Integration, the VLSI Journal 69 (2019) 309–320

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

Computer vision algorithms and hardware implementations: A survey

Xin Feng [a,b], Youni Jiang [a], Xuejiao Yang [a], Ming Du [c,**], Xin Li [b,*]

[a] Computer Science and Engineering Department, Chongqing University of Technology, Chongqing, 400054, China
[b] Data Science Research Center, Duke Kunshan University, Kunshan, Jiangsu, 215316, China
[c] Donghua University, Shanghai, 200051, China

ARTICLE INFO

Keywords:
Computer vision
Hardware accelerator
Deep convolutional neural network
Artificial intelligence

ABSTRACT

The field of computer vision is experiencing a great-leap-forward development today. This paper aims at providing a comprehensive survey of the recent progress on computer vision algorithms and their corresponding hardware implementations. In particular, the prominent achievements in computer vision tasks such as image classification, object detection and image segmentation brought by deep learning techniques are highlighted. On the other hand, review of techniques for implementing and optimizing deep-learning-based computer vision algorithms on GPU, FPGA and other new generations of hardware accelerators are presented to facilitate real-time and/or energy-efficient operations. Finally, several promising directions for future research are presented to motivate further development in the field.

1. Introduction

The recent progress of scientific technologies is producing a "Cambrian explosive" [1] in developing new techniques that lead the world entering promptly into the new artificial intelligence (AI) era. Computer systems injected by the new AI techniques are intelligent to perceive and understand the visual world, and even smarter than humans in a number of specific tasks. The ability of being smart is primarily provided by a computer vision system, including both the algorithms and their hardware implementations, which gives us the ability to teach a computer to understand the physical world from vision.

Computer vision tasks seek to enable computer system automatically to see, identify and understand the visual world, simulating the same way that human vision does [2]. Researchers in computer vision aspired to develop algorithms for such visual perception tasks including (i) object recognition in order to determine whether image data contains a specific object, (ii) object detection in order to localize instances of semantic objects of a given class, and (iii) scene understanding to parse an image into meaningful segments for analyzing. Given the broad range of mathematics being covered and the intrinsically difficult nature of recovering unknowns from insufficient information to fully specify the solution, the aforementioned tasks in the computer vision field are extremely challenging. Studying these problems is both theoretically and practically important.

Early efforts have made a great contribution to the philosophy of human vision and the basic computational theory of computer vision by exploiting well-designed features and feature descriptors combined with classical machine learning methods [3,4]. Although researchers have spent several decades to teach machines how to see, the most advanced machine at that time could only perceive common objects and struggled at recognizing numbers of natural objects with infinite shape variations similar to toddlers [5]. Fortunately, researchers have believed that computer systems can go beyond regular object recognition and learn to reveal details and insights of the visual world by training them to see trillions of images and videos generated from Internet. To nourish the computer brain, the largest image classification dataset "ImageNet" [6] that contains 15 million images across 22,000 classes of objects was created, upon which the well-known "deep learning" technology has demonstrated its overwhelming superiority over traditional computer vison algorithms that treat objects as a collection of shape and color features.

Deep learning is a particular class of machine learning algorithm, which typically simplifies the process of feature extraction and description through a multi-layer convolutional neural network (CNN). CNN aims to transform the high-dimension input image into low-dimension yet highly-abstracted semantic output. Powered by the massive data

* Corresponding author.
** Corresponding author.
E-mail addresses: xfeng@cqut.edu.cn (X. Feng), naria_petrova@163.com (Y. Jiang), 1064699383@qq.com (X. Yang), duming@dhu.edu.cn (M. Du), xinli.ece@duke.edu (X. Li).

https://doi.org/10.1016/j.vlsi.2019.07.005
Received 11 April 2019; Received in revised form 23 June 2019; Accepted 27 July 2019
Available online 6 August 2019

from ImageNet and the modern central processing units (CPUs) and graphics processing units (GPUs), methods based on deep neural network (DNN) achieve the state-of-the-art performance and bring an unprecedented development of computer vision in both algorithms and hardware implementations. In recent years, CNN has become the *de-facto* standard computation framework in computer vision. Numbers of deeper and more complicated networks are developed to make CNNs deliver near-human accuracy in many computer vision applications, such as classification, detection and segmentation. The high accuracy, however, comes at the price of large computational cost. As a result, dedicated hardware platforms, from the general-purpose GPUs to application-specific processors, are investigated to optimize for DNN-based workloads.

In this paper, we look into this rapid evolution of computer vision field by presenting a brief survey on the key algorithms that make computer systems perceivable and the underlying hardware platforms that make these algorithms applicable. In particular, we will discuss how the recent DNN algorithms accomplish the computer vision tasks (i.e. image classification, object detection and image segmentation) with high perception accuracy, and summarize the notable hardware units including GPUs, field-programmable gate arrays (FPGAs) and other advanced mobile hardware platforms that are adapted or designed to accelerate DNN-based computer vision algorithms. According to our knowledge, there are recent summaries in the literature that discuss the DNN-based algorithms for particular tasks, including image classification [7], object detection [8], image segmentation [9], and the corresponding hardware accelerators such as FPGAs [10]. There is no comprehensive survey that covers both algorithm and hardware simultaneously. A thorough review of existing works from both topics is essential for researchers to understand the entire picture and motivate further progress in the computer vision field.

The reminder of this paper is organized as follows. In Section 2, we overview the computer vision algorithms for three visual perception tasks: image classification, object detection and image segmentation. Important hardware platforms including GPUs, FPGAs and other hardware accelerators for implementing the DNN-based algorithms are discussed in Section 3. Finally, we conclude in Section 4.

2. Computer vision algorithms

2.1. Image classification

Image classification is a kind of biologically primary ability of human visual perception system. It has been an active task and plays a crucial role in the field of computer vision, which aims to automatically classify images into pre-defined classes. For decades, researchers have laid path in developing advanced techniques to improve the classification accuracy. Traditionally, classification models can perform well only on small datasets such as CIFAR-10 [11] and MNIST [12]. The great-leap-forward development of image classification occurred when the large-scale image dataset "ImageNet" was created by Feifei Li in 2009 [6]. It was almost the same time when the well-known deep learning technologies started to show great performance in classification and stepped onto the stage of computer vision.

Before the explosion of deep learning methods, research works put lots of efforts in designing scale-invariant features (e.g. SIFT [13], HOG [14], GIST [15]), feature representations (e.g. Bag-of-Features [16], Fisher Kernel [17]) and classifiers (e.g. SVM [18]) for image classification [19,20]. However, these manually crafted features work against objects in natural images with complicated background, variant color, texture, illumination and ever-changing poses and view factors. At the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, AlexNet [21] won the first prize by a significant margin over the second place that was based on SIFT and Fisher Vectors (FVs) [20]. It demonstrates that the classification model based on deep CNN performs much more robustly than other conventional methods in the presence of large-scale variations. It also represents a remarkable milestone in the modern history of neural network after a long trough period.

A typical deep CNN model consists of several convolution layers followed by activation functions and pooling layers, and several fully connected layers before prediction. It comes into deep structure to facilitate filtering mechanisms by performing convolutions in multi-scale feature maps, leading to highly abstract and discriminative features.

AlexNet has 8 convolution layers, 3 pooling layers and 3 fully connected layers, with a total of 60 million parameters. It successfully uses ReLU as the activation function instead of sigmoid. Furthermore, data augmentation and dropout are widely used today as efficient learning strategies. AlexNet is hence known as the foundation work of modern deep CNN.

Inspired by AlexNet, VGGNet [22] and GoogleNet [23] focus on designing deeper networks to further improve accuracy. They were the runner-up and winner of ILSVRC in 2014 respectively. By repeatedly stacking 3×3 convolutional kernels and 2×2 maximum pooling layers, VGGNet successfully constructs a convolutional neural network of 16–19 layers. GoogleNet has 22 layers, but its floating-point operations and number of parameters are much less than those of AlexNet and VGGNet by removing the fully-connected layers and optimizing the operations of sparse matrices.

Although deeper networks offer better accuracy, simply increasing the number of layers cannot continuously improve accuracy because of vanishing/exploding gradient information during network training. ResNet [24], which makes another great progress of deep network structure, proposes to use a shortcut connection between residual blocks to make full use of information from previous layers and keep the gradients during backward propagation. By using this residual block, ResNet successfully trains very deep networks with up to 152 layers and was the winner of ILSVRC in 2015. Following the idea of ResNet, DenseNet [25] establishes connections between all previous layers and the current layer. It concatenates and, therefore, reuses the features from all previous layers. DenseNet presents with great advantage in classification accuracy on ImageNet, as we can see in Table 1. Based on these works in the literature, connecting different network layers has shown promising improvement in learning representations of deeper networks.

By using ResNet or DenseNet as the major backbone structure, researchers focus on improving the functionality of neural network blocks. SENet [26], which was the winner of ILSVRC 2017, proposes a "squeeze-and-excitation" (SE) unit by taking channel relationship into account. It learns to recalibrate channel-wise feature maps by explicitly modeling the interdependencies among channels, which is consequently exploited to enhance informative channels and suppress other useless channels.

Despite the high classification performance of the aforementioned CNN models, appropriately designing the optimal network structure

Table 1
Summary of different CNN models on ImageNet classification task.

Model	Time	Accuracy	Num. of Parameters	Num. of FLOPs	Num. of Layers
AlexNet [21]	2012	57.2%	60 M	720 M	8
VGGNet [22]	2014	71.5%	138 M	15,300 M	16
GoogleNet [23]	2014	69.8%	6.8 M	1,500 M	22
ResNet [24]	2015	78.6%	55 M	2,300 M	152
DenseNet [25]	2017	79.2%	25.6 M	1,150 M	190
SENet [26]	2017	82.7%	145.8 M	42,300 M	–
NASNet [27]	2018	82.7%	88.9 M	23,800 M	–
SqueezeNet [29]	2016	57.5%	1.2 M	833 M	–
MobileNet [30]	2017	70.6%	4.2 M	569 M	28
ShuffleNet [31]	2018	73.7%	4.7 M	524 M	–
ShiftNet-A [32]	2018	70.1%	4.1 M	1,400 M	–
FE-Net [33]	2019	75.0%	5.9 M	563 M	–

often requires significant engineering work. NASNet [27] studies a paradigm to learn the optimal convolutional architecture based on training data. It adopts a neural architecture search (NAS) framework derived from reinforcement learning [28]. In addition, it designs a new search space to enable network mapping from a proxy dataset (e.g. CIFAR-10) to ImageNet, and a regularization technique for generalization purpose. Offering less computational complexity than SENet, NASNet achieves the state-of-the-art accuracy on ImageNet, as shown in Table 1.

The aforementioned deep networks facilitate classification to be more accurate. However, in many real-life classification applications, such as robotics, autonomous driving, smartphone, etc., the classification task is highly constrained by the computational resources that are available. The problem thus becomes to pursue the optimal accuracy subject to a limited computational budget (i.e. memory and/or MFLOPs). Therefore, a set of lightweight networks such as SqueezeNet [29], MobileNet [30], ShuffleNet [31], ShiftNet [32] and FE-Net [33] start a wave.

SqueezeNet substitutes most 3×3 filters by 1×1 filters and cuts down the numbers of input channels for 3×3 filters to reduce the network complexity. To maximize the accuracy with a limited number of network parameters, it delays the down-sampling operation to avoid information loss in early layers. SqueezeNet is 50 $\times$ smaller than AlexNet. If combined with deep compression [34], it can even be reduced to be 510 $\times$ smaller than AlexNet. Depth-wise separable convolution is employed to decompose the standard convolution into depth-wise convolution and point-wise convolution in MobileNet [30]. Depth-wise convolution performs convolution on each input channel with one filter, while point-wise convolution combines those separate channels by using 1×1 convolution. This novel design of convolution reduces both the computational complexity and the number of parameters.

ShuffleNet [31] uses point-wise group convolution that divides the input feature maps into groups and performs convolutions separately on each group to reduce computational cost. However, because the grouping operations limit the communication between different channels, ShuffleNet further shuffles the channels and feeds each group in the following layer with multiple channels from different groups in order to distribute information across channels.

In addition to the strategies adopted to reduce the computational cost of spatial convolution (e.g., depth-wise convolution), ShiftNet [32] presents a parameter-free, FLOP-free shift operation to replace expensive spatial convolutions. The proposed shift operation is able to provide spatial information communication by shifting feature maps, making it possible to aggregate spatial information by the following point-wise convolutional layer. More recently, FE-Net [33] further finds that only a few shift operations are sufficient to provide spatial information communication. A sparse shift layer (SSL) is proposed to perform shift operations on a small portion of feature maps only. With only 563 M FLOPs, FE-Net achieves the state-of-the-art performance among all major lightweight classification models on ImageNet, as shown in Table 1.

The aforementioned network models are briefly summarized in Table 1. In addition to the conventional image classification problem with thousands of classes and complex scenes, multi-label classification (e.g. face attributions [35,36]) and fine-grained classification (e.g. Stanford Dogs classification [37]) are also of great interest in the computer vision area.

Furthermore, the great success of deep learning in image domain has stimulated a variety of techniques to learn robust feature representations for video classification, where the semantic contents such as human actions [38] or complex events [39] are automatically categorized. Early works often treat a video clip as a collection of frames. Video classification is implemented by aggregating frame-level CNN features by averaging or encoding [40]. Standard classifiers, such as SVM, are finally used for recognition [41,42].

In contrast to the frame-level classification methods, there are a number of other approaches applying end-to-end CNN models to learn the hidden spatio-temporal patterns in video. For example, the typical C3D features [43] are derived from a deep 3-D convolutional network trained on the large-scale UCF101 dataset. Moreover, a two-stream approach [44] is proposed to factorize the learning problem of video representation into spatial and temporal cues separately. Specifically, a spatial CNN is adopted to model the appearance information from RGB frames, while a temporal CNN is used to learn the motion information from the dense optical flow among adjacent frames.

Since the two-stream approach only depicts movements within a short time window and fails to consider the temporal order of different frames, several recurrent connection models for sequential data, including recurrent neural networks (RNNs) and long short-term memory (LSTM) models, are leveraged to model the temporal dynamics for videos. In Ref. [45], two two-layer LSTM networks are trained with features from the two-stream approach for action recognition. In Ref. [46], the LSTM model and CNN model are combined to jointly learn spatial-temporal cues for video classification. In Refs. [47,48], attention mechanism is introduced for convolutional LSTM models to discover relevant spatio-temporal volumes for video classification.

2.2. Object detection

Object detection, which is to determine and locate the object instances either from a large number of predefined categories in natural images or for a given particular object (e.g., Donald Trump's face, the distorted area in an image, etc.), is another important and challenging task in computer vision. Object detection and image classification share a similar technical challenge: both of them must handle a large number of highly variable objects. However, object detection is more difficult than image classification, as it must identify the accurate localization of the object of interest.

Historically, most research efforts have focused on detecting a single category of given objects such as pedestrian [14,49] and face [50] by designing a set of appropriate features (e.g. HOG [14,49], Harr-like [50], LBP [51], etc.). In these works, objects are detected by using a set of predefined feature templates matching with each location in the image or feature pyramids. Standard classifiers such as SVM [14,49] and Adaboost [50] are often used for this purpose.

In order to build a general-purpose, robust object detection system, research community has started to develop large-scale, multi-class datasets in recent years. Pascal-VOC 2007 [52] with 20 classes and MS-COCO [53] with 80 object categories are two iconic object detection datasets. In these two datasets, detection results are evaluated by two possible metrics: (i) Average Precision (AP) by counting the correctly detected bounding boxes for which the overlap ratio exceeds 0.5, and (ii) mean Average Precision (mAP) by averaging the AP values associated with different thresholds of the overlap ratio.

Recently, deep learning has substantially advanced the object detection field. As shown in Fig. 1, striking improvements in object detection accuracy have been demonstrated over both Pascal-VOC 2007 and MS-COCO by taking advantages of deep learning techniques.

R-CNN [54] was the first two-stage method among the earliest CNN-based generic object detection techniques. It adopts AlexNet to extract a fixed-length feature vector from each resized region proposal, which is the object candidate generated by selective search algorithm [55]. Each region is then classified by a set of category-specific linear SVMs. The method shows significant improvement in mAP over the traditional state-of-the-art DPM detector [49]. It is, however, not elegant and inefficient, due to its multistage complex pipeline and the redundant CNN feature extraction from numerous region proposals.

Inspired by the spatial pyramid pooling in SPPnet [56] that leverages fixed-length feature outputs for arbitrary input image sizes, Fast R-CNN [57] incorporates a ROI pooling layer before the fully-connected layer to obtain a fixed-length feature vector for each proposed region, so that only a single convolution operation is required for the input image. Fast R-CNN substantially improves the detection efficiency over R-CNN and SPPnet. However, it requires expensive computation for external region

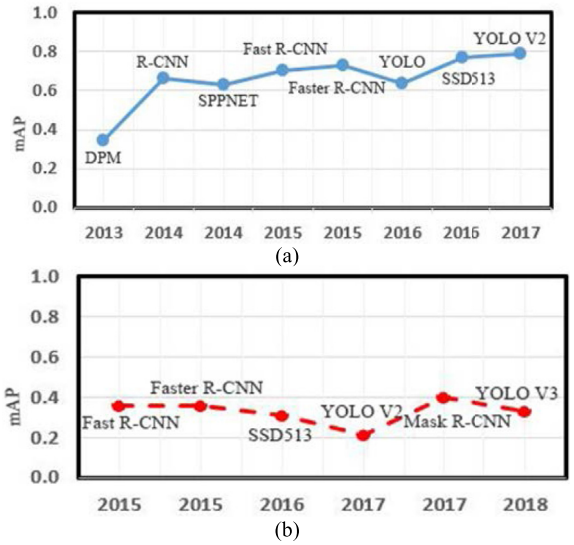

Fig. 1. Object detection accuracy on (a) Pascal-VOC 2007 and (b) MS-COCO for different object detection techniques based on deep learning.

proposal, which now becomes the major bottleneck in overall efficiency.

To address the aforementioned challenge associated with computational complexity, Faster R-CNN [58] further proposes a Region Proposal Network (RPN). It then integrates both RPN (for proposal generation) and Fast-RCNN (for region classification) into a unified, end-to-end network structure. RPN and Fast-RCNN share most of the convolution layers, and the features from the last shared layer are used for two separate tasks (i.e. proposal generation and region classification). With this highly efficient architecture, Faster R-CNN achieves 6 FPS inference speed on a GPU and the state-of-the-art detection accuracy on Pascal-VOC 2007.

As a follow-up work on Faster R-CNN, Mask R-CNN [59] was later proposed to combine object detection and pixel-level instance segmentation based on Faster R-CNN. By using ResNet101-FPN as the backbone network, Mask R-CNN demonstrated the best detection accuracy on MS-COCO in 2017.

The two-stage R-CNN architectures are able to offer superior accuracy. However, real-time efficiency is required for object detection by many real-world applications. In this regard, the simple one-stage architectures are often preferred. YOLO [60] is the first one-stage method that casts detection task as a regression problem. It divides an image into a number of $S \times S$ grids and proposes B bounding boxes for each grid. Next, by using the CNN features of the input image globally, it directly predicts the coordinates, confidence scores and C-class probabilities for these bounding boxes. Without the proposal generation step, YOLO can achieve the real-time speed of 45 FPS. On the other hand, since YOLO investigates few prediction candidates, it is less accurate than the two-stage models, especially for small objects detection.

The Single Shot MultiBox Detector (SSD) [61] follows a similar one-stage strategy. It outperforms YOLO in accuracy due to two major improvements. First, SSD extracts important features from multi-scale CNN feature maps. Second, it adopts a number of default bounding boxes by following the concept of anchor proposed by Faster R-CNN.

YOLOv2 absorbs the merit of SSD and Faster R-CNN by introducing the anchor mechanism [62]. The new YOLOv2 model both improves detection accuracy and reduces inference time by a large margin over YOLO on Pascal-VOC 2007. However, its accuracy is still worse than the two-stage methods for generic detection tasks (e.g. MS-COCO).

In the latest implementation of YOLOv3 [63], several anchor boxes are assigned on three different scaled feature maps, thereby producing much more proposals than YOLO and YOLOv2. Small objects can thus be accurately detected from the anchors in low-level feature maps with small receptive fields. In addition, it uses a powerful backbone network darknet-53 with several sets of residual blocks. YOLOv3 achieves similar accuracy as Faster R-CNN, while maintaining real-time efficiency. It is the current state-of-the-art object detection framework for real-time applications.

Table 2
Summary of different object detection architectures (FPS based on a Pascal Titan X GPU).

Architecture	mAP (Pascal-VOC 2007)	mAP (MS-COCO)	Num. of FLOPs	FPS
R-CNN [54]	66.0%	–	–	0.1
SPPnet [56]	63.1%	–	–	1
Fast R-CNN [57]	70.0%	35.9%	–	0.5
Faster R-CNN [58]	73.2%	36.2%	–	6
Mask R-CNN [59]	–	39.8%	–	3.3
YOLO [60]	63.4%	–	–	45
SSD513 [61]	76.8%	31.2%	–	8
YOLOv2 (608) [62]	78.6%	21.6%	62.94 G	67
YOLOv3 (416) [63]	–	33.0%	65.86 G	35

We summarize the performance metrics of the aforementioned detection models in Table 2. In addition to the one-stage and two-stage architectures, there are several other spotlights for object detection. For example, the relationship of different objects is considered by designing an object relation module in Ref. [64]. Generative Adversarial Network (GAN) is used to generate the super-resolution of small object patches [65] or features [66] to help with the detection of small objects.

In contrast to the significant progress in object detection focusing on still images, video object detection has received less attention. Generally, object detection for videos is realized by fusing the results of object detection on the current frame and object tracking from the previous frames. Deep SORT [67] is a typical tracking-by-detection algorithm for multi-object tracking in videos. By integrating appearance information extracted from a CNN-based object detector with the original SORT [68] algorithm, it is able to achieve real-time tracking. Recently, several approaches have been proposed for end-to-end video object detection. In Ref. [69], temporal feature aggregation is performed to improve feature quality and recognition accuracy. In Ref. [70], data redundancy between consecutive frames is exploited. These seminal works have been adopted by the winner of ImageNet Video Object Detection Challenge 2017 [71].

2.3. *Image segmentation*

Image classification should recognize *what* objects are in the visual scene (as shown by the example in Fig. 2(a)), while object detection reveals *where* the objects are (as shown by the example in Fig. 2(b)). In this sub-section, we further focus on the problem of *how* the objects are exactly presented in the visual scene by using image segmentation.

Image segmentation is regarded as pixel-level classification, which aims at dividing an image into meaningful regions by classifying each pixel into a specific entity. In traditional image segmentation, the idea of unsupervised local region merging and splitting has been extensively explored based on clustering [72], optimizing global criteria [73], or user interaction [74]. The blooming deep learning technologies have promoted large-scale supervised classification moving from image-level object classification to box-level object localization, and further to pixel-wise object segmentation. Therefore, today's image segmentation is object oriented and can be divided into two subtle branches: (i) semantic segmentation, which assigns each pixel in an image to a semantic object class, as shown in Fig. 2 (c), and (ii) instance segmentation, which predicts different labels for different object instances as a further improvement to semantic segmentation, as shown in Fig. 2 (d).

In addition to classification and detection, the challenges of Pascal-VOC 2012 [76] and MS-COCO provide segmentation competitions as

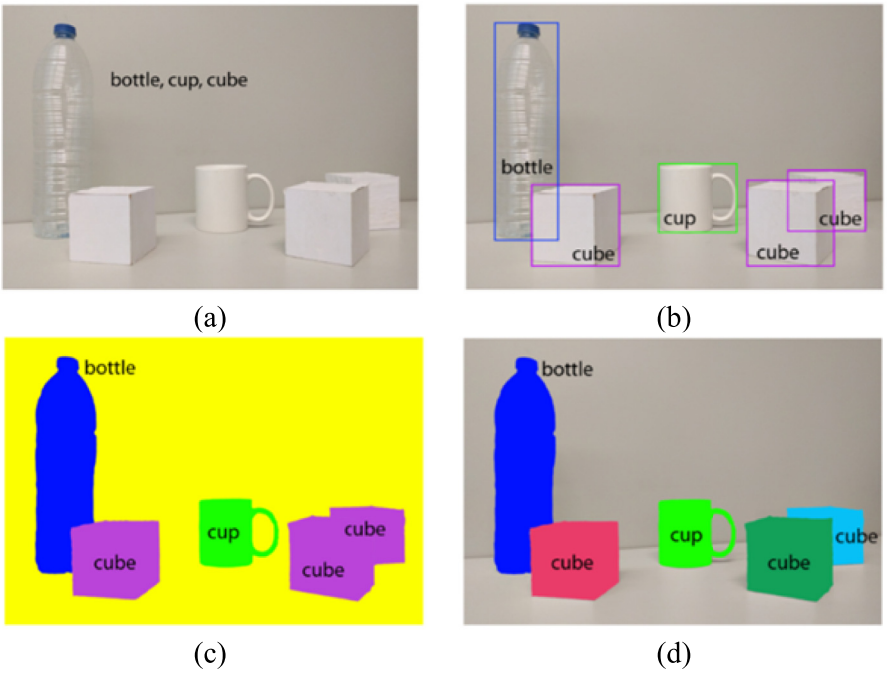

Fig. 2. An example of different visual perception problems [75]: (a) image classification, (b) object detection, (c) semantic segmentation, and (d) instance segmentation.

well. They are two important datasets for image segmentation on which most research works are evaluated. In addition to them, Cityscapes [77] and MVD [78] provide street scenes with a large number of traffic objects in each image. The popular metrics to evaluate pixel-level segmentation accuracy are pixel accuracy (PA, i.e., the proportion of pixels predicted correctly), mean pixel accuracy (mPA) of all classes, mean intersection over union (mIOU), and frequency weighted intersection over union (FWIOU). Among them, mIOU is often preferred for its simplicity and representativeness.

Early segmentation techniques based on deep learning usually apply CNN as the feature descriptor for each pixel that is described by its surrounding patch [79,80]. This CNN-based framework is problematic in efficiency and is not sufficiently accurate for redundant feature extraction. Fully convolutional network (FCN) [81] is the forerunner that successfully implements pixel-wise dense predictions for semantic segmentation in an end-to-end CNN structure. It replaces the fully-connected layers of the well-known classification architectures (e.g.VGG [22], GoogleNet [23], etc.) with convolution layers to facilitate inputs of arbitrary sizes, and outputs a heatmap rather than a vector to indicate classification scores. Prediction loss is then measured by the pixel-wise loss between the upsampled heatmap using deconvolution and the labeled image of original size. FCN shows great improvement in pixel accuracy over traditional segmentation methods on Pascal-VOC 2012. However, the basic FCN structure fails to capture a large number of features and it does not consider spatial consistency between pixels, which hinders its application to certain problems and scenarios.

In any wise, the success of FCN architecture makes it popular and it has been actively followed by many subsequent segmentation works [82–84]. Generally, using classification models without fully-connected layers as the backbone network to produce low-resolution feature maps is referred to as the *encoder*, while the symmetric mapping from the low-resolution image to pixel-wise classification outcome is termed as the *decoder*. With the well-known backbone network as encoder, alternative CNN-based segmentation works are usually variant in decoder implementation. For example, the decoding stage of SegNet [82] uses the max-pooling indices from corresponding feature maps in its encoder for upsampling. The resultant maps are then convolved with a set of filters to generate the restored feature maps for dense prediction. In another typical encoder-decoder framework U-Net [83] that is designed for biomedical image segmentation, the upsampling layer directly concatenates with a set of cropped duplicates of corresponding feature maps in encoder to enhance resolution during the decoder process.

Image segmentation is a difficult problem that requires both good pixel-level accuracy, which relies on fine-grained local features, and classification accuracy, for which global context of the image is crucial to resolve local ambiguities. However, the pooling strategy in classic CNN architectures is a defect for losing the detailed information when multi-pooling steps are performed.

One possible and common solution to integrate context knowledge is to refine the output to have fine-grained details for accurate segmentation by Conditional Random Field (CRF) [85–87]. Alternatively, instead of using pooling strategy, the problem can be solved by expanding receptive fields in which each neuron is connected to a subset of neurons in the previous layer. Dilated convolution [88], which is a regular convolution with upsampled filters or dilated filters, is proposed to exponentially expand receptive fields without sacrificing resolution, as shown in Fig. 3. The DeepLab [85] model takes advantages of both dilated convolution and CRF refinement by post-processing to integrate context knowledge. As can be seen in Fig. 4, it achieves much higher prediction accuracy than SegNet and FCN and is thus considered as a milestone work for semantic segmentation.

Several recently-proposed methods, such as RefineNet [84] and PSPNet [89], try to avoid or restore the loss of down-sampling in encoder by fusing low-level and high-level features. RefineNet designs a decoder module by using both short-range and long-range residual connections to capture rich contextual information. It has achieved the state-of-the-art performance on 7 public datasets. In PSPNet, a pyramid pooling module is proposed to aggregate different region-based context information to exploit the capability of global context information.

Inspired by the spatial pyramid pooling, DeepLab V2 [87] investigates an atrous spatial pyramid pooling (ASPP) module by incorporating dilated convolution with different sampling rates and spatial pyramid pooling to capture multi-level context information. In the latest DeepLab V3+ [90], an Xception module [91] is introduced to the

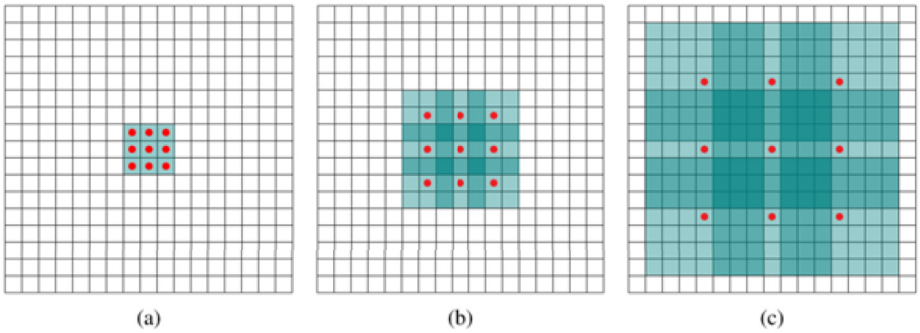

Fig. 3. The receptive field of (a) 1-dilated convolution, (b) 2-dilated convolution, and (c) 4-dilated convolution [88].

encoder, together with the improved ASPP module. It obtains the state-of-the-art performance on Pascal-VOC 2012, as shown in Fig. 4.

The aforementioned end-to-end architectures mainly focus on semantic segmentation. Comparatively, most works on instance segmentation follow the pipeline that segmentation precedes recognition. For example, DeepMask [92] and the instance-sensitive fully convolutional network [93] use Fast-RCNN to classify the learned segment proposals. The fully convolutional instance segmentation (FCIS) combines the segment proposal system [93] and object detection in Ref. [94] to predict object classes, boxes, and masks simultaneously. However, this method is not as accurate as Mask-RCNN [59] on the MS-COCO instance segmentation challenge. Mask-RCNN extends Faster-RCNN by adding a mask prediction branch in addition to bounding box regression and class recognition. The very recent path aggregation network (PANet) [95] enhances the feature hierarchy by a bottom-up path augmentation. With subtle computation overhead, it reaches the first place in the MS-COCO challenge of instance segmentation task and also represents the state-of-the-art on MVD and Cityscapes.

Although pixel-wise image segmentation is progressing rapidly with superior accuracy, it is still far from practical usage, such as video semantic segmentation, due to the high complexity of dense prediction. Therefore, developing highly efficient image segmentation framework is one of the grand challenges in the computer vision community.

3. Hardware implementation

The recent breakthroughs in developing computer vision algorithms are not only driven by deep learning technologies and large-scale datasets, but also relying on the major leaps of hardware acceleration that provides powerful parallel computing architectures to enable the efficient training and inference of large-scale, complex and multi-layered neural networks.

Hardware acceleration takes advantage of computer hardware to perform computing tasks with lower latency and higher throughput than the conventional software implementation running on general-purpose CPUs [98]. Historically, the von-Neumann-style compute-centric architectures (e.g. CPUs) are primarily designed for effective serial computations with complex task scheduling. It suffers from high energy consumption and low memory bandwidth for data movement when evaluating the deep CNN networks that require parallel dense computation, high data reusability and large memory bandwidth [99].

Fig. 5(a) compares neural network with other approaches in terms of accuracy and scale (i.e. data/model size). The traditional machine learning methods, such as decision tree, SVM, etc., are referred as "Other Approaches" in Fig. 5(a). They are generally based on manually designed features. Due to the limited learning capability of these traditional methods, their accuracy cannot continuously increase with data/model scale. On the other hand, deep neural networks are highly scalable in their learning capability when deeper network structures and larger data sets are adopted.

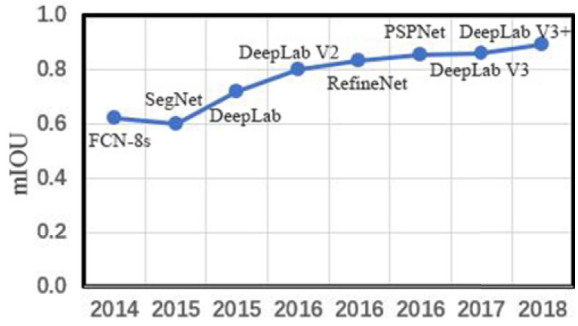

Fig. 4. mIOU for different semantic segmentation methods on Pascal-VOC 2012.

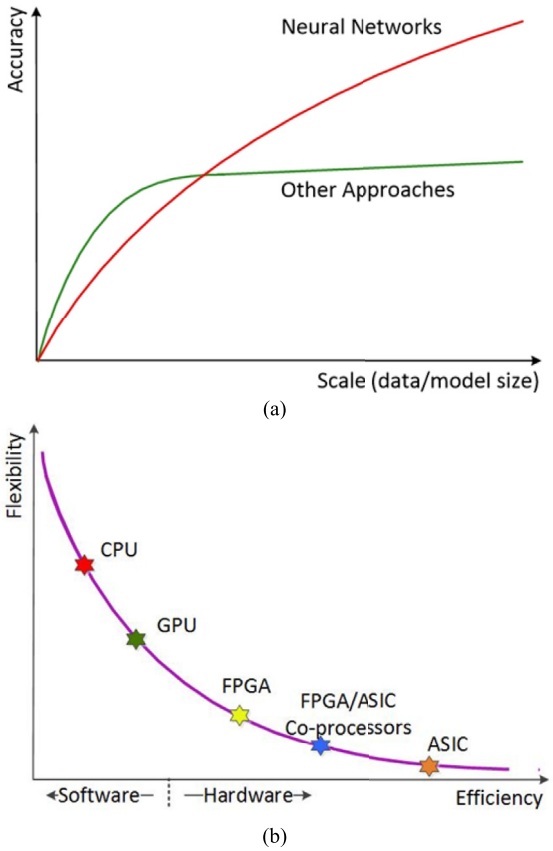

Fig. 5. (a) Comparison between neural networks and other approaches in terms of accuracy and scale (i.e. data/model size) [96], and (b) trade-off between flexibility and efficiency for different hardware implementations [97].

Practically, an operation can be computed faster in the hardware platform that is application-specifically designed and/or programmed. Hardware acceleration thus steps forward for heavy customization in processing capability by allowing great parallelism, having specific datapaths for temporal variants, and reducing the overhead of instruction control [98]. For decades, hardware customization in the form of GPUs, FPGAs, and application-specific integrated circuits (ASICs) offer a promising path in trading flexibility for computation efficiency, as seen in Fig. 5(b). In this section, we review a number of popular hardware implementations including GPUs, FPGAs and other application-specific accelerators.

3.1. Graphics processing units (GPUs)

GPUs were initially developed to accelerate graphics processing. A GPU is particularly designed for integrated transform, lighting, triangle setup/clipping, and rendering [100]. A modern GPU is not only a powerful graphics engine but also a highly parallelized computing processor featuring high throughput and high memory bandwidth for massive parallel algorithms, which is dubbed as GPU computing or general-purpose computing on GPU (GPGPU).

In contrast to multicore CPUs that are typically out-of-order, multi-instructional, running at high frequencies and using large-size caches to minimize the latency of a single-thread, GPGPUs consist of thousands of cores that are in-order, operating at lower frequencies and relying on smaller-size caches. To create high performance GPU-accelerated applications with parallel programming, a variety of development platforms such as compute unified device architecture (CUDA) [101] and open computing language (OpenCL) [102], are studied and utilized for GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms and high-performance computing (HPC) servers.

A number of hardware vendors have produced GPUs. Among them, Intel, Nvidia and AMD/ATI have been the market share leaders [100]. As shown in Fig. 6, the evolution of GPGPUs began in 2007, when Nvidia released its CUDA development environment. A great variety of GPUs have been designed for a specific usage, such as Nvidia GeForce GTX and AMD Radeon HD GPUs for powerful gaming, and Nvidia Quadro and Titan X series for professional workstation [100]. More recently, the emergence of deep learning technology ushers in significant advances in GPU computing.

Taking CNN as an example, it can take advantages of the nature of algorithmic parallelism in the following aspects [103]: (i) the convolution operation of an $n \times n$ matrix using a $k \times k$ kernel can be in parallel; (ii) the subsampling/pooling operation can be parallelized by executing different pooling operations separately; (iii) the activation of each neuron in a fully connected layer can be parallelized by creating a binary-tree multiplier. With great parallel processing structures and strong floating-point capabilities, GPGPUs have been recognized to be a good fit to accelerate deep learning. A number of GPU-based CNN libraries have been developed to facilitate highly optimized CNN implementation on GPUs, including cuDNN [104], Cuda-convnet [105] and several other libraries built upon the popular deep learning frameworks, such as Caffe [106], Torch [107], Tensorflow [108], etc.

Computational throughput, power consumption and memory efficiency are three important metrics when implementing deep learning on GPUs. Fig. 6 summarizes the peak performance of recent Nvidia GPUs for single-precision floating-point (FP32) arithmetic measured by GFLOPs and power consumption gauged by Thermal Design Power (TDP). The GeForce 10 series, based on the most powerful GPU architecture "Nvidia Pascal", is a set of consumer graphics cards released by Nvidia in 2016 [110]. With an inexpensive GeForce GTX 1060, composed of 1280 CUDA cores delivering 3855 GFLOPs in computational throughput, one can get into deep learning with affordable cost.

For professional usage, Titan V and Tesla V100 are much more powerful and scalable than the GeForce 10 series based on the Pascal and a new Volta architecture, which integrates CUDA cores with the new Tensor core technology. Tensor cores are especially designed for deep learning, which offer an extremely wide memory bus. Compared to CUDA cores, they improve the peak performance by up to 12 × for training and up to 6 × for inference. In addition to their high throughput, Tensor cores allow efficient computation with 16-bit word-length, implying that the amount of transferred data can be doubled over 32-bit arithmetic with the same memory bandwidth.

Nvidia Jetson is a leading low-power embedded platform that enables server-grade computing performance on edge devices. Jetson TX2 is based on the 16 nm NVIDIA Tegra "Parker" system-on-a-chip (SoC), which delivers 1 TFLOPs of throughput in a credit-card-sized module. A new series of RTX gaming cards (i.e., RTX 2070/2080/2080Ti) with Turing architectures were unveiled in August 2018. They have Tensor cores on board and support unrestricted 16-bit floating-point (FP16), 8-bit integer (INT8) and 4-bit integer (INT4) arithmetic. Among them, RTX 2080Ti offers the promising performance with more than 100 TFLOPs in FP16.

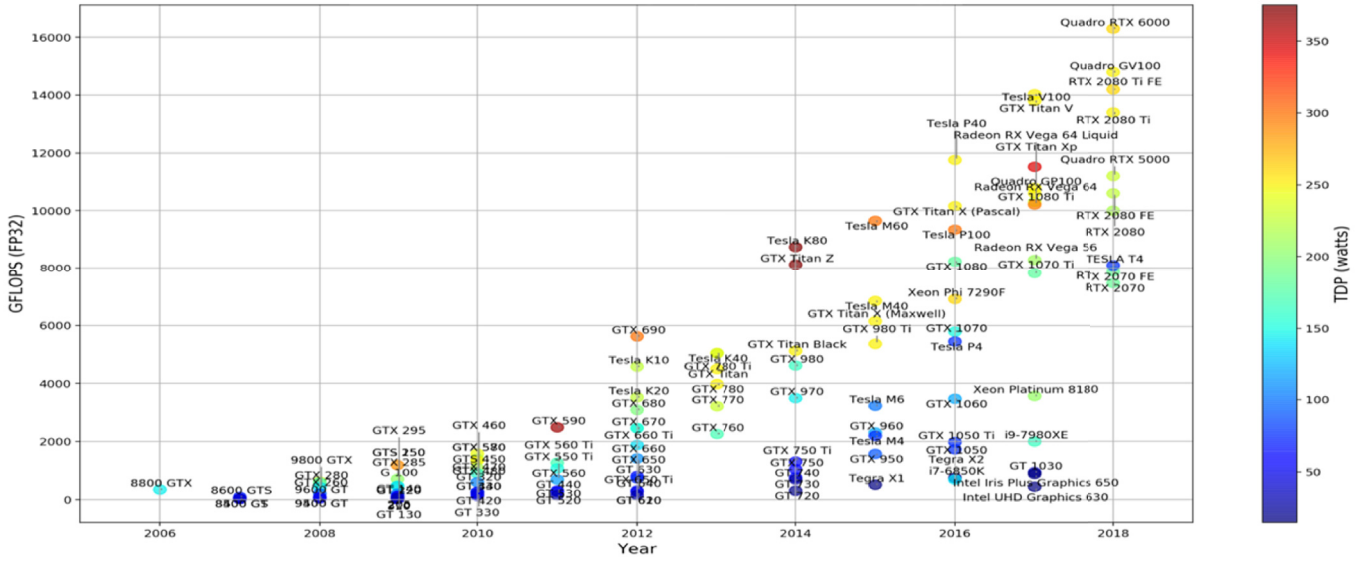

Fig. 6. Computational throughput in terms of GFLOPS and power consumption in terms of TDP (watts) for single-precision floating-point arithmetic [109].

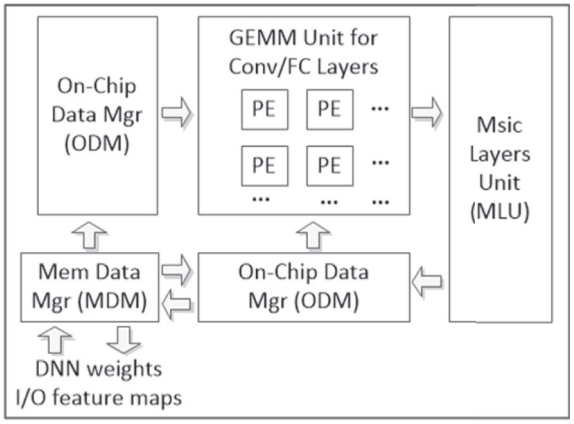

Fig. 7. A typical FGPA architecture to implement DNNs [115].

When evaluating memory efficiency for GPUs, memory size and memory bandwidth are two important metrics. Today, it is common to have 11–12 GB memory on start-of-the-art gaming cards. Many Tesla GPUs have 16 GB memory, and several Turing architecture Quadro models have 24 GB memory (e.g. RTX 6000) or 64 GB memory (e.g. RTX 8000). For a given deep learning task, the peak performance of GPU is often far from the actual performance. In most practical applications, the throughput of a GPU is about 15–20% of its peak performance [111]. It, in turn, implies that evaluating DNNs is actually limited by memory bandwidth, instead of computing power. Adopting GPUs with high memory bandwidth is a valid strategy to speed up both training and inference. For example, GTX Titan XP (10,709 FP32 GFLOPS and 548 GB/s) can be faster than GTX 1080Ti (10,609 FP32 GFLOPS and 484 GB/s) by up to 13% on bandwidth-limited tasks. GTX Titan V (652.8 GB/s), Tesla V100 (900 GB/s) and P100 (720 GB/s) are even faster than GTX Titan XP [109]. In addition to Nvidia, AMD has also released its high-end Vega GPUs with high memory bandwidth similar to Titan V [100].

In order to address the fundamental issue of limited computing throughput and memory bandwidth, multi-GPU systems allow single machine and multi-GPUs or even distributed multi-system, multi-GPU configurations. For a system with single machine and multi-GPUs working on separate tasks, one can directly access any available GPU without coding in CUDA. On the other hand, for multi-GPUs working on shared tasks such as training several models with different hyper-parameters, distributed training is needed. Nvidia has a collective communications library (NCCL) [112] that implements multi-GPU and multi-node collective communication primitives to make full use of all GPUs within and across multi-nodes with maximum bandwidth. Distributed training is now supported by many popular deep learning frameworks such as Tensorflow, Caffe, etc. These techniques reduce computational time linearly with the number of GPUs [112].

3.2. Field-programmable gate arrays (FPGAs)

While GPUs have been demonstrated to offer extremely high throughput and are broadly used for hardware acceleration of DNNs, they are often not preferred for energy/power-constrained applications, such as IoT devices, due to their high power consumption. DNN acceleration thus moves towards an alternative solution based on energy-efficient FPGAs. FPGA allows us to implement irregular parallelism, customized data type and application-specific hardware architecture, offering great flexibility to accommodate the recent DNN models that are featured with increased sparsity and compact network structures. Further, FPGA can be reprogrammed after manufacturing for desired functions and applications. Due to these attractive features, a large number of FPGA-based accelerators have been proposed for both HPC data centers [113] and embedded applications [114].

FPGA is a component that contains an array of programmable logic blocks connected via a hierarchy of reconfigurable interconnects. Today's FPGAs usually contain (i) digital signal processing units (DSPs) for multiply-add-accumulation (MAC) operations, (ii) lookup tables (LUTs) for combinatorial logic operations, and (iii) block RAMs for on-chip data storage [10]. Fig. 7 shows a typical FPGA architecture to implement DNNs [115]. It consists of memory data management (MDM) unit, on-chip data management (ODM) unit, general matrix multiply (GEMM) unit that is implemented by a set of processing elements (PEs) to perform one or more MAC operations, and msic-layers (MLU) unit where ReLU pooling and batch normalization are computed.

The FPGA architecture in Fig. 7 executes a DNN model by following several major steps. It first fetches DNN weights and input feature maps into an on-chip buffer (i.e. ODM) from MDM. Next, the GEMM unit performs matrix operations and transfers the outcomes to MLU for ReLU/batch normalization/pooling. The output of MLU goes to another ODM unit and will be accessed by the subsequent convolutional and/or fully connected layers. If the on-chip buffer does not have sufficiently large capacity, the intermediate results must be temporarily stored in on-chip or off-chip memory.

Historically, an FPGA system is often specified at register-transfer level (RTL) by using hardware description language (HDL) such as Verilog or VHDL. This low-level design methodology needs substantial efforts and hardware expertise to carefully describe the detailed hardware architecture, including the massive concurrency between different hardware modules. Recently, high-level synthesis (HLS) tools have been successfully developed to facilitate efficient FPGA design by using high-level programming language such as C and C++, and automatically compile high-level description to generate low-level specification (i.e. HDL) [116]. With the aforementioned synthesis flow, the design cost of FPGA accelerators can be significantly reduced. However, there is an important tradeoff between RTL and HLS approaches in terms of design cost and system performance.

In practice, a DNN model is often trained or fine-tuned on a high-performance computing platform such as GPU, while FPGA acceleration is implemented for DNN inference to process given input data based on the pre-trained DNN model. As illustrated in Tables 1 and 2, computer vision algorithms based on DNNs are often associated with high computational workload (i.e., a large number of FLOPs) and large memory storage (i.e., a large number of network parameters), while the memory bandwidth of FPGAs is often less than 10% of that of GPUs. Hence, the grant challenge is to find an efficient mapping from the pre-trained complex DNN model to the limited hardware resources (i.e., high-density logic and memory blocks) offered by FPGAs. Such a technical challenge has been tackled via hardware-friendly algorithmic optimizations, including: (i) algorithmic operation, (ii) data-path optimization and (iii) model compression.

Algorithmic operation: Computational transforms, such as GEMM, fast Fourier transform (FFT) and Winograd transform, may be applied to feature maps and/or convolutional kernels to reduce the number of arithmetic operations during inference. GEMM is a popular way of processing DNNs in CPUs and GPUs, which vectorizes the computation of convolutional and fully connected layers [117]. FFT casts 2D convolution to element-wise matrix multiplication, thereby reducing the arithmetic complexity. It is highly efficient for large kernel size (>5) because of the large number of convolutional operations between the feature maps and kernels [117,118]. For small kernels where FFT is not preferred, Winograd transform [119] provides an alternative way to reduce the number of multiplications by reusing the intermediate results. It can offer 7.28 $\times$ runtime speed-up compared to GEMM when running VGGNet on a Titan X GPU [118], and deliver the throughput of 46 GOPs when running AlexNet on FGPA [120].

Data-path optimization: In order to fully exploit the parallelism, data path is optimized by unrolling the convolutional layers in CNNs and mapping them onto a limit number of PEs. In early FPGA

implementations, PEs are arranged in a 2D grid as a systolic array [121–123]. Because such a simple architecture limits the CNN kernel size and does not offer data caching, it cannot achieve extremely high performance. Recently, loop optimization techniques, including loop reordering, unrolling, pipelining and tiling, have been proposed to address the aforementioned issue. Loop reordering tries to prevent redundant memory access between loops to increase cache usage efficiency [124]. Loop unrolling and pipelining maximize the utilization of FPGA resources by exploring the parallelism of loop iterations [115,125,126]. Loop tiling deals with the issue posed by insufficient on-chip memory of FPGAs. It partitions the feature maps and weights of each layer fetched from memory into chunks, also referred to as tiles, to fit them into on-chip buffers [124,125,127].

Model compression: DNNs often carry a significant number of redundant parameters and are mainly used for error-tolerant applications. Hence, a lot of efforts have been made to simplify DNN models and, consequently, reduce the complexity of their hardware implementation. These model compression methods can be broadly classified into three different categories: (i) pruning [115,128–131], (ii) low-rank approximation [114,132], and (iii) quantization [114,121,133–137].

First, pruning is usually the first step to reduce model redundancy by removing the least-important connections and/or parameters. Taking CNN as an example, we can remove its weights that are extremely small [34] and/or cause high energy consumption [130]. After pruning, the CNN model is highly sparse and can be efficiently implemented with FPGA by masking the zero weights for multiplications. Second, low-rank approximation decomposes the weight matrix of a convolutional or fully connected layer to a set of low-rank filters that can be evaluated with low computational cost [114]. Finally, because fixed-point arithmetic requires less computational resources than floating-point arithmetic, feature maps, weight matrices and/or convolutional kernels can be quantized by using a fixed-point representation to further reduce the computational cost.

A straightforward approach is to encode each numerical value with the desired word-length according to its range [114,121]. Alternatively, a dynamic scheme may be adopted to assign different scaling factors to different numerical parameters within the same network [133,134]. When the dynamic quantization method is applied to both the convolutional and fully-connected layers of AlexNet without fine-tuning, the classification accuracy is almost unchanged (<1%) [135].

In the extreme case, a DNN may use binary weights and activations, resulting in an extremely compact representation that is referred to as binary neural network (BNN) [115,133,136,137]. A BNN can be evaluated with extremely low computational cost, as binary addition and multiplication can both be implemented with simple logic gates. Other than BNN, a ternary DNN [138] sets its weights to +1, 0 or −1, allowing each weight to be represented by 2 bits. On the other hand, the numerical operations in all neurons are implemented with floating-point arithmetic (FP32).

Table 3 summarizes the performance of several typical CNN models deployed on FPGA using different model optimization methods. CNN models based on quantized arithmetic are highly efficient in terms of hardware utilization and power consumption; however, their accuracy is often compromised. On the other hand, CNN models based on low-rank approximation (e.g. SVD) and pruning carry a smaller number of weights, while simultaneously achieving high classification accuracy. The ternary ResNet [115], implemented with Intel StratixTM 10 FPGA [140], achieves a throughput of 12 TGOP/s, outperforming the throughput of Titan X Pascal GPU by 10%.

To make a comprehensive comparison of the state-of-the-art hardware accelerators, we present the key performance metrics of several mainstream accelerators (i.e. CPU, GPU and FPGA) with different network models (i.e. VGG and ResNet) in Table 4 [141]. The FPGA cluster in Table 4 is composed of 15 FPGA chips, as described in Ref. [142]. Two important observations can be made from the data in Table 4. First, the throughput of FPGA is substantially higher than that of CPU, but it is often lower than the throughput of GPU. Second, among FPGA, CPU and GPU, FPGA offers the highest energy efficiency.

3.3. Application-specific hardware accelerators

A typical computer system is often heterogeneous, composed of a

Table 3
Performance comparison of FPGA-based CNN accelerators.

CNN Model	FPGA Device	Optimization Method	Accuracy (Top-5)	# of Param (M)	Computation (GOP)	Precision	Frequency (MHz)	Throughput (GOP/s)	Power (W)
VGG-19 [139]	Arria10 GX1150	–	90.1%	138	30.8	float32	370	866	41.7
VGG-16 [114]	Zynq 7Z045	SVD	87.96%	50.2	30.5	fixed16	150	137	9.6
VGG-16 [134]	Arria10 GX1150	Dynamic	88.1%	138	30.8	fixed8	150	645	–
BNN: XNOR-Net [137]	Stratix5 GSD8	Binary	66.8%	87.1	2.3	fixed1	150	1,964	26.2
Ternary ResNet [115]	Stratix10	Ternary, pruning	79.7%	61	1.4	float32	500	12,000	141.2

Table 4
Performance comparison of neural networks on difference hardware platforms [141].

Model	Platform	Device	Precision	Frequency. (MHz)	Throughput (GOP/s)	Energy Efficiency (GOP/j)	Power (W)
VGG-19	CPU	Xeon E5-2650v2	float32	2,600	119	0.63	95
	GPU	GTX TITAN X	float32	1,002	1,704	6.82	250
	Cluster w/15 FPGAs	XC7 VX690T	fixed16	–	1,220[a]	38.13	–
VGG-16	Cluster w/15 FPGAs	XC7 VX690T	fixed16	–	1,197[a]	37.88	–
	FPGA	Stratix-V GSD8	fixed8	120	117.8	19.1	–
ResNet-152	CPU	Xeon E5-2650v2	float32	2,600	119	0.63	95
	GPU	GTX TITAN	float32	1,002	1,661	0.60	250
	FPGA	Stratix 10 GX 2800	fixed8/16	300	789.44	–	–
	FPGA	Arria10 GX1150	fixed8/16	240	697.09	–	–
	FPGA	Arria10 GX1150	float16	150	315.5	–	–
ReNet-50	FPGA	Arria10 GX1150	float16	150	285.07	–	–
	FPGA	Arria10 GX1150	fixed8/16	240	599.61	–	–

[a] Measured throughput value of a single FPGA.

heterogeneous suit of processors, such as CPUs added with other dissimilar processors, to meet the specific computing requirement. Processors that complement CPUs are known as application-specific coprocessors. In addition to the notable coprocessors such as GPUs and FPGAs, there are several specialized hardware units in the form of either stand-alone devices or coprocessors that are particularly developed for deep learning and/or other AI applications.

Tensor processing unit (TPU) [143,144], a customized ASIC developed by Google, is a stand-alone device specifically designed for neural networks and tailored for the Google Tensorflow framework [108]. TPU targets a high volume of low-precision (e.g., 8-bit) arithmetic. It has already powered many applications at Google, such as the search engine and AlphaGo [144]. Intel Nervana neural network processor (NNP) [145] is designed to provide the required flexibility of deep learning primitives while making its core hardware components as efficient as possible. Mobileye EyeQ [146] is a family of SoC devices specialized for vision processing in autonomous driving. It shows the ability of handling complex and computationally intensive vision tasks, while maintaining low power consumption.

Various AI coprocessors for mobile platforms are developed recently, where an arms race seems around the corner. The mobile processor Qualcomm Snapdragon 845 contains a Hexagon 685 DSP core that supports sophisticated, on-device AI processing in camera, voice and gaming applications [147]. Imagination Technologies [148] develops a series of neural network accelerators (NNAs), such as PowerVR Series 2NX/3NX NNA. They are intellectual property (IP) cores that are designed to deliver high performance computation and low power consumption for embedded and mobile devices. The new "neural engine" by Apple [149], incorporating Apple A11/A12 Bionic SoC, is a pair of processing cores dedicated for specific machine learning algorithms, including Face ID, augmented reality, etc. HiSilicon Kirin 970 is the first mobile AI platform developed by HUAWEI [150]. With a dedicated neural processing unit (NPU), its new heterogeneous computing architecture improves the throughput and energy efficiency by up to 25 × and 50 × respectively over a quad-core Cortex-A73 CPU cluster.

4. Conclusions

As a scientific discipline, computer vision has been a challenging research area and received significant attention. With the emergence of big data, advanced deep learning algorithms and powerful hardware accelerators, modern computer vision systems have dramatically evolved. In this paper, we conduct a comprehensive survey on computer vision techniques. Specially, we have highlighted the recent accomplishments in both the algorithms for a variety of computer vision tasks such as image classification, object detection and image segmentation, and the promising hardware platforms to implement DNNs efficiently for practical applications, such as GPUs, FPGAs and other new generation of hardware accelerators.

In the future, increasingly compact and efficient DNNs are needed for real-time and embedded applications. In addition, weakly supervised or unsupervised DNN schemes must be investigated to perceive all object categories in all open world scenes. Furthermore, highly energy-efficient hardware engines are required to extend the existing accelerators to a broad spectrum of challenging scenarios. To address the aforementioned grand challenges, massive innovations of computer vision systems, in terms of both algorithm developments and hardware designs, are expected over the next five or even ten years.

References

[1] James Kobielus, Powering AI: the Explosion of New AI Hardware Accelerator, InfoWord, 2018 [Online]. Available: https://www.infoworld.com/article/3290104/powering-ai-the-explosion-of-new-ai-hardware-accelerators.html.

[2] Computer vision, WikiPedia. [Online]. Available: https://en.wikipedia.org/wiki/Computer_vision.

[3] Richard Szeliski, Computer Vision: Algorithms and Applications, Springer Science & Business Media, 2010.

[4] Wilson Geisler, Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, Psyccritiques, 1983, pp. 581–582.

[5] Laura McClure, Building an AI with the Intelligence of a Toddler: Fei-Fei Li at TED2015, TED, 2015 [Online]. Available: https://blog.ted.com/building-an-ai-with-the-intelligence-of-a-toddler-fei-fei-li-at-ted2015/.

[6] Deng Jia, et al., Imagenet: a large-scale hierarchical image database, in: Conference on Computer Vision and Pattern Recognition, 2009.

[7] M. Sornam, Kavitha Muthusubash, V. Vanitha, A survey on image classification and activity recognition using deep convolutional neural network architecture, in: International Conference on Advanced Computing, 2017.

[8] Li Liu, et al., Deep Learning for Generic Object Detection: a Survey, arXiv: 1809.02165, 2018.

[9] Alberto Garcia-Garcia, et al., A Review on Deep Learning Techniques Applied to Semantic Segmentation, arXiv:1704.06857, 2017.

[10] Sparsh Mittal, A survey of FPGA-based accelerators for convolutional neural networks, Neural Comput. Appl. (2018) 1–31.

[11] Alex Krizhevsky, Geoffrey Hinton, Technical Report, Learning Multiple Layers of Features from Tiny Images, vol. 1, University of Toronto, 2009. No. 4.

[12] Yann LeCun, et al., Gradient-based learning applied to document recognition, Proc. IEEE (1998) 2278–2324.

[13] David Lowe, Object recognition from local scale-invariant features, in: International Conference on Computer Vision, Vol.2, 1999.

[14] Navneet Dalal, Bill Triggs, Histograms of oriented gradients for human detection, in: Conference on Computer Vision and Pattern Recognition, vol. 1, 2005.

[15] Aude Oliva, Antonio Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vis. (2001) 145–175.

[16] Fei-Fei Li, Pietro Perona, A bayesian hierarchical model for learning natural scene categories, in: Conference on Computer Vision and Pattern Recognition, 2005.

[17] Kostas Daniilidis, Patros Maragos, Nikos Paragios, Improving the Fisher kernel for large-scale image classification, in: European Conference on Computer Vision, 2010, pp. 143–156.

[18] Corinna Cortes, Vladimir Vapnik, Support-vector Networks, Machine Learning, 1995, pp. 273–297.

[19] Jianchao Yang, et al., Linear spatial pyramid matching using sparse coding for image classification, in: Conference on Computer Vision and Pattern Recognition, 2009.

[20] Jorge Sánchez, Florent Perronnin, High-dimensional signature compression for large-scale image classification, in: Conference on Computer Vision and Pattern Recognition, 2011.

[21] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.

[22] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556, 2014.

[23] Christian Szegedy, et al., Going deeper with convolutions, in: Conference on Computer Vision and Pattern Recognition, 2015.

[24] Kaiming He, et al., Deep residual learning for image recognition, in: Conference on Computer Vision and Pattern Recognition, 2016.

[25] Gao Huang, et al., Densely connected convolutional networks, in: Conference on Computer Vision and Pattern Recognition, 2017.

[26] Jie Hu, Li Shen, Gang Sun, Squeeze-and-excitation networks, in: Conference on Computer Vision and Pattern Recognition, 2017.

[27] Barret Zoph, et al., Learning transferable architectures for scalable image recognition, in: Conference on Computer Vision and Pattern Recognition, 2018.

[28] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, in: International Conference on Learning Representations, 2017.

[29] Forrest Iandola, et al., Squeezenet: Alexnet-Level Accuracy with 50x Fewer Parameters and < 0.5 Mb Model Size, arXiv:1602.07360, 2016.

[30] Andrew Howard, et al., Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv:1704.04861, 2017.

[31] Xiangyu Zhang, et al., ShuffleNet: an extremely efficient convolutional neural network for mobile devices, in: Conference on Computer Vision and Pattern Recognition, 2018.

[32] Bichen Wu, et al., Shift: a zero flop, zero parameter alternative to spatial convolutions, in: Conference on Computer Vision and Pattern Recognition, 2018.

[33] Weijie Chen, et al., All you need is a few shifts: designing efficient convolutional neural networks for image classification, in: Conference on Computer Vision and Pattern Recognition, 2019.

[34] Song Han, Huizi Mao, William J. Dally, Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, in: International Conference on Learning Representations, 2016.

[35] Shuo Yang, et al., From facial parts responses to face detection: a deep learning approach, in: International Conference on Computer Vision, 2015.

[36] Bart Thomee, et al., YFCC100M: the New Data in Multimedia Research, arXiv: 1503.01817, 2015.

[37] Aditya Khosla, et al., Novel dataset for fine-grained image categorization: stanford dogs, in: Conference on Computer Vision and Pattern Recognition Workshop on Fine-Grained Visual Categorization, 2011.

[38] K. Soomro, A.R. Zamir, M. Shah, UCF101: a dataset of 101 human actions classes from videos in the wild, arXiv:1212.0402 (2012).

[39] TREC Video Retrieval Evaluation Multimedia event detection. [Online]. Available: https://trecvid.nist.gov/.

[40] Herve Jegou, et al., Aggregating local descriptors into a compact image representation, in: Conference on Computer Vision and Pattern Recognition, 2010.

[41] Zhongwen Xu, Yi Yang, G. Alex, Hauptmann, A discriminative CNN video representation for event detection, in: Conference on Computer Vision and Pattern Recognition, 2015.
[42] Hongteng Xu, Y. Zhen, H. Zha, Trailer generation via a point process-based visual attractiveness model, in: International Joint Conference on Artificial Intelligence, 2015.
[43] Du Tran, et al., Learning spatiotemporal features with 3D convolutional networks, in: International Conference on Computer Vision, 2015.
[44] Karen Simonyan, Andrew Zisserman, Two-stream convolutional networks for action recognition in videos, Adv. Neural Inf. Process. Syst. (2014) 568–576.
[45] Jeff Donahue, et al., Long-term recurrent convolutional networks for visual recognition and description, Trans. Pattern Anal. Mach. Intell. (2014) 677–691.
[46] Zuxuan Wu, et al., Modeling spatial-temporal clues in a hybrid deep learning framework for video classification, in: International Conference on Multimedia, 2015.
[47] Shikhar Sharma, Kiros Ryan, Ruslan Salakhutdinov, Action Recognition Using Visual Attention, arXiv:1511.04119, 2015.
[48] Z. Li, et al., Video LSTM convolves, attends and flows for action recognition, Comput. Vis. Image Understand. 166 (2018) 41–50.
[49] Pedro F. Felzenszwalb, et al., Object detection with discriminatively trained part-based models, Trans. Pattern Anal. Mach. Intell. (2010) 1627–1645.
[50] Viola Paul, Michael J. Jones, Robust real-time face detection, Int. J. Comput. Vis. (2004) 137–154.
[51] Timo Ahonen, Abdenour Hadid, Matti Pietikainen, Face description with local binary patterns: application to face recognition, Trans. Pattern Anal. Mach. Intell. (2006) 2037–2041.
[52] Mark Everingham, et al., The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. (2010) 303–338.
[53] Tsung-Yi Lin, et al., Microsoft coco: common objects in context, in: European Conference on Computer Vision, 2014.
[54] Ross Girshick, et al., Rich feature hierarchies for accurate object detection and semantic segmentation, in: Conference on Computer Vision and Pattern Recognition, 2014.
[55] Jasper Uijlings, et al., Selective search for object recognition, Int. J. Comput. Vis. (2013) 154–171.
[56] Kaiming He, et al., Spatial pyramid pooling in deep convolutional networks for visual recognition, Trans. Pattern Anal. Mach. Intell. (2015) 1904–1916.
[57] Ross Girshick, Fast R-CNN, in: International Conference on Computer Vision, 2015.
[58] Shao Ren, Kai He, Ross Girshick, et al., Faster R-CNN: towards real-time object detection with region proposal networks, Trans. Pattern Anal. Mach. Intell. (2017) 1137–1149.
[59] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al., Mask R-CNN, in: International Conference on Computer Vision, 2017.
[60] Redmon Joseph, Santosh Divvala, Ross Girshick, et al., You only look once: unified, real-time object detection, in: Conference on Computer Vision and Pattern Recognition, 2016.
[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al., SSD: single shot multibox detector, in: European Conference on Computer Vision, 2016.
[62] Redmon Joseph, Farhadi Ali, YOLO9000: better, faster, stronger, in: Conference on Computer Vision and Pattern Recognition, 2017.
[63] Redmon Joseph, Farhadi Ali, Yolov3: an Incremental Improvement, arXiv: 1804.02767, 2018.
[64] Hu Han, et al., Relation networks for object detection, in: Conference on Computer Vision and Pattern Recognition, 2018.
[65] Yancheng Bai, et al., SOD-MTGAN: small object detection via multi-task generative adversarial network, in: European Conference on Computer Vision, 2018, pp. 8–14.
[66] Yancheng Bai, et al., Finding tiny faces in the wild with generative adversarial network, in: Conference on Computer Vision and Pattern Recognition, 2018.
[67] Alex Bewley, et al., Simple online and realtime tracking with a deep association metric, in: International Conference on Image Processing, 2017.
[68] Alex Bewley, et al., Simple online and realtime tracking, in: International Conference on Image Processing, 2016.
[69] Xizhou Zhu, et al., Flow-Guided feature aggregation for video object detection, in: International Conference on Computer Vision, 2017.
[70] Xizhou Zhu, et al., Deep feature flow for video recognition, in: Conference on Computer Vision and Pattern Recognition, 2017.
[71] Jonathan Huang, et al., Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors, 2017 [Online]. Available: http://image-net.org/challenges/talks_2017/Imagenet2017VID.pdf.
[72] Ron Ohlander, Keith Price, D. Raj Reddy, Picture segmentation using a recursive region splitting method, Comput. Graph. Image Process. (1978) 313–333.
[73] Pedro F. Felzenszwalb, Daniel P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vis. (2004) 167–181.
[74] Wenxian Yang, et al., User-friendly interactive image segmentation through unified combinatorial user inputs, Trans. Image Process. (2010) 2470–2479.
[75] Alberto Garcia-Garcia, et al., A Review on Deep Learning Techniques Applied to Semantic Segmentation, arXiv:1704.06857, 2017.
[76] Mark Everingham, et al., The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. (2015) 98–136.
[77] Marius Cordts, et al., The cityscapes dataset for semantic urban scene understanding, in: Conference on Computer Vision and Pattern Recognition, 2016.
[78] Gerhard Neuhold, et al., The mapillary vistas dataset for semantic understanding of street scenes, in: International Conference on Computer Vision, 2017.
[79] R. Giuly, M. Martone, M. Ellisman, Method: automatic segmentation of mitochondria utilizing patch classification, contour pair classification, and automatically seeded level sets, BMC Bioinf. 13 (1) (2012) 29.
[80] Holger Roth, et al., Deeporgan: multi-level deep convolutional networks for automated pancreas segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015.
[81] Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully convolutional networks for semantic segmentation, in: Conference on Computer Vision and Pattern Recognition, 2015.
[82] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, Segnet: a deep convolutional encoder-decoder architecture for image segmentation, Trans. Pattern Anal. Mach. Intell. (2017) 2481–2495.
[83] Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015.
[84] Guosheng Lin, et al., RefineNet: multi-path refinement networks for high-resolution semantic segmentation, in: Conference on Computer Vision and Pattern Recognition, 2017.
[85] Liang Chieh Chen, et al., Semantic image segmentation with deep convolutional nets and fully connected CRFs, Comput. Sci. (2014) 357–361.
[86] Shuai Zheng, et al., Conditional random fields as recurrent neural networks, in: International Conference on Computer Vision, 2015.
[87] Liang-Chieh Chen, et al., Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, Trans. Pattern Anal. Mach. Intell. (2018) 834–848.
[88] Yu Fisher, Vladlen Koltun, Multi-scale Context Aggregation by Dilated Convolutions, arXiv:1511.07122, 2015.
[89] Hengshuang Zhao, et al., Pyramid scene parsing network, in: Conference on Computer Vision and Pattern Recognition, 2017.
[90] Liang-Chieh Chen, et al., Encoder-decoder with atrous separable convolution for semantic image segmentation, in: European Conference on Computer Vision, 2018.
[91] François Chollet, Xception: deep learning with depthwise separable convolutions, in: Conference on Computer Vision and Pattern Recognition, 2017.
[92] P.O. Pinheiro, R. Collobert, P. Dollár, Learning to segment object candidates, Adv. Neural Inf. Process. Syst. (2015) 1990–1998.
[93] Jifeng Dai, et al., Instance-sensitive fully convolutional networks, in: European Conference on Computer Vision, 2016.
[94] J. Dai, et al., R-FCN: object detection via region-based fully convolutional networks, Adv. Neural Inf. Process. Syst. (2016) 379–387.
[95] Shu Liu, et al., Path aggregation network for instance segmentation, in: Conference on Computer Vision and Pattern Recognition, 2018.
[96] Jeff Dean, Keynote: Recent Advances in Artificial Intelligence via Machine Learning and the Implications for Computer System Design, Hotchips, 2017.
[97] Shaaban, "Advanced computer architecture: digital signal processing (DSP): architecture & processors," Lecture EECC722, [Online]. Available:http://meseec.ce.rit.edu/eecc722-fall2012/722-10-10-2012.pdf.
[98] Hardware acceleration, WikiPedia. [Online]. Available: https://en.wikipedia.org/wiki/Hardware_acceleration.
[99] S. Mittal, J.S. Vetter, A survey of CPU-GPU heterogeneous computing techniques, ACM Comput. Surv. 47 (4) (2015) 69.
[100] Graphics processing unit, WikiPedia. [Online]. Available: https://en.wikipedia.org/wiki/Graphics_processing_unit.
[101] Michael Garland, et al., Parallel computing experiences with CUDA, IEEE Micro (2008) 13–27.
[102] John E. Stone, D. Gohara, G. Shi, OpenCL: a parallel programming standard for heterogeneous computing systems, Comput. Sci. Eng. (2010) 66–73.
[103] Magnus Halvorsen, Hardware Acceleration of Convolutional Neural Networks, MS thesis, Norwegian University of Science Technology, 2015.
[104] Sharan Chetlur, et al., CUDNN: Efficient Primitives for Deep Learning, arXiv: 1410.0759, 2014.
[105] Alex Krizhevsky, Cudaconvet2. [Online]. Available: https://code.google.com/archive/p/cuda-convnet2/.
[106] Yangqing Jia, et al., Caffe: convolutional architecture for fast feature embedding, in: International Conference on Multimedia, 2014.
[107] Ronan Collobert, Koray Kavukcuoglu, Clément Farabet, Torch7: a matlab-like environment for machine learning, in: Conference on Neural Information Processing System Workshop, 2011.
[108] TensorFlow, URL: https://www.tensorflow.org/.
[109] Grigory Sapunov, Hardware for Deep Learning, Intento, 2018 [Online]. Available: https://blog.inten.to/hardware-for-deep-learning-current- state-and-trends-51c01ebbb6dc.
[110] PASCAL GPU Architecture, URL: https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture/.
[111] Eugenio Culurciello, Computation and Memory Bandwidth in Deep Neural Networks, A Medium Corporation, 2017 [Online]. Available: https://medium.com/@culurciello/computation-and-memory-bandwidth-in-deep-neural-networks-16cbac63ebd5.
[112] NVIDIA Collective Communications Library (NCCL), URL: https://developer.nvidia.com/nccl.
[113] Kalin Ovtcharov, et al., Accelerating Deep Convolutional Neural Networks Using Specialized Hardware, Microsoft Research Whitepaper, 2015, pp. 1–4.
[114] Jiantao Qiu, et al., Going deeper with embedded fpga platform for convolutional neural network, in: International Symposium on Field-Programmable Gate Arrays, 2016.

[115] Eriko Nurvitadhi, et al., Can FPGAs beat GPUs in accelerating next-generation deep neural networks?, in: International Symposium on Field-Programmable Gate Arrays, 2017.
[116] Ritchie Zhao, et al., Accelerating binarized convolutional neural networks with software-programmable FPGAs, in: International Symposium on Field-Programmable Gate Arrays, 2017.
[117] Jeremy Bottleson, et al., CLCAFFE: OpenCL accelerated CAFFE for convolutional neural networks, in: International Parallel and Distributed Processing Symposium Workshops, 2016.
[118] Andrew Lavin, Scott Gray, Fast algorithms for convolutional neural networks, in: Conference on Computer Vision and Pattern Recognition, 2016.
[119] Shmuel Winograd, Arithmetic Complexity of Computations, vol. 33, Society for Industrial and Applied Mathmatics, 1980.
[120] Roberto DiCecco, et al., Caffeinated FPGAs: FPGA Framework for Convolutional Neural Networks, Field-Programmable Technology, 2016.
[121] Murugan Sankaradas, et al., A Massively Parallel Coprocessor for Convolutional Neural Networks, Application-specific Systems, Architectures and Processors, 2009.
[122] Srimat Chakradhar, et al., A dynamically configurable coprocessor for convolutional neural networks, Comput. Architect. News (2010) 247–257.
[123] Clément Farabet, et al., Neuflow: a runtime reconfigurable dataflow processor for vision, in: Conference on Computer Vision and Pattern Recognition, 2011.
[124] Chen Zhang, et al., Optimizing FPGA-based accelerator design for deep convolutional neural networks, in: International Symposium on Field-Programmable Gate Arrays, 2015.
[125] Atul Rahman, et al., Design Space Exploration of FPGA Accelerators for Convolutional Neural Networks, Design, Automation & Test in Europe, 2017.
[126] Y. Li, et al., A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks, J. Emerg. Technol. Comput. Syst. 14 (2) (2018) 18.
[127] Steven Derrien, Sanjay Rajopadhye, Loop tiling for reconfigurable accelerators, in: Conference on Field Programmable Logic and Applications, 2001.
[128] Baoyuan Liu, et al., Sparse convolutional neural networks, in: Conference on Computer Vision and Pattern Recognition, 2015.
[129] Xiaofan Zhang, et al., High-performance video content recognition with long-term recurrent convolutional network for FPGA, in: Conference on Field Programmable Logic and Applications, 2017.
[130] Tien-Ju Yang, Yu-Hsin Chen, Vivienne Sze, Designing energy-efficient convolutional neural networks using energy-aware pruning, in: Conference on Computer Vision and Pattern Recognition, 2017.
[131] P. Adam, et al., SPARCNet: a hardware accelerator for efficient deployment of sparse convolutional networks, J. Emerg. Technol. Comput. Syst. 13 (3) (2017) 31.
[132] Roberto Rigamonti, et al., Learning separable filters, in: Conference on Computer Vision and Pattern Recognition, 2013.
[133] Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David, Training Deep Neural Networks with Low Precision Multiplications, arXiv:1412.7024, 2014.
[134] Yufei Ma, et al., Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks, in: International Symposium on Field-Programmable Gate Arrays, 2017.
[135] Naveen Suda, et al., Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks, in: International Symposium on Field-Programmable Gate Arrays, 2016.
[136] Matthieu Courbariaux, et al., Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained To+ 1 Or-1, arXiv: 1602.02830, 2016.
[137] Shuang Liang, et al., FP-BNN: binarized neural network on FPGA, Neurocomputing (2018) 1072–1086.
[138] Fengfu Li, B. Zhang, B. Liu, Ternary Weight Networks, arXiv:1605.04711vol. 2, 2016.
[139] Jialiang Zhang, Li Jing, Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network, in: International Symposium on Field-Programmable Gate Arrays, 2017.
[140] Intel FPGA, Intel® Stratix® 10 Variable Precision DSP Blocks User Guide, Technical report, Intel FPGA Group, 2017.
[141] Teng Wang, et al., A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities, arXiv:1901.04988, 2018.
[142] Tong Geng, et al., A framework for acceleration of CNN training on deeply-pipelined FPGA clusters with work and weight load balancing, in: Conference on Field Programmable Logic and Applications, 2018.
[143] Joe Osborne, Google's Tensor Processing Unit Explained: This Is what the Future Computing Looks like, TechRadar, 2016 [Online]. Available: https://www.techradar.com/news/computing-components/processors/google-s-tensor-processing-unit-explained-this-is-what-the-future-of-computing-looks-like-1326915.
[144] Norm Jouppi, Google Supercharges Machine Learning Tasks with TPU Custom Chip, Google Could, 2017 [Online]. Available: https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip.
[145] Naveen Rao, Intel Nervana Neural Network Processor (NNP) Redefine AI Silicon, Intel Website, 2017 [Online]. Available: https://www.intel.ai/intel-nervana-neural-network-processors-nnp-redefine-ai-silicon/#gs.1s250i.
[146] MobilEYE, The Evolution of EyeQ, Mobileye website, 2018 [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq- chip/.
[147] Qualcomm, "Snapdragon 845 Mobile Platform," URL: https://www.qualcomm.com/products/snapdragon-845-mobile-platform.
[148] Imagination Technology, "PowerVR vision & AI," URL: https://www.imgtec.com/vision-ai/.
[149] James Vincent, The iPhone X's New Neural Engine Exemplifies Apple's Approach to AI, The Verge, 2017 [Online]. Available: https://www.theverge.com/2017/9/13/16300464/apple-iphone-x-ai-neural-engine.
[150] HUAWEI, HUAWEI Reveals the Future of Mobile AI and IFA 2017, HUAWEI website, 2017 [Online]. Available: https://www.huawei.com/en/press-events/news/2017/9/mobile-ai-ifa-2017.