Chapter 6

Efficient Deep Learning in Network Compression and Acceleration

Shiming Ge

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79562

Abstract

While deep learning delivers state-of-the-art accuracy on many artificial intelligence tasks, it comes at the cost of high computational complexity due to large parameters. It is important to design or develop efficient methods to support deep learning toward enabling its scalable deployment, particularly for embedded devices such as mobile, Internet of things (IOT), and drones. In this chapter, I will present a comprehensive survey of several advanced approaches for efficient deep learning in network compression and acceleration. I will describe the central ideas behind each approach and explore the similarities and differences between different methods. Finally, I will present some future directions in this field.

Keywords: deep learning, deep neural networks, network compression, network acceleration, artificial intelligence

1. Introduction

With the rapid development of modern computing power and large data collection technique, deep neural networks (DNNs) have pushed artificial intelligence limits in a wide range of inference tasks, including but not limited to visual recognition [1], face recognition [2], speech recognition [3], and Go game [4]. For example, visual recognition method proposed in [5] achieves 3.57% top-5 test error on the ImageNet LSVRL-2012 classification dataset, while face recognition system [6] achieves over 99.5% accuracy on the public face benchmark LFW [7], which both have surpassed human-level performance (5.1% on ImageNet [8] and 97.53% on LFW [9], respectively).

These powerful methods usually rely on DNNs containing millions or even billions of parameters. For example, the "very deep" VGG-16 [10], which achieves very impressive performance on ImageNet LSVRC 2014, uses a 16-layer deep network containing 138 million parameters and takes more than 500 MB in storing the model. Beyond the remarkable performance, there is increasing concern that the larger number of parameters consumes considerable resources (e.g., storage, memory, and energy), which hinders their practical deployment. First, for a deep neural network (DNN) usage on mobile, the storage bandwidth is very critical both for model size and data computation. For example, the mobile-first companies (such as Facebook and Baidu) are very care about the sizes of the uploaded file, while mobile sensor data companies (such as Google and Microsoft) usually build largely cloud powered systems with limited mobile computation. Second, for a DNN usage in cloud, memory bandwidth demand is very important to save transmission and power. Therefore, smaller models via DNN compression at least mean that they (1) are easier to download from App Store, (2) need less bandwidth to update to an autonomous car, (3) are easier to deploy on embedded hardware with limited memory, (4) need less communication across servers during distributed training, and (5) need less energy cost to perform face recognition.

The objective of efficient methods is to improve the efficiency of deep learning through smaller model size, higher prediction accuracy, faster prediction speed, and lower power consumption. Toward this end, a feasible solution is performing model compression and acceleration to optimized well-trained networks. In this chapter, I will first introduce some background of deep neural networks in Section 2, which provides us the motivation toward efficient algorithms. Then, I will present a comprehensive survey of recent advanced approaches for efficient deep learning in network compression and acceleration, which are mainly grouped into five categories, including network pruning category in Section 3, network quantization category in Section 4, network parameter structuring category in Section 5, network distillation category in Section 6, and compact network design category in Section 7. After that, I will discuss some future directions in this field in Section 8. Finally, Section 9 gives the conclusion.

2. Background

In this section, a brief introduction and analysis are given with some classic networks as examples on the structure of deep networks, computation and storage complexity, weight distribution, and memory bandwidth. This analysis inspires the behind motivation of model compression and acceleration approaches.

Recently, deep convolutional neural networks (CNNs) have become very popular due to their powerful representational capacity. A deep convolutional neural network (CNN) usually has a hierarchical structure of a number of layers, containing multiple blocks of convolutional layers, activation layers, and pooling layers, followed by multiple fully connected layers. **Figure 1** gives the structures of two classic CNNs, where (a) AlexNet [1] and (c) VGG-16 [10] consist of eight and sixteen layers, respectively. The two networks are larger than 200 MB and 500 MB, which makes them difficult to deploy on mobile devices. The convolutional layers dominate most of the computational complexity since they need a lot of multiplication-and-addition

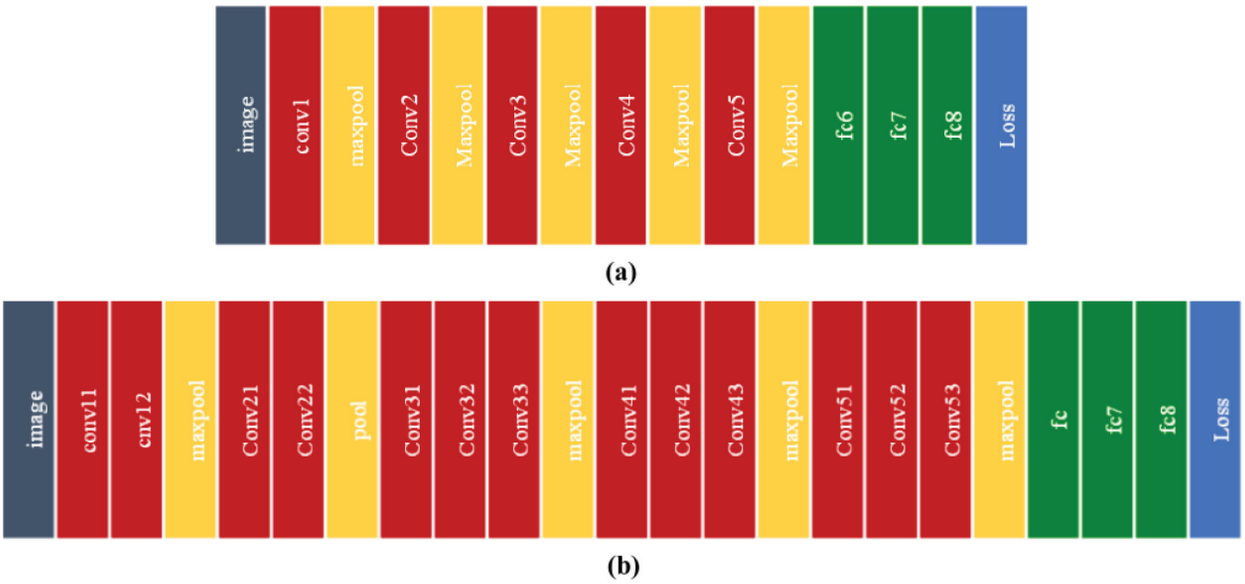

Figure 1. The structures of two classic deep networks. AlexNet (a) and VGGNet (b) all contain multiple convolutional layers (red), activation layers, and pooling layers (yellow), followed by multiple fully connected layers (green). The input and loss layer are marked in mazarine and blue, respectively.

(MAC) operations to extract local pattern, while they contain less weights due to weight sharing and local connectivity. By contrast, fully connected layers contain most of the weights since dense matrix-vector multiplications are very resource-intense. In addition, an activation layer (such as ReLU) contains a nonlinear function to activate or suppress some neurons. It can make the network more sparse and robust again to over-fitting while reducing the number of connections. A pooling layer is followed by a convolutional layer and aims to merge semantically similar features to reduce the memory.

As shown in **Table 1**, the complexity of CNNs could be spitted into two parts: (1) the computational complexity of a CNN is dominated by the convolutional layers and (2) the number of parameters is mainly related to the fully connected layers. Therefore, most model acceleration approaches focus on decreasing the computational complexity of the convolutional layers, while the model compression approaches mainly try to compress the parameters of the fully connected layers.

Network	Computational complexity			Parameter complexity		
	MACs	Conv (%)	FC (%)	Size	Conv (%)	FC (%)
AlexNet	724 M	91.9	8.1	61 M	3.8	96.2
VGG-16	15.5G	99.2	0.8	138 M	10.6	89.4
GoogleNet	1.6G	99.9	0.1	6.9 M	85.1	14.9
ResNet-50	3.9G	100	0	25.5 M	100	0

Table 1. The computational and parameter complexities and distributions for deep CNNs.

DNNs are known to be over-parameterized for facilitating convergence to good local minima of the loss function during model training [11]. Therefore, such redundancy can be removed from the trained networks in the test or inference time. Moreover, each layer contains lots of weights near zero value. **Figure 2** shows the probability distribution of the weights in two layers of AlexNet and VGG-16, respectively, where the weights are scaled and quantized into [−1, 1] with 32 levels to convenient visual display. It can be seen that the distribution is biased: most of the (quantized) weights on each layer are distributed around zero-value peak. This observation demonstrates that the weights can be reduced through weight coding, such as Huffman coding.

The memory bandwidth of a CNN model refers to the inference processing and greatly impacts the energy consumption, especially when running on embedded or mobile devices. To analyze the memory bandwidth of a trained CNN model, a simple but effective way is applied here by performing forward testing on multiple images and then analyzing the range of each layer output. The memory of each layer is dependent on bit width of each feature and

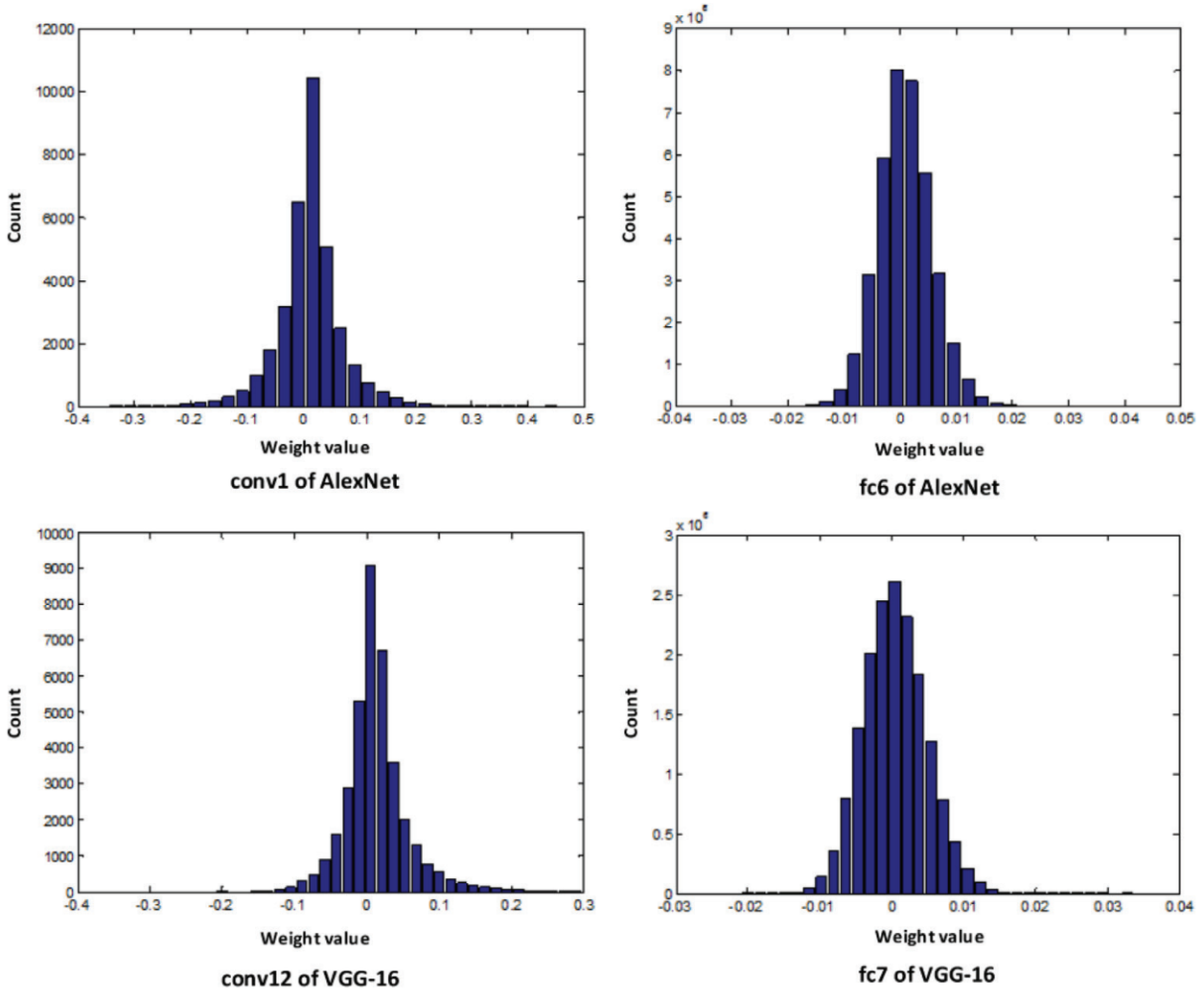

Figure 2. The probability distribution of weights in two layers of AlexNet and VGG-16. It is shown that the weight distribution is around a zero-value peak.

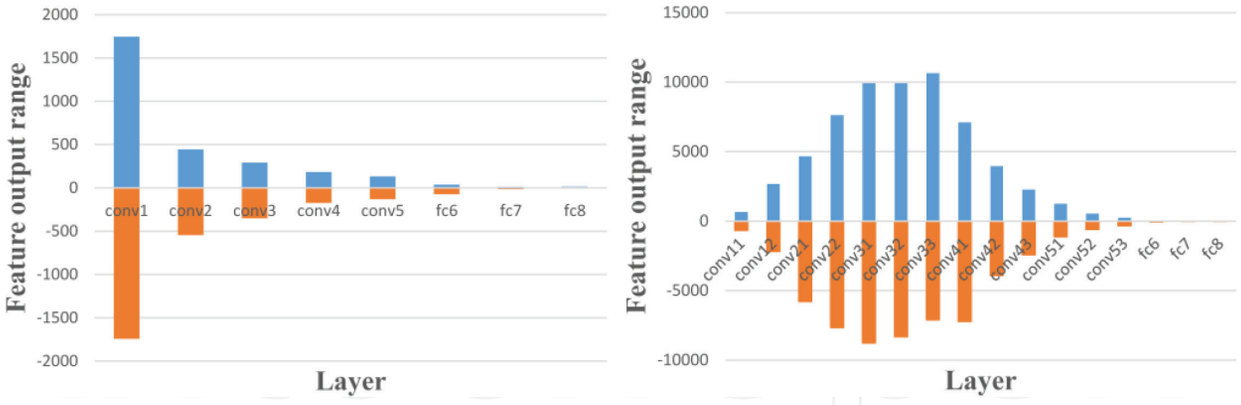

Figure 3. The memory bit-width range of each layer of AlexNet (left) and VGG-16 (right).

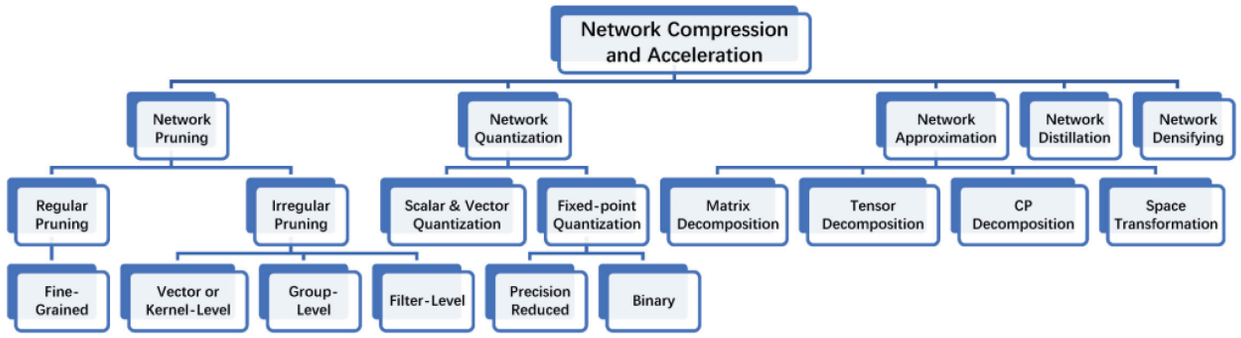

Figure 4. The main categories of network compression and acceleration approaches.

the number of output features. 1000 images from ImageNet dataset are randomly selected to perform inference with AlexNet and VGG-16, respectively; the mean range of output features on each layer are shown in **Figure 3**. It shows that the ranges of memory bandwidths in each layer are different and variable. Inspired by that, network compression and acceleration approaches can be designed to dynamically control the memory allocation in network layers by evaluating the ranges of each layer. Following these observations, many efficient methods for network compression and acceleration have been proposed, and several survey papers could be found in [12–14]. As shown in **Figure 4**, these approaches are grouped into five main categories according to their scheme for processing deep networks: pruning, quantization, approximation, distillation, and densification. In the following sections, I will introduce the advanced approaches in these categories.

3. Network pruning

DNNs are known to be over-parameterized for facilitating convergence to good local minima of the loss function during network training [11]. Therefore, the optimally trained deep networks usually contain redundancy on parameters. Inspired by that, network pruning category aims to remove such redundancy from the pre-trained networks in the inference time. In this

way, pruning approaches are applied to prune the unimportant or unnecessary parameters to significantly increase the sparsity of the parameters. Recently, many approaches are proposed, which consist of regular pruning approaches and irregular pruning approaches. As stated in [13], regular pruning refers to fine-gained pruning, while irregular pruning approaches are further categorized into four classes according to the pruning levels: vector level, kernel level, group level, and filter level. **Figure 5** shows different pruning methods. The core of network pruning is measuring the importance of weights or parameters.

Fine-grained pruning is the most popular approaches used in network pruning. It removes any unimportant parameters in convolutional kernels by using an irregular manner. In early work, LeCun et al. proposed optimal brain damage, a fine-grained pruning technique that estimates the saliency of the parameters by using the approximate second-order derivatives of the loss function w.r.t the parameters and then removes the parameters at a low saliency. This technique shows to work better than the naive approach. Later, Hassibi and Stork [15] came up with optimal brain surgeon, which performed much better than optimal brain damage although costing much more computational consumption. Recently, Chaber and Lawrynczuk [16] applied optimal brain damage for pruning recurrent neural models. Han et al. developed a method to prune unimportant connection and then retrain the weights to reduce storage and computation [17]. Later, they proposed a hybrid method, called Deep Compression [18], to compress deep neural networks with pruning, quantization, and Huffman coding. On the ImageNet dataset, the method reduced the storage required by AlexNet by 35× from 240 MB to 6.9 MB and VGG-16 by 49× from 552 MB to 11.3 MB both without loss of accuracy. Recently, Guo et al. [19] improved Deep Compression via dynamic network surgery which incorporated connection splicing into the whole process to avoid incorrect pruning. For face recognition, Sun et al. [20] proposed to iteratively learn sparse ConvNets. Instead of removing individual weights, Srinivas et al. [21] proposed to remove one neuron at a time. They presented a systematic way to remove the redundancy by wiring similar neurons together. In general, these irregular pruning approaches could achieve efficient compression of model sizes, but the memory footprint still has not been saved.

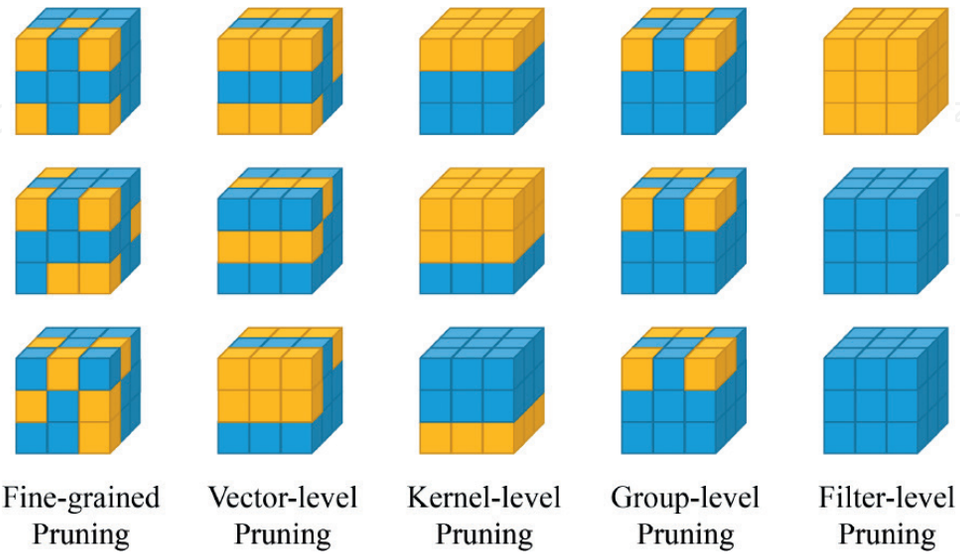

Figure 5. Different pruning methods for a convolutional layer that has three convolutional filters of size 3 × 3 × 3 [13].

Different from fine-grained pruning, vector-level and kernel-level pruning remove vectors in the convolutional kernels and 2D-convolutional kernels in the filters in a regular manner, respectively. Anwar et al. [22] proposed pruning a vector in a fixed stride via intra-kernel strided pruning. Mao et al. [23] explored different granularity levels in pruning. Group-level pruning aims to remove network parameters according to the same sparse pattern on the filters. In this way, convolutional computation can be efficiently implemented with reduced matrix multiplication. Lebedev and Lempitsky [24] revised brain damage-based pruning approach in a group-wise manner. Their approach added group-wise pruning to the training process to speed up the convolutional operations by using group-sparsity regularization. Similarly, Wen et al. [25] pruned groups of parameters by using group Lasso. Filter-level pruning aims to remove the convolutional filters or channels to thin the deep networks. Since the number of input channels is reduced after a filter layer is pruned, such pruning is more efficient for accelerating network inference. Polyak and Wolf proposed two compression strategies [26]: one based on eliminating lowly active channels and the other on coupling pruning with the repeated use of already computed elements. Luo et al. [27] proposed ThiNet to perform filter-level pruning. The pruning is guided by feature map, and the channels are selected by minimizing the construction error between two successive layers. Similarly, He et al. [28] applied an iterative two-step algorithm to prune filters by minimizing the feature maps. Generally speaking, these regular pruning (vector-level, kernel-level, group-level, and filter-level) approaches are more suitable for hardware implementations.

4. Network quantization

Typically, DNNs apply floating-point (such as 32-bit) precision for training and inference, which may lead to a large cost in memory, storage, and computation. To save the cost, network quantization category uses reduced precision to approximate network parameters. These approaches consist of scalar or vector quantization and fixed-point quantization (see **Figure 4**).

Scalar or vector quantization techniques are originally designed for data compression, where a codebook and a set of quantization codes are used to represent the original data. Considering that the size of codebook is much smaller than the original data, the original data could be efficiently compressed via quantization. Inspired by that, scalar or vector quantization approaches are applied to represent the parameters or weights of a deep network for compressing. In [29], Gong et al. applied k-means clustering to the weights or conducting product quantization and achieved a very good balance between model size and recognition accuracy. They achieved 16–24× compression of the network with only 1% loss of accuracy on ImageNet classification task. Wu et al. [30] proposed quantized CNN to simultaneously speedup the computation and reduce the storage and memory overhead of CNN models. This method obtains 4–6× speedup and 15–20× compression with 1% loss of accuracy on ImageNet. With the quantized CNN model, even mobile devices can accurately classify images within 1 second. Soulié et al. [31] proposed compressing deep network during the learning phase by adding an extra regularization term and combining product quantization of the network parameters.

Different from scalar and vector quantization approaches, fixed-point quantization approaches directly reduce the precision of parameters without codebooks. In [32], Dettmers proposed 8-bit approximation algorithms to make better use of the available bandwidth by compressing 32-bit gradients and nonlinear activations to 8-bit approximations, which obtains a speedup of 50×. In [33], Gupta et al. used only 16-bit wide fixed-point number representation when using stochastic rounding and incur little to no degradation in the classification accuracy. Lin et al. [34] proposed a fixed-point quantizer design by formulating an optimization problem to identify optimal fixed-point bit-width allocation across network layers. The approach offered larger than 20% reduction in the model size without performance loss, and the performance continued to improve after fine-tuning. Beyond fixed-point quantization with reduced precision, an alternative is using binary or ternary precision to lower the parameter representation. Soudry et al. [35] proposed Expectation Propagation (EP) algorithm to train multilayer neural networks. The algorithm has the advantages of parameter-free training and discrete weights, which are useful for large-scale parameter tuning and efficient training implementation on precision limited hardware, respectively. Courbariaux et al. [36] introduced BinaryConnect to provide deep neural network learning with binary weights. BinaryConnect acts as regularizer like other dropout schemes. The approach obtained near state-of-the-art results on permutation-invariant MNIST, CIFAR-10, and SVHN. Esser et al. [37] proposed training with standard backpropagation in binary precision by treating spikes and discrete synapses as continuous probabilities. They trained a sparse connected network running on the TrueNorth chip, which achieved a high accuracy of 99.42% on MNIST dataset with ensemble of 64 and 92.7% accuracy with ensemble of 1. Hubara et al. [38] introduced a method to train Binarized Neural Networks (BNNs) at run-time. In trained neural networks, both the weights and activations are binary. BNNs achieved near state-of-the-art results on MNIST, CIFAR-10, and SVHN. Moreover, BNNs achieved competitive results on the challenging ImageNet dataset (36.1%, top 1 using AlexNet) while drastically reducing memory consumption (size and number of accesses) and improving the speed of matrix multiplication at seven times. Later, Rastegari et al. [39] proposed XNOR-Net for ImageNet classification. They proposed two approximations to standard CNNs with binary-weight-networks and XNOR-Networks. The first approximation achieved 32× memory saving by replacing 32-bit floating-point weights with binary values. The second approximation enabled both the filters and the activations being binary. Moreover, it approximated convolutions using primarily binary operations. In this way, it achieved 58× faster convolutional operations and 32× memory savings while a much higher classification accuracy (53.8%, top 1 using AlexNet) than BNNs. Beyond the great reductions on network sizes and convolutional operations, these binarization schemes are based on simple matrix approximations and ignore the effect of binarization on the loss. To address this problem, Hou et al. [40] recently proposed a loss-aware binarization method by directly minimizing the loss w.r.t. the binarized weights with a proximal Newton algorithm with diagonal Hessian approximation. This method achieved good binarization performance and was robust for wide and deep networks. Motivated by local binary patterns (LBP), Xu et al. [41] proposed an efficient alternative to convolutional layers called local binary convolution (LBC) for facilitate binary network training. Compared to a standard convolutional layer, the LBC layer affords significant parameter savings, 9×–169× in the parameter numbers, as well as 9×–169× savings in model size. Moreover, the resulting CNNs with LBC layers

achieved comparable performance on a range of visual classification tasks, such as MNIST, SVHN, CIFAR-10, and ImageNet. Targeting at more faithful inference and better trade-off for practical applications, Guo et al. [42] introduced network sketching for pursuing binary-weight CNNs. They applied a coarse-to-fine model approximation by directly exploiting the binary structure in pre-trained filters and generated binary-weight models via tensor expansion. Moreover, an associative implementation of binary tensor convolutions was proposed to further speedup the generated models. After that, the resulting models outperformed the other binary-weight models on ImageNet large-scale classification task (55.2%, top 1 by using AlexNet). In order to reduce the accuracy loss or even improve accuracy, Zhu et al. [43] proposed Trained Ternary Quantization (TTQ) to reduce the precision of weights in neural networks to ternary values. TTQ trained the models from scratch with both ternary values and ternary assignment, while network inference only needed ternary values (2-bit weights) and scaling factors. The resulting models achieved an improved accuracy of 57.5%, top 1, using AlexNet on ImageNet large-scale classification task against full-precision model (57.2%, top 1, using AlexNet). TTQ was argued to be viewed as sparse binary-weight networks, which can potentially be accelerated with custom circuit. Generally speaking, the binary or ternary quantization approaches can greatly save the costs on model sizes, memory footprint, and computation, which make them friendly for hardware implementations. However, the accuracy needs to be improved especially in large-scale classification problems.

5. Network approximation

As stated in Section 2, the most computational cost of network inference comes from the convolution operators. In general, the convolutional kernel of a convolutional layer is represented with a 4D tensor, such as $K \in \mathbf{R}^{w \times h \times c \times s}$, where w and h are the width and height of the kernel filter, c is the number of input channels, and s indicates the target number of feature maps. The convolutional operation is performed by first transforming the kernel into a t-D ($t = 1,2,3,4$) tensor and then computed with efficient mathematical algorithm, such as by using Basic Linear Algebra Subprograms (BLAS). Inspired by that, network approximation aims to approximate the operation with low-rank decomposition.

Some approaches approximate 2D tensor by using singular value decomposition (SVD). Jaderberg et al. [44] decomposed the spatial dimension $w \times h$ into $w \times 1$ and $1 \times h$ filters, which achieved a 4.5× speedup for a CNN trained on a text character recognition dataset, with only an accuracy drop of 1%. Observing that the computation is dominated by the convolution operations in the lower layers of the network, Denton et al. [45] exploited the redundancy present within the convolutional filters to derive approximations that significantly reduce the required computation. The approach delivered 2× speedup on both CPU and GPU while keeping the accuracy within 1% of the original network on object recognition tasks. In [46], the authors proposed using a sparse decomposition to reduce the redundancy in model parameters. They obtained maximum sparsity by exploiting both interchannel and intrachannel redundancy and performing fine-tuning to minimize the recognition loss, which zeros out more than 90% of parameters and with a less than 1% loss of accuracy on the ImageNet.

Inspired by that the convolutional layer can be calculated with matrix-matrix multiplication; Figurnov et al. [47] used loop perforation technique to eliminate redundant multiplication, which allows to reduce the inference time by 50%.

By successive 2D tensor decompositions, 3D tensor decompositions can be obtained directly. Zhang et al. [48] applied the strategy that conducts a 2D decomposition on the first weight tensor after SVD. Their approach had been used to accelerate very deep networks for object classification and detection tasks. Another 3D tensor decomposition, Tucker decomposition [49], was proposed to compress deep CNNs for mobile applications by performing SVD along the input channel dimension for the first tensor after 2D decomposition. To further reduce complexity, a block-term decomposition [50] method based on low-rank and group sparse decomposition was proposed by approximating the original weight tensor by the sum of some smaller subtensors. By rearranging these subtensors, the block-term decomposition can be seen as a Tucker decomposition where the second decomposed tensor is a block diagonal tensor.

4D tensor decomposition can be obtained by exploring the low-rank property along the channel dimension and the spatial dimension. This is used in [51], and the decomposition is CP decomposition. The CP decomposition can achieve a very high speedup, for example, as 4.5× speedup for the second layer of AlexNet at only 1% accuracy drop.

Beyond low-rank tensor decomposition approaches which are performed in original space domain, there are some network approximation approaches by processing parameter approximation in transformation domain. In [52], Wang et al. proposed CNNPack to compress the deep networks in frequency domain. CNNPack treated convolutional filters as images and then decomposed their representations in the frequency domain as common parts shared by other similar filters and their individual private parts (i.e., individual residuals). In this way, a large number of low-energy frequency coefficients in both parts can be discarded to produce high compression without significantly compromising accuracy. Moreover, the computational burden of convolution operations in CNNs was relaxed by linearly combining the convolution responses of discrete cosine transform (DCT) bases. Later, Wang et al. [53] extended frequency domain method to the compression of feature maps. They proposed to extract intrinsic representation of the feature maps and preserve the discriminability of the features. The core is employing circulant matrix to formulate the feature map transformation. In this way, both online memory and processing time were reduced. Another transformation domain scheme is hashing, such as HashedNets [54] and FunHashNN [55].

In general, network approximation category focuses on accelerating network inference and reducing network sizes at a minimal performance drop. However, the memory footprint usually cannot be reduced.

6. Network distillation

Different from the above approaches which compress a pretrained deep network, network distillation category aims to train a smaller network to simulate the behaviors of a more complex

pre-trained deep network or the ensemble of multiple pre-trained deep networks. The training applies a teacher-student learning manner. Early work proposed by [56] proposed model compression, where the main idea was to use a fast and compact model to approximate the function learned by a slower, larger but better-performing model. Later, Hinton et al. proposed knowledge distillation [57] that trained a smaller neural network (called student network) by taking the output of a large, capable, but slow pre-trained one (called teacher network). The main strength of this idea comes from using the vast network to take care of the regularization process facilitating subsequent training operations. However, this method requires a large pre-trained network to begin with which is not always feasible. In [58], the authors extended this method to allow the training of a student that is deeper and thinner than the teacher, using not only the outputs but also the intermediate representations learned by the teacher as hints to improve the training process and final performance of the student. Inspired by these methods, Luo et al. [59] proposed to utilize the learned knowledge of a large teacher network or its ensemble as supervision to train a compact student network. The knowledge is represented by using the neurons at the higher hidden layer, which preserve as much information as the label probabilities but are more compact. When using an ensemble of DeepID2+ as teacher, a mimicked student is able to outperform it and achieves 51.6× compression ratio and 90× speedup in inference, making this model applicable on mobile devices. Lu et al. [60] investigated the teacher-student training for small-footprint acoustic models. Shi et al. [61] proposed a task-specified knowledge distillation algorithm to derive a simplified model with preset computation cost and minimized accuracy loss, which suits the resource-constraint front-end systems well. The knowledge distillation method relied on transferring the learned discriminative information from a teacher model to a student model. The method first analyzed the redundancy of the neural network related to a priori complexity of the given task and then trains a student model by redefining the loss function from a subset of the relaxed target knowledge according to the task information. Recently, Yom et al. [62] defined the distilled knowledge in terms of flow between layers and computed it with the inner product between features from two layers. The knowledge distillation idea was used to compress networks for object detection tasks [63–65].

Generally speaking, the network distillation approaches achieve a very high compression ratio. Another advantage of these approaches is making the resulting deep networks more interpretable. One issue needed to be addressed is reducing the accuracy drop ever improving the accuracy.

7. Network densifying

Another direct category for obtaining network compression and acceleration is to design more efficient but low-cost network architecture. I call this category as "network densifying," which aims to design compact deep networks to provide high accurate inference. In recent years, several approaches have been proposed following this line. The general ideas to achieve this goal include the usage of small filter kernels, grouping convolution, and advanced regularization.

Lin et al. [66] proposed Network-In-Network (NIN) architecture, where the main idea is using 1 × 1 convolution to increase the network capacity while making the computational complexity small. NIN also removed the fully connected layers instead of a global average pooling to reduce the storage requirement. The idea of 1 × 1 convolution is spread wide used in many advanced networks such as GoogleNet [67], ResNet [68], and DenseNet [69]. In [70], Iandola et al. designed a small DNN architecture termed SqueezeNet that achieves AlexNet-level accuracy on ImageNet but with 50× fewer parameters. In addition, with model compression techniques, SqueezeNet can be compressed to less than 1 MB (461× smaller than AlexNet). By using multiple group convolution, ResNeXt [71] achieved much higher accuracy than ResNet when costing the same computation. MobileNet [72] applied depth-wise convolution to reduce the computation cost, which achieved a 32× smaller model size and a 27× faster speed than VGG-16 model with comparable accuracy on ImageNet. ShuffleNet [73] introduced the channel shuffle operation to increase the information change within the multiple groups. It achieved about 13× speedup over AlexNet with comparable accuracy. DarkNet [74, 75] was proposed to facilitate object detection tasks, which applied most of the small convolutional kernels.

Moreover, some advanced regularization techniques are used to enhance the sparsity and robustness of deep networks. Dropout [76] and DropConnect [77] are widely exploited in many networks to increase the sparsity of activations for memory saving and weights for model size reduction, respectively. The activations neurons, including rectified Liner Unit (ReLU) [1], and its extends such as P-ReLU [78] are used to increase the sparsity of activations for memory saving while provide a speedup for model training, therefore they can facilitate the design of more compact networks.

8. Conclusions and future directions

It is necessary to develop efficient methods for deep learning via network compression and acceleration for facilitating the real-world deployment of advanced deep networks. In this chapter, I give a survey of recent network compression and acceleration approaches in five categories. In the following, I further introduce a few directions in the future in this literature including hybrid scheme for network compression, network acceleration for other visual tasks, hardware-software codesign for on-device applications, and more efficient distillation methods.

- **Hybrid scheme for network compression.** Current network compression approaches mainly focus on one single scheme, such as by using network quantization and network approximation. This processing leads to insufficient compression or large accuracy loss. It is necessary to exploit a hybrid scheme to combine the advantages from each network compression category. Some attempts can be found in [18, 79], which have demonstrated good performance.
- **Network acceleration for other visual tasks.** Most current approaches aim to compress and accelerate deep networks for image classification tasks, such as ImageNet large-scale

object classification, MNIST handwriting recognition, CIFAR object recognition, and so on. Very little effort have been attempted for other visual tasks, such as object detection, object tracking, semantic segmentation, and human pose estimation. Generally, direct using network acceleration approaches for image classification in these visual tasks may encounter a sharp drop on performance. The reason may come from that these visual tasks requires more complex feature representation or richer knowledge than image classification. The work has provided an attempt in facial landmark localization [80]. Therefore, this challenging problem on network acceleration for other visual tasks is one of the future directions.

- **Hardware-software codesign for on-device applications.** To realize the practical deployment on resource-limited devices, the network compression and acceleration algorithms should take the hardware design into consideration besides software algorithm modeling. The requirements from recent on-device applications such as autopiloting, video surveillance, and on-device AI enable tht it is highly desirable to design hardware-efficient deep learning algorithm according to the specific hardware platforms. This co-design scheme will be one future direction.
- **More effective distillation methods.** Network distillation methods have proven efficient for model compression in widespread fields beyond image classification, for example, machine translation [81]. However, these methods usually suffer from accuracy drop in inference, especially for complex inference tasks. Considering its efficacy, it is necessary to develop more effective distillation methods to extend their applications. Recent works [82–84] have given some attempts. Therefore, developing more effective distillation methods is one of the future directions in this field.

Acknowledgements

This work was partially supported by grants from National Key Research and Development Plan (2016YFC0801005), National Natural Science Foundation of China (61772513), and the International Cooperation Project of Institute of Information Engineering at Chinese Academy of Sciences (Y7Z0511101). Shiming Ge is also supported by Youth Innovation Promotion Association, Chinese Academy of Sciences.

Author details

Shiming Ge

Address all correspondence to: geshiming@iie.ac.cn

Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

References

[1] Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NIPS '12); 3-6 December 2012; Lake Tahoe; 2012. pp. 1097-1105

[2] Taigman Y, Yang M, Ranzato M, et al. DeepFace: Closing the gap to human-level performance in face verification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14); 24-27 June 2014; Columbus. New York: IEEE; 2014. pp. 1701-1708

[3] Amodei D, Ananthanarayanan S, Anubhai R, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning (ICML '16); 19-24 June 2016; New York: IEEE; 2016. pp. 173-182

[4] Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;**529**:484-489. DOI: 10.1038/nature16961

[5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 770-778

[6] Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15); 8-10 June 2015; Boston. New York: IEEE; 2015. pp. 815-823

[7] Learned-Miller E, Huang GB, RoyChowdhury A, et al. Labeled faces in the wild: A survey. In: Kawulok M, Celebi M, Smolka B, editors. Advances in Face Detection and Facial Image Analysis. Cham: Springer; 2016. pp. 189-248. DOI: 10.1007/978-3-319-25958-1_8

[8] Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. International Journal of Computer Vision (IJCV). 2015;**115**(3):211-252. DOI: 10.1007/s11263-015-0816-y

[9] Kumar N, Berg AC, Belhumeur PN, et al. Attribute and simile classifiers for face verification. In: IEEE International Conference on Computer Vision (ICCV '09); 29 Sepetember–2 October 2009; Kyoto. New York: IEEE; 2009. pp. 365-372

[10] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR '15); 7-9 May, 2015; San Diego; 2015

[11] Denil M, Shakibi B, Dinh L, et al. Predicting parameters in deep learning. In: Neural Information Processing Systems (NIPS '13); 5-10 December 2013; Lake Tahoe; 2013. pp. 2148-2156

[12] Sze V, Chen YH, Yang TJ, et al. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE. 2017;**105**(12):2295-2329. DOI: 10.1109/JPROC.2017.2761740

[13] Cheng J, Wang P, Li G, et al. Recent advances in efficient computation of deep convolutional neural networks. Frontiers of Information Technology & Electronic Engineering. 2018;**19**(1):64-77. DOI: 10.1631/FITEE.1700789

[14] Cheng Y, Wang D, Zhou P, et al. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine. 2018;**35**(1):126-136. DOI: 10.1109/MSP.2017.2765695

[15] Babak Hassibi, David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In: Neural Information Processing Systems (NIPS '93); 1993; Denvor. San Francisco: Morgan Kaufmann; 1994. pp. 164-171

[16] Chaber P, Lawrynczuk M. Pruning of recurrent neural models: An optimal brain damage approach. Nonlinear Dynamics. 2018;**92**(2):763-780. DOI: 10.1007/s11071-018-4089-1

[17] Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural networks. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1135-1143

[18] Han S, Mao H, Dally W. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In: International Conference on Learning Representations (ICLR '16); 2-4 May 2016; San Juan; 2016

[19] Guo Y, Yao A, Chen Y. Dynamic network surgery for efficient DNNs. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 1379-1387

[20] Sun Y, Wang X, Tang X. Sparsifying neural network connections for face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 4856-4864

[21] Srinivas S, Babu RV. Data-free parameter pruning for deep neural networks. In Proceedings of the British Machine Vision Conference (BMVC '15); 7-10 September 2015; Swansea: BMVA Press; 2015. pp. 31.1-31.12

[22] Anwar S, Hwang K, Sung W. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC). 2017;**13**(3):32. DOI: 10.1145/3005348

[23] Mao H, Han S, Pool J, et al. Exploring the Regularity of Sparse Structure in Convolutional Neural Networks [Internet]. Available from: https://arxiv.org/pdf/1707.06342 [Accessed: May 12, 2018]

[24] Lebedev V, Lempitsky V. Fast convnets using group-wise brain damage. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 2554-2564

[25] Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 2074-2082

[26] Polyak A, Wolf L. Channel-level acceleration of deep face representations. IEEE Access. 2015;**3**:2163-2175. DOI: 10.1109/access.2015.2494536

[27] Luo JH, Wu J, Lin W. ThiNet: A filter level pruning method for deep neural network compression. In: IEEE International Conference on Computer Vision (ICCV '17); 22-29 October 2017; Venice Italy; 2017. pp. 5058-5066

[28] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: IEEE International Conference on Computer Vision (ICCV '17); 22-29 October 2017; Venice. New York: IEEE; 2017. pp. 1389-1397

[29] Gong Y, Liu L, Yang M, et al. Compressing deep convolutional networks using vector quantization. In: International Conference on Learning Representations (ICLR '15); 7-9 May 2015; San Diego; 2015

[30] Wu J, Leng C, Wang Y, et al. Quantized convolutional neural networks for mobile devices. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 4820-4828

[31] Soulié G, Gripon V, Robert M. Compression of deep neural networks on the fly. In: International Conference on Artificial Neural Networks; 6-9 September 2016; Barcelona. Cham: Springer; 2016. pp. 153-160

[32] Dettmers T. 8-bit approximations for parallelism in deep learning. In: International Conference on Learning Representations (ICLR '16); 2-4 May 2016; Caribe Hilton; 2016

[33] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision. In: International Conference on Machine Learning (ICML '15); 6-11 July 2015; Lille. JMLR; 2015. pp. 1737-1746

[34] Lin DD, Talathi SS, Annapureddy VS. Fixed point quantization of deep convolutional networks. In: Proceedings of The International Conference on Machine Learning (ICML '16); 19-24 June 2016; New York. JMLR; 2016. pp. 2849-2858

[35] Soudry D, Hubara I, Meir R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In: Neural Information Processing Systems (NIPS'14); 8-13 December 2014; Montreal; 2014. pp. 963-971

[36] Courbariaux M, Bengio Y, David JP. Binaryconnect: Training deep neural networks with binary weights during propagations. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 3123-3131

[37] Esser SK, Appuswamy R, Merolla P, et al. Backpropagation for energy-efficient neuromorphic computing. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1117-1125

[38] Hubara I, Courbariaux M, Soudry D, et al. Binarized neural networks. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 4107-4115

[39] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision (ECCV '16); 8-16 October 2016; Amsterdam. Cham: Springer; 2016. pp. 525-542

[40] Hou L, Yao Q, Kwok JT. Loss-aware binarization of deep networks. In: International Conference on Learning Representations (ICLR '17); 24-26 April 2017; Palais des Congrès Neptune, Toulon, France; 2017

[41] Juefei-Xu F, Boddeti VN, Savvides M. Local binary convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 19-28

[42] Guo Y, Yao A, Zhao H, Chen Y. Network sketching: Exploiting binary structure in deep CNNs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 5955-5963

[43] Zhu C, Han S, Mao H, Dally WJ. Trained ternary quantization. In: International Conference on Learning Representations (ICLR '17); 24-26 April 2017; Palais des Congrès Neptune, Toulon, France; 2017

[44] Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions. In: Proceedings of the British Machine Vision Conference (BMVC '15); 7-10 September 2015; Swansea: BMVA Press; 2015

[45] Denton EL, Zaremba W, Bruna J, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1269-1277

[46] Liu B, Wang M, Foroosh H, et al. Sparse convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15); 8-10 June 2015; Boston. New York: IEEE; 2015. pp. 806-814

[47] Figurnov M, Ibraimova A, Vetrov D P, et al. Perforated CNNs: Acceleration through elimination of redundant convolutions. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 947-955

[48] Zhang X, Zou J, He K, et al. Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016;**38**(10):1943-1955. DOI: 10.1109/tpami.2015.2502579

[49] Kim YD, Park E, Yoo S, et al. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications [Internet]. Available from: https://arxiv.org/pdf/1511.06530 [Accessed: May 12, 2018]

[50] Wang P, Cheng J. Accelerating convolutional neural networks for mobile applications. In: Proceedings of the 2016 ACM on Multimedia Conference; 15-19 Oct. 2016; Amsterdam, The Netherlands; 2016. p. 541-545

[51] Lebedev V, Ganin Y, Rakhuba M, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In: International Conference on Learning Representations (ICLR '15); 7-10 May 2015; San Diego; 2015

[52] Wang Y, Xu C, You S, et al. Cnnpack: Packing convolutional neural networks in the frequency domain. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 253-261

[53] Wang Y, Xu C, Xu C, Tao D. Beyond filters: Compact feature map for portable deep model. In: Proceedings of The International Conference on Machine Learning (ICML '17); 8-11 August 2017; Sydney. JMLR; 2017. pp. 3703-3711

[54] Chen W, Wilson J, Tyree S, et al. Compressing neural networks with the hashing trick. In: Proceedings of the International Conference on Machine Learning (ICML '15); 6-11 July 2015; Lille. JMLR; 2015. pp. 2285-2294

[55] Shi L, Feng S. Functional Hashing for Compressing Neural Networks [Internet]. Available from: https://arxiv.org/pdf/1605.06560 [Accessed: May 12, 2018]

[56] Bucilu C, Caruana R, Niculescu-Mizil A. Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 20-23 August 2006; Philadelphia. ACM; 2006. pp. 535-541

[57] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network [Internet]. Available from: https://arxiv.org/pdf/1503.02531 [Accessed: May 12, 2018]

[58] Romero A, Ballas N, Kahou SE, et al. FitNets: Hints for thin deep nets. In: International Conference on Learning Representations (ICLR '15); 7-10 May 2015; San Diego; 2015

[59] Luo P, Zhu Z, Liu Z, et al. Face model compression by distilling knowledge from neurons. In: Thirtieth AAAI Conference on Artificial Intelligence (AAAI '16); 12-17 February. 2016; Phoenix, Arizona, USA; 2016. pp. 3560-3566

[60] Lu L, Guo M, Renals S. Knowledge distillation for small-footprint highway networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '17); 5-9 March 2017; New Orleans. New York: IEEE; 2017. pp. 4820-4824

[61] Shi M, Qin F, Ye Q, et al. A scalable convolutional neural network for task-specified scenarios via knowledge distillation. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 5-9 March 2017; New Orleans. New York: IEEE; 2017. pp. 2467-2471

[62] Yim J, Joo D, Bae J, et al. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7130-7138

[63] Shen J, Vesdapunt N, Boddeti VN, et al. In Teacher We Trust: Learning Compressed Models for Pedestrian Detection [Internet]. Available from: https://arxiv.org/pdf/1602.00478 [Accessed: May 12, 2018]

[64] Li Q, Jin S, Yan J. Mimicking very efficient network for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7341-7349

[65] Chen G, Choi W, Yu X, et al. Learning efficient object detection models with knowledge distillation. In: Neural Information Processing Systems (NIPS '17); 4-9 December 2017; Long Beach, CA; 2017. pp. 742-751

[66] Lin M, Chen Q, Yan S. Network in Network [Internet]. Available from: https://arxiv.org/pdf/1312.4400 [Accessed: May 12, 2018]

[67] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 1-9

[68] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 770-778

[69] Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 4700-4708

[70] Iandola FN, Han S, Moskewicz MW, et al. SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and < 0.5 MB Model Size [Internet]. Available from: https://arxiv.org/pdf/1602.07360 [Accessed: May 12, 2018]

[71] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 5987-5995

[72] Howard AG, Zhu M, Chen B, et al. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Internet]. Available from: https://arxiv.org/pdf/1704.04861 [Accessed: May 12, 2018]

[73] Zhang X, Zhou X, Lin M, et al. Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [Internet]. Available from: https://arxiv.org/pdf/1707.01083 [Accessed: May 12, 2018]

[74] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 779-788

[75] Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7263-7271

[76] Srivastava N, Hinton GE, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014;**15**(1):1929-1958

[77] Wan L, Zeiler MD, Zhang S, et al. Regularization of neural networks using DropConnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML '13); 16-21 June 2013; Atlanta, GA; 2013. pp. 1058-1066

[78] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: IEEE International Conference on Computer Vision (ICCV '15). 7-13 December 2015; Santiago, Chile. IEEE; 2015. pp. 1-9

[79] Ge S, Luo Z, Zhao S, et al. Compressing deep neural networks for efficient visual inference. In: IEEE International Conference on Multimedia and Expo (ICME '17). 10-14 July 2017; Hong Kong. IEEE: 2017. pp. 667-672

[80] Zeng D, Zhao F, Shen W, Ge S. Compressing and accelerating neural network for facial point localization. Cognitive Computation. 2018;**10**(2):359-367. DOI: 10.1007/s12559-017-9506-0

[81] Kim Y, Rush AM. Sequence-level knowledge distillation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '16). 1-4 November 2016; Austin, Texas; 2016. pp. 1317-1327

[82] Lopez-Paz D, Bottou L, Schölkopf B, Vapnik V. Unifying distillation and privileged information. In: International Conference on Learning Representations (ICLR '16). 2-4 May 2016; San Juan; 2016. pp. 1-10

[83] Hu Z, Ma X, Liu Z, et al. Harnessing deep neural networks with logic rules. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL '16). 7-12 August 2016; Berlin, Germany; 2016. pp.1-11

[84] Luo Z, Jiang L, Hsieh JT, et al. Graph Distillation for Action Detection with Privileged Information [Internet]. Available from: https://arxiv.org/abs/1712.00108 [Accessed: December 30, 2017]