A Survey of Model Compression and Acceleration for Deep Neural Networks

Yu Cheng, Duo Wang, Pan Zhou, *Member, IEEE,* and Tao Zhang, *Senior Member, IEEE*

Abstract—Deep convolutional neural networks (CNNs) have recently achieved great success in many visual recognition tasks. However, existing deep neural network models are computationally expensive and memory intensive, hindering their deployment in devices with low memory resources or in applications with strict latency requirements. Therefore, a natural thought is to perform model compression and acceleration in deep networks without significantly decreasing the model performance. During the past few years, tremendous progress has been made in this area. In this paper, we survey the recent advanced techniques for compacting and accelerating CNNs model developed. These techniques are roughly categorized into four schemes: parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. Methods of parameter pruning and sharing will be described at the beginning, after that the other techniques will be introduced. For each scheme, we provide insightful analysis regarding the performance, related applications, advantages, and drawbacks etc. Then we will go through a few very recent additional successful methods, for example, dynamic capacity networks and stochastic depths networks. After that, we survey the evaluation matrix, the main datasets used for evaluating the model performance and recent benchmarking efforts. Finally, we conclude this paper, discuss remaining challenges and possible directions on this topic.

Index Terms—Deep Learning, Convolutional Neural Networks, Model Compression and Acceleration,

I. INTRODUCTION

In recent years, deep neural networks have recently received lots of attention, been applied to different applications and achieved dramatic accuracy improvements in many tasks. These works rely on deep networks with millions or even billions of parameters, and the availability of GPUs with very high computation capability plays a key role in their success. For example, the work by Krizhevsky *et al.* [1] achieved breakthrough results in the 2012 ImageNet Challenge using a network containing 60 million parameters with five convolutional layers and three fully-connected layers. Usually, it takes two to three days to train the whole model on ImagetNet dataset with a NVIDIA K40 machine. Another example is the top face verification results on the Labeled Faces in the Wild (LFW) dataset were obtained with networks containing hundreds of millions of parameters, using a mix of convolutional, locally-connected, and fully-connected layers

Yu Cheng is a Researcher from Microsoft AI & Research, One Microsoft Way, Redmond, WA 98052, USA.

Duo Wang and Tao Zhang are with the Department of Automation, Tsinghua University, Beijing 100084, China.

Pan Zhou is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China.

[2], [3]. It is also very time-consuming to train such a model to get reasonable performance. In architectures that rely only on fully-connected layers, the number of parameters can grow to billions [4].

As larger neural networks with more layers and nodes are considered, reducing their storage and computational cost becomes critical, especially for some real-time applications such as online learning and incremental learning. In addition, recent years witnessed significant progress in virtual reality, augmented reality, and smart wearable devices, creating unprecedented opportunities for researchers to tackle fundamental challenges in deploying deep learning systems to portable devices with limited resources (e.g. memory, CPU, energy, bandwidth). Efficient deep learning methods can have significant impacts on distributed systems, embedded devices, and FPGA for Artificial Intelligence. For example, the ResNet-50 [5] with 50 convolutional layers needs over 95MB memory for storage and over 3.8 billion floating number multiplications when processing an image. After discarding some redundant weights, the network still works as usual but saves more than 75% of parameters and 50% computational time. For devices like cell phones and FPGAs with only several megabyte resources, how to compact the models used on them is also important.

Achieving these goal calls for joint solutions from many disciplines, including but not limited to machine learning, optimization, computer architecture, data compression, indexing, and hardware design. In this paper, we review recent works on compressing and accelerating deep neural networks, which attracted a lot of attention from the deep learning community and already achieved lots of progress in the past years.

We classify these approaches into four categories: parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. The parameter pruning and sharing based methods explore the redundancy in the model parameters and try to remove the redundant and uncritical ones. Low-rank factorization based techniques use matrix/tensor decomposition to estimate the informative parameters of the deep CNNs. The approaches based on transferred/compact convolutional filters design special structural convolutional filters to reduce the parameter space and save storage/computation. The knowledge distillation methods learn a distilled model and train a more compact neural network to reproduce the output of a larger network.

In Table I, we briefly summarize these four types of methods. Generally, the parameter pruning & sharing, low-rank factorization and knowledge distillation approaches can

TABLE I
SUMMARIZATION OF DIFFERENT APPROACHES FOR MODEL COMPRESSION AND ACCELERATION.

Theme Name	Description	Applications	More details
Parameter pruning and sharing	Reducing redundant parameters which are not sensitive to the performance	Convolutional layer and fully connected layer	Robust to various settings, can achieve good performance, can support both train from scratch and pre-trained model
Low-rank factorization	Using matrix/tensor decomposition to estimate the informative parameters	Convolutional layer and fully connected layer	Standardized pipeline, easily to be implemented, can support both train from scratch and pre-trained model
Transferred/compact convolutional filters	Designing special structural convolutional filters to save parameters	Convolutional layer only	Algorithms are dependent on applications, usually achieve good performance, only support train from scratch
Knowledge distillation	Training a compact neural network with distilled knowledge of a large model	Convolutional layer and fully connected layer	Model performances are sensitive to applications and network structure only support train from scratch

be used in DNN models with fully connected layers and convolutional layers, achieving comparable performances. On the other hand, methods using transferred/compact filters are designed for models with convolutional layers only. Low-rank factorization and transfered/compact filters based approaches provide an end-to-end pipeline and can be easily implemented in CPU/GPU environment, which is straightforward. while parameter pruning & sharing use different methods such as vector quantization, binary coding and sparse constraints to perform the task. Generally it will take several steps to achieve the goal.

Regarding the training protocols, models based on parameter pruning/sharing low-rank factorization can be extracted from pre-trained ones or trained from scratch. While the transferred/compact filter and knowledge distillation models can only support train from scratch. These methods are independently designed and complement each other. For example, transferred layers and parameter pruning & sharing can be used together, and model quantization & binarization can be used together with low-rank approximations to achieve further speedup. We will describe the details of each theme, their properties, strengths and drawbacks in the following sections.

II. PARAMETER PRUNING AND SHARING

Early works showed that network pruning is effective in reducing the network complexity and addressing the over-fitting problem [6]. After that researcher found pruning originally introduced to reduce the structure in neural networks and hence improve generalization, it has been widely studied to compress DNN models, trying to remove parameters which are not crucial to the model performance. These techniques can be further classified into three sub-categories: quantization and binarization, parameter sharing, and structural matrix.

A. Quantization and Binarization

Network quantization compresses the original network by reducing the number of bits required to represent each weight. Gong *et al.* [6] and Wu et al. [7] applied k-means scalar quantization to the parameter values. Vanhoucke *et al.* [8] showed that 8-bit quantization of the parameters can result in significant speed-up with minimal loss of accuracy. The work in [9] used 16-bit fixed-point representation in stochastic rounding based CNN training, which significantly reduced
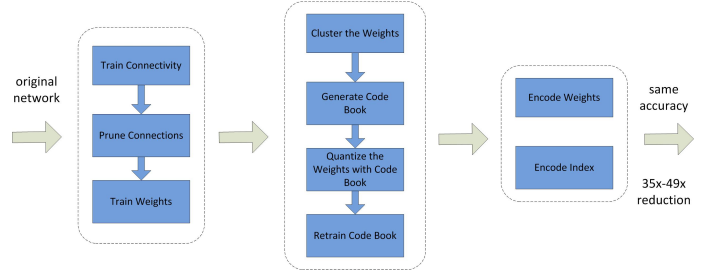

Fig. 1. The three-stage compression method proposed in [10]: pruning, quantization and encoding. The input is the original model and the output is the compression model.

memory usage and float point operations with little loss in classification accuracy.

The method proposed in [10] quantized the link weights using weight sharing and then applied Huffman coding to the quantized weights as well as the codebook to further reduce the rate. As shown in Figure 1, it started by learning the connectivity via normal network training, followed by pruning the small-weight connections. Finally, the network was retrained to learn the final weights for the remaining sparse connections. This work achieved the state-of-art performance among all parameter quantization based methods. It was shown in [11] that Hessian weight could be used to measure the importance of network parameters, and proposed to minimize Hessian-weighted quantization errors in average for clustering network parameters.

In the extreme case of the 1-bit representation of each weight, that is binary weight neural networks. There are many works that directly train CNNs with binary weights, for instance, BinaryConnect [12], BinaryNet [13] and XNORNetworks [14]. The main idea is to directly learn binary weights or activation during the model training. The systematic study in [15] showed that networks trained with back propagation could be resilient to specific weight distortions, including binary weights.

Drawbacks: the accuracy of the binary nets is significantly lowered when dealing with large CNNs such as GoogleNet. Another drawback of such binary nets is that existing binarization schemes are based on simple matrix approximations and ignore the effect of binarization on the accuracy loss.

To address this issue, the work in [16] proposed a proximal Newton algorithm with diagonal Hessian approximation that directly minimizes the loss with respect to the binary weights. The work in [17] reduced the time on float point multiplication in the training stage by stochastically binarizing weights and converting multiplications in the hidden state computation to significant changes.

B. *Pruning and Sharing*

Network pruning and sharing has been used both to reduce network complexity and to address the over-fitting issue. An early approach to pruning was the Biased Weight Decay [18]. The Optimal Brain Damage [19] and the Optimal Brain Surgeon [20] methods reduced the number of connections based on the Hessian of the loss function, and their work suggested that such pruning gave higher accuracy than magnitude-based pruning such as the weight decay method. The training procedure of those methods followed the way training from scratch manner.

A recent trend in this direction is to prune redundant, non-informative weights in a pre-trained CNN model. For example, Srinivas and Babu [21] explored the redundancy among neurons, and proposed a data-free pruning method to remove redundant neurons. Han *et al.* [22] proposed to reduce the total number of parameters and operations in the entire network. Chen *et al.* [23] proposed a HashedNets model that used a low-cost hash function to group weights into hash buckets for parameter sharing. The deep compression method in [10] removed the redundant connections and quantized the weights, and then used Huffman coding to encode the quantized weights. In [24], a simple regularization method based on soft weight-sharing was proposed, which included both quantization and pruning in one simple (re-)training procedure. The above pruning schemes typically produce connections pruning in CNNs.

There is also growing interest in training compact CNNs with sparsity constraints. Those sparsity constraints are typically introduced in the optimization problem as l_0 or l_1-norm regularizers. The work in [25] imposed group sparsity constraint on the convolutional filters to achieve structured brain Damage, i.e., pruning entries of the convolution kernels in a group-wise fashion. In [26], a group-sparse regularizer on neurons was introduced during the training stage to learn compact CNNs with reduced filters. Wen *et al.* [27] added a structured sparsity regularizer on each layer to reduce trivial filters, channels or even layers. In the filter-level pruning, all the above works used $l_{2,1}$-norm regularizers. The work in [28] used l_1-norm to select and prune unimportant filters.

Drawbacks: there are some potential issues of the pruning and sharing. First, pruning with l_1 or l_2 regularization requires more iterations to converge than general. In addition, all pruning criteria require manual setup of sensitivity for layers, which demands fine-tuning of the parameters and could be cumbersome for some applications.

C. *Designing Structural Matrix*

In architectures that contain fully-connected layers, it is critical to explore this redundancy of parameters in fully-connected layers, which is often the bottleneck in terms of memory consumption. These network layers use the nonlinear transforms $f(\mathbf{x}, \mathbf{M}) = \sigma(\mathbf{Mx})$, where $\sigma(\cdot)$ is an element-wise nonlinear operator, $\mathbf{x}$ is the input vector, and $\mathbf{M}$ is the $m \times n$ matrix of parameters [29]. When $\mathbf{M}$ is a large general dense matrix, the cost of storing mn parameters and computing matrix-vector products in $O(mn)$ time. Thus, an intuitive way to prune parameters is to impose $\mathbf{x}$ as a parameterized structural matrix. An $m \times n$ matrix that can be described using much fewer parameters than mn is called a structured matrix. Typically, the structure should not only reduce the memory cost, but also dramatically accelerate the inference and training stage via fast matrix-vector multiplication and gradient computations.

Following this direction, the work in [30], [31] proposed a simple and efficient approach based on circulant projections, while maintaining competitive error rates. Given a vector $\mathbf{r} = (r_0, r_1, \cdots, r_{d-1})$, a circulant matrix $\mathbf{R} \in \mathbf{R}^{d \times d}$ is defined as:

$$\mathbf{R} = \mathrm{circ}(\mathbf{r}) := \begin{bmatrix} r_0 & r_{d-1} & \dots & r_2 & r_1 \\ r_1 & r_0 & r_{d-1} & & r_2 \\ \vdots & r_1 & r_0 & \ddots & \vdots \\ r_{d-2} & & \ddots & \ddots & r_{d-1} \\ r_{d-1} & r_{d-2} & \dots & r_1 & r_0 \end{bmatrix}. \quad (1)$$

thus the memory cost becomes $\mathcal{O}(d)$ instead of $\mathcal{O}(d^2)$. This circulant structure also enables the use of Fast Fourier Transform (FFT) to speed up the computation. Given a d-dimensional vector $\mathbf{r}$, the above 1-layer circulant neural network in Eq. 1 has time complexity of $\mathcal{O}(d \log d)$.

In [32], a novel Adaptive Fastfood transform was introduced to reparameterize the matrix-vector multiplication of fully connected layers. The Adaptive Fastfood transform matrix $\mathbf{R} \in \mathbf{R}^{n \times d}$ was defined as:

$$\mathbf{R} = \mathbf{SHG\Pi HB} \quad (2)$$

where $\mathbf{S}$, $\mathbf{G}$ and $\mathbf{B}$ are random diagonal matrices. $\mathbf{\Pi} \in \{0, 1\}^{d \times d}$ is a random permutation matrix, and $\mathbf{H}$ denotes the Walsh-Hadamard matrix. Reparameterizing a fully connected layer with d inputs and n outputs using the Adaptive Fastfood transform reduces the storage and the computational costs from $\mathcal{O}(nd)$ to $\mathcal{O}(n)$ and from $\mathcal{O}(nd)$ to $\mathcal{O}(n \log d)$, respectively.

The work in [29] showed the effectiveness of the new notion of parsimony in the theory of structured matrices. Their proposed method can be extended to various other structured matrix classes, including block and multi-level Toeplitz-like [33] matrices related to multi-dimensional convolution [34]. Following this idea, [35] proposed a general structured efficient linear layer for CNNs.

Drawbacks: one problem of this kind of approaches is that the structural constraint will hurt the performance since the constraint might bring bias to the model. On the other hand, how to find a proper structural matrix is difficult. There is no theoretical way to derive it out.

III. LOW-RANK FACTORIZATION AND SPARSITY

Convolution operations contribute the bulk of most computations in deep CNNs, thus reducing the convolution layer
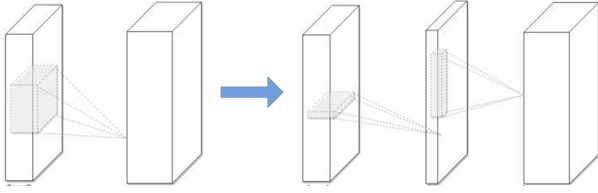
Fig. 2. A typical framework of the low-rank regularization method. The left is the original convolutional layer and the right is the low-rank constraint convolutional layer with rank-K.

TABLE II
COMPARISONS BETWEEN THE LOW-RANK MODELS AND THEIR BASELINES ON ILSVRC-2012.

Model	TOP-5 Accuracy	Speed-up	Compression Rate
AlexNet	80.03%	1.	1.
BN Low-rank	80.56%	1.09	4.94
CP Low-rank	79.66%	1.82	5.
VGG-16	90.60%	1.	1.
BN Low-rank	90.47%	1.53	2.72
CP Low-rank	90.31%	2.05	2.75
GoogleNet	92.21%	1.	1.
BN Low-rank	91.88%	1.08	2.79
CP Low-rank	91.79%	1.20	2.84

would improve the compression rate as well as the overall speedup. For the convolution kernels, it can be viewed as a 4D tensor. Ideas based on tensor decomposition is derived by the intuition that there is a significant amount of redundancy in the 4D tensor, which is a particularly promising way to remove the redundancy. Regarding the fully-connected layer, it can be view as a 2D matrix and the low-rankness can also help.

It has been a long time for using low-rank filters to accelerate convolution, for example, high dimensional DCT (discrete cosine transform) and wavelet systems using tensor products to be constructed from 1D DCT transform and 1D wavelets respectively. Learning separable 1D filters was introduced by Rigamonti *et al.* [36], following the dictionary learning idea. Regarding some simple DNN models, a few low-rank approximation and clustering schemes for the convolutional kernels were proposed in [37]. They achieved $2\times$ speedup for a single convolutional layer with 1% drop in classification accuracy. The work in [38] proposed using different tensor decomposition schemes, reporting a $4.5\times$ speedup with 1% drop in accuracy in text recognition.

The low-rank approximation was done layer by layer. The parameters of one layer were fixed after it was done, and the layers above were fine-tuned based on a reconstruction error criterion. These are typical low-rank methods for compressing 2D convolutional layers, which is described in Figure 2. Following this direction, Canonical Polyadic (CP) decomposition of was proposed for the kernel tensors in [39]. Their work used nonlinear least squares to compute the CP decomposition. In [40], a new algorithm for computing the low-rank tensor decomposition for training low-rank constrained CNNs from scratch were proposed. It used Batch Normalization (BN) to transform the activation of the internal hidden units. In general, both the CP and the BN decomposition schemes in [40] (BN Low-rank) can be used to train CNNs from scratch. However, there are few differences between them. For example, finding the best low-rank approximation in CP decomposition is an ill-posed problem, and the best rank-K (K is the rank number) approximation may not exist sometimes. While for the BN scheme, the decomposition always exists. We perform a simple comparison of both methods shown in Table II. The actual speedup and the compression rates are used to measure their performances.

As we mentioned before, the fully connected layers can be viewed as a 2D matrix and thus the above mentioned methods can also be applied there. There are several classical works on exploiting low-rankness in fully connected layers.

For instance, Misha *et al.* [41] reduced the number of dynamic parameters in deep models using the low-rank method. [42] explored a low-rank matrix factorization of the final weight layer in a DNN for acoustic modeling. In [3], Lu *et al.* adopted truncated SVD (singular value decomposition) to decompsite the fully connected layer for designing compact multi-task deep learning architectures.

Drawbacks: low-rank approaches are straightforward for model compression and acceleration. The idea complements recent advances in deep learning, such as dropout, rectified units and maxout. However, the implementation is not that easy since it involves decomposition operation, which is computationally expensive. Another issue is that current methods perform low-rank approximation layer by layer, and thus cannot perform global parameter compression, which is important as different layers hold different information. Finally, factorization requires extensive model retraining to achieve convergence when compared to the original model.

IV. TRANSFERRED/COMPACT CONVOLUTIONAL FILTERS

CNNs are parameter efficient due to exploring the translation invariant property of the representations to the input image, which is the key to the success of training very deep models without severe over-fitting. Although a strong theory is currently missing, a large number of empirical evidence support the notion that both the translation invariant property and the convolutional weight sharing are important for good predictive performance. The idea of using transferred convolutional filters to compress CNN models is motivated by recent works in [43], which introduced the equivariant group theory. Let $\mathbf{x}$ be an input, $\Phi(\cdot)$ be a network or layer and $\mathcal{T}(\cdot)$ be the transform matrix. The concept of equivalence is defined as:

$$\mathcal{T}'\Phi(\mathbf{x}) = \Phi(\mathcal{T}\mathbf{x}) \tag{3}$$

indicating that transforming the input $\mathbf{x}$ by the transform $\mathcal{T}(\cdot)$ and then passing it through the network or layer $\Phi(\cdot)$ should give the same result as first mapping $\mathbf{x}$ through the network and then transforming the representation. Note that in Eq. (10), the transforms $\mathcal{T}(\cdot)$ and $\mathcal{T}'(\cdot)$ are not necessarily the same as they operate on different objects. According to this theory, it is reasonable applying transform to layers or filters $\Phi(\cdot)$ to compress the whole network models. From empirical observation, deep CNNs also benefit from using a large set of convolutional filters by applying certain transform $\mathcal{T}(\cdot)$ to a

small set of base filters since it acts as a regularizer for the model.

Following this direction, there are many recent reworks proposed to build a convolutional layer from a set of base filters [43]–[46]. What they have in common is that the transform $\mathcal{T}(\cdot)$ lies in the family of functions that only operate in the spatial domain of the convolutional filters. For example, the work in [45] found that the lower convolution layers of CNNs learned redundant filters to extract both positive and negative phase information of an input signal, and defined $\mathcal{T}(\cdot)$ to be the simple negation function:

$$\mathcal{T}(\mathbf{W}_x) = \mathbf{W}_x^- \qquad (4)$$

where $\mathbf{W}_x$ is the basis convolutional filter and $\mathbf{W}_x^-$ is the filter consisting of the shifts whose activation is opposite to that of $\mathbf{W}_x$ and selected after max-pooling operation. By doing this, the work in [45] can easily achieve $2\times$ compression rate on all the convolutional layers. It is also shown that the negation transform acts as a strong regularizer to improve the classification accuracy. The intuition is that the learning algorithm with pair-wise positive-negative constraint can lead to useful convolutional filters instead of redundant ones.

In [46], it was observed that magnitudes of the responses from convolutional kernels had a wide diversity of pattern representations in the network, and it was not proper to discard weaker signals with a single threshold. Thus a multi-bias non-linearity activation function was proposed to generates more patterns in the feature space at low computational cost. The transform $\mathcal{T}(\cdot)$ was define as:

$$\mathcal{T}`\Phi(\mathbf{x}) = \mathbf{W}_x + \delta \qquad (5)$$

where δ were the multi-bias factors. The work in [47] considered a combination of rotation by a multiple of $90°$ and horizontal/vertical flipping with:

$$\mathcal{T}`\Phi(\mathbf{x}) = \mathbf{W}^{T_\theta} \qquad (6)$$

where $\mathbf{W}^{T_\theta}$ was the transformation matrix which rotated the original filters with angle $\theta \in \{90, 180, 270\}$. In [43], the transform was generalized to any angle learned from data, and θ was directly obtained from data. Both works [47] and [43] can achieve good classification performance.

The work in [44] defined $\mathcal{T}(\cdot)$ as the set of translation functions applied to 2D filters:

$$\mathcal{T}`\Phi(\mathbf{x}) = T(\cdot, x, y)_{x,y\in\{-k,...,k\},(x,y)\neq(0,0)} \qquad (7)$$

where $T(\cdot, x, y)$ denoted the translation of the first operand by (x, y) along its spatial dimensions, with proper zero padding at borders to maintain the shape. The proposed framework can be used to 1) improve the classification accuracy as a regularized version of maxout networks, and 2) to achieve parameter efficiency by flexibly varying their architectures to compress networks.

Table III briefly compares the performance of different methods with transferred convolutional filters, using VGGNet (16 layers) as the baseline model. The results are reported on CIFAR-10 and CIFAR-100 datasets with Top-5 error. It is observed that they can achieve reduction in parameters with little or no drop in classification accuracy.

TABLE III
A SIMPLE COMPARISON OF DIFFERENT APPROACHES ON CIFAR-10 AND CIFAR-100.

Model	CIFAR-100	CIFAR-10	Compression Rate
VGG-16	34.26%	9.85%	1.
MBA [46]	33.66%	9.76%	2.
CRELU [45]	34.57%	9.92%	2.
CIRC [43]	35.15%	10.23%	4.
DCNN [44]	33.57%	9.65%	1.62

Drawbacks: there are few issues to be addressed for approaches that apply transform constraints to convolutional filters. First, these methods can achieve competitive performance for wide/flat architectures (like VGGNet) but not thin/deep ones (like GoogleNet, Residual Net). Secondly, the transfer assumptions sometimes are too strong to guide the learning, making the results unstable in some cases.

Using a compact filter for convolution can directly reduce the computation cost. The key idea is to replace the loose and over-parametric filters with compact blocks to improve the speed, which significantly accelerate CNNs on several benchmarks. Decomposing 3×3 convolution into two 1×1 convolutions was used in [48], which achieved significant acceleration on object recognition. SqueezeNet [49] was proposed to replace 3×3 convolution with 1×1 convolution, which created a compact neural network with about 50 fewer parameters and comparable accuracy when compared to AlexNet.

V. KNOWLEDGE DISTILLATION

To the best of our knowledge, exploiting knowledge transfer (KT) to compress model was first proposed by Caruana *et al.* [50]. They trained a compressed/ensemble model of strong classifiers with pseudo-data labeled, and reproduced the output of the original larger network. But the work is limited to shallow models. The idea has been recently adopted in [51] as knowledge distillation (KD) to compress deep and wide networks into shallower ones, where the compressed model mimicked the function learned by the complex model. The main idea of KD based approaches is to shift knowledge from a large teacher model into a small one by learning the class distributions output via softmax.

The work in [52] introduced a KD compression framework, which eased the training of deep networks by following a student-teacher paradigm, in which the student was penalized according to a softened version of the teacher's output. The framework compressed an ensemble of teacher networks into a student network of similar depth. The student was trained to predict the output and the classification labels. Despite its simplicity, KD demonstrates promising results in various image classification tasks. The work in [53] aimed to address the network compression problem by taking advantage of depth neural networks. It proposed an approach to train thin but deep networks, called FitNets, to compress wide and shallower (but still deep) networks. The method was extended the idea to allow for thinner and deeper student models. In order to learn from the intermediate representations of teacher network, FitNet made the student mimic the full feature maps

of the teacher. However, such assumptions are too strict since the capacities of teacher and student may differ greatly.

All the above approaches are validated on MNIST, CIFAR-10, CIFAR-100, SVHN and AFLW benchmark datasets, and experimental results show that these methods match or outperform the teacher's performance, while requiring notably fewer parameters and multiplications.

There are several extension along this direction of distillation knowledge. The work in [54] trained a parametric student model to approximate a Monte Carlo teacher. The proposed framework used online training, and used deep neural networks for the student model. Different from previous works which represented the knowledge using the soften label probabilities, [55] represented the knowledge by using the neurons in the higher hidden layer, which preserved as much information as the label probabilities, but are more compact. The work in [56] accelerated the experimentation process by instantaneously transferring the knowledge from a previous network to each new deeper or wider network. The techniques are based on the concept of function-preserving transformations between neural network specifications. Zagoruyko et al. [57] proposed Attention Transfer (AT) to relax the assumption of FitNet. They transferred the attention maps that are summaries of the full activations.

Drawbacks: KD-based approaches can make deeper models thinner and help significantly reduce the computational cost. However, there are a few disadvantages. One of those is that KD can only be applied to classification tasks with softmax loss function, which hinders its usage. Another drawback is the model assumptions sometimes are too strict to make the performance competitive with other type of approaches.

VI. OTHER TYPES OF APPROACHES

We first summarize the works utilizing attention-based methods. Note that attention-based mechanism [58] can reduce computations significantly by learning to selectively focus or "attend" to a few, task-relevant input regions. The work in [59] introduced the dynamic capacity network (DCN) that combined two types of modules: the small sub-networks with low capacity, and the large ones with high capacity. The low-capacity sub-networks were active on the whole input to first find the task-relevant areas, and then the attention mechanism was used to direct the high-capacity sub-networks to focus on the task-relevant regions. By dong this, the size of the CNNs model has been significantly reduced.

Following this direction, the work in [60] introduced the conditional computation idea, which only computes the gradient for some important neurons. It proposed a sparsely-gated mixture-of-experts Layer (MoE). The MoE module consisted of a number of experts, each a simple feed-forward neural network, and a trainable gating network that selected a sparse combination of the experts to process each input. In [61], dynamic deep neural networks (D2NN) were introduced, which were a type of feed-forward deep neural network that selected and executed a subset of D2NN neurons based on the input.

There have been other attempts to reduce the number of parameters of neural networks by replacing the fully connected layer with global average pooling [44], [62]. Network architecture such as GoogleNet or Network in Network, can achieve state-of-the-art results on several benchmarks by adopting this idea. However, these architectures have not been fully optimized the utilization of the computing resources inside the network. This problem was noted by Szegedy et al. [62] and motivated them to increase the depth and width of the network while keeping the computational budget constant.

The work in [63] targeted the Residual Network based model with a spatially varying computation time, called stochastic depth, which enabled the seemingly contradictory setup to train short networks and used deep networks at test time. It started with very deep networks, while during training, for each mini-batch, randomly dropped a subset of layers and bypassed them with the identity function. Following this direction, thew work in [64] proposed a pyramidal residual networks with stochastic depth. In [65], Wu et al. proposed an approach that learns to dynamically choose which layers of a deep network to execute during inference so as to best reduce total computation. Veit et al. exploited convolutional networks with adaptive inference graphs to adaptively define their network topology conditioned on the input image [66].

Other approaches to reduce the convolutional overheads include using FFT based convolutions [67] and fast convolution using the Winograd algorithm [68]. Zhai et al. [69] proposed a strategy call stochastic spatial sampling pooling, which speed-up the pooling operations by a more general stochastic version. Saeedan et al. presented a novel pooling layer for convolutional neural networks termed detail-preserving pooling (DPP), based on the idea of inverse bilateral filters [70]. Those works only aim to speed up the computation but not reduce the memory storage.

VII. BENCHMARKS, EVALUATION AND DATABASES

In the past five years the deep learning community had made great efforts in benchmark models. One of the most well-known model used in compression and acceleration for CNNs is Alexnet [1], which has been occasionally used for assessing the performance of compression. Other popular standard models include LeNets [71], All-CNN-nets [72] and many others. LeNet-300-100 is a fully connected network with two hidden layers, with 300 and 100 neurons each. LeNet-5 is a convolutional network that has two convolutional layers and two fully connected layers. Recently, more and more state-of-the-art architectures are used as baseline models in many works, including network in networks (NIN) [73], VGG nets [74] and residual networks (ResNet) [75]. Table IV summarizes the baseline models commonly used in several typical compression methods.

The standard criteria to measure the quality of model compression and acceleration are the compression and the speedup rates. Assume that a is the number of the parameters in the original model M and a^* is that of the compressed model M^*, then the compression rate $\alpha(M, M^*)$ of M^* over M is

$$\alpha(M, M^*) = \frac{a}{a^*}. \tag{8}$$

TABLE IV
SUMMARIZATION OF BASELINE MODELS USED IN DIFFERENT
REPRESENTATIVE WORKS OF NETWORK COMPRESSION.

Baseline Models	Representative Works
Alexnet [1]	structural matrix [29], [30], [32]
	low-rank factorization [40]
Network in network [73]	low-rank factorization [40]
VGG nets [74]	transferred filters [44]
	low-rank factorization [40]
Residual networks [75]	compact filters [49], stochastic depth [63]
	parameter sharing [24]
All-CNN-nets [72]	transferred filters [45]
LeNets [71]	parameter sharing [24]
	parameter pruning [20], [22]

Another widely used measurement is the index space saving defined in several papers [30], [35] as

$$\beta(M, M^*) = \frac{a - a^*}{a^*}, \qquad (9)$$

where a and a are the number of the dimension of the index space in the original model and that of the compressed model, respectively.

Similarly, given the running time s of M and s^* of M^*, the speedup rate $\delta(M, M^*)$ is defined as:

$$\delta(M, M^*) = \frac{s}{s^*}. \qquad (10)$$

Most work used the average training time per epoch to measure the running time, while in [30], [35], the average testing time was used. Generally, the compression rate and speedup rate are highly correlated, as smaller models often results in faster computation for both the training and the testing stages.

Good compression methods are expected to achieve almost the same performance as the original model with much smaller parameters and less computational time. However, for different applications with different CNN designs, the relation between parameter size and computational time may be different. For example, it is observed that for deep CNNs with fully connected layers, most of the parameters are in the fully connected layers; while for image classification tasks, float point operations are mainly in the first few convolutional layers since each filter is convolved with the whole image, which is usually very large at the beginning. Thus compression and acceleration of the network should focus on different type of layers for different applications.

VIII. DISCUSSION AND CHALLENGES

We summarized recent efforts on compressing and accelerating deep neural networks (DNNs). Here we discuss more details about how to choose different compression approaches, technique challenges and possible solutions in this area.

A. General Suggestions

There is no golden rule to measure which approach is the best. How to choose the proper method really depends on the applications and requirements. Here are some general guidance we can provide:

- If the applications need compacted models from pre-trained deep nets, you can choose either pruning &

sharing or low rank factorization based methods. If you need end-to-end solutions for your problem, the low rank and transferred convolutional filters approaches should be considered.

- For applications in some particular domains (e.g., medical images), methods with human prior (like the transferred convolutional filters, structural matrix) sometimes have benefits. For example, when doing medical images classification, transferred convolutional filters could work well as medical images (like organ) do have the rotation transformation property.

- The approaches of pruning & sharing generally give reasonable compression rate while not hurt the accuracy. Thus for applications which requires stable model performance, it is better to utilize pruning & sharing.

- If your problem involves small/medium size datasets, you can try the knowledge distillation approaches. The compressed student model can take the benefit of transferring knowledge from teacher model, achieving robust performance when datasets are not large.

- As we mentioned before, these aforementioned techniques are orthogonal. It is reasonable to combine two or three of them to maximize the performance. For some specific applications, like object detection, which requires both convolutional and fully connected layers, you can compress the convolutional layers with low rank based method and the fully connected layers with a pruning technique.

B. Technique Challenges

Techniques for deep model compression and acceleration are still in the early stage and the following challenges still need to be addressed.

- Most of the current state-of-the-art approaches build on well-designed CNN models, which have limited freedom to change the configuration (e.g., network architectures, hyper-parameters). To handle more complicated tasks, the furture work should provide more plausible ways to configure the compressed models.

- Hardware constraints in various of small platforms (e.g., mobile, robotic, self-driving car) are still a major problem to hinder the extension of deep CNNs. How to make full use of the limited computational source and how to design special compression methods for such platforms are still challenges that need to be addressed.

- Pruning is an effective way to compress and accelerate CNNs. The current pruning techniques are mostly designed to eliminate connections between neurons. On the other hand, pruning channel can directly reduce the feature map width and shrink the model into a thinner one. It is efficient but also challenging because removing channels might dramatically change the input of the following layer.

- As we mentioned before, methods of structural matrix and transferred convolutional filters impose prior human knowledge to the model, which could significantly affect the performance and stability. It is critical to investigate how to control the impact of those prior knowledge.

- The methods of knowledge distillation provide many benefits such as directly accelerating model without special hardware or implementations. It is still worthy developing KD-based approaches and exploring how to improve their performances.
- Despite the great achievements of these compression approaches, the black box mechanism is still the key barrier to the adoption. For example, why some neurons/connections are pruned is not clear. Exploring the knowledge interpret-ability is still an important challenge.

C. Possible Solutions

To solve the hyper-parameters configuration problem, we can rely on the recent learning-to-learn strategies [76], [77]. This framework provides a mechanism allowing the algorithm to automatically learn how to exploit structure in the problem of interest. Very recently, leveraging reinforcement learning to efficiently sample the design space and improve the model compression has also been tried [78].

Regarding the use of CNNs in different hardware platforms, proposing some hardware-aware approaches is one direction. Wang *et al.* [79] proposed the Hardware-Aware Automated Quantization (HAQ) to take the hardware accelerator's feedback in the design loop. Similar idea can be applied to make CNNs more applicable for different platforms. The work in [80] directly learn the architectures for large-scale target tasks and target hardware platforms.

Channel pruning provides the efficiency benefit on both CPU and GPU because no special implementation is required. But it is also challenging to handle the input configuration. One possible solution is to use the training-based channel pruning methods [81], which focus on imposing sparse constraints on weights during training. However, training from scratch for such method is costly for very deep CNNs. In [82], the authors provided an iterative two-step algorithm to effectively prune channels in each layer. The work in [83] associated a scaling factor with each channel and imposed regularization on these scaling factors during training to automatically identify unimportant channels. On the other hand, Liu *et al.* [84] showed that pruned architecture itself is more crucial and pruning can be useful as an architecture search paradigm.

Exploring new types of knowledge in the teacher models and transferring it to the student models is useful for the knowledge distillation (KD) approaches. Instead of directly reducing and transferring parameters, passing selectivity knowledge of neurons could be helpful. One can derive a way to select essential neurons related to the task [85], [86]. The intuition is that if a neuron is activated in certain regions or samples, that implies these regions or samples share some common properties that may relate to the task.

For methods with the convolutional filters and the structural matrix, we can conclude that the transformation lies in the family of functions that only operations on the spatial dimensions. Hence to address the imposed prior issue, one solution is to provide a generalization of the aforementioned approaches in two aspects: 1) instead of limiting the transformation to belong to a set of predefined transformations, let it be the whole family of spatial transformations applied on 2D filters or matrix, and 2) learn the transformation jointly with all the model parameters.

Despite the classification task, people are also adapting the compacted models in other tasks [87]–[89]. We would like to see more work for applications with larger deep nets (e.g., video and image frames [90], [91]). There is also some work about deep natural language models [92].

IX. ACKNOWLEDGMENTS

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.

[3] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. S. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," *CoRR*, vol. abs/1611.05377, 2016.

[4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014.

[7] Y. W. Q. H. Jiaxiang Wu, Cong Leng and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[9] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 1737–1746.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[11] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," *CoRR*, vol. abs/1612.01543, 2016.

[12] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 3123–3131.

[13] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.

[14] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.

[15] P. Merolla, R. Appuswamy, J. V. Arthur, S. K. Esser, and D. S. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," *CoRR*, vol. abs/1606.01981, 2016.

[16] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *CoRR*, vol. abs/1611.01600, 2016.

[17] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," *CoRR*, vol. abs/1510.03009, 2015.

[18] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed., 1989, pp. 177–185.

[19] Y. L. Cun, J. S. Denker, and S. A. Solla, "Advances in neural information processing systems 2," D. S. Touretzky, Ed., 1990, ch. Optimal Brain Damage, pp. 598–605.

[20] B. Hassibi, D. G. Stork, and S. C. R. Com, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993, pp. 164–171.

[21] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, 2015, pp. 31.1–31.12.

[22] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, ser. NIPS'15, 2015.

[23] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick." JMLR Workshop and Conference Proceedings, 2015.

[24] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," *CoRR*, vol. abs/1702.04008, 2017.

[25] V. Lebedev and V. S. Lempitsky, "Fast convnets using group-wise brain damage," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 2554–2564.

[26] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *European Conference on Computer Vision*, Amsterdam, the Netherlands, October 2016, pp. 662–677.

[27] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2074–2082.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *CoRR*, vol. abs/1608.08710, 2016.

[29] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 3088–3096.

[30] Y. Cheng, F. X. Yu, R. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *International Conference on Computer Vision (ICCV)*, 2015.

[31] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. N. Choudhary, and S. Chang, "Fast neural networks with circulant projections," *CoRR*, vol. abs/1502.03436, 2015.

[32] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang, "Deep fried convnets," in *International Conference on Computer Vision (ICCV)*, 2015.

[33] J. Chun and T. Kailath, *Generalized Displacement Structure for Block-Toeplitz, Toeplitz-block, and Toeplitz-derived Matrices*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 215–236.

[34] M. V. Rakhuba and I. V. Oseledets, "Fast multidimensional convolution in low-rank tensor formats via cross approximation," *SIAM J. Scientific Computing*, vol. 37, no. 2, 2015.

[35] M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas, "Acdc: A structured efficient linear layer," in *International Conference on Learning Representations (ICLR)*, 2016.

[36] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 2013, pp. 2754–2761.

[37] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 1269–1277.

[38] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[39] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *CoRR*, vol. abs/1412.6553, 2014.

[40] C. Tai, T. Xiao, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," vol. abs/1511.06067, 2015.

[41] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. D. Freitas, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., 2013, pp. 2148–2156.

[Online]. Available: http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips26/1053.pdf

[42] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2013.

[43] T. S. Cohen and M. Welling, "Group equivariant convolutional networks," *arXiv preprint arXiv:1602.07576*, 2016.

[44] S. Zhai, Y. Cheng, and Z. M. Zhang, "Doubly convolutional neural networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 1082–1090.

[45] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," *arXiv preprint arXiv:1603.05201*, 2016.

[46] H. Li, W. Ouyang, and X. Wang, "Multi-bias non-linear activation in deep neural networks," *arXiv preprint arXiv:1604.00676*, 2016.

[47] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, "Exploiting cyclic symmetry in convolutional neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, 2016.

[48] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning." *CoRR*, vol. abs/1602.07261, 2016.

[49] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," *CoRR*, vol. abs/1612.01051, 2016.

[50] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, 2006, pp. 535–541.

[51] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 2654–2662.

[52] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[53] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2014.

[54] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian dark knowledge," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 3420–3428.

[55] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, "Face model compression by distilling knowledge from neurons," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 3560–3566.

[56] T. Chen, I. J. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *CoRR*, vol. abs/1511.05641, 2015.

[57] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *CoRR*, vol. abs/1612.03928, 2016.

[58] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[59] A. Almahairi, N. Ballas, T. Cooijmans, Y. Zheng, H. Larochelle, and A. C. Courville, "Dynamic capacity networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 2549–2558.

[60] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017.

[61] D. Wu, L. Pigou, P. Kindermans, N. D. Le, L. Shao, J. Dambre, and J. Odobez, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1583–1597, 2016.

[62] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[63] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, *Deep Networks with Stochastic Depth*, 2016.

[64] Y. Yamada, M. Iwamura, and K. Kise, "Deep pyramidal residual networks with separated stochastic depth," *CoRR*, vol. abs/1612.01230, 2016.

[65] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *CVPR*, 2018.

[66] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," 2018.

[67] M. Mathieu, M. Henaff, and Y. Lecun, *Fast training of convolutional networks through FFTs*, 2014.

[68] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 4013–4021.

[69] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. S. Feris, "S3pool: Pooling with stochastic spatial sampling," *CoRR*, vol. abs/1611.05138, 2016.

[70] F. Saeedan, N. Weber, M. Goesele, and S. Roth, "Detail-preserving pooling in deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[71] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[72] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014.

[73] M. Lin, Q. Chen, and S. Yan, "Network in network," in *ICLR*, 2014.

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[76] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Neural Information Processing Systems (NIPS)*, 2016.

[77] D. Ha, A. Dai, and Q. Le, "Hypernetworks," in *International Conference on Learning Representations 2016*, 2016.

[78] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[79] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[80] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019.

[81] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," pp. 2270–2278, 2016.

[82] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[83] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *ICCV*, 2017.

[84] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *ICLR*, 2019.

[85] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," *ECCV*, 2018.

[86] Y. Chen, N. Wang, and Z. Zhang, "Darkrank: Accelerating deep metric learning via cross sample similarities transfer," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 2852–2859.

[87] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 742–751.

[88] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[89] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 3296–3297.

[90] Y. Cheng, Q. Fan, S. Pankanti, and A. Choudhary, "Temporal sequence modeling for video event detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[91] L. Cao, S.-F. Chang, N. Codella, C. V. Cotton, D. Ellis, L. Gong, M. Hill, G. Hua, J. Kender, M. Merler, Y. Mu, J. R. Smith, and F. X. Yu, "Ibm research and columbia university trecvid-2012 multimedia event detection (med), multimedia event recounting (mer), and semantic indexing (sin) systems," 2012.

[92] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
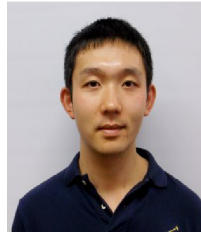
Yu Cheng (yu.cheng@microsoft.com) currently is a Researcher at Microsoft. Before that, he was a Research Staff Member at IBM T.J. Watson Research Center. Yu got his Ph.D. from Northwestern University in 2015 and bachelor from Tsinghua University in 2010. His research is about deep learning in general, with specific interests in the deep generative model, model compression, and transfer learning. He regularly serves on the program committees of top-tier AI conferences such as NIPS, ICML, ICLR, CVPR and ACL.

Duo Wang (d-wang15@mail.tsinghua.edu.cn) received the B.S. degree in automation from the Harbin Institute of Technology, China, in 2015. Currently he is purchasing his Ph.D. degree at the Department of Automation, Tsinghua University, Beijing, P.R. China. Currently his research interests are about deep learning, particularly in few-shot learning and deep generative models. He also works on a lot of applications in computer vision and robotics vision.

Pan Zhou (panzhou@hust.edu.cn) is currently an associate professor with School of Electronic Information and Communications, Wuhan, China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology in 2011. Before that, he received his B.S. degree in the *Advanced Class* of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. His current research interest includes big data analytics and machine learning, security and privacy, and information networks.

Tao Zhang (taozhang@mail.tsinghua.edu.cn) obtained his B.S., M.S., and Ph.D. degrees from Tsinghua University, Beijing, China, in 1993, 1995, and 1999, respectively, and another Ph.D. degree from Saga University, Saga, Japan, in 2002, all in control engineering. He is currently a Professor with the Department of Automation, Tsinghua University. He serves the Associate Dean, School of Information Science and Technology and Head of the Department of Automation. His current research interests include artificial intelligence, robotics, image processing, control theory, and control of spacecraft.