams of typical detector pipelines. (a) Two-stage detector. (b) One-stage detector.

The functional diagrams of the two baselines are illustrated in Fig. 11. The two-stage detector pipeline is shown in Fig. 11 (a), where region proposal network block (Proposal Generation) feeds region proposals into the classifier and localization modules. These detectors differ from the one-stage detectors, depicted in Fig. 11 (b), in the operation of the

RoI (Region of Interest) pooling layer. Unlike the two-stage detectors, where the prediction of the bounding-boxes is carried out by the region proposal stage (Proposal Generation), in the one-stage detectors, it is implemented directly from the input images, which allows them to perform faster. This is why one-stage detectors are more preferred in real-time applications despite their low accuracy performance compared to the two-stage ones. Weighing the merits of the two schemes, it could be said that the accuracy of object recognition and localization is higher in two-stage detectors, whereas the inference speed is better in one-stage detectors [124].

Fig. 10 displays the entire timeline of both the traditional object detectors and the CNN-based designs of the last decade. It could be seen from Fig. 10 that the two-stage CNN-based detectors preceded the development of the one-stage detectors. Therefore, in the following review of the existing object detectors with CNNs, first, the two-stage designs are introduced in Section C, and then the one-stage detectors - in Section D.

C. TWO-STAGE CNN DETECTORS

1) R-CNN

In 2014, Girchick *et al.* [120] introduced a detection algorithm called Region-based Convolutional Neural Network (R-CNN), which featured a high object detection performance. Fig. 12 illustrates the architectures of the three best-known R-CNN models: R-CNN [120], Fast R-CNN [121] and Faster R-CNN [122] during both the testing and the training phase. The diagrams related to the testing phase are depicted using only amber colour. The diagrams of the training phase are depicted using both amber and purple colour for the Fast R-CNN and the Faster R-CNNs in Fig. 12 (b) and Fig. 12 (c), respectively, which include multi-task loss functions.

As shown in Fig. 12 (a), the R-CNN method starts by identifying category-independent ∼2k region proposals containing potential objects, and then, proceeds each region to the backbone network AlexNet [28] to extract 4096-dimensional feature representations. Lastly, while an SVM is deployed for classification, fine adjustments of the bounding boxes are provided by a Bounding-Box regression and a greedy non-maximum suppression (NMS) method [125]. Using such a design, R-CNN achieved an improved mean average precision (mAP) of 58.5% on the Pascal VOC dataset [35]. Despite its accuracy performance, R-CNN had a few design problems. For instance, it was slow in processing the images, taking a second per image or 1 frame per second (fps), and required many GPU days to be trained, as ∼2k region proposals per image had to be classified in the network. The algorithm, therefore, could not be utilised in real-time. In order to solve these drawbacks, new versions of R-CNN were proposed afterwards.

2) SPP-NET

He *et al.* [126] proposed a new algorithm, SPP-Net, with a strategy, which is referred to as spatial pyramid pooling (SSP) eliminating the singly passing of ∼2k region proposals to the backbone network in R-CNN. SPP-Net firstly computes a convolutional feature map of the whole input image and following that, classifies each object proposal by taking advantage of the spatial pyramid pooling layer. Thus, SPP-Net accelerated the testing time of R-CNN by between 10 and 100 times. Another superior side of SPP-Net was that it could be used for increasing the performance of all CNN-based tasks, like image classification. ZF-Net [74] was used as a backbone to measure the mAP performance of SPP-Net on Pascal VOC [35] dataset and it achieved 60.9%, which was better than the R-CNN performance with a lower computational load. In spite of its advantages, SPP-Net also had several drawbacks of [120]. The training was a multi-stage pipeline that included extracting features from the input image, which required fine-tuning of the network with log loss, using of SVMs, and providing bounding boxes for the regression stage. All those signalled new approaches in the field.

3) FAST R-CNN

In an attempt to resolve the problems of R-CNN [120] and SPP-Net [126], outlined in Sections 1) and 2) above, Girshick [121], from Microsoft Research, proposed a new algorithm called Fast R-CNN at the ICCV 2015 conference. By introducing several innovations in the Fast R-CNN design, the training and testing speed of R-CNN was improved, and its detection accuracy was increased as well. A novel RoI pooling layer was fitted to the network as the single-level spatial pyramid pooling layer in SPP-Net.

Fig. 12 (b) represents the architecture, where region proposals are first created by a Selective Search algorithm [107] and then, mapped onto the feature maps. Following that, the RoI Pooling layer prepares different feature regions as fixed-size feature vectors, which are then used as inputs in FCLs. Finally, whereas object categories are predicted by the softmax operation [127] with log loss [128] in contrast to R-CNN employing SVM [114], object locations are located by a bounding-box regression with a smooth L1 (absolute) loss [129]. Thus, its performance on the Pascal VOC 2007, 2010, and 2012 datasets on training with the backbone network VGG-16 [75] was measured as 70.0%, 68.8%, and 68.4% mAP, respectively. Fast R-CNN was trained on the VGG-16 base network 9× faster than R-CNN and was 213× faster during testing. As for SPP-Net, Fast R-CNN was 3× faster in training, 10× faster in testing, and more accurate when training on VGG-16. Besides improving training, testing, and accuracy performance, there were three other contributions which are: end-to-end training, no need for disk storage, and updates on all network layers.

4) FASTER R-CNN

In R-CNN [120] and Fast R-CNN [121], region proposals were being generated by different blocks, rather than using only one convolutional neural network. That was time-consuming, i.e., ∼2.3s for making predictions and ∼2s for

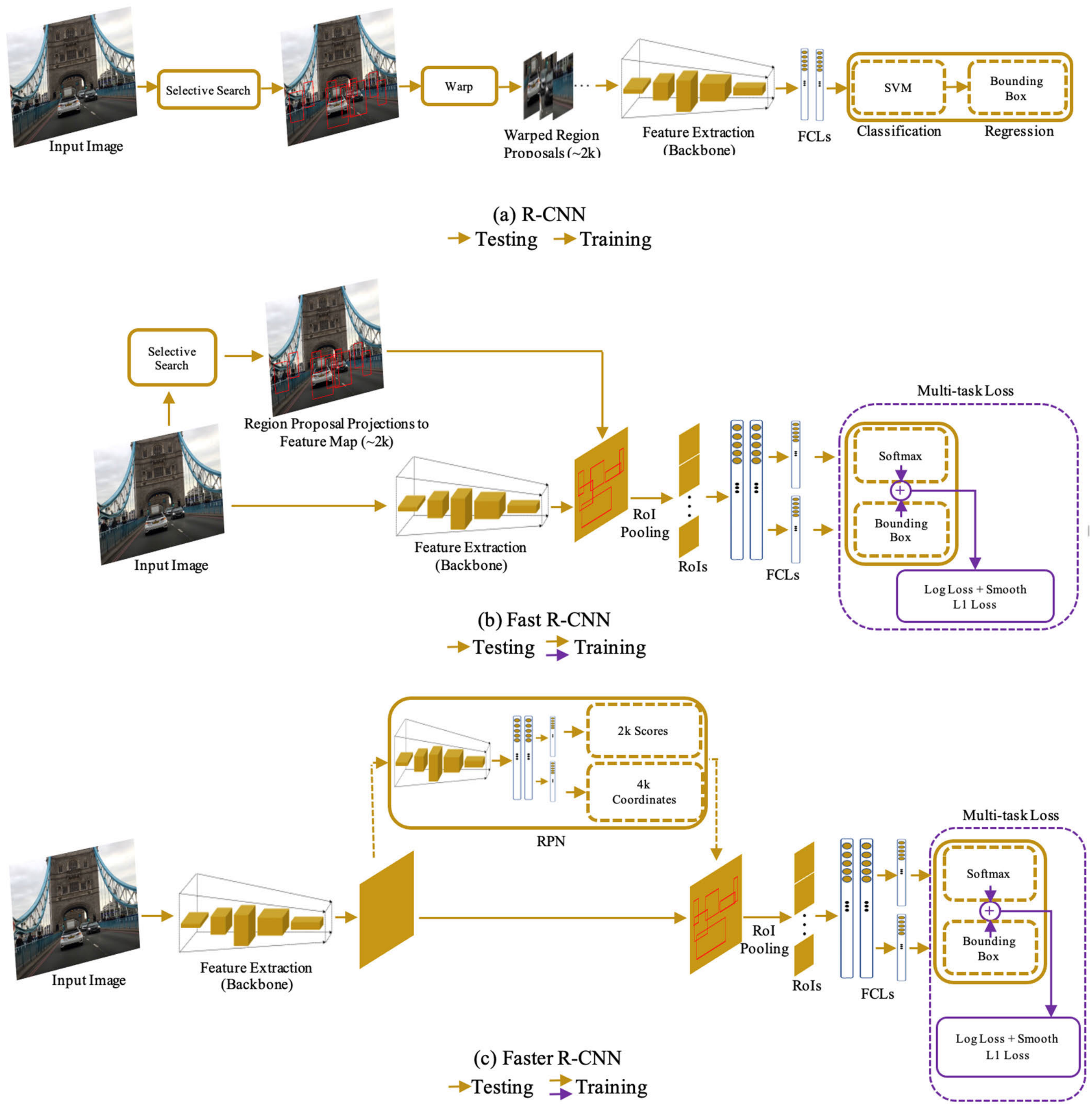


FIGURE 12. Architectural demonstrations of the three best-known R-CNN models R-CNN, Fast R-CNN and Faster R-CNN [120]–[122].

generation of 2k RoIs, representing a bottleneck in system performance. To solve the drawback, Ren *et al.* [122] proposed a novel Region Proposal Network (RPN), shown in Fig. 12 (c), which was constructed by using CNN-based layers, where the region proposals are generated right after the backbone network. Unlike its ancestors, the number of RoIs is not a constant value and is defined by the size of the feature map. Thus, the region proposals were implemented on GPUs with nearly free of computation cost compared to previous baselines. The performance results of the algorithm are detailed in Table 7.

5) RFCN

Dai *et al.* [130] addressed the basic shortcoming of Faster R-CNN [122], which was related to the RoI layer being located between the backbone network and the object detection modules. Such a design caused a lack of translational invariance in the detector as the RoI was converting the multi-dimensional feature map to fixed size FCLs. To overcome the problem, they proposed position-sensitive score maps in the design of RFCN, based on a fully convolutional network. Thus, all learnable parameters were convolutional and sharable in the kernels, which was increasing the translational invariance. It achieved 80.5% mAP on the Pascal VOC [35] dataset with a ResNet101 [17] backbone network. Further details are represented in Table 7.

6) MASK R-CNN

Mask R-CNN [119] is a result of extending Faster R-CNN [122]. It has been primarily designed as a framework to perform a segmentation task in addition to object detection. The Mask network as a modernised version of FCNs [131] is placed in parallel to the detection pipeline for the generation of split masks for each RoI. Due to an alignment problem between the feature map and the original image in the RoI pooling layer, which uses integer quantization, a RoIAlign layer based on a bilinear interpolation method was employed. This allows preserving the alignment at pixel-level in the exact spatial locations between the feature map and input image. It also enables to achieve a better detection accuracy. Thus, by deploying ResNeXt-101 [78] as a base network, a rise in performance was achieved compared to the previous detectors, as shown in Table 7.

7) FPN

In 2017, Lin *et al.* [117] proposed the Feature Pyramid Network (FPN) that fuses features and enhances the system detection performance. The algorithms, proposed until then, were detecting either a top-level feature or were performing an independent detection in feature layers. This disallows to combine classification and location information. However, FPN introduced a way of laterally connecting layers top-down and bottom-up, evoking a pyramidal operation. FPN was incorporated in Faster R-CNN [122] and achieved state-of-art results on the MS COCO dataset [36], as detailed in Table 7.

8) NASNet

NASNet [77] was not solely designed for the purpose of image classification. It was also targeted at object detection tasks, as the learned features from the classification process can be used in detectors. Thus, NASNet used the Faster R-CNN [122] framework for detection and obtained a remarkable accuracy of 43.1% mAP compared to previous baselines, such as MobileNetV2 [99] with 22.1% mAP on MS COCO [36] dataset.

9) DETR

Carion *et al.* [132] from the Facebook AI team developed a new CNN model, called DETR, which was designed as a solution to the direct set prediction problem for object detection. The model does not include anchor generation [133] and non-maximum suppression [125] as they prevent the use of prior information in the detection pipeline.

There are two main contributions. The first one is a set-based global loss, which enables powerful predictions through partite matching and the other one is a transformer encoder-decoder architecture. Via these strategies, the model has improved a bit further the accuracy rate of the two-stage detectors to 44.9% mAP with 10 fps in the COCO dataset [36].

10) DYNAMIC R-CNN

In 2020, Zhang *et al.* [134] published an advanced R-CNN algorithm. Their idea was based on addressing the inconsistencies between the dynamic training process and the fixed network adjustments, which significantly affect performance. To tackle the problem, Dynamic R-CNN provides an automatic and dynamic adjustment on both the shape of the regression loss function, e.g., the parameters of Smooth L1 Loss [129], and the label assignment criteria, e.g., the IoU threshold [135], by using the region proposal statistics in training. Moreover, no extra cost has been added to the model. Thus, the model performance was improved considerably achieving 49.2% mAP on the COCO dataset.

D. ONE-STAGE CNN DETECTORS

1) YOLO

Even though the proposed RPN in Faster-RCNN [122] reduces the region proposal number and the overlaps between them, the repetitive computations due to inevitable overlaps still cause an essential bottleneck in the performance of these detectors. So as to deal with the problem, Redmon *et al.* [116] introduced a new hybrid CNN-based architecture, named YOLO (You Only Look Once), which is illustrated in Fig. 13 (a). It combines the region proposals and detection branches into one single stage in contrast to R-CNN [120], Fast R-CNN [121] and Faster R-CNN [122], as seen in Fig. 13 (a), where the classification and bounding boxes regression are responsible to detect the object centres in the grid cells.

YOLO detectors first divide the input images into $S \times S$ grids and then, each grid is responsible to predict C class probabilities, B bounding boxes, and the confidence value for those boxes. Thus, the input is encoded as a tensor with $S \times S \times (5B + C)$ dimension. The end-to-end architecture is built by 24 convolutional and 2 FCLs. As a result, YOLOv1 has achieved a high processing performance of 45 fps, which satisfies real-time implementation requirements. Further details are presented in Table 7.

Despite its success, the YOLO model has also exhibited a few limitations. As a major problem, its detection ability on dense small objects is inefficient, since each grid predicts two bounding boxes from the same class at most. Another shortcoming is its generalization ability in case of different aspect ratios of objects on different images during testing.

Lastly, the loss function is in a need of improvement, as it affects the detection score.

2) SSD

As discussed in Sections C) and D.1) above, the R-CNN series [120]–[122] and YOLO [116] come with their superior and inferior features, trading-off between accuracy and speed. Liu *et al.* [123] proposed the Single Shot MultiBox Detector (SSD), which has advantages on both sides. In the SSD design, VGG-16 [75] is used as the backbone network, in which the fully connected sixth and seventh layers are replaced with convolutional layers as well as four more convolutional layers are placed at the end.

The entire design, which comprises six-stages, is hierarchical, and represents a single forward pass network. The reason for such a design is to provide hierarchical extraction of features, where each hierarchical layer provides object classification and bounding-box detection with different semantic information levels, as it could be seen from Fig. 13 (b). Moreover, each stage applies a fast non-maximum suppression (NMS) technique [125], aimed at post-processing of redundant bounding boxes. The purpose is to eliminate overlaps in bounding boxes at every stage, and it also leads to reducing the amount of computation without compromising the accuracy. Thus, SSD512 outperforms Faster R-CNN on both accuracy and speed, and SSD300 shows a processing performance of 59 fps, which is higher than that of YOLO.

3) SqueezeDet

In 2017, another CNN model was proposed by the creators of SqueezeNet [96], named SqueezeDet [136], which is a fully convolutional neural network designed for object detection. In fact, it received inspiration from both YOLO [116] and SqueezeNet. SqueezeDet is aimed at satisfying real-time constraints of embedded deployment in terms of model size, energy efficiency, as well as a high level of accuracy, as discussed in Section V.D. below.

Due to its single forward pass neural network pipeline, the SqueezeDet architecture is effectively fast, small model sized, accurate and energy efficient. By keeping the same accuracy as previous baselines, such as YOLO, it enables 8MB model size, which is 30× smaller compared to Fast RCNN + AlexNet [121]. The energy consumption of 1.4J per fps in NVIDIA Titan X GPU is 35× lower compared to [121], and, the inference processing speed of 57 fps, is 20× faster compared to [121]. It also requires fewer DRAM (Dynamic Random-Access Memory) accesses and enables the best average precision in all three difficulty levels of cyclist detection of the KITTI dataset [37], which is introduced in Section II.D.1).

4) YOLOV2 (YOLO9000)

As stated in Section 1) above, YOLO has several limitations, causing lower localization and recall performances. To address the problems, the YOLOv2 [137] model introduces a number of improvements on YOLO. First, it enhances the generalization capability by means of batch normalization [138], which speeds up the optimization process. The anchor idea [122], aimed at increasing the generalization capability in different aspect ratios, is also incorporated in YOLOV2, and thus, more scale and aspect ratio can be predicted by each grid cell. The second improvement is that it trains high-resolution classifiers to locate images with higher resolutions. Third, in order to increase the detection ability, the K-means clustering algorithm [139] is deployed to automatically find the bounding boxes. Lastly, it handles the instability of the YOLO model [116] by limiting the ground truth offset in regard to the grid coordinates. By upgrading YOLO with these advancements and designing a new base network, Darknet-19, based on VGG-16 [75], YOLOV2 has achieved a 78.6% mAP on the COCO dataset [36] and a higher number of learned object categories.

5) MobileNetV2-SSDLite

MobileNetV2-SSDLite is based on the classification CNN MobileNetV2 [99], which was discussed in Section III.B.10) above. It includes an architecture for object detection using a modified version of the SSD (Single Shot Detector) model [123] that is a mobile variant of SSD, whereby the main idea comes from the separation of regular convolutions. All the spatial 2D convolutions in the SSD prediction layers are replaced with separable convolutions (i.e., depth-wise convolutions, followed by point-wise convolutions). As a result, its performance was even better than that of YOLOv2 [137] with 10× fewer parameter number and 20× less computational load.

6) YOLOv3

YOLOv3 [141] inherits the superior sides of the previous YOLOv1 and YOLOv2 models and handles their shortcoming balancing accuracy and speed. To actualize the object, YOLOv3 combines residual networks [17], FPN [117], and binary cross-entropy/log loss [128]. As a result, the model can detect complex multi-size objects with more categories achieving performance of 28.2% mAP at 22ms or 33.0% mAP at 51ms on the COCO dataset [36].

7) CenterNet

In 2019, Duan *et al.* [142] proposed an algorithm, CenterNet, whose motivation comes from the realisation that faults in the detection of bounding boxes are the main problems in architectures, and if the cropped regions are looked into again, the accuracy performances could level up. CenterNet, therefore, focused on minimising that kind of errors. They built their framework being inspired by CornerNet [140], which is a typical one-stage key-point based detector. Unlike a pair of key-points in CornerNet, each object in CenterNet is detected as a triplet of key-points, which allows to improve precision and recall accuracy.

As seen from Fig. 14, a convolutional backbone network is used to define a two-corner feature map and centre key-point features on the map by employing cascade corner and centre

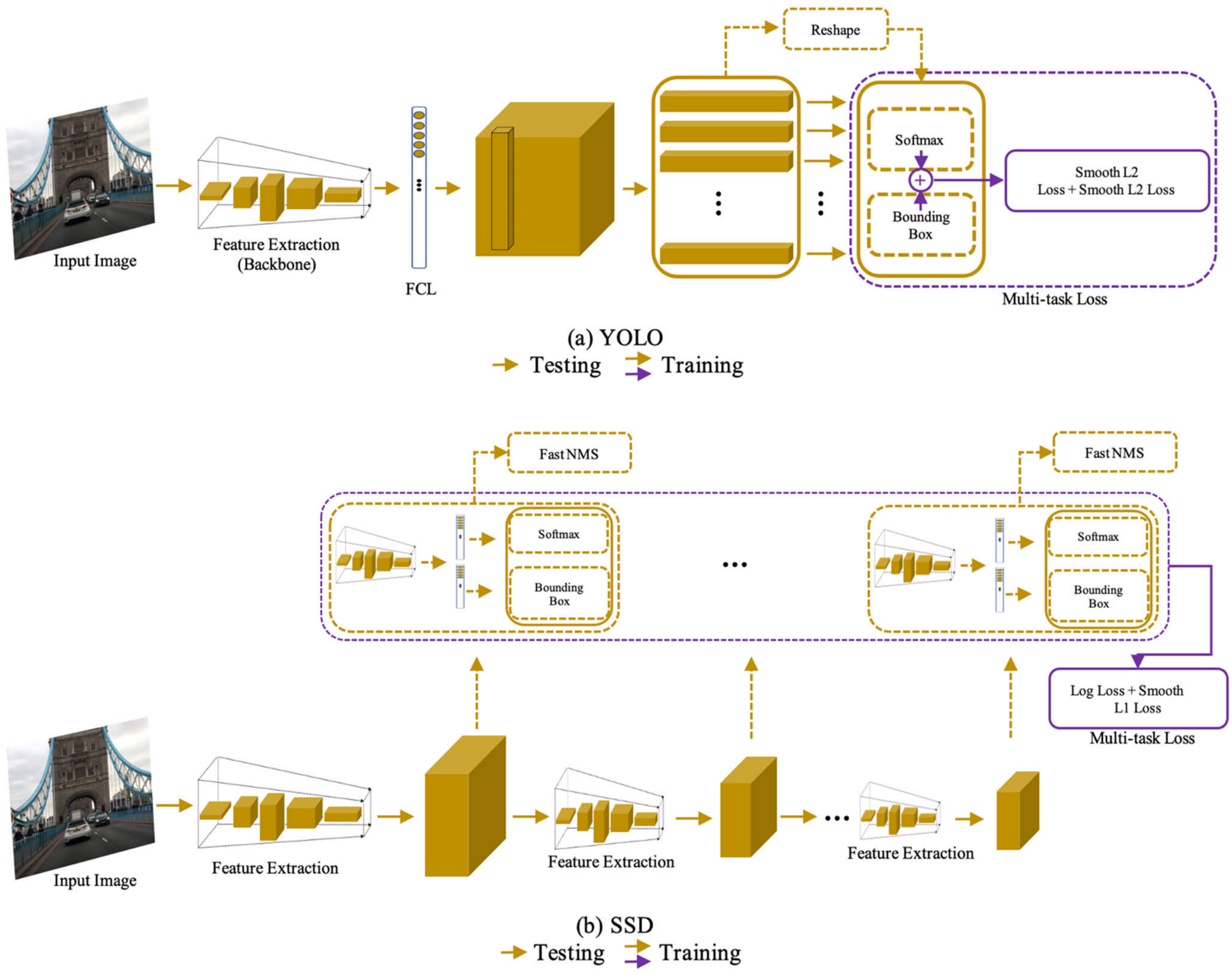


FIGURE 13. **Illustrations of the single-stage pipelines of the YOLO [116] and SSD [123] architectures.**

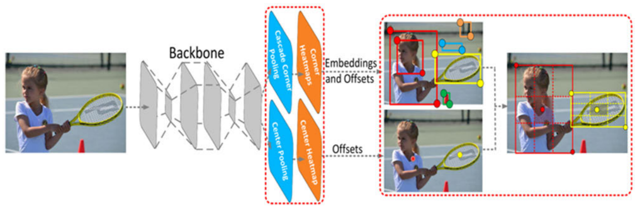


FIGURE 14. **CenterNet network [142].**

pooling modules, respectively. By using these feature maps, CenterNet detects the final bounding boxes. It outperformed on the COCO dataset state-of-the-art detectors such as Faster R-CNN [120], YOLO [116], SSD [123] and CornerNet, achieving an accuracy performance of 47.0% mAP.

8) YOLOv4

Bochkovskiy *et al.* [144] introduced YOLOv4 at CVPR 2020 as the latest version of the YOLO series. It incorporates several existing universal and unique strategies, such as DropBlock regularization, CIoU loss, Mosaic data augmentation, Mish activation, etc., into the architecture. CSPDarknet53 [145] is used as the backbone network as well as SPP [126] and PAN [147] are employed as neck layers. These are transition layers between the backbone and head blocks (classification and bounding boxes), which are inserted to improve detection accuracy. Consequently, its real-time processing speed and accuracy are quite good with 43.5% mAP at 65 fps, which is better than the previous YOLO version, YOLOv3 [141].

E. DESIGN TRENDS ON MS COCO AND PASCAL VOC

The performance of the leading CNN-based detectors is summarized in Table 7, in which the names of the two-stage models are shaded in light grey colour, while the names of the one-stage models – in dark grey colour. For each model, the following data are provided: literature source, backbone network, year of introduction, model size in MB, processing speed in frames per second (fps), implementation platform,

TABLE 7. Performances of CNN-based object detectors.

Model (*Two-stage* / *One-stage*)	Backbone Network	Year	~Model Size (MB)	fps	Platform	mAP Pascal VOC [35]	mAP MS COCO [36]
DPM [106]	-	2010	37	2.0	2.8 GHz Intel Xeon	33.7%	-
R-CNN [120]	AlexNet [28]	2014	-	1.0	Titan	58.5%	-
SPP-Net [126]	ZFNet [74]	2015	143	2.6	Titan	60.9%	-
Fast R-CNN [121]	VGG-16 [75]	2015	240	0.5	Titan X	70.0%	19.7%
Faster R-CNN [122]	VGG-16 [75]	2015	490	0.7	Titan X	73.2%	21.9%
RFCN [130]	ResNet101 [17]	2016	435	9.0	Titan X	80.5%	29.9%
Mask R-CNN [119]	ResNeXt-101 [78]	2017	-	5.1	Tesla M40	-	39.8%
Faster R-CNN with FPN [117]	ResNet-50 [17]	2017	137	6.7	Tesla M40	-	33.9%
NASNet + Faster R-CNN [77]	NasNet [77]	2018	-	-	-	-	43.1%
Detr [132]	ResNet-101 [17]	2020	720	10.0	Tesla V100	-	44.9%
Dynamic R-CNN [134]	ResNet-101 [17]	2020	550	13.9	RTX 2080TI	-	49.2%
YOLOv1 [116]	-	2016	-	45	Titan X	66.4%	-
SSD [123]	VGG-16 [75]	2016	-	59	Titan X	76.8%	28.8%
YOLOv2 [137]	DarkNet-19	2017	106	40	Titan X	78.6%	21.6%
SqueezeDet [136] [a]	SqueezeNet [96]	2017	8	57.2	Titan X	76.7%	-
RetinaNet800 [140]	ResNet-101 [17]	2017	500	13.7	Tesla M40	-	39.1%
MobileNetV2-SSDLite [99]	MobileNetV2 [99]	2018	18	5.0	CPU	-	22.1%
YOLOv3 [141]	DarkNet-53	2018	320	45.5	Tesla M40	-	33.0%
CenterNet + [142]	Hourglass [143]	2019	512	3.7	Tesla P100	-	47.0%
YOLOv4 [144]	CSPDarknet-53 [145]	2020	416	65.0	Tesla V100	-	43.5%
Dave-2 System [146]	-	2016	-	30.0	NVIDIA DRIVE PX	-	-

[a] Accuracy value was obtained on KITTI dataset [37].

and mean average precision (mAP) on the Pascal VOC [35] and MS COCO [36] pascal datasets.

It can be observed from Table 7 that the models that have performed best in terms of accuracy in their baseline class are Detr [132] and Dynamic R-CNN [134] for two-stage detectors and CenterNet [142] and YOLOv4 [144] for one-stage detectors. It can also be seen from Table 7 that the model sizes of these detectors are quite high among models. Thus, it seems that heavyweight detectors have provided higher accuracies on the test datasets like the image classification models, discussed in Section III.

As for the low model size detectors, the winners are SqueezeDet [136] and MobileNetV2-SSDLite [99] having the outstandingly small size of around 8MB and 18MB, respectively, according to the literature. However, their accuracy levels are remarkably low for any real-time implementation, compared to other state-of-the-art detectors. So, lightweight CNN-based object detection solutions have shown the same tendency as classification CNNs.

It is not surprising that the evolution of the object detection CNN-based models and the CNNs for image classification have shown similar tendencies. This is because they have a close historical and architectural relationship.

V. AUTONOMOUS DRIVING SYSTEMS

The accumulation of knowledge on vehicle dynamics, the ground-breaking improvements on computer vision through the emergence of deep learning, and the advent of newly designed sensor modalities energized the R&D activities among researchers and developers of Autonomous Driving Systems (ADSs). From the first large-scaled automated driving competitions, DARPA Grand [148] and Urban [149] Challenges, to this day, numerous approaches have been proposed, and common system architectures have been established. In addition, substantial tasks in ADSs have been divided into subcategories, and explicit dominance of deep learning (DL) models has been seen in a number of subcategories [150]. Nevertheless, robust ADSs in urban environments have not been implemented yet [151].

Owing to the extraordinary advances in CNNs as a subbranch of DL, they have become a promising choice for the implementation of visual tasks in different modules of Autonomous Driving Systems, aimed at reducing human interventions in driving. This section first presents a review of the most recently published ADS related works that are focused on CNN-based image classification and object detection. Then, an outline of the current industry status is given. Next, the architecture and components of computational pipelines in the latest ADSs are discussed. Following that, application constraints are introduced, and, lastly, future directions are summarised.

A. REVIEW OF ADSs RELATED WORKS ON OBJECT CLASSIFICATION AND DETECTION

This review is targeted at the perception part of ADSs, which is one of the subsystems of self-driving cars, among others, as detailed in Section V.C. below. It is responsible for understanding scenes of the driving environment that include objects from multiple different categories. Consequently, multi-label image classification [152] is employed in ADSs rather than the traditional single-label image classification, which handles images as containing one category of an object per image. Multi-level image classification is also referred to as object classification in the rest of the paper.

The image classification and object detection algorithms in Tables 5 and 7 are specifically designed for general-purpose datasets, such as ImageNet [31], Pascal VOC [35], MS COCO [36], which can be utilised in different fields, as discussed in Section II.D.1). To enhance the object classification and detection capabilities of ADSs, a diverse set of naturalistic driving scenarios have to be addressed, e.g., all weather conditions, day/night time, pedestrians, traffic lights, cyclists, traffic density, etc., which are not properly represented in domain-general datasets. Thus, it is necessary to use domain-specific datasets (exemplified in Section II.D.1)).

In this section we overview ADSs related works on object classification and detection, which are published in the last three years, by analysing them in terms of dataset collection scenarios, sensors, and detection types.

1) SELF-DRIVING SCENARIOS

Understanding the complexity of naturalistic road scenes is vitally important when there are diverse self-driving scenarios, such as day/night time and weather conditions. In fact, handling diverse driving conditions represents the most challenging research part of ADS design. Hence, enhancements in object classification accuracy of such scenes can be seen as a foremost solution to overcoming the difficulties.

In this direction, Li *et al.* [153] proposed a deep adaptive neural network for multi-label image classification, ML-ANet, which enhances the accuracy performance in cross-domain adaptations. Aiming at an effective knowledge transfer between similar but different domains, the proposed approach exploits the technique of transfer learning from a fully labelled to a limited or unlabelled domain. In order to achieve this, the domain discrepancies are reduced by distributing feature maps of source and target domains via multiple-kernel variants of maximum mean discrepancies (MK-MMD) loss. The experimental work is primarily focused on domain changes arising from the conversion between clear and hazy weather conditions. It is shown that the proposed multi-label classifier network outperforms existing state-of-the-art methods [154]–[156] on three commonly used domain-specific datasets: KITTI [37], Cityscapes [39], and Foggy Cityscapes [157]. The ML-ANet neural network is also capable of adapting to round the clock illuminations in diverse weather conditions. In addition, it offers a reduced development cycle, as there is no need of fully labelled training data.

Three models aimed at a detection of pedestrians in hazy weather, which are based on the YOLO CNNs (Sections IV.D.1) and IV.D.4)) are proposed in [158]. In one of the models, named MNPrioriBoxes-Yolo, the detection precision is increased by employing a new weighted combination layer. The network features a high speed and low computational cost due to the use of depth-wise separable convolutions and linear bottlenecks. In addition, a modified priori boxes method [122] is employed to enhance precision and detection speed. Evaluation results based on a purpose-built hazy weather pedestrian dataset, which was created using six different augmentation techniques, have shown that the proposed model outperforms state-of-the-art methods in terms of accuracy and speed.

Fog is also an adverse weather condition that can occur in the driving environment. The object detection performance of Faster R-CNN [122] in four levels of foggy weather: clear (no fog), light, medium, heavy is analysed in [159]. Other works could be found in [160]–[162].

A large number of works have been addressing rainy and snowy weather conditions. The performance of the Faster R-CNN [122] and YOLO-v3 [141] detectors under clear and rainy conditions is analysed in [163]. The impact on performance of mitigating the effect of rain is investigated using various techniques, such as image translation [164], domain adaptation [165], and deraining (i.e., raindrop removal) [166]. The BDD100K dataset [38] is used in the experiments, as it offers image tagging for weather types. Thus, each image weather condition, such as foggy, rainy, and so on, is labelled. The evaluation results show that the mitigating techniques have a positive effect on the rainy weather detection performance of the employed CNNs.

Regarding snowy weather, Bernuth *et al.* [167] apply an adverse weather augmentation approach to reuse existing, well-organised, and labelled datasets, KITTI [37] and Cityscapes [39]. The approach produces lifelike and physically correct images to be added to existing ready-to-use datasets created with real-world images. Rendering the scenes with snowflakes for realistic images is provided by using OpenGL. Then, these generated images are evaluated in the proposed object detection algorithm to verify their efficiency. Other recently published works regarding rainy and snowy weather are [162] and [168].

Driving at night-time under low-light conditions causes a lack of details in the driver's field of vision and increases accident risks. Thus, many research works have been targeted at finding possible solutions. Li *et al.* [169] proposed an image enhancement network, LE-net based on CNN, which specifically addresses the exceedingly low-light conditions at night-time, such as rural areas without street lighting. Then, the study follows four sequential steps: (i) building a novel pipeline to generate low-light images from existing daytime images, extracted from BDD100K [38]; (ii) generation of naturalistic image pairs via the proposed pipeline for model development; (iii) training and validation of the designed network with the generated low-light images; (iv) testing of the network, LE-net, on real night surroundings with variable low-lights. It is shown that on the BDD100K dataset, the proposed network outperforms other models [170]–[172]. The EuroCity [44] dataset is used as a second evaluation metric, whereby a better performance is also observed in comparison to other studies [171]–[173].

To enhance the accuracy performance of object classification tasks at night, a dataset is created in [174] by combining two well-known datasets, MMSP [175] and BDD100K [38]. Following that, the created dataset samples are subclassified into daytime and night-time, and similar samples in the dataset are removed to prevent overfitting. The evaluation results have shown an improved night-time object classification performance. Other works addressing night-time object classification and detection are [176] and [162].

2) SENSORS

The works introduced in the section above mostly use camera-based image data, which are susceptible to level changes in the lighting conditions due to different seasons, intemperate weather, and shifting shadows. Changes in illumination could lead to failure of algorithms, affecting badly the quality of perception. Alternative sensors for perception tasks in ADSs are lidar and radar, however lidar struggles with foggy and snowy weather [177], while radar lacks sufficient resolution for perception tasks [178].

Sensor fusion is currently employed to improve precision and prevent any single point of failure [179]. Furthermore, dynamic conditions in camera-only systems [180] and low lighting conditions in thermal infrared imaging [181] are dealt with by data processing techniques that extract lighting invariant features [182] and assess feature quality [183]. Despite that, perception quality remains a central issue and prevents the ADSs from becoming prevalent in vehicles.

In addition to illumination changes, another issue is the image space for camera-based perception, as the scale of the image scene is unknown in advance. In fact, making use of scale information in dynamic tasks, such as obstacle avoidance is feasible by means of a single camera [184], although multi-view or stereo systems are more preferable in terms of robustness [185]. However, they lead to a considerable amount of computational load to an already complicated perception pipeline. As a relatively new and alternative sensing method for 3D perception, 3D lidars exist to solve the scale problem as they depend far less on illumination changes and intemperate weather. Nevertheless, their owning cost remains as a noteworthy problem as stated in Section I.A.

In view of the above, numerous recent works have been concerned with enhancing the perception quality of self-driving scenes on different collection scenarios and improving the object classification and detection performances in real-time. For instance, Gao *et al.* [186] have fused lidar and vision data in an object classification task. Point clouds of lidar data and RGB images from the KITTI [37] dataset are used in the study. They initially upsample and convert the point cloud data into depth feature maps at pixel-level, and the RGB images are also converted to depth feature maps. Following that, the integrated depth and RGB data are fed into a CNN. Thus, experiments on the KITTI [37] dataset exhibit superior object classification accuracy compared to using only depth or RGB data. Moreover, acceleration in feature learning and convergence has been achieved using lidar data.

Another fusion-based system [166], which employs a cheaper 4-beam lidar, rather than an expensive 64-beam one, and a stereo camera for 3D object detection, handles the cost problem of lidars. Due to a significant enhancement of the depth estimation technique, the proposed method has shown an improved 3D object detection performance on the KITTI dataset [37].

In [187], the performance of point clouds of lidar data on 3D object detection is addressed by simulating clear weather to foggy weather compared to other sensor types. Details about more sophisticated recently published works can be found in VoxelNet [188], SECOND [189], IPOD [190], PointPillars [191], PointRCNN [192], F-PointNet [193].

3) DETECTION TYPES

So far, the review of ADS related works has focused on the overall content of the self-driving scenes, or in other words, the big picture. On the other hand, enhancements of object classification and detection of specific features of the scene, such as pedestrian, cyclist, traffic sign and light, pavement marking, etc. are also quite important, e.g. from a safety point of view. Here we call these sub-classification and detection types.

In a recent study on pedestrian detection, Boyuan and Muqing [194] placed a novel Spatial Pyramid Pooling (SPP) network to YOLOv4 [144] detector for an improvement of the detection accuracy, where a Mish activation function [195] is used instead of Leaky ReLU. Then, the anchor parameters of YOLOv4 were optimized with a K-means clustering algorithm. As a result, an excellent pedestrian detection accuracy of 84.7% mAP was obtained with 36.6 fps in real-time, on the Caltech [43] dataset (introduced in Table 3). In [196], thermal imaging, enabling day/night time and illumination-independent data collection, was used to implement a robust pedestrian and cyclist detection using the Faster R-CNN [122]. Details of other related approaches are available in [197], [198], and [199].

Detection of traffic signalization, comprising elementarily traffic signs, traffic lights, and road surface markings in the surrounding of self-driving cars allows the ADSs to make correct decisions according to the traffic rules. There are several most recent studies regarding this research topic. Henchri and Mtibaa [200] propose a two-stage approach for traffic sign detection. In the first stage, only the shape of the signs, which are circular or triangular, are detected and classified by using HOG [109] features and support vector machines (SVM). Then, the second stage by means of a CNN attempts to classify these detected shapes into their own subclasses. Finally, the approach is tested on the GTSDB benchmark [201] with improved results.

Another significant object for ADSs in road scenes is traffic lights. An SSD [123] detector is exploited for an adaptation study to detect traffic lights and small objects in [202]. The Inception-v3 CNN (Sec. III.B.2)) is employed as the base network instead of the originally enlisted VGG CNN (Sec. III.A.3)) to increase speed and accuracy. The study adapts a prior box generation allowing smaller strides in the latter network layers, which enables detections of smaller object. Non-maximum suppression (NMS) is also adapted to prevent multiple detections for a single object. Finally, an additional block is inserted to classify the states of the traffic lights (i.e., red, amber, or green). The model showed a good performance on the DriveU [203] traffic light dataset.

Road surface markings as an important component of traffic signalization have been also studied in recent years. Ye *et al.* [204] have addressed partly distorted and worn road markings by means of a two-stage model based on YOLO-v2 [137]. The first stage is responsible for the detection of initial road markings with coordinates and class confidences of bounding boxes by using the YOLO-v2 CNN (Sec. IV.D.4)). In the second stage a novel, lightweight, and transformation-invariant classification network, RM-Net, is used for road markings detection, which is able to tackle the distortions and surface wears. An annotated road marking detection dataset for public use is developed, consisting of ~12k high-resolution images, grouped into 13 classes, which are collected under various weather conditions in day/night time. The model achieved 86.5% mAP on this dataset, outperforming other existing frameworks.

Details of other most recently performed works could be found in [205] and [206] for traffic sign detection, [207] and [208] for traffic light detection, [34] and [209] for road surface marking detection. Readers are referred to [23] for information about more works on traffic signalisation.

B. CURRENT INDUSTRY ACTIVITIES

1) TAXONOMY OF VEHICLE AUTOMATION

In 2017, the US National Highway Traffic Safety Authority [210] released guidelines for ADSs, in which six levels of automation were specified [211]. These are summarised in Table 8 and are outlined below.

- LEVEL 0: NO AUTOMATION
  All driving tasks must be completed by a human driver, which also means zero autonomy.
- LEVEL 1: DRIVER ASSISTANCE
  The vehicle is mainly controlled by a driver. Only under limited driving conditions, some driving assistance features, such as steering, acceleration/deceleration, are shared with the automated system.

TABLE 8. Automation levels.

Levels of Automation	Steering	Acceleration /Deceleration	Lane Change	Certain Conditions	All Conditions	Human Driver
Level 0	-	-	-	-	-	√
Level 1	√	√	-	√	-	√
Level 2	√	√	-	-	√	√
Level 3	√	√	√	-	√	√
Level 4	√	√	√	√	-	-
Level 5	√	√	√	-	√	-

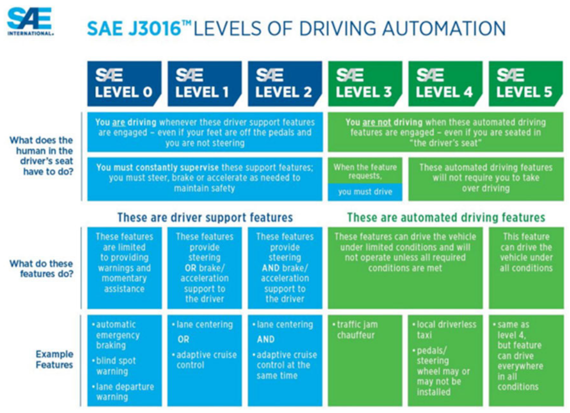
- LEVEL 2: PARTIAL AUTOMATION
  The vehicle fully controls the steering and acceleration/deceleration even under limited conditions. However, drivers must remain focused on other driving tasks and monitor vehicle surroundings at all times.
- LEVEL 3: CONDITIONAL AUTOMATION
  The human driver is not required to monitor the environment as the automated system takes on all driving tasks, including lane change as well. What the driver must still do is to be in a state of readiness to respond to requests arising from the automated system.
- LEVEL 4: HIGH AUTOMATION
  Under certain conditions, the vehicle is capable to handle all the driving tasks. A human driver might have an option to take part in the control.
- LEVEL 5: FULL AUTOMATION
  In this level of autonomy, vehicles are capable of performing all the driving tasks under all driving conditions. There is no task assigned to the human driver.

In brief, as seen from Table 8, whereas in levels 1 and 2 drivers are still responsible for a few tasks, in levels 3-5 the responsibility for all driving tasks, even lane change, is supposed to be handled automatically. Fig. 15 shows recent updates to the level definitions above by the Society of Automotive Engineers (SAE) [211].

2) CURRENT STATUS QUO IN INDUSTRY

This subsection presents the results of a literature survey on the current status and activities of industry leaders in terms of the achieved level of automation as well as used sensors and computing platforms in the vehicle, in an attempt to establish where automakers stand. A summary of the findings is presented in Table 9.

It could be seen from Table 9 that even Tesla [12], Audi [212], and Mobileye [13], [14] which are leading automotive companies, have only achieved automation levels 2 or 3, in which the driver must be highly involved in completing the driving tasks. It would appear that the production process relating to ADSs of many automakers is still under experimentation. In addition, some of them have imposed certain conditions on the use of their ADSs, i.e., driving in confined areas such as specific cities, particular highways, etc. Currently, levels 3-4 can be operated only in limited Operational Design Domains (ODDs) like highways,


SAE INTERNATIONAL

SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/ acceleration support to the driver	These features provide steering AND brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	•automatic emergency braking •blind spot warning •lane departure warning	•lane centering OR •adaptive cruise control	•lane centering AND •adaptive cruise control at the same time	•traffic jam chauffeur	•local driverless taxi •pedals/ steering wheel may or may not be installed	•same as level 4, but feature can drive everywhere in all conditions

FIGURE 15. SAE recent updates on automation levels [211].

and Audi claims to be the first company which have produced a level-3 vehicle under the condition that it is driven on a restricted highway [213].

In 2020, Volvo and Lidar-maker Luminar [214] stated that they aim to deliver genuine hands-free level-3 driving without monitoring surroundings by drivers in the real-world environment, rather than under certain conditions, by 2023. Although, as shown in Table 9, Waymo [215] have reached level 4 driving under certain conditions, there has not been yet any production of a vehicle satisfying levels 3-5 driving on any road in an urban environment. Furthermore, the Toyota Research Institute [216] have stated that there is no one in the industry that has been even close to attaining level 5. Thus, the details in Table 9 show that there are still challenges in the production of highly automated vehicles, especially at levels 3-5, which motivates the research community to actively investigate this emerging area.

As for the used sensors, as seen from Table 9, Nvidia/Audi [212], Waymo [215], and Navya [217], who have achieved automation levels 3-4 differ from the others by using lidars, which enable high precision as sensing devices send light beams to surroundings. However, the reason why the industry tends to mount cameras instead of lidars is their overwhelming cost, as mentioned in Section I.A. There is also a general tendency for using a combination of SoCs and GPUs based high-performance computing platforms, as shown in Table 9, as these can support high-speed processing of large volumes of sensor data.

TABLE 9. Current situation of autonomous driving under experimentation.

Leading Automakers	Tesla [12]	Nvidia/ Audi [212]	Waymo [215]	Mobileye [13], [14]	Navya [217]
Automation	level 2	level 3 [a]	level 4 [a]	level 2	level 4 [a]
Platform	SoCs GPUs	SoCs GPUs	SoCs GPUs	SoCs FPGAs	SoCs GPUs
Sensor	camera radar	camera lidar, radar	camera lidar, radar	camera	camera lidar, radar

[a] Under certain conditions.

C. SYSTEM ARCHITECTURES AND COMPONENTS

System architectures of ADSs defining working order can be classified into two types by their connectivity status and algorithmic design. By connectivity status, architectures could be subdivided into two classes: ego-only, i.e., a single vehicle [26], [218] or connected vehicles, i.e., multi-agent systems [219]–[221]. Depending on their algorithmic design ADSs architectures could also be subdivided into two alternative classes, which are: modular [26], [218], [46], [178], [222]–[227] or end-to-end driving systems [146], [228]–[236].

1) CONNECTIVITY IN ADSs ARCHITECTURES

The idea behind the connected systems is based on a communication network among the vehicles (agents). It is expected that this type of system would allow to advance autonomous driving in a more hierarchical way. Even though there is not any kind of operational system of this type at present, in academia, it is notably believed that this emerging technology will take a high proportion of the future ADSs [219]–[221]. Current ever-developing design approaches to connected systems are: Vehicular Ad hoc NETwork (VANETs) [219], vehicle to everything (V2X) [237],

Information-Centric Networking (ICN) [219], and Internet of Vehicles (IoV) [219].

Contrary to connected systems depending on other vehicles on the roads, the ego-only system brings a different approach, whereby all the necessary automated driving operations are managed by a self-sufficient vehicle. At present, this approach is the most preferred system type [26], [218], [46], [178], [222]–[226], as it may be that the enormous development challenges force the industry to focus on this system type for now.

2) ALGORITHMIC DESIGNS FOR ADSs ARCHITECTURES

Modular systems consist of separate linked components that form the pipeline of ADSs, including connections from the sensory inputs to the actuator outputs [150]. It is referred to as a mediated approach in [228]. Fig. 16 (a) demonstrates a state-of-the-art modular system design, which is based on recent publications [15], [16], [238]. The individual modules in Fig. 16 (a) are described in Section 3) below.

Unlike modular systems, end-to-end systems generate ego-motion information by using directly the input data provided by the sensors. Ego motion is defined as either the continuous operations on the steering wheel and driving pedals or a set of discrete actions, e.g., acceleration and turning left/right [16]. It is also referred to as a direct reception in [228]. The pipeline of an end-to-end system is represented in Fig. 16 (b). Neuroevolution [235], [236], direct supervised deep learning [228]–[232], and deep reinforcement learning [233], [234] are three different main approaches for these systems.

Table 10 summarises the pros and cons of the two algorithmic designs. The key factor of why automakers prefer

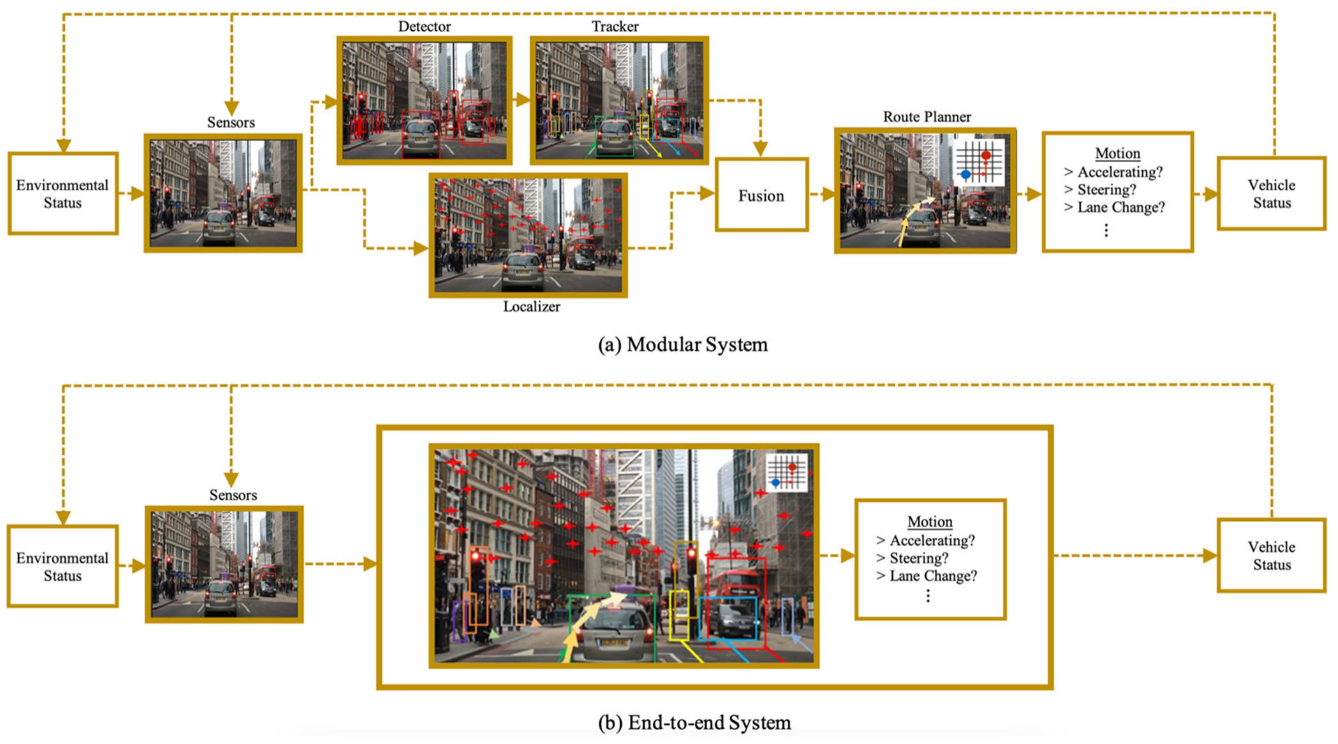


FIGURE 16. Algorithmic design types: (a) modular system and (b) end-to-end system.

modular designs compared to end-to-end design is existing literature, which enables easier production even though there are still many shortcomings, primarily error propagations. End-to-end designs still need many efforts for near-future production. Particularly, hardcoded safety measures and interpretability remain crucial topics for these designs.

TABLE 10. Comparison of design types.

Type	Pros	Cons
Modular	> Convenience in solving the problems [239] > Existing literature for modules enabling transferable info. > Feasible integration with reliable architectures [240]	> Error propagation in the pipeline [150] > Complexity [241]
End-to-end	> Offline training option > Better performance during operation	> No real-world implementation yet > Hard coded safety measures [242] > Interpretability [239]

3) MODULES

A variety of sensors are used in the onboard data processing pipelines of cutting-edge ADSs. Commonly used types of sensors are exteroceptive sensors, e.g., ultrasonic, radar, lidar, and cameras, which are responsible for perceiving the vehicle surroundings. They are employed in both modular and algorithmic designs, as seen from Fig. 16 (a) and (b). As previously stated in Section I.A., many leading companies, such as Tesla [12] and Mobileye [13], [14] have mainly focused on camera-based approaches. However, others have used lidar sensors as well, e.g., the Uber car (XC90) [225] deploys 20 cameras and 8 lidars, and VisLab's BRAiVE [222] has 10 cameras and 5 lidars. The data-flow diagram for modular systems is shown in Fig. 16 (a).

Data collected from sensors are utilised for two different purposes: (i) perception and (ii) localisation & mapping. The first requires detector and tracker modules, while the second requires a localizer module, as illustrated in Fig. 16 (a). The perception modules perform the crucial task of perceiving the vehicle environment and extracting meaningful information from the objects. They also play a significant role in a vehicle's navigation system. Perception has some core subtasks which are object detection [116], semantic segmentation [119], road and lane detection [243], [244], and object tracking [245]–[248]. Information extracted by the detector and tracker in the pipeline is combined with the localizer in the fusion module. Then, this information is used in the behaviour prediction module, also known as path planning.

The localization & mapping modules are responsible for finding the vehicle's ego-position relative to a map or a reference frame taken in the surroundings [249]. By means of localization, the vehicle finds its precise relationship with all of the objects on the map. This task is crucial for ADSs similarly to for any other mobile robotic system [250]. This is because vehicles need to estimate their ego-position with a centimetre-level precision [16]. In the industry, mainly the following five localization techniques are used: SLAM [251], Absolute positioning sensors [252], Odometry/dead reckoning [253], GPS-IMU fusion [254], prior map-based [255]–[257], and many of them are lidar-based approaches. There have been suggestions that the future of localization may be shaped by camera-based approaches as their deployment is more cost-efficient [16].

Lastly, a significant part of the ADSs pipeline is responsible for behaviour prediction and planning, which are produced in a route planner module as shown on Fig. 16 (a). Planning modules realise two sub-tasks, global planning and local planning. As the name suggests, a global planner finds a route from point of departure to point of arrival by using the road network, whereas local planning tries to perform a defined global plan with as lower as a possible error. In other words, ADSs strive to reach the final destination by finding trajectories avoiding obstacles and satisfying optimization criteria.

In addition, many modern cars are equipped with navigation systems, which plot a global route by utilising GPS or offline maps. Both, the academy community and industry have proposed a number of global planning methods, such as goal-directed path [258], separator based[259], hierarchical [260], bounded-hop [261]. In addition to these methods, some hybrid studies have been published as well [262], [263]. Recently, several methods on local planning have been proposed too, following different approaches, e.g., graph search [264], sampling-based [265], [266], curve interpolation [267], [268], numerical optimization [269]. Apart from these traditional methods, new approaches to planning have emerged based on DL and reinforcement learning [269], [270], which still have many shortcomings that need to be overcome such as interpretability [239], hard-coded safety measures [242], generalization, training data, etc.

In contrast to modular systems, end-to-end systems aim to produce a similar result in one step, as shown in Fig. 16 (b). However, these systems are still an ongoing research topic and need a lot of development efforts.

D. ADSs DESIGN CONSTRAINTS

According to official information, 94% of the total number of road accidents are due to human errors [271]. Against these grim statistics, ADSs promise to reduce traffic accidents, driving-related stress, emissions, and many others [272]. However, ADSs have been the cause of accidents as well. For instance, Google's ADS [273] hit a bus when changing lanes, as the system did not accurately estimate the speed of the bus. As another example, Tesla's Autopilot [274] failed to detect a truck and it caused a fatal accident resulting in the death of the driver. There are many other such collisions with tragic ends [275], [276]. These unfortunate events show that there are still many issues that need to be addressed during the design process of such systems.

The purpose of this section is to discuss the main criteria that should be handled to realise the object detection algorithms designed for ADSs.

Design of ADSs is a trade-off decision process taking into account a number of requirements on different parts of the system, in terms of performance, predictability of performance, power consumption, storage, thermal constraints, among others [15]. The object detection part of ADSs, for example, must satisfy real-time implementation requirements. In this section, ADSs design constraints will be evaluated based on the open literature, after which a trade-off analysis between accuracy and model size will be discussed.

Mainly, four parameters that affect directly the performance of the vehicle are considered, when designing a new CNN for object classification or object detection, namely accuracy, model size, inference speed, and efficiency. As seen from Table 5 and 7 increasing the accuracy level of CNNs for both classification and detection has been a mainstream research goal, apart of several works, such as [70], [96], [98], [101] on classification and [122], [116] on object detection. Moreover, what is another trend exemplified by Tables 5 and 7 is that the rises in the accuracy have been followed by rises in the model sizes as well. However, while the former is regarded as an added value, the latter is not a favourable result for real-time implementations.

Without a doubt, a novel CNN architecture featuring both low model size and a high accuracy level would bring many advantages to ADSs systems. Some of the benefits of a small model size are low power consumption (efficiency), low latency and high inference speed. It also facilitates over-the-air updates [96], online and distributed training [96], [277], FPGA-based implementation, use of low memory bandwidth [96] and it even leads to lower fuel consumption [15].

Some technical terms about latency and memory types of processors (CPUs) are clarified next. There are two types of CPU memory – internal (on-chip) and external (off-chip) where the former could be of two types: ROM (non-volatile) or RAM (volatile). When the processor needs to process data, it fetches it from an on-chip memory component that is embedded in the processor itself, as shown in Fig. 17. The on-chip RAM is of an SRAM type and the off-chip memory is of a DRAM type. When the processor needs to process data not available in the on-chip memory, it fetches it from an off-chip memory component that is not a part of the processor itself.

The on-chip SRAM memory allows reducing the latency (delay time), as generally, it supports a single-cycle access time. SRAM performs as a cache memory for the off-chip DRAM as well. The off-chip DRAM has a much greater latency of 1000× cycles more compared to the on-chip SRAM [278] as illustrated in Fig. 17. Due to its high cost, SRAM memory is preferred at low sizes. Although DRAMs are low-priced memories, they are quite power-hungry due to their internal design features [279].

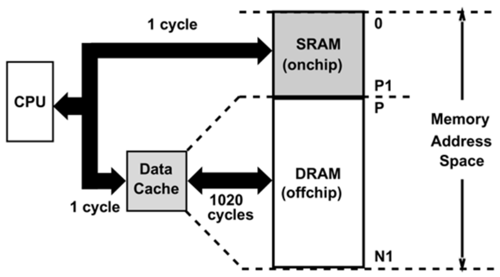


FIGURE 17. Memory hierarchy in processors [278].

1) PERFORMANCE AND PREDICTABILITY

Being candidates for preventing car accidents, ADSs need the capability of understating their surroundings and reacting to them in an acceptably fast way. Despite the promise for such a potential, the real-time performance requirements for ADSs are still largely undefined.

According to Mody [280], the reaction time of ADSs is defined by two criteria, frame rate and processing latency. While frame rate denotes how fast the data coming from real-time sensors can be fed into ADSs, the processing latency is defined by how fast the system reaction to every piece of fed data (frame) is. Following that, Lin *et al.* [15] state that ADSs should be able to react or process current traffic events within the latency of 100ms and support a frequency of at least 10 fps in object detection, as well. These metrics are based on the actual real-time performances of the human driver [271], [281]–[284].

In [15], the reaction performance of human drivers is used to systematize the ADSs real-time performance. Even though the human being is a very complicated non-linear system, and its behaviour cannot be assessed easily [285], [286], it is shown in [287], [288] that human ability for image classification outperforms DNNs indisputably. Ideally, in operation, a 100% accuracy rate is expected from object detectors in finding objects of interest. However, such an accuracy, which is on par with human abilities, is yet to be achieved by ADSs, and it is a great challenge in the field. In summary, the only agreed so far target to be reached by ADSs is up to %100 accuracy rate, and the evaluation of human behaviours on detection accuracy is yet to be finalised.

Three performance targets of ADSs were defined so far: processing latency, frame rate, and accuracy. The predictability constraint is determined by the reliability and quickness of the ADSs reactions to the real-time traffic conditions. In general, they are defined with both temporal aspects, like timing deadlines, and functional aspects, like making the correct operational decisions. Not being able to respond reliably within a specific deadline may endanger passengers and could result in fatal accidents. The performance of ADSs, therefore, needs to be exceedingly predictable to be adopted in real driving scenarios.

2) POWER

As mentioned previously in Section I.A., Intel [10] has stated that fully developed ADSs will be required to process approximately 1GB of data per second of its real-time operation. This data needs to be processed very fast so that vehicles can react to their surroundings by keeping real-time performance constraints. Consequently, that results in huge power consumption as well. In addition, as stated in [15], a power-hungry system could cause a significant downgrade of the fuel efficiency of the vehicle, i.e., up to 11.5%.

3) STORAGE

In localization tasks of ADSs, tens of TBs data are required to store the prior maps [15], [255]–[257]. For instance, a prior map of the USA is 41 TB [289], which need to be stored while driving. The reasons why such prior maps are needed are the lack of precision and accessibility in GPS technology [290], [291].

Apart from the above, there are also many other constraints, such as thermal [15], privacy [10], hardware reliability, etc.

4) TRADING-OFF ACCURACY AND SIZE

The low model size of CNN-based object detectors enables many advantages, but their accuracy is not so high. In view of this, here we evaluate the trade-off between high accuracy and low model size.

The benefits of low model size CNN designs come from the feasibility of their implementation on embedded computing platforms, as previously pointed out with ablation experiments in Section III.D. and summarized in Table 11. The high performance of GPUs in terms of processing latency is mostly due to the use of large on-chip memories, which have a very short access time. Nevertheless, despite their quite low latency, they have a high-power consumption. This is also confirmed by [15] with experimental results based on testing of a number of real-time implementations of CNN-based object detection, tracking, and extraction using CPUs, GPUs, and FPGAs. Moreover, [15] reveals that GPUs are outperformed by FPGAs in respect of energy consumption.

At the same time, processing latency of CNN FPGA implementations is restricted by the size of the on-chip memory, which often amounts to less than 10MB [96]. Examples of automotive-grade FPGA families are XA Spartan-3A (576KB) [292], XA Artix-7 XA7A12T (720KB) [293], XA Spartan-7 XA7S75 (3.2MB) [294] for XILINX and Intel Cyclone IV E EP4CE1150 (3.9MB) [295], Intel MAX 10 10M50 (1,6MB) [296], Intel Cyclone 10 10CL055 (2,3MB) [297] for Intel. Even though these FPGAs support access to external off-chip memory, performance of CNN models would be limited by the amount of the on-chip SRAM memory.

In conclusion, there are many benefits of reducing the model size of CNN-based detectors as well as making them capable of state-of-the-art accuracy rates as seen in Table 11.

Therefore, there is a need of improving low model size CNN designs by increasing their accuracy to a favourable level. Such developments would expedite the use of CNNs in the ADSs design process by allowing CNN deployment in low-power FPGA-based implementations despite the low on-chip memory resources.

TABLE 11. Advantages of accuracy and model size in CNNs.

High Accuracy	Low Model Size
> High real-time reaction time [116] > High Security [15] > High Predictability [15]	> Low power consumption (efficiency) > Low fuel consumption [15] > Low latency and inference speed > Capable of over-the-air updates [96] > Capable of online and distributed training [277] > Feasible with FPGAs having limited on-chip memory > Capable to work with low memory bandwidth [96]

E. PROMISING DIRECTIONS

Here we discuss the emerging trends in the two main areas covered above: object detection and ADSs.

1) OBJECT DETECTION

Several prominent directions could be noted in the area of object detection.

- Firstly, as seen from the object detection designs in Section IV, while the two-stage detectors are only focused on accuracy, the one-stage detectors are concerned primarily with the model size and efficiency. Thus, each of them sacrifices a design parameter in their design. From this perspective, utilizing one-stage and two-stage detectors by combining their superior sides is a big challenge enabling higher accuracy and lower latency together in the detectors.
- Another trend is troubleshooting video data streaming problems, such as video defocus, motion blur, intense target movements, etc. so that their detrimental effects on performance are removed.
- The third direction is related to the used machine learning method in CNN training. To avoid the long time and inefficiency of the training process in supervised learning, unsupervised learning methods have started to be employed as an emerging research direction.
- Another non-CNN based trend is using GAN-based detectors [298]. The necessity of a large amount of data in deep learning has several drawbacks. To overcome this problem, GAN-based detection methods are expected to become a promising and ever-increasing area.
- Finally, decreasing the model size of detectors brings a large number of advantages. However, the design process has to be managed efficiently so that the accuracy level is improved as well.

Apart from these, there are also many other trends, such as multi-task learning [299], assistance with multi-source information, e.g., social media, big data, etc.

2) ADSs

Similarly to detectors, the area of ADSs has several promising directions, as follows.

- The first is 3D object detection, which is a challenging area due to the cost of the sensors. Although lidars are quite efficient in 3D object detection, their costs at present are too high to be deployed in the ADSs. However, the approach is highly innovative.
- Connected systems promise a huge potential in improving the level of autonomy, but they require communication among road users, which is a complex and challenging problem that is yet to be addressed.
- The third one is human-machine interaction, which is an attractive area, required in designs at automation level 3 or upper levels. However, mutual understanding between cars and human requires further research work.
- Lastly, establishing ADSs requirements for real-time applications is another important area. As discussed in Section V.D., despite the fact that human behaviours are not easy to be evaluated, investigations of parallels between humans and machines have already been undertaken towards formulating commonly held ADSs regulations in real-time applications [15].

In addition to the above, other ADS trends are deep learning-based route planners [300], multi-task networks in ADSs [301], etc.

VI. CONCLUSION

This paper provides a comprehensive review of the literature on the contemporary state-of-the-art of Convolutional Neural Networks for image classification and detection as well as Autonomous Driving Systems. Layer-based details of CNNs along with parameter and floating-point operation number calculations are outlined.

Using an evolutionary approach, the latest CNN models for image classification are discussed covering the most up-to-date developments. A novel timing analysis aimed at assessing the impact of commonly used convolution types on CNN performance is presented running a reference CNN model on a desktop computer and two powerful GPUs. This extensive experimental study provides a new insight into the performance of each convolution type in terms of training time, inference time, and layer level decomposition.

Building blocks for CNN-based object detection are also discussed, such as backbone networks and baseline types, and then representative state-of-the-art designs are outlined. Experimental results from the literature are summarised for each of the reviewed models based on particular image datasets.

Most recent ADSs related works on CNN-based object classification and detection, current ADSs technologies and promising directions are outlined. In addition, a comprehensive trade-off analysis of ADSs from a human-machine perspective is presented. Thus, the paper deals with a broader study area compared to previous reviews, covering new trends and improvements that are expected in the near future, as well. In particular, it is highlighted that CNN models that feature low model sizes and achieve satisfactory accuracy have a promising future. It is our strong belief that with the necessary real-time hardware capabilities and successful resolution of design constraints, CNNs would enable a safer and more efficient self-driving cars and a rise in their popularity.

APPENDIX: DESIGN SPACE EXPLORATION

Sections III and IV above have highlighted that it is by innovating and perfecting the existing design approaches and techniques that researchers have been advancing so effectively the CNN state-of-the-art. The investigation of the literature reveals that the success of CNN models depends on the design strategies in regard to the CNN micro/macro architecture parameters as well as optimization algorithms, activation functions, compression techniques, learning types, normalization [302], regularization [303], loss functions [304], data augmentation [305], RoI feature extractors [122], Region proposal algorithms [120], etc. Therefore, in this section, we review some of the most important areas of design space exploration for classification and object detection CNNs.

A. OPTIMIZATION

Optimization algorithms have played a significant role in the training of machine learning models in recent years. In particular, iterative or coordinate descent methods, have been instrumental in increasing model performance. However, the optimization field has gradually faced more challenges with the growth in model complexity and data size, as reported in [19]. A summary of the algorithms, employed in CNN, is presented below, dividing them into three main categories: first-order, high-order, and heuristic derivative-free.

1) FIRST-ORDER ALGORITHMS

Gradient descent algorithms are the most used algorithms in first-order optimization methods. They depend on the first-order derivatives of the loss functions, represented in Jacobian matrices. In its simplest form, the weight matrice θ is defined by

$$\theta = \theta - \mu \cdot \nabla J(\theta), \tag{14}$$

where μ is the learning rate and $\nabla J(\theta)$ is the Jacobian matrix of the cost function J(θ).

Three gradient descent variants are used depending on the number of examples (m) from a dataset, which the algorithm uses to compute the gradient objective in each iteration. The first variant, Batch gradient descent, uses all m examples in every iteration and is calculated with

$$\theta = \theta - \mu \cdot \nabla_\theta \mathrm{J}(\theta; \mathrm{x}^{(i:i+m)}; \mathrm{y}^{(i:i+m)}), \tag{15}$$

where x represents the input features of the i^{th} training example and y is the real value for the i^{th} example. Its drawback is a possible slowness, as it calculates the gradients for the whole dataset.

The second one is Stochastic gradient descent (SGD) that uses only one example in each iteration. Its limitation is that SGD performs updates frequently with a high variance that produces intense fluctuations in the objective function. Its function is

$$\theta = \theta - \mu \cdot \nabla_\theta \mathrm{J}(\theta; \mathrm{x}^{(i)}; \mathrm{y}^{(i)}). \tag{16}$$

The last variation is Mini-batch gradient descent that uses $n = m/x$ examples, where x is selected according to the application, in each iteration. The method, which is defined by the expression below, enhances performance, as it reduces the variance effects and computational load.

$$\theta = \theta - \mu \cdot \nabla_\theta \mathrm{J}(\theta; \mathrm{x}^{(i:i+n)}; \mathrm{y}^{(i:i+n)}). \tag{17}$$

Among these three algorithms, the SGD method [306], [307] and its variants have been widely used in the last years, as a representative of first-order optimization methods. It is crucial to state that the SGD term can be interchangeably used when mini-batches are used as well. It is also quite significant to pay attention to the application scope and characteristics of these methods while selecting them for application in DNNs.

There is a long list of gradient descent algorithms that fall in the category of first-order optimization methods. Methods have pros and cons towards each other. Further details could be found in Nesterov Accelerated Gradient Descent (NAG) [308], AdaGrad [309], AdaDelta [310], RMSProp [311], Adaptive moment estimation (Adam) [312], Stochastic Average Gradient (SAG) [313], Stochastic Variance Reduction Gradient Since (SVRG) [314], etc.

2) HIGH-ORDER ALGORITHMS WITH HESSIAN MATRIX

Compared to the first-order methods, the high-order optimization methods [315]–[317] converge most correctly to the local or global minimum as the curvature information of these methods is more powerful. In spite of the attractive features, they face multiple challenges. Second-order algorithms present problems with the computation and storage of the inverse matrices of the Hessian matrices, which are used in the high-order optimization to hold the second or higher-order derivatives, in each iteration of the training process.

Newton's methods have been applied to overcome the problem [318], [319]. Following that the Stochastic quasi-Newton method and its improved versions [320]–[322] has been proposed, to allow the use of high-order methods on large-scale datasets. Several other methods have been proposed too, such as Conjugate Gradient [323], Newton's Method [324], Quasi-Newton Method [325], Stochastic Quasi-Newton Method [326], Hessian Free Method [318], Sub-sampled Hessian Free Method [327], Natural Gradient [328].

3) HEURISTIC DERIVATIVE-FREE ALGORITHMS
The heuristic derivative-free algorithms in [322], [329] have been proposed to address the case when the derivative information of the target functions may not exist or be hard to calculate. They are based on two root ideas. While the first one is a heuristic investigation with empirical rules [330], the second one is trying to fit the function with samples [331].

B. CNN MICRO/MACRO ARCHITECTURE
Microarchitecture parameters of CNN are related to layer-based adjustments, such as filter dimensions, pooling operation types, etc. There is a variety of those parameters among CNN architectures. For instance, while VGGNet [75] employs 3×3 filters, Inception-v1 [29] makes use of different sizes of filters, such as 1×1, 3×3, and 5×5. In addition, as CNNs tend to go deeper, manual adjustments of kernel types and input/output dimensions become impractical in individual layers. To address the problem, higher-level predesigned blocks or modules, which involve different sizes and types of kernels, bypass connections, activation functions, etc., have been proposed to lighten designers' effort. An example of modules is the inception modules, which were used in the design of the Inception-v1 CNN, also known as GoogLeNet, where many of them are combined with ad-hoc layers to construct a complete network.

Whereas the properties of individual layers and modules represent microarchitecture, CNN macro architecture concerns the system-level organization in its entirety or the end-to-end algorithm pipeline. The selection of depth, i.e., the number of layers or modules, and the choice of connection types across multiple layers or blocks can be listed as the two primary areas of the CNN macro architecture. In VGGNet, increasing the number of layers and modules gives a better accuracy performance at the system level. An example of connection types between the layers is the FCLs that are visualised in Fig. 1 for AlexNet [28]. In addition, several macro architecture strategies were proposed in ResNet-50 [17] that advanced the state-of-the-art in its publication year. Several other CNN designs [103], [116], and [122], which are introduced before, have gained success with innovative macro architecture-based strategies as well.

C. COMPRESSION
Although Deep Neural Networks (DNNs) have obtained great success in various computer vision tasks, many existing designs are computationally intensive and require a large amount of memory. This prevents their implementation on hardware with limited memory like FPGAs and uses in applications that run on strict latency constraints. For these reasons, compression of a pre-trained algorithm into a lightweight one without ceding from the model performance is a frequent technique to enable real-time deployments of the models. Four approaches exist, which are pruning, tensor decomposition, quantization, and knowledge distillation.

1) PRUNING
Pruning reduces the redundant parameters, which have no effect on the performance. Thus, weight matrices are turned into spare ones. Parameter pruning is quite robust over various settings in the layer and can support training from scratch or pre-trained models. Additionally, it helps with the overfitting problem. More information on pruning could be found in [332]–[334].

2) QUANTIZATION
Quantization is to adopt low-bit number representations instead of floating-point ones for every weight parameter. Currently, the main research tendency on quantization is model binarization. For instance, Binarized neural networks [335] quantize the weight values to 1-bit representations based on BinnaryConnect as well as changing the activation values to 1 bit. Thus, it reduces memory usage and simplifies many multiply-add operations into bitwise operations XNOR-Count. Another example is XNOR-Net [336], which achieves $32\times$ compression, providing a $52\times$ speed rise. Several other techniques [337]–[342] have been proposed as well.

3) KNOWLEDGE DISTILLATION
The idea behind the knowledge distillation approach is related to a type of migration learning, in which a complex architecture, called teacher model, is used to train a compact algorithm, called student model, by means of distilling knowledge from the teacher. The knowledge distillation approach is used in both classification and detection algorithms. However, the technique is quite sensitive to the range of applications and there is no pre-trained model option. An example of distillation is EfficientNet-L2 [80], in which it is applied quite effectively. Improved versions of the technique can be found in [343]–[351].

4) TENSOR DECOMPOSITION
The aim of tensor decomposition is to exploit channel or spatial-wise redundancies of convolution filters and to seek the lower rank approximations of them. Further details are reported in [352]–[356].

D. MACHINE LEARNING METHODS
Here we briefly touch upon the current machine learning methods, as choosing the right machine learning approach in ADSs pipelines may allow improving their performance [80], [85].

Based upon the addressed problem and data type, machine learning approaches can be broadly divided into three groups: primary approaches, hybrid approaches, and other types. Primary approaches comprise supervised [357], unsupervised [358], and reinforcement learning [359]. Hybrid approaches, the use of which has become quite prevalent in the last few years [81], could be subdivided into three groups: semi-supervised learning [81], self-supervised learning [360], and self-taught learning [361].

Numerous other types of machine learning methods have also been developed, such as active [362], online [277], transfer [363], federated [364], adversarial [365], ensemble [366], meta [367], targeted [368], multi-task [369], and multi-modal [370] methods.

ACKNOWLEDGMENT

This work used the ALICE High-Performance Computing Facility at the University of Leicester.

REFERENCES

[1] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962, doi: 10.1113/jphysiol.1962.sp006837.

[2] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: 10.1007/BF00344251.

[3] A. Carruthers and J. Carruthers, "Introduction," *Dermatologic Surg.*, vol. 39, no. 1, p. 149, Jan. 2013, doi: 10.1111/dsu.12130.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: http://ieeexplore.ieee.org/document/726791/#full-text-section.

[5] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Proc. Int. Joint Conf. Neural Netw.*, 1989, pp. 593–605.

[6] D. Steinkraus, I. Buck, and P. Y. Simard, "Using GPUs for machine learning algorithms," in *Proc. 8th Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 2, Aug. 2005, pp. 1115–1120, doi: 10.1109/ICDAR.2005.251.

[7] K. Chellapilla, S. Puri, and P. Simard. *High Performance Convolutional Neural Networks for Document Processing*. Accessed: Oct. 19, 2021. [Online]. Available: https://hal.inria.fr/inria-00112631.

[8] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[9] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, no. 1, Jan. 2007, pp. 153–160, doi: 10.7551/mitpress/7503.003.0024.

[10] (2014). Intel. *Technology and Computing Requirements for Self-Driving Cars*. Intel Corporation. Accessed: Jul. 15, 2021. [Online]. Available: http://www.intel.my/content/dam/www/public/us/en/documents/white-papers/automotive-autonomous-driving-vision-paper.pdf

[11] (2021). Velodyne LiDAR. *HDL-64E Durable Surround LiDAR Sensor*. Accessed: Jul. 12, 2021. [Online]. Available: https://velodynelidar.com/products/hdl-64e/

[12] (2021). Tesla. *Autopliot*. Tesla. Accessed: Jul. 10, 2021. [Online]. Available: https://www.tesla.com/autopilot

[13] (2021). Mobileye. *True Redundancy*. Accessed: Jul. 13, 2021. [Online]. Available: https://www.mobileye.com/true-redundancy/

[14] (2017). Mobileye. *Mobileye Press Conference CES 2017*. Accessed: Jul. 10, 2021. [Online]. Available: https://www.mobileye.com/live/ces-2017/

[15] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. S. M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," *ACM SINGPLAN Notices*, vol. 53, no. 2, pp. 751–766, 2018, doi: 10.1145/3173162.3173191.

[16] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 19, no. 2, Jun. 2016, pp. 770–778, doi: 10.1016/0141-0229(95)00188-3.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009, vol. 34, no. 4.

[19] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[20] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[21] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: Object recognition when the signal gets weaker," 2017, *arXiv:1706.06969*.

[22] S. Kuutti, R. Bowden, Y. C. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2018, doi: 10.1109/TITS.2019.2962338.

[23] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816, doi: 10.1016/J.ESWA.2020.113816.

[24] H. H. Tan and K. H. Lim, "Review of second-order optimization techniques in artificial neural networks backpropagation," in *Proc. IOP Conf. Mater. Sci. Eng.*, vol. 495, no. 1, 2019, p. 12003, doi: 10.1088/1757-899X/495/1/012003.

[25] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 3759–3768, doi: 10.1109/BigData.2016.7841045.

[26] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 163–168, doi: 10.1109/IVS.2011.5940562.

[27] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, 2020, doi: 10.1145/3377454.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012, doi: 10.1145/3065386.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9, doi: 10.1109/cvpr.2015.7298594.

[30] R. Duncan, "Individual choice behavior: A theoretical analysis," *J. Roy. Stat. Soc.*, vol. 123, no. 4, pp. 486–488, 1960, doi: 10.2307/2343282.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[32] S. Rajpal, N. Lakhyani, A. K. Singh, R. Kohli, and N. Kumar, "Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images," *Chaos, Solitons Fractals*, vol. 145, Apr. 2021, Art. no. 110749.

[33] M. O. Ramkumar, S. Sarah Catharin, V. Ramachandran, and A. Sakthikumar, "Cercospora identification in spinach leaves through resnet-50 based image processing," *J. Phys., Conf.*, vol. 1717, no. 1, Jan. 2021, Art. no. 012046.

[34] T. M. Hoang, S. H. Nam, and K. R. Park, "Enhanced detection and recognition of road markings based on adaptive region of interest and deep learning," *IEEE Access*, vol. 7, pp. 109817–109832, 2019, doi: 10.1109/ACCESS.2019.2933598.

[35] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and W. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009, doi: 10.1007/s11263-009-0275-4.

[36] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, B. Lubomir, and R. Girshick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 8693, no. 5, 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361, doi: 10.1109/CVPR.2012.6248074.

[38] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2633–2642, doi: 10.1109/CVPR42600.2020.00271.

[39] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Apr. 2016, pp. 3213–3223.

[40] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, and T. Fernandez. (2020). *A2D2: Audi Autonomous Driving Dataset*. Accessed: Oct. 6, 2021. [Online]. Available: https://www.a2d2.audi

[41] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11618–11628.

[42] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape open dataset for autonomous driving and its application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2702–2719, Oct. 2020.

[43] P. Dollár, C. Wojek, B. Schiele, and P. Perona. *Pedestrian Detection: A Benchmark*. Accessed: Oct.19, 2021. [Online]. Available: https://www.vision.caltech.edu/Image

[44] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "The EuroCity persons dataset: A novel benchmark for object detection," pp. 1–18, 2018, *arXiv:1805.07193*, doi: 10.1109/TPAMI.2019.2897684.

[45] J.-L. Déziel, P. Merriaux, F. Tremblay, D. Lessard, D. Plourde, J. Stanguennec, P. Goulet, and P. Olivier, "PixSet : An opportunity for 3D computer vision to go beyond point clouds with a full-waveform LiDAR dataset," 2021, *arXiv:2102.12010*.

[46] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km?: The Oxford RobotCar dataset," *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 1–15, 2017, doi: 10.1177/0278364916679498.

[47] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2443–2451.

[48] Udacity. *Udacity Dataset*. Accessed: Dec. 23, 2021. [Online]. Available: https://github.com/udacity/self-driving-car/tree/master/datasets

[49] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "TorontoCity: Seeing the world with a million eyes," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3028–3036.

[50] H. Yin and C. Berger, "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–8, doi: 10.1109/ITSC.2017.8317828.

[51] K. Takeda, J. H. L. Hansen, P. Boyraz, L. Malta, C. Miyajima, and H. Abut, "International large-scale vehicle corpora for research on driver behavior on the road," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1609–1623, Dec. 2011, doi: 10.1109/TITS.2011.2167680.

[52] M. Benmimoun, A. P?tz, A. Zlocki, and L. Eckstein, "Eurofot: Field operational test and impact assessment of advanced driver assistance systems: Final results," in *Proc. FISITA*, vol. 197, 2013, pp. 537–547.

[53] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and R. Kern, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[54] W. McKinney, "Data structures for statistical computing in Python," in *Proc. Python Sci. Conf.*, 2010, pp. 56–61, doi: 10.25080/Majora-92bf1922-00a.

[55] W. McKinney. (2019). *Pandas: Powerful Python Data Analysis Toolkit*. [Online]. Available: https://pandas.pydata.org/docs/pandas.pdf

[56] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, May/Jun. 2007, doi: 10.1109/MCSE.2007.55.

[57] M. Waskom, "Seaborn: Statistical data visualization," *J. Open Source Softw.*, vol. 6, no. 60, p. 3021, Apr. 2021, 2021, doi: 10.21105/joss.03021.

[58] (2015). Plotly Technologies Inc. *Collaborative Data Science*. Accessed: Jul. 15, 2021. [Online]. Available: https://plot.ly

[59] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Sympo Operating Syst. Design Implement.*, vol. 101, 1983, pp. 582–598, doi: 10.1016/0076-6879(83)01039-3.

[60] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, and Y. Bengio, "Theano: A Python framework for fast computation of mathematical expressions," Tech. Rep., May 2016.

[61] S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent, "Chainer: A deep learning framework for accelerating the research cycle," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2002–2011, doi: 10.1145/3292500.3330756.

[62] F. Chollet. (2015). *Keras*. GitHub Repos. [Online]. Available: https://github.com/fchollet/keras

[63] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[64] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[65] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678, doi: 10.1145/2647868.2654889.

[66] (2021). Paperswithcode.com. *Papers With Code: Trends*. Paperswithcode. Accessed: Jul. 12, 2021. [Online]. Available: https://paperswithcode.com/trends

[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011, doi: 10.5555/1953048.2078195.

[68] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497, doi: 10.1109/ICCV.2015.510.

[69] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[70] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[71] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," 2014, *arXiv:1412.5474*.

[72] Y. LeCun, C. Cortes, and C. Burges. (2010). *MNIST Handwritten Digit Database*. ATT Labs. [Online]. Available: https://yann.lecun.com/exdb/mnist.

[73] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[74] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 8689, 2014, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.

[75] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, vol. 75, no. 6, 2015, pp. 398–406, doi: 10.2146/ajhp170251.

[76] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze- and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Apr. 2020, doi: 10.1109/TPAMI.2019.2913372.

[77] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710, doi: 10.1109/CVPR.2018.00907.

[78] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *Computer Vision* (Lecture Notes in Computer Science), vol. 11206. Cham, Switzerland: Springer, 2018, pp. 185–201, doi: 10.1007/978-3-030-01216-8_12.

[79] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jul. 2020, doi: 10.1109/JPROC.2020.3004555.

[80] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy Student improves ImageNet classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10684–10695, doi: 10.1109/cvpr42600.2020.01070.

[81] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020, doi: 10.1007/s10994-019-05855-6.
[82] H. Henry Mao, "A survey on self-supervised pre-training for sequential transfer learning in neural networks," 2020, *arXiv:2007.00800*.
[83] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[84] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 10691–10700.
[85] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy: FixEfficientNet," 2020, *arXiv:2003.08237*.
[86] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–14.
[87] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5999–6009.
[89] I. Bello, "LambdaNetworks: Modeling long-range interactions without attention," in *Proc. ICLR*, 2021, p. 16.
[90] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, "Meta pseudo labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11557–11568.
[91] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852, doi: 10.1109/ICCV.2017.97.
[92] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.
[93] S. Arora, A. Bhaskara, R. Ge, and T. Ma, "Provable bounds for learning some deep representations," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, vol. 1, 2014, pp. 883–891.
[94] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.
[95] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 4278–4284.
[96] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5MB model size," 2016, *arXiv:1602.07360*.
[97] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.
[98] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, "Shift: A zero FLOP, zero parameter alternative to spatial convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9127–91355, doi: 10.1109/CVPR.2018.00951.
[99] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
[100] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856, doi: 10.1109/CVPR.2018.00716.
[101] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "SqueezeNext: hardware-aware neural network design," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1719–1728, doi: 10.1109/CVPRW.2018.00215.
[102] S. N. Gowda and C. Yuan, "ColorNet: Investigating the importance of color spaces for image classification," in *Computer Vision* (Lecture Notes in Computer Science), vol. 11364. Cham, Switzerland: Springer, 2018, pp. 581–596, doi: 10.1007/978-3-030-20870-7_36.
[103] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
[104] D. Gschwend, "ZynqNet: An FPGA—Accelerated embedded convolutional neural network," M.S. thesis, Dept. Inf. Technol. Elect. Eng., ETH Zürich, Zürich, Switzerland, 2020.
[105] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995, doi: 10.1109/CVPR.2017.634.
[106] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010, doi: 10.1109/TPAMI.2009.167.
[107] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013, doi: 10.1007/S11263-013-0620-5.
[108] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 606–613, doi: 10.1109/ICCV.2009.5459183.
[109] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, no. 1, Jun. 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.
[110] Y. Yu, J. Zhang, Y. Huang, S. Zheng, W. Ren, and C. Wang, "Object detection by context and boosted HOG-LBP," Nat. Lab. Pattern Recognit. Inst. Automat., Beijing, China, Tech. Rep., 2010.
[111] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Jun. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.
[112] P. Viola and M. J. Jones, "Robust real-time face detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Jul. 2001, p. 747, doi: 10.1109/ICCV.2001.937709.
[113] N. M. Ali, S. W. Jun, M. S. Karis, M. M. Ghazaly, and M. S. M. Aras, "Object classification and recognition using bag-of-words (BoW) model," in *Proc. IEEE 12th Int. Colloq. Signal Process. Appl. (CSPA)*, vol. 2016, Jul. 2016, pp. 216–220, doi: 10.1109/CSPA.2016.7515834.
[114] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY, USA: Springer, 2008.
[115] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[116] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
[117] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.
[118] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "DetNet: A backbone network for object detection," 2018, *arXiv:1804.06215*.
[119] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2017.
[120] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.
[121] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.
[122] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
[123] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision* (Lecture Notes in Computer Science), vol. 9905. Cham, Switzerland: Springer, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.
[124] Y. Xiao, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du, and X. Lan, "A review of object detection based on deep learning," *Multimedia Tools Appl.*, vol. 79, nos. 33-34, pp. 23729–23791, 2020.
[125] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2006, pp. 850–855, doi: 10.1109/ICPR.2006.479.
[126] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015, doi: 10.1109/TPAMI.2015.2389824.

[127] X. Qi, T. Wang, and J. Liu, "Comparison of support vector machine and softmax classifiers in computer vision," in *Proc. 2nd Int. Conf. Mech., Control Comput. Eng. (ICMCCE)*, Dec. 2017, pp. 151–155, doi: 10.1109/ICMCCE.2017.49.

[128] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Ann. Oper. Res.*, vol. 134, no. 1, pp. 19–67, Jan. 2005, doi: 10.1007/S10479-005-5724-Z.

[129] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *Schedae Informaticae*, vol. 25, pp. 49–59, Feb. 2017.

[130] Y. Li, K. He, J. Sun, and others, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*. Barcelona, Spain: Neural Information Processing Systems, 2016, pp. 379–387. [Online]. Available: http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf

[131] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017, doi: 10.1109/TPAMI.2016.2572683.

[132] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2020, pp. 213–229.

[133] S. Li, L. Yang, J. Huang, X.-S. Hua, and L. Zhang, "Dynamic anchor feature selection for single-shot object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6608–6617, doi: 10.1109/ICCV.2019.00671.

[134] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, "Dynamic R-CNN: Towards high quality object detection via dynamic training," 2020, *arXiv:2004.06002*.

[135] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 237–242, doi: 10.1109/IWSSIP48289.2020.9145130.

[136] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 446–454, doi: 10.1109/CVPRW.2017.60.

[137] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.

[138] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 1, Jul. 2015, pp. 448–456.

[139] K. Wagstaff, C. Cardie, S. Rogers, and S. Schr?dl, "Constrained K-means clustering with background knowledge," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 577–584.

[140] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," *Int. J. Comput. Vis.*, vol. 128, pp. 642–656, 2020, doi: 10.1007/s11263-019-01204-1.

[141] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[142] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6568–6577, doi: 10.1109/ICCV.2019.00667.

[143] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision* (Lecture Notes in Computer Science), vol. 9912. Cham, Switzerland: Springer, 2016, pp. 483–499, doi: 10.1007/978-3-319-46484-8_29.

[144] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[145] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1571–1580, doi: 10.1109/CVPRW50498.2020.00203.

[146] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Müller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[147] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768, doi: 10.1109/CVPR.2018.00913.

[148] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, 1st ed. Berlin, Germany: Springer, 2007.

[149] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, 1st ed. Berlin, Germany: Springer, 2009.

[150] R. McAllister, Y. Gal, A. Kendall, M. van der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 4745–4753, doi: 10.24963/ijcai.2017/661.

[151] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," *Computer Vision* (Lecture Notes in Computer Science), vol. 11211. Cham, Switzerland: Springer, Mar. 2018, pp. 449–468.

[152] L. Chen, W. Zhan, W. Tian, Y. He, and Q. Zou, "Deep integration: A multi-label architecture for road scene recognition," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4883–4898, May 2019, doi: 10.1109/TIP.2019.2913079.

[153] G. Li, Z. Ji, Y. Chang, S. Li, X. Qu, and D. Cao, "ML-ANet: A transfer learning approach using adaptation network for multi-label image classification in autonomous driving," *Chin. J. Mech. Eng.*, vol. 34, no. 1, pp. 1–11, Aug. 2021, doi: 10.1186/S10033-021-00598-9.

[154] L. Song, J. Liu, B. Qian, M. Sun, K. Yang, M. Sun, and S. Abbas, "A deep multi-modal CNN for multi-instance multi-label image classification," *IEEE Trans. Image Process.*, vol. 27, no. 12, pp. 6025–6038, Aug. 2018, doi: 10.1109/TIP.2018.2864920.

[155] J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen, "Balanced distribution adaptation for transfer learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 1129–1134.

[156] G. Li, Y. Wang, F. Zhu, X. Sui, N. Wang, X. Qu, and P. Green, "Drivers' visual scanning behavior at signalized and unsignalized intersections: A naturalistic driving study in China," *J. Saf. Res.*, vol. 71, pp. 219–229, Dec. 2019, doi: 10.1016/j.jsr.2019.09.012.

[157] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 973–992, Sep. 2018, doi: 10.1007/s11263-018-1072-8.

[158] G. Li, Y. Yang, and X. Qu, "Deep learning approaches on pedestrian detection in hazy weather," *IEEE Trans. Ind. Electron.*, vol. 67, no. 10, pp. 8889–8899, Oct. 2020, doi: 10.1109/TIE.2019.2945295.

[159] Z. Liu, Y. He, C. Wang, and R. Song, "Analysis of the influence of foggy weather environment on the detection effect of machine vision obstacles," *Sensors*, vol. 20, no. 2, p. 349, Jan. 2020, doi: 10.3390/S20020349.

[160] Y. He and Z. Liu, "A feature fusion method to improve the driving obstacle detection under foggy weather," *IEEE Trans. Transport. Electrific.*, vol. 7, no. 4, pp. 2505–2515, Dec. 2021, doi: 10.1109/TTE.2021.3080690.

[161] S.-C. Huang, T.-H. Le, and D.-W. Jaw, "DSNet: Joint semantic learning for object detection in inclement weather conditions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2623–2633, Aug. 2021, doi: 10.1109/TPAMI.2020.2977911.

[162] M. Haris and A. Glowacz, "Road object detection: A comparative study of deep learning-based algorithms," *Electronics*, vol. 10, no. 16, p. 1932, Aug. 2021, doi: 10.3390/electronics10161932.

[163] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles: A review of state-of-the-art and emerging techniques," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 53–67, Jan. 2021, doi: 10.1109/msp.2020.2984801.

[164] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 701–709.

[165] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3339–3348.

[166] N. A. M. Mai, P. Duthon, L. Khoudour, A. Crouzil, and S. A. Velastin, "Sparse LiDAR and stereo fusion (SLS-Fusion) for depth estimationand 3D object detection," 2021, *arXiv:2103.03977*.

[167] A. V. Bernuth, G. Volk, and O. Bringmann, "Simulating photo-realistic snow and fog on existing images for enhanced CNN training and evaluation," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 41–46, doi: 10.1109/ITSC.2019.8917367.

[168] X. Z. Chen, C. M. Chang, C. W. Yu, and Y. L. Chen, "A real-time vehicle detection system under various bad weather conditions based on a deep learning model without retraining," *Sensors*, vol. 20, no. 20, pp. 1–22, 2020, doi: 10.3390/s20205731.

[169] G. Li, Y. Yang, X. Qu, D. Cao, and K. Li, "A deep learning based image enhancement approach for autonomous driving at night," *Knowl.-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106617, doi: 10.1016/j.knosys.2020.106617.

[170] Q. Chen, J. Xu, and V. Koltun, "Fast image processing with fully-convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2516–2525.

[171] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention* (Lecture Notes in Computer Science). vol. 9351. Cham, Switzerland: Springer, May 2015, pp. 234–241.

[172] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Computer Vision*. Cham, Switzerland: Springer, 2018, pp. 833–851.

[173] D. J. Jobson, Z.-U. Rahman, and G. A. Woodell, "Properties and performance of a center/surround Retinex," *IEEE Trans. Image Process.*, vol. 6, no. 3, pp. 451–462, Mar. 1997, doi: 10.1109/83.557356.

[174] C.-E. Wu, Y.-M. Chan, C.-H. Chen, W.-C. Chen, and C.-S. Chen, "IMMVP: An efficient daytime and nighttime on-road object detector," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–5.

[175] J.-I. Guo, C.-C. Tsai, Y.-H. Yang, H.-W. Lin, B.-X. Wu, T. T. Kuo, and L.-J. Wang, "Summary embedded deep learning object detection model competition," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–5, doi: 10.1109/MMSP.2019.8901733.

[176] A. Bhandari, A. Kafle, P. Dhakal, P. R. Joshi, and D. B. Kshatri, "Image enhancement and object recognition for night vision surveillance," 2020, *arXiv:2006.05787*.

[177] R. H. Rasshofer and K. Gresser, "Automotive radar and LiDAR systems for next generation driver assistance functions," *Adv. Radio Sci.*, vol. 3, pp. 205–209, May 2005.

[178] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2013, pp. 763–770, doi: 10.1109/IVS.2013.6629559.

[179] P. Radecki, M. Campbell, and K. Matzen, "All weather perception: Joint data association, tracking, and classification for autonomous ground vehicles," 2016, *arXiv:1605.02196*.

[180] N. Carlevaris-Bianco and R. M. Eustice, "Learning visual feature descriptors for dynamic lighting conditions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2769–2776.

[181] P. Hurney, P. Waldron, F. Morgan, E. Jones, and M. Glavin, "Review of pedestrian detection techniques in automotive far-infrared video," *IET Intell. Transp. Syst.*, vol. 9, no. 8, pp. 824–832, 2015.

[182] V. Peretroukhin, W. Vega-Brown, N. Roy, and J. Kelly, "PROBE-GK: Predictive robust estimation using generalized kernels," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 817–824.

[183] W. Maddern, A. D. Stewart, C. Mcmanus, B. Upcroft, W. Churchill, and P. Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," Tech. Rep., 2014.

[184] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6850–6859.

[185] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2361–2379, Oct. 2020.

[186] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4224–4230, Sep. 2018, doi: 10.1109/TII.2018.2822828.

[187] M. Hahner, C. Sakaridis, D. Dai, L. Van Gool, L. V. Be, and E. Zürich, "Fog simulation on real LiDAR point clouds for 3D object detection in adverse weather," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 15283–15292.

[188] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[189] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[190] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018, *arXiv:1812.05276*.

[191] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12689–12697.

[192] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[193] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[194] W. Boyuan and W. Muqing, "Study on pedestrian detection based on an improved YOLOv4 algorithm," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2020, pp. 1198–1202, doi: 10.1109/ICCC51575.2020.9344983.

[195] D. Misra, "Mish: A self regularized non-monotonic activation function," 2019, *arXiv:1908.08681*.

[196] N. Annapareddy, E. Sahin, S. Abraham, M. M. Islam, M. DePiro, and T. Iqbal, "A robust pedestrian and cyclist detection method using thermal images," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Apr. 2021, pp. 1–6, doi: 10.1109/SIEDS52267.2021.9483730.

[197] Z. Ahmed, R. Iniyavan, and M. M. P., "Enhanced vulnerable pedestrian detection using deep learning," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 971–974, doi: 10.1109/ICCSP.2019.8697978.

[198] T. T. Feng and H. Y. Ge, "Pedestrian detection based on attention mechanism and feature enhancement with SSD," in *Proc. 5th Int. Conf. Commun., Image Signal Process. (CCISP)*, Nov. 2020, pp. 145–148, doi: 10.1109/CCISP51026.2020.9273507.

[199] M. García-Venegas, D. A. Mercado-Ravell, L. A. Pinedo-Sánchez, and C. A. Carballo-Monsivais, "On the safety of vulnerable road users by cyclist detection and tracking," *Mach. Vis. Appl.*, vol. 32, no. 5, p. 109, Sep. 2021, doi: 10.1007/s00138-021-01231-4.

[200] A. Hechri and A. Mtibaa, "Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks," *IET Image Process.*, vol. 14, no. 5, pp. 939–946, Apr. 2020, doi: 10.1049/IET-IPR.2019.0634.

[201] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, "An automatic traffic sign detection and recognition system based on colour segmentation, shape matching, and SVM," *Math. Probl. Eng.*, vol. 2015, Nov. 2015, Art. no. 250461, doi: 10.1155/2015/250461.

[202] J. Müller and K. Dietmayer, "Detecting traffic lights by single shot detection," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 266–273, doi: 10.1109/ITSC.2018.8569683.

[203] A. Fregin, J. Müller, U. Krebel, and K. Dietmayer, "The DriveU traffic light dataset: Introduction and comparison with existing datasets," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 3376–3383, doi: 10.1109/ICRA.2018.8460737.

[204] X.-Y. Ye, D.-S. Hong, H.-H. Chen, P.-Y. Hsiao, and L.-C. Fu, "A two-stage real-time YOLOv2-based road marking detector with lightweight spatial transformation-invariant classification," *Image Vis. Comput.*, vol. 102, Oct. 2020, Art. no. 103978, doi: 10.1016/J.IMAVIS.2020.103978.

[205] Y. Wu, Z. Li, Y. Chen, K. Nai, and J. Yuan, "Real-time traffic sign detection and classification towards real traffic scene," *Multimedia Tools Appl.*, vol. 79, pp. 18201–18219, Mar. 2020, doi: 10.1007/S11042-020-08722-Y.

[206] Y.-C. Chiu, H.-Y. Lin, and W.-L. Tai, "Implementation and evaluation of CNN based traffic sign detection with different resolutions," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Dec. 2019, pp. 1–2, doi: 10.1109/ISPACS48206.2019.8986319.

[207] M. Weber, M. Huber, and J. M. Zollner, "HDTLR: A CNN based hierarchical detector for traffic lights," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 255–260, doi: 10.1109/ITSC.2018.8569794.

[208] Z. Ouyang, J. Niu, Y. Liu, and M. Guizani, "Deep CNN-based real-time traffic light detector for self-driving vehicles," *IEEE Trans. Mobile Comput.*, vol. 19, no. 2, pp. 300–313, Feb. 2020, doi: 10.1109/TMC.2019.2892451.

[209] C. Gu, X. Wu, H. Ma, and L. Yang, "An end-to-end practical system for road marking detection," in *Image and Graphics*. Cham, Switzerland: Springer, 2019, pp. 85–93.

[210] (Sep. 2016). NHTSA. *Federal Automated Vehicles Policy: Accelerating the Next Revolution in Roadway Safety*. U.S. Dep. Transp. [Online]. Available: https://www.transportation.gov/sites/dot.gov/files/docs/AV policy guidance PDF.pdf

[211] J. Shuttleworth. (2019). *SAE J3016 Automated-Driving Graphic*. SAE International. Accessed: Jul. 29, 2021. [Online]. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

[212] (2021). NVIDIA. *Audi and NVIDIA | NVIDIA*. Accessed: Jul. 27, 2021. [Online]. Available: https://www.nvidia.com/en-gb/self-driving-cars/partners/audi/
[213] P. E. Ross. (Jul. 11, 2017). *The Audi A8: The World's First Production Car to Achieve Level 3 Autonomy—IEEE Spectrum*. Accessed: Jul. 27, 2021. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/the-audi-a8-the-worlds-first-production-car-to-achieve-level-3-autonomy
[214] L. Ulrich. (May 7, 2020). *Volvo and LiDAR-Maker Luminar to Deliver Hands-free Driving by 2022—IEEE Spectrum*. Accessed: Jul. 27, 2021. [Online]. Available: https://spectrum.ieee.org/cars-that-think/sensors/automotive-sensors/volvo-and-lidarmaker-luminar-to-deliver-handsfree-driving-by-2022
[215] (2021). *Waymo*. Accessed: Jul. 27, 2021. [Online]. Available: https://waymo.com/
[216] E. Ackerman. (Jan. 23, 2017). *Toyota's Gill Pratt on Self-Driving Cars and the Reality of Full Autonomy—IEEE Spectrum*. Accessed: Jul. 27, 2021. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/toyota-gill-pratt-on-the-reality-of-full-autonomy
[217] (2021). Navya. *Navya Reaches a New Milestone in Autonomous Mobility With the First Fully Autonomous Level 4 Operation on a Restricted Site—NAVYA*. Accessed: Jul. 27, 2021. [Online]. Available: https://navya.tech/en/navya-reaches-a-new-milestone-in-autonomous-mobility-with-the-first-fully-autonomous-level-4-operation-on-a-closed-site/
[218] C. Urmson, M. Gerla, G. Pau, U. Lee, and J.-H. Lim, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 33, no. 1, pp. 1–17, 2014, doi: 10.1002/rob.
[219] E.-K. Lee, M. Gerla, G. Pau, U. Lee, and J.-H. Lim, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular fogs," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 9, Sep. 2016, Art. no. 155014771666550, doi: 10.1177/1550147716665500.
[220] M. Gerla, E. K. Lee, G. Pau, and U. Lee, "Internet of Vehicles: From intelligent grid to autonomous cars and vehicular clouds," *Arch. Phys. Med. Rehabil.*, vol. 46, pp. 198–199, Sep. 1965.
[221] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: A survey and future perspectives," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 98–104, Feb. 2016, 2016, doi: 10.1109/MCOM.2016.7402268.
[222] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, "Extensive tests of autonomous driving technologies," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1403–1415, May 2013, doi: 10.1109/TITS.2013.2262331.
[223] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya, "Autonomous driving based on accurate localization using multilayer LiDAR and dead reckoning," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317797.
[224] E. Guizzo. (2011). *How Google's Self-Driving Car Works*. IEEE Spectrum. Accessed: May 10, 2021. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works
[225] H. Somerville, P. Lienert, and A. Sage. *Uber's Use of Fewer Safety Sensors Prompts Questions After Arizona Crash*. Reuters. Accessed: May 10, 2021. [Online]. Available: https://www.reuters.com/article/us-uber-selfdriving-sensors-insight/ubers-use-of-fewer-safety-sensors-prompts-questions-after-arizona-crash-idUSKBN1H337Q
[226] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, and E. Kaus, "Making bertha drive—An autonomous journey on a historic route," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, Apr. 2014, doi: 10.1109/MITS.2014.2306552.
[227] GitHub. *GitHub—ApolloAuto/Apollo: An Open Autonomous Driving Platform*. Accessed: Jul. 27, 2021. [Online]. Available: https://github.com/ApolloAuto/apollo
[228] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2722–2730, doi: 10.1109/ICCV.2015.312.
[229] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network (technical report CMU-CS-89-107)," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313. [Online]. Available: http://dl.acm.org/citation.cfm?id=89851.89891.
[230] Y. LeCun, U. Müller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 739–746.
[231] L. Chen, Q. Wang, X. Lu, D. Cao, and F.-Y. Wang, "Learning driving models from parallel end-to-end driving data set," *Proc. IEEE*, vol. 108, no. 2, pp. 262–273, Feb. 2020, doi: 10.1109/JPROC.2019.2952735.
[232] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3530–3538, doi: 10.1109/CVPR.2017.376.
[233] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Perez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–18, 2021, doi: 10.1109/TITS.2021.3054625.
[234] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8248–8254, doi: 10.1109/ICRA.2019.8793742.
[235] S. Baluja, "Evolution of an artificial neural network based autonomous land vehicle controller," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 26, no. 3, pp. 450–463, Jun. 1996, doi: 10.1109/3477.499795.
[236] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "Evolving large-scale neural networks for vision-based reinforcement learning," in *Proc. 15th Annu. Conf. Genetic Evol. Comput. Conf. - GECCO*, 2013, pp. 1061–1068, doi: 10.1145/2463372.2463509.
[237] S. Behere and M. Törngren, "A functional architecture for autonomous driving," in *Proc. 1st Int. Workshop Automot. Softw. Archit.*, May 2015, pp. 3–10, doi: 10.1145/2752489.2752491.
[238] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, Nov./Dec. 2015, doi: 10.1109/MM.2015.133.
[239] L. Chi and Y. Mu, "Learning end-to-end autonomous steering model from spatial and temporal visual cues," in *Proc. Workshop Vis. Anal. Smart Connected Communities*, Oct. 2017, pp. 9–16, doi: 10.1145/3132734.3132737.
[240] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2012, pp. 383–388, doi: 10.1109/IVS.2012.6232153.
[241] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of Deep-neural-network-driven autonomous cars," 2017, *arXiv:1708.08559*.
[242] M. Wood, P. Robbel, M. Maass, R. D. Tebbens, M. Meijs, M. Harb, J. Reach, and K. Robinson. (2019). *Safety First For Automated Driving*. [Online]. Available: https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf
[243] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: A survey," *Mach. Vis. Appl.*, vol. 25, no. 3, pp. 727–745, 2014, 2014, doi: 10.1007/s00138-011-0404-2.
[244] C. Fernández, D. Fernández-Llorca, and M. A. Sotelo, "A hybrid vision-map method for urban road detection," *J. Adv. Transp.*, vol. 2017, pp. 1–21, 2017, doi: 10.1155/2017/7090549.
[245] M.-P. Dubuisson and A. K. Jain, "A modified Hausdorff distance for object matching," in *Proc. 12th Int. Conf. Pattern Recognit.*, Dec. 2002, pp. 566–568, doi: 10.1109/ICPR.1994.576361.
[246] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon, "Fast multiple objects detection and tracking fusing color camera and 3D LIDAR for intelligent vehicles," in *Proc. 13th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, Aug. 2016, pp. 234–239, doi: 10.1109/URAI.2016.7625744.
[247] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 154–165, Mar. 2012, doi: 10.1109/TITS.2011.2165705.
[248] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha—A local, continuous method," in *Proc. IEEE Intell. Vehicles Symp. Proc.*, Jun. 2014, pp. 450–457, doi: 10.1109/IVS.2014.6856581.
[249] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018, doi: 10.1109/JIOT.2018.2812300.

[250] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017, 2017, doi: 10.1109/TIV.2017.2749181.
[251] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006, 2006, doi: 10.1109/MRA.2006.1678144.
[252] M. A. Al Khedher, "Hybrid GPS-GSM localization of automobile tracking system," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 6, pp. 75–85, Dec. 2011, doi: 10.5121/ijcsit.2011.3606.
[253] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Proc. Int. Conf. Robot. Automat.*, vol. 4, Apr. 1997, pp. 2783–2788, doi: 10.1109/robot.1997.606708.
[254] C. P. Urmson, N. A. J. Graham, M. S. Pratchett, G. P. Jones, and N. V. C. Polunin, "High speed navigation of unrehearsed terrain: Red team technology for grand challenge," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-04-37, 2004, doi: 10.1111/j.1365-2486.2006.01252.x.
[255] A. Hata and D. Wolf, "Road marking detection using LIDAR reflective intensity data and its application to vehicle localization," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 584–589, doi: 10.1109/ITSC.2014.6957753.
[256] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2040–2047, doi: 10.1109/ICRA.2018.8460519.
[257] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 4372–4378, doi: 10.1109/ROBOT.2010.5509700.
[258] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.
[259] D. Van Vliet, "Improved shortest path algorithms for transport networks," *Transp. Res.*, vol. 12, no. 1, pp. 7–20, 1978, doi: 10.1016/0041-1647(78)90102-8.
[260] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," *Transp. Sci.*, vol. 46, no. 3, pp. 388–404, 2012. 10.1287//TRSC.1110.0401.
[261] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels," *SIAM J. Comput.*, vol. 32, no. 5, pp. 1338–1355, Jan. 2003, doi: 10.1137/S0097539702403098.
[262] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, "Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm," *ACM J. Experim. Algorithmics*, vol. 15, pp. 1–26, Mar. 2010, doi: 10.1145/1671970.1671976.
[263] A. Uçar, Y. Demir, and C. Güzeliç, "Object recognition and detection with deep learning for autonomous driving applications," *Simulation*, vol. 93, no. 9, pp. 759–769, 2017, doi: 10.1177/0037549717709932.
[264] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *Proc. 8th Int. Symp. Artif. Intell., Robot. Automat. Space (iSAIRAS)*, no. 603, 2005, pp. 249–255.
[265] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991, doi: 10.1177/027836499101000604.
[266] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, Jun. 1999, pp. 473–479, doi: 10.1109/ROBOT.1999.770022.
[267] H. Fuji, J. Xiang, Y. Tazaki, B. Levedahl, and T. Suzuki, "Trajectory planning for automated parking using multi-resolution state roadmap considering non-holonomic constraints," *IEEE Intell. Veh. Symp. Proc.*, vol. 407, Jun. 2014, pp. 407–413, doi: 10.1109/IVS.2014.6856433.
[268] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 510–515, doi: 10.1109/IVS.2014.6856526.
[269] J. Ren, K. A. McIsaac, and R. V. Patel, "Modified Newton's method applied to potential field-based navigation for mobile robots," *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 384–391, Apr. 2006, doi: 10.1109/TRO.2006.870668.
[270] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "LIDAR-based driving path generation using fully convolutional neural networks," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317618.
[271] D. V. McGehee, E. N. Mazzae, and G. H. S. Baldwin, "Driver reaction time in crash avoidance research: Validation of a driving simulator study on a test track," in *Proc. Hum. Factors Ergonom. Soc. Annu. Meeting*, vol. 44, no. 20, 2000, pp. 3–323, doi: 10.1177/154193120004402026.
[272] T. J. Crayton and B. M. Meier, "Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy," *J. Transp. Health*, vol. 6, pp. 245–252, Sep. 2017, doi: 10.1016/j.jth.2017.04.004.
[273] A. Tilley. (2016). *Google's Self-Driving Car Caused its First Accident*. Forbes. Accessed: May 12, 2021. [Online]. Available: https://www.forbes.com/sites/aarontilley/2016/02/29/googles-self-driving-car-caused-its-first-accident/?sh=3b9b1b2f538d
[274] E. Fernandez. (2018). *Who is Responsible in a Crash With a Self-Driving Car?*. Forbes. Accessed: May 12, 2021. [Online]. Available: https://www.forbes.com/sites/fernandezelizabeth/2020/02/06/who-is-responsible-in-a-crash-with-a-self-driving-car/?sh=7b5dbd974b2b
[275] D. Lavrinc. (2014). *This is How Bad Self-Driving Cars Suck in Rain*. Jalopnik. Accessed: May 9, 2021. [Online]. Available: https://jalopnik.com/this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433
[276] (2020). *Uber's Self-Driving Operator Charged Over Fatal Crash*. BBC. Accessed: Jul. 29, 2021. [Online]. Available: https://www.bbc.co.U.K./news/technology-54175359
[277] V. Losing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, Jan. 2018, doi: 10.1016/j.neucom.2017.06.084.
[278] P. R. Panda, N. D. Dutt, and A. Nicolau, "On-chip vs. off-chip memory: The data partitioning problem in embedded processor-based systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 5, no. 3, pp. 682–704, Jul. 2000, doi: 10.1145/348019.348570.
[279] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254, doi: 10.1109/ISCA.2016.30.
[280] M. Mody. (2016). *ADAS Front Camera: Demystifying Resolution and Frame-Rate*. EETimes. Accessed: Mar. 10, 2021. [Online]. Available: https://www.eetimes.com/adas-front-camera-demystifying-resolution-and-frame-rate/#
[281] G. Johansson and K. Rumar, "Drivers' brake reaction times," *Hum. Factors*, vol. 13, no. 1, pp. 23–27, 1971, doi: 10.1177/001872087101300104.
[282] A. Newell and S. K. Card, "The prospects for psychological science in human-computer interaction," *Hum. Computer Interact.*, vol. 1, no. 3, pp. 209–242, Sep. 1985, doi: 10.1207/s15327051hci0103_1.
[283] S. J. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, no. 6582, pp. 520–522, 1996, doi: 10.1038/381520a0.
[284] (2017). Mobileye. *Mobileye C2-270 Essentials*. [Online]. Available: https://manualzz.com/doc/23526946/mobileye-c2-270-essentials?_cf_chl_jschl_tk_=pmd_b91f7eab6b66ebe7b1bf5a63e43ccefe471565b0-1627407908-0-gqNtZGzNAfijcnBszQii
[285] R. Rajalingham, E. B. Issa, P. Bashivan, K. Kar, K. Schmidt, and J. J. DiCarlo, "Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and State-of-the-Art deep artificial neural networks," *J. Neurosci.*, vol. 38, no. 33, pp. 7255–7269, Aug. 2018, doi: 10.1523/JNEUROSCI.0388-18.2018.
[286] R. M. Cichy, A. Khosla, D. Pantazis, A. Torralba, and A. Oliva, "Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence," *Sci. Rep.*, vol. 6, no. 1, pp. 1–13, Sep. 2016, doi: 10.1038/srep27755.
[287] L. E. van Dyck and W. R. Gruber, "Seeing eye-to-eye? A comparison of object recognition performance in humans and deep convolutional neural networks under image manipulation," 2020, *arXiv:2007.06294*.
[288] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–7, doi: 10.1109/ICCCN.2017.8038465.
[289] (2015). U. S. Department of Trasnportation. *Highway Statistics 2015*. [Online]. Available: https://www.fhwa.dot.gov/policyinformation/statistics/2015/
[290] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," *Robot. Sci. Syst.*, vol. 3, pp. 121–128, Jun. 2008, doi: 10.15607/rss.2007.iii.016.
[291] C. Berger and B. Rumpe, "Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system," in *Proc. 10th Workshop Automot. Softw. Eng. (ASE)*, vol. 208, 2012, pp. 789–798.

[292] *XA Spartan-3A Automotive FPGA Family Data Sheet*, vol. 681, Xilinx, San Jose, CA, USA, 2021, pp. 1–57.

[293] (2017). Xilinx. *XA Artix-7 FPGAs Data Sheet: Overview*. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds197-xa-artix7-overview.pdf

[294] (2017). Xilinx. *XA Spartan-7 Automotive FPGA Data Sheet: Overview*. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds171-xa-spartan7-overview.pdf.

[295] *Cyclone IV FPGAs Features*. Intel. Accessed: Jul. 29, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/iv/features.html

[296] *Intel Max 10 FPGA*. Intel. Accessed: Jul. 29, 2021. [Online]. Available: https://www.intel.co.U.K./content/www/U.K./en/products/details/fpga/max/10.html

[297] *Intel Cyclone 10 LP FPGA Devices*. Intel. Accessed: Jul. 29, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/10/lp.html

[298] X. Sun, D. Li, Z. Li, H. Song, H. Jiang, Y. Chen, G. Miao, and Z. Zhang, "High spectral response of self-driven GaN-based detectors by controlling the contact barrier height," *Sci. Rep.*, vol. 5, no. 1, p. 16819, Dec. 2015, doi: 10.1038/srep16819.

[299] J. Xu, L. Zhao, S. Zhang, C. Gong, and J. Yang, "Multi-task learning for object keypoints detection and classification," *Pattern Recognit. Lett.*, vol. 130, pp. 182–188, Feb. 2020, doi: 10.1016/j.patrec.2018.08.013.

[300] Y. Geng, E. Liu, R. Wang, and Y. Liu, "Deep reinforcement learning based dynamic route planning for minimizing travel time," 2020, *arXiv:2011.01771*.

[301] I. Leang, G. Sistu, F. Burger, A. Bursuc, and S. Yogamani, "Dynamic task weighting methods for multi-task networks in autonomous driving systems," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–8.

[302] J. Sun, X. Cao, H. Liang, W. Huang, Z. Chen, and Z. Li, "New interpretations of normalization methods in deep learning," in *Proc. 34th AAAI Conf. Artif. Intell.*, Jun. 2020, pp. 5875–5882.

[303] R. Moradi, R. Berangi, and B. Minaei, "A survey of regularization strategies for deep models," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 3947–3986, Aug. 2020, doi: 10.1007/s10462-019-09784-7.

[304] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Ann. Data Sci.*, vol. 20, pp. 1–26, Apr. 2020, doi: 10.1007/s40745-020-00253-5.

[305] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu, "How does data augmentation affect privacy in machine learning?" 2020, *arXiv:2007.10567*.

[306] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, 1951, doi: 10.1214/aoms/1177729586.

[307] P. Jain, P. Netrapalli, S. M. Kakade, R. Kidambi, and A. Sidford, "Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification," *J. Mach. Learn. Res.*, vol. 18, pp. 1–42, Jan. 2018.

[308] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," Tech. Rep., 1983.

[309] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for (parallel) stochastic optimization," in *Proc. IEEE 51st IEEE Conf. Decis. Control (CDC)*, Dec. 2012, pp. 5442–5444, doi: 10.1109/CDC.2012.6426698.

[310] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*.

[311] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude," *COURSERA Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[312] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," in *Proc. 32nd Int. Conf. Mach. Learn. Res.*, vol. 37, Jul. 2015, pp. 1462–1471.

[313] N. L. Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, 2012, pp. 2663–2671.

[314] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1, no. 3, 2013, pp. 1–9.

[315] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comput.*, vol. 24, no. 111, pp. 647–656, 1970.

[316] L. Devroye, J. King, and C. Mcdiarmid, "Random hyperlane search trees," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2411–2425, 2009.

[317] J. Pajarinen, H. L. Thai, R. Akrour, J. Peters, and G. Neumann, "Compatible natural gradient policy search," *Mach. Learn.*, vol. 108, nos. 8–9, pp. 1443–1466, Sep. 2019, doi: 10.1007/s10994-019-05807-0.

[318] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 735–742.

[319] J. E. Dennis and J. J. Moree, "Quasi-Newton methods, motivation and theory," *Soc. Ind. Appl. Math.*, vol. 19, no. 1, pp. 46–89, 2010. [Online]. Available: https://www.jstor.org/stable/2029325.

[320] F. Roosta-Khorasani and M. W. Mahoney, "Sub-sampled Newton methods II: Local convergence rates," 2016, *arXiv:1601.04738*.

[321] P. Xu, J. Yang, F. Roosta-Khorasani, C. Ré, and M. W. Mahoney, "Sub-sampled Newton methods with non-uniform sampling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3008–3016.

[322] R. Bollapragada, R. H. Byrd, and J. Nocedal, "Exact and inexact subsampled Newton methods for optimization," *IMA J. Numer. Anal.*, vol. 39, no. 2, pp. 545–548, 2019, doi: 10.1093/imanum/dry009.

[323] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards*, vol. 49, no. 6, p. 409, Dec. 1952, doi: 10.6028/jres.049.044.

[324] S. Akram and Q. U. Ann, "Newton Raphson method," *Int. J. Sci. Eng. Res.*, vol. 6, no. 7, pp. 1748–1752, Jul. 2015.

[325] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[326] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, May 2018, doi: 10.1137/16M1080173.

[327] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal, "On the use of stochastic hessian information in optimization methods for machine learning," *SIAM J. Optim.*, vol. 21, no. 3, pp. 977–995, Jan. 2011.

[328] S. I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, Feb. 1998, doi: 10.1162/089976698300017746.

[329] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: A review of algorithms and comparison of software implementations," *J. Global Optim.*, vol. 56, no. 3, pp. 1247–1293, 2013, doi: 10.1007/s10898-012-9951-y.

[330] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.

[331] R. Fletcher and E. S. de la Maza, "Nonlinear programming and nonsmooth optimization by successive linear programming," *Math. Program.*, vol. 43, nos. 1–3, pp. 235–256, Jan. 1989, doi: 10.1007/BF01582292.

[332] S. Srinivas and R. Venkatesh Babu, "Data-free parameter pruning for deep neural networks," 2015, *arXiv:1507.06149*.

[333] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[334] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, vol. 37, pp. 2275–2284.

[335] X. Wu, Y. Wu, and Y. Zhao, "Binarized neural networks on the ImageNet classification task," 2016, *arXiv:1604.03058*.

[336] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Computer Vision* (Lecture Notes in Computer Science), vol. 9908. Cham, Switzerland: Springer, Mar. 2016, pp. 525–542.

[337] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*.

[338] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4820–4828, doi: 10.1109/CVPR.2016.521.

[339] V. Vanhoucke, A. Senior, and M. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop*, 2011, pp. 1–8. [Online]. Available: https://www.semanticscholar.org/paper/Improving-the-speed-of-neural-networks-on-CPUs-Vanhoucke-Senior/fbeaa499e10e98515f7e1c4ad89165e8c0677427.

[340] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1737–1746.

[341] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[342] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," in *Proc. 5th Int. Learn. Represent. Conf. (ICLR)*, Dec. 2016, pp. 1–14.

[343] B. B. Sau and V. N. Balasubramanian, "Deep model compression: Distilling knowledge from noisy teachers," 2016, *arXiv:1610.09650*.

[344] G. Hinton, O. Vinyals, and J. Dean, ''Distilling the knowledge in a neural network,'' 2015, *arXiv:1503.02531*.
[345] C. Bucil and R. Caruana, ''Model Compression,'' in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, vol. 54, no. 1, 2006, pp. 535–541.
[346] L. Ba and R. Caruana, ''Do deep nets really need to be deep?'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 3, Jan. 2014, pp. 2654–2662.
[347] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, ''FitNets: Hints for thin deep nets,'' 2014, *arXiv:1412.6550*.
[348] A. Korattikara, V. Rathod, K. Murphy, and M. Welling, ''Bayesian dark knowledge,'' 2015, *arXiv:1506.04416*.
[349] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, ''Face model compression by distilling Knowledge from neurons,'' in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 3560–3566.
[350] T. Chen, I. Goodfellow, and J. Shlens, ''Net2Net: Accelerating learning via knowledge transfer,'' 2015, *arXiv:1511.05641*.
[351] S. Zagoruyko and N. Komodakis, ''Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer,'' 2016, *arXiv:1612.03928*.
[352] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, ''Exploiting linear structure within convolutional networks for efficient evaluation,'' 2014, *arXiv:1404.0736*.
[353] M. Jaderberg, A. Vedaldi, and A. Zisserman, ''Speeding up convolutional neural networks with low rank expansions,'' Tech. Rep., 2014, doi: 10.5244/c.28.88.
[354] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, ''Compression of deep convolutional neural networks for fast and low power mobile applications,'' in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–16.
[355] C. Tai, T. Xiao, Y. Zhang, X. Wang, and E. Weinan, ''Convolutional neural networks with low-rank regularization,'' in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, vol. 1, 2016, pp. 1–11.
[356] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, ''Speeding-up convolutional neural networks using fine-tuned CP-decomposition,'' in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–11.
[357] S. B. Kotsiantis, ''Supervised machine learning: A review of classification techniques,'' *Informatica*, vol. 31, no. 1, pp. 249–268, 2007.
[358] Y. Bengio, A. Courville, and P. Vincent, ''Representation learning: A review and new perspectives,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.
[359] V. François-lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, ''An introduction to deep reinforcement learning,'' *Found. trends Mach. Learn.*, vol. 2, no. 3, pp. 1–140, 2018, doi: 10.1561/2200000071.Vincent.
[360] L. Jing and Y. Tian, ''Self-supervised visual feature learning with deep neural networks: A survey,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2021, doi: 10.1109/TPAMI.2020.2992393.
[361] K. Huang, Z. Xu, I. King, M. R. Lyu, and C. Campbell, ''Supervised self-taught learning: Actively transferring knowledge from unlabeled data,'' in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 1272–1277, doi: 10.1109/IJCNN.2009.5178647.
[362] S. Pratapa, K. Umaroh, and E. Weddakarti, ''Microstructural and decomposition rate studies of periclase-added aluminum titanate–corundum functionally-graded materials,'' *Mater. Lett.*, vol. 65, no. 5, pp. 854–856, Mar. 2011, doi: 10.1016/j.matlet.2010.11.072.
[363] S. J. Pan and Q. Yang, ''A survey on transfer learning,'' *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.
[364] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, ''Towards federated learning at scale: System design,'' 2019, *arXiv:1902.01046*.
[365] D. Lowd and C. Meek, ''Adversarial learning,'' in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2005, pp. 641–647, doi: 10.1145/1081870.1081950.
[366] R. Polikar, ''Ensemble learning,'' in *Ensemble Machine Learning*, C. Zhang and Y. Ma, Eds. Boston, MA, USA: Springer, 2012, pp. 1–34.
[367] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, ''Meta-learning in neural networks: A survey,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 11, 2021, doi: 10.1109/TPAMI.2021.3079209.
[368] M. J. van der Laan and S. Rose, *Targeted Learning*. New York, NY, USA: Springer, 2011.
[369] Y. Zhang and Q. Yang, ''A survey on multi-task learning,'' *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 31, 2021, doi: 10.1109/TKDE.2021.3070203.
[370] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, ''Multimodal machine learning: A survey and taxonomy,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019, doi: 10.1109/TPAMI.2018.2798607.

TOLGA TURAY was born in Erzurum, Turkey, in 1991. He received the B.S. degree in electric/electronic engineering from Ataturk University, Turkey, in 2014, and the M.Sc. degree in embedded systems and control from the University of Leicester, U.K., in 2018, where he is currently pursuing the Ph.D. degree with the School of Engineering.

His research interests include computer vision tasks for autonomous vehicles with deep learning techniques and optimization methods.

TANYA VLADIMIROVA (Senior Member, IEEE) was born in Sofia, Bulgaria. She received the M.Eng. and Ph.D. degrees in computer systems engineering from St. Petersburg Electrotechnical University (LETI), Saint Petersburg, Russia, and the M.Sc. degree in applied mathematics and informatics from the Technical University of Sofia, Bulgaria.

From 1998 to 2011, she was a Reader at the Surrey Space Centre, Department of Electronic Engineering, University of Surrey, U.K. Since 2011, she has been a Professor with the School of Engineering, University of Leicester, Leicester, U.K. She is the author and coauthor of more than 200 articles. Her research interests include in the area of intelligent high-performance embedded computing for autonomous mission critical systems, and AI and ML for image and data processing, intelligent sensing, distributed computing, reconfigurable systems, hardware acceleration, and wireless sensor networks.