Neural Network Compression and Acceleration: An Overview

Kai Yi
Department of Software Engineering
Xi'an Jiaotong University
yikai2015@stu.xjtu.edu.cn

Xinyu Jiang
Department of Public Management
Xi'an Jiaotong University
kirinna@163.com

Yanlan Lou
Department of Management
Xi'an Jiaotong University
banansorbananas@gmail. com

Yile Qiu
Department of Software Engineering
Xi'an Jiaotong University
851606177qyl@gmail.com

Shuanghe Yu
Department of Software Engineering
Xi'an Jiaotong University
13961835623@qq.com

Jianye Pang
Department of Software Engineering
Xi'an Jiaotong University
1158230985@qq.com

Abstract

A large number of fields (e.g. natural language processing, computer vision) have witnessed the power of deep convolutional neural networks(CNN). However, both training and doing inference are computationally expensive and memory intensive. Therefore, a growing number of researches are attempting to compress convolutional neural networks, which aimed at accelerating CNN-based models without losing accuracy. These techniques are roughly categorized into four schemes: neural network configuration reset, parameter pruning and sharing, low-rank factorization and sparsity, and knowledge distillation. For methods of each scheme, we provide a roughly insightful analysis regarding their performance, related applications, advantages and drawbacks. After that, we survey the evaluation benchmarks and datasets. Finally, we conclude this paper, discuss remaining challenges and our opinions about the direction in this topic.

1 Introduction

In recent years, deep neural networks have recently received lots of attentions, been applied to different applications and won several contests. However, the success of deep neural networks heavily rely on the high computation capability of GPUs. AlexNet [1] introduced convolutional neural networks aimed at handling image classification problems and achieved breakthrough results in the 2012 ImageNet Challenge using a network containing over 60 million parameters with five convolutional layers and three fully-connected layers. What's more, for example, VGG16 [2], a widely used robust and powerful model, has almost one billion parameters. Usually, it takes several days, even more than one week to train the whole model on ImageNet dataset with a NVIDIA TITAN.

Nowadays, it has a tendency to design deeper and wider neural network with more layers and connection between them [1, 2, 3, 4]. Reducing the computation is a vital task to give the ability of deep neural network to plant on mobile devices and embedded systems. Nowadays deep learning on cell phones is a hot topic. Several successful trials has been made to make deep neural network light-weight. Andrew et al. Flattened networks [5] build a network out of fully factorized convolutions and showed the potential of extremely factorized networks. [6] uses depth-wise separable convolutions to build light weight deep neural networks. Zhang et al. [7] utilized point-wise group convolution and channel shuffle to reduce computation cost while maintaining accuracy.

We classify these approaches trying to compress neural network model without losing too much accuracy into four categories: neural network configuration reset, parameter pruning and sharing, low-rank factorization and sparsity, and knowledge distillation. The neural network configuration reset approaches aimed at change the set of architecture’ s configuration such as pooling size and the connection ways among feature maps with different depth. 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 information parameters of the deep CNNs. The knowledge distillation methods learn a distilled model and train a more compact neural network to reproduce the output of a larger network.

2 Neural Network Configuration Reset

Neural network configuration reset means doing some slight changes on the former architecture. Experimental results indicate it always really increase the performance of neural network, including speed or model size. We classify the neural network configuration reset problem into three categories: CNNs traditional components reset, Connection among feature maps reset and inception module reset.

2.1 CNNs Traditional Components Reset

The CNNs Traditional Components, except input and output, can be briefly summarized into three parts: filter size, pooling and feature map. CNNs traditional components reset means changing them to make neural network more efficient.

2.1.1 Filter Size

[1] achieve great success on image classification problems with a high efficient net- work architectures with five convolutional layer with filter size of 7*7 and 5*5 at that time. [8] uses three smaller 3*3 filter to replace a 7*7 one and replace a 5*5 filter with 3*3 one. The result shows these methods really decreased the connection parameters and achieved better accuracy. [3] proposed that 1*1 filter is more efficient. [9], the AlexNet-level accuracy model with 50x fewer parameters and < 1MB, proposed the combination with 1*1 filter and 3*3 filter make the model parameters decreased, which achieved 9x fewer parameters than using 3*3 filter alone.

2.1.2 Pooling

As pooling is not very computational, I will focus on how to use different ways of pooling to get better neural network model with fewer parameters. AlexNet [1] proposed three max pooling layer and max pooling was widely used after then. [8] found that doing less max pooling and modularization can be parameter-saving without losing accuracy. [3, 10, 7] adopted this opinion and got better performance. Meanwhile, [11] also proposed that the highly computational-consuming fully connect layer can be replaced by GAP(Global Average Pooling), which was really elegant with fewer parameters and was widely used as well.

2.1.3 Feature Map

Deeper and smaller feature maps have larger receptive fields but detailed features are hard to extract from them. Meanwhile, larger feature maps contain more features but are computation-consuming and the robustness of them are always not good enough. [10] proposed smaller feature maps. Even they are deeper than [2], they still get better performance on image classification tasks with fewer parameters. [12] also used smaller and well-designed feature maps leading to state-of-the-art performance.

Table 1: Error rate on CIFAR and SVHN datasets

Method	Params	C10	C10+	C100	C100+	SVHN
ResNet110	1.7M	-	6.61	-	-	-
ResNet with Stochastic Depth	1.7M	11.66	5.23	37.80	24.58	1.75
FractalNet	3.6M	10.18	5.22	35.34	23.30	2.01
DenseNet-BC (K = 12)	0.8M	5.92	4.51	24.15	22.27	1.76

2.2 Connection Among Feature Maps Reset

With neural network going deeper, it become very hard to train a model because of gradient vanish. ResNet [10] connected the current convolutional outcome with previous one and it was the champion of ImageNet2016 with a pretty low error rate even excel our human beings. [13] proposed interacting subpaths of different length among feature maps, which surpass the accuracy of ResNet with higher speed. The success of FractalNet shows there are many bootless parameters in the connection of hidden layer. What’s more, DenseNet [12] attempted to connect every feature map with any others. Due to the conclusion drawn above, it achieved better efficiency with nearly half parameters than ResNet.

Table 1 is the error rate of the model mentioned above on CIFAR and SVHN. The result shows modifying the CNNs traditional components can really increase the accuracy with fewer parameters.

2.3 Inception Module Reset

Modularization is experimentally proved that it can lead to better performance. Nowadays, almost all the powerful and successful neural network models, including [2, 11, 3, 4] are modular. For example, the Inception module with dimension reductions in GoogLeNet [3] contains three 1*1 convolutions to reduce parameters and increase the depth of neural network, one 3*3 convolutions and one 5*5 convolutions to maintain the property of convolutional neural network. One the one hand, the well-organized neural network modules are easy to train, the size of model is smaller as well.

3 Parameter Pruning and Sharing

Early works showed that network pruning is effective in reducing the network complexity and addressing the over-fitting problem. After that researcher found pruning originally introduced to reduce the structure in neural networks and hence improve generalization, can be applied to the problem of compression and speed-up. It has been widely studied to compress DNN models, trying to remove parameters which are not crucial to the model performance. According to the way to reduce the redundant (e.g., information redundant or parameter space redundant), these techniques can be further classified into two categories: model quantization and binarization and parameter sharing.

3.1 Quantization and Binarization

Network quantization compresses the original network by reducing the number of bits required to represent each weight. Gong et al. [14] and Wu et al. [15] applied k-means scalar quantization to the parameter values. Vanhoucke et al. [16] showed that 8-bit quantization of the parameters can result in significant speed-up with minimal loss of accuracy. The work in [17] used 16-bit fixed-point representation in stochastic rounding based CNN training, which significantly reduced memory usage and float point operations with little loss in classification accuracy.

The method proposed in [18] first pruned the unimportant connections and retrained the sparsely connected networks. Then it 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. The work achieved the state-of-the-art performance among all parameter quantization based methods. It was shown in [19] 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 1-bit representation of each weight, that is, binary weight neural networks, there are also many works that directly train CNNs with binary weights, for instance, BinaryNet [20] and XNOR Networks [21]. The main idea is to directly learn binary weights or activations during the model training. THe systematic study in [22] showed that networks trained with bak propagation could be resilient to specific weight distortions, including binary weights.

3.2 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 [23]. The optimal brain damage [24] and the optimal brain surgeon [25] 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. These methods followed the way training from scratch manner.

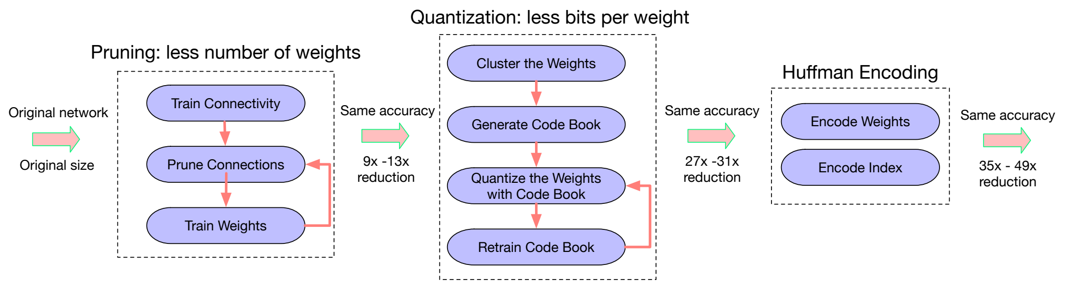

Figure 1: The three-stage compression method proposed in [18]: pruning, quantizatioin and encoding. The input is the original model and the output is the compression model.

A recent trend in this direction is to prune redundant, non-informative weights in a pre-trained CNN model. For example, Srinivasasnd Babu [26] explored the redundancy among neurons, and proposed a data-free pruning method to remove redundant neurons. Han et al. [27] proposed to reduce the total number of parameters and operations in the entire network. Chen et al. [28] 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 [18] removed the redundant connections and quantized the weights, and then used Huffman coding to encode the quantized weights. In [29], a simple regularization method based on soft weight-sharing was proposed, which included both quantization and pruning in one simple training procedure. It is worthy to note that, the above 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 l1 or l2 regularization. The work in [30] 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 [31] , a group-sparse regularizer on neurons was introduced during the training stage to learn compact CNNs with reduced filters. Wen et al. [32] 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 l2-l1 norm regularizers. The work in [33] used l1 norm to select and prune unimportant filters.

3.3 Main-Stream methods Evaluation

As for the quantization and binarization method, the accuracy of such binary nets is significantly lowered when dealing with large CNNs such as GoogLeNet and ResNet. Another drawback of these 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 [34] proposed a proximal Newton algorithm with diagonal Hessian approximation that directly minimizes the loss with respect to binary weights. The work in [35] reduced the time on float point multiplication in the training stage by stochastically binarizing weights and converting multiplication in the hidden state computation to sign changes.

The essential drawbacks of pruning and sharing methods are diversified. First, pruning with l1 or l2 regularization requires more iterations to converge. Furthermore, all pruning criteria require manual setup of sensitivity for layers, which demands fine-tuning of the parameters and could be cumbersome for some questions.

4 Low-Rank Factorization and Sparsity

Convolution operations contribute the bulk of all computations in deep CNNs, thus reducing the convolution layer 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 4D tensor, which is a particularly promising way to remove the redundancy. Regarding to the fully-connected layer, it can be view as a 2D matrix and the low-rankness can also help.

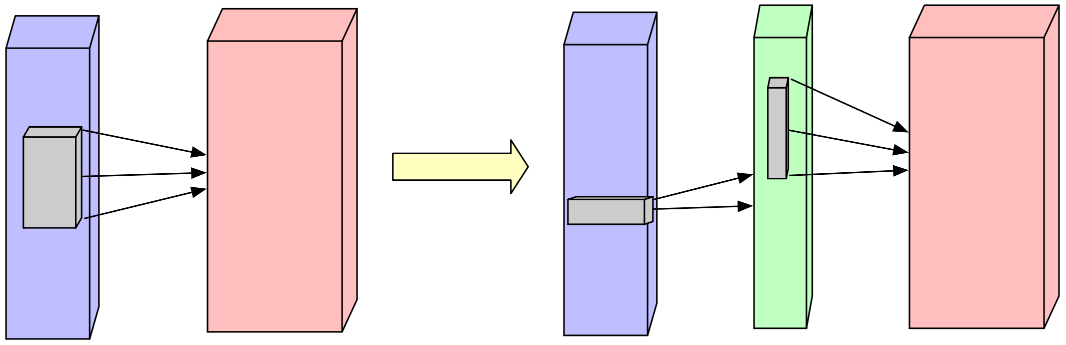

Figure 2: A typical framework of the low-rank regularization method. The left is the original convolutional layer and the right is the low-rank constraint convolutional layer with rank-K.

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 model, a few low-rank approximation and clustering schemes for the convolutional kernels were proposed in [37]. They achieved 2x speedup for a single convolutional layer with 1% drop in classification accuracy. The work in [38] proposed using different tensor decomposition schemes, reporting a 4.5x 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 activations 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 approximation may not exist sometimes. While for the BN scheme, the decomposition always exists. We also compare the performance of both methods in Table 2. The actual speedup and the compression rates are used to measure their performance. We can see that the BN method can achieve slightly better speedup rate while the CP version give higher compression rates.

Note that 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 ow-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.

Table 2: 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	5.0
VGG-16	90.60%	1	1.0
BN Low-rank	90.46%	1.53	2.81
CP Low-rank	90.31%	2.05	2.75
GoogLeNet	92.21%	1.0	1.0
BN Low-rank	91.88%	1.08	2.79
CP Low-rank	91.79%	1.20	2.84

5 Knowledge Distillation

To the best of our knowledge, exploiting knowledge transfer(KT) to compress model was first proposed by Caruana et al. [43]. They trained a compressed/ensemble model of strong classifiers with pseudo-data labeled, and reproduced the output of the original larger network. However, the work is limited to shallow models. The idea has been recently adopted in [44] 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 softened softmax.

The work in [45] 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 teachers output. The framework compressed an ensemble of deep networks (teacher) into a student network of similar depth. To do so, the student was trained to predict the output of the teacher, as well as the true classification labels. Despite its simplicity, KD demonstrates promising results in various image classification tasks. The work in [46] 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. In certain circumstances, FitNet may adversely affect the performance and convergence. All the above methods are validated on MNIST, CIFAR-10, CIFAR-100, SVHN and AFLW benchmark datasets, and simulation results show that these methods match or outperform the teachers performance, while requiring notably fewer parameters and multiplications.

There are several extension along this direction of distillating knowledge . The work in [47] 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, [48] 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 [49] 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. [15] proposed Attention Transfer (AT) to relax the assumption of FitNet. They transferred the attention maps that are summaries of the full activations.

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

6 Evaluation and Datasets

In the past five years, the deep learning community had made great efforts in benchmarks models. Several widely-used models such as VGG16 [2] and ResNet [10] is very robust now.

Due to the excellent performance of AlexNet [1] on handling image classification problem and achieved breakthrough results in the 2012 ImageNet Challenge. A growing number of researchers tends to evaluate whether a novel deep neural network model works well on image classification problems.

6.1 CIFAR-10 & CIFAR-100

One of the most small-scale datasets is CIFAR-10 and CIFAR-100. CIFAR-10 consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. And there are 50000 training images and 10000 test images. CIFAR-10 dataset is divided into five train- ing batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Be- tween them, the training batches contain exactly 5000 images from each class. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. Those dataset is very excellent at evaluating the performance of a newly proposed neural network architecture quickly.

6.2 COCO

COCO [50] is a dataset aimed at object classification in the context of the broader question of scene understanding. This dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images.

6.3 PASCAL VOC

PASCAL VOC [51] is a publicly available dataset of images together with ground truth annotation and standardized evaluation software. PASCAL VOC datasets including PASCAL VOC 07 and PASCAL VOC 12. PASCAL VOC 07 constrains 9,963 images containing 24,640 an- notated objects. And all the objects are classified into 20 classes. As far PASCAL VOC 12, The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentation.

6.4 ImageNet

ImageNet is a widely used dataset which mainly targeted on handling classification problems. ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a ”synonym set” or ”synset”. There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). In ImageNet, they aim to provide on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated.

7 Discussion and Challenges

We only summary the techniques in the perspective of algorithms and new architectures instead of hardware compression and acceleration. The parameter storage compression methods aimed at optimal the storage of data in memory including bit-compression methods and increase the compatibility between neural network architecture and hardware is beyond our scope.

7.1 Typical Rules

1. Well-organized module of neural network with delicate changes on traditional neural network (e.g. the combination of 1*1 filters and 3*3 filters, replace fully connected layer with global average pooling) tends to get a model with fewer parameters with a better performance.

2. Usually the approach of pruning and weight sharing could give reasonable compression rate while not hurt the accuracy. Thus for applications which requires stable model accuracy, it it better to utilize pruning and weight sharing.

3. With neural network going deeper or wider, novel designs with robust efficiency is desired. We often need a lot of data to train our model well. If we only have a small/medium size datasets, the knowledge distillation approaches are great choice. The compressed student model can take the benefit of transferring knowledge from teacher model, making it robust dataset which are not large.

4. Techniques mentioned above are some experimental methods, we can combine two or more to maximize the compression rates and acceleration rates. However, it is a trade-off between model size and speed.

7.2 Challenges

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

1. Most of the current state-of-the-art approaches are built on well-designed CNN models, which have limited free- dom to change the configuration (e.g., network structural, hyper- parameters). To handle more complicated tasks, it should provide more plausible ways to configure the compressed models.

2. Pruning is an effective way to compress and accelerate CNNs. 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. It is important to address how to address this issue.

3. The methods of knowledge distillation (KD) 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 the performance.

4. 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 available and how to design special compression methods for such platforms are still challenges that need to be addressed.

7.3 The Most State-of-the-art Attemptions

Recently, a lot of novel attempts trying to do neural network compression and acceleration were proposed. They are fancy but they remain to be examined whether the potential conclusions extracted from them can lead to the right direction, and whether they are robust and can be transferred successfully. In the end of this paper, I will discuss them to the largest scope of my understanding.

7.3.1 Divide and Conquer

Divide and Conquer is a very powerful algorithm, and it also can be applied into neural network compression and acceleration. [1] used this idea for the first time, but the main motivation about it was to alleviate the heavy load of computation by separate this architecture into two parts.

[6] proposed novel compression model based on depth-wise separable convolutions which is a form of factorized convolutional which factorize a standard convolution into a depth-wise convolution and a 1*1 convolution called a point-wise convolution. For MobileNets the depth-wise convolution applies a single filter to each input channel. The point-wise convolution then applies a 1*1 convolution to combine the outputs the depth-wise convolution. This method provided a promising future of applying light-weight and high-efficient deep neural network on mobile devices. [7] proposed a more powerful light-weight deep neural network by using point-wise group convolution and channel shuffle.

However, the bottleneck of these methods mentioned above are not accurate as larger one. Al- though the division decreased the computation, it still lost many information.

7.3.2 Channel Pruning

Pruning channels is simple but challenging because removing channels in one layer might dramatically change the input of the following layer. Recently, training-based channel pruning works [52, 32] have focused on imposing sparse constrain on weights during training, which could adaptively determine hyper-parameters. However, training from scratch is very costly and results for very deep CNNs on ImageNet have been rarely reported. Inference-time attempts [53, 33] have focused on analysis of the importance of individual weight. The reported speed-up ratio is very limited. [54] propose a new inference-time approach for channel pruning, utilizing redundancy inter channels, based on LASSO regression.

Theoretically, for VGG-16, the method proposed by He et al. [54] achieve $4\times$ acceleration with 1.0% increase of top 5 error. It could also speed up ResNet-50 and Xception-50 by $2\times$ with 1.4%, 1.0% accuracy loss respectively. But it remains to be seen whether this method is robust and really works well in practice.

7.3.3 Using CNNs Model Without Pretraining

Although knowledge distillation works when we only have a small/medium dataset, it is not very elegant and robust in some degree. Nowadays, almost all the mainstream neural network models are pretrained on ImageNet or some other datasets. In order to accelerate the training procedure, we may need to using CNNs model without pretraining. Shen et al [55] proposed DSOD, which attempts to learn deeply supervised object detectors from scratch.

But this paper didn’t describe the detailed information about it so far, thus leads to the confusion whether the fancy result was trained from scratch. Meanwhile, no convincing comparasion was conducted between this method and well pretrained models.

7.3.4 Model Architecture Generation Directly on the Dataset of Interest

A great neural network architecture may heavily rely on the fine-tuning to reach a state-of-the-art performance, which is really tricky. Whether we can generate model architecture directly on the dataset of interest is a really fancy direction. Barret et al [56] proposed a new method to automate this engineering process by learning the model architectures directly on the dataset of interest. When evaluated at different levels of computational cost, accuracies of networks constructed from the convolutional layers exceed those of the state-of-the-art human-designed models.

However, whether this work can be very fast at inference-time remains to be seen. Meanwhile, It really too costly for it to used 500+ GPUs to train a automatic generated model.

Acknowledgments

We would like to thank the support of Institute of Artificial Intelligence and Robotics in China. Prof. Nanning Zheng helped us a lot about deciding the promising research direction. Dr. Chen as well as other graduate students have taught me a lot of skills of fine-tuning. What’s more, the daily discussion part really pushed us to understand the state-of-the-art method well and inspired us. Besides, thanks Tutor Zhang for his kind suggestions and sincere help, which really deeply motivated us.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[4] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016.

[5] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[7] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[8] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *Computer Science*, 2013.

[9] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. 2016.

[13] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[14] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[15] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[16] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4. Citeseer, 2011.

[17] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[18] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[19] Yoojin Choi, Mostafa Elkhamy, and Jungwon Lee. Towards the limit of network quantization. 2016.

[20] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[21] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[22] Paul Merolla, Rathinakumar Appuswamy, John Arthur, Steve K Esser, and Dharmendra Modha. Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv preprint arXiv:1606.01981*, 2016.

[23] Stephen José Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pages 177–185, 1989.

[24] John Fulcher. Computational intelligence: an introduction. In *Computational intelligence: a compendium*, pages 3–78. Springer, 2008.

[25] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

[26] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[27] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[28] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[29] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[30] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2554–2564. IEEE, 2016.

[31] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[32] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[33] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[34] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

[35] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

[36] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2754–2761. IEEE, 2013.

[37] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[38] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[39] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[40] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

[41] Babak Shakibi, Babak Shakibi, Marc'Aurelio Ranzato, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *International Conference on Neural Information Processing Systems*, pages 2148–2156, 2013.

[42] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[43] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, Pa, Usa, August*, pages 535–541, 2006.

[44] Jimmy Ba Lei and Rich Caruana. Do deep nets really need to be deep? In *International Conference on Neural Information Processing Systems*, pages 2654–2662, 2014.

[45] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[46] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[47] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

[48] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, Xiaoou Tang, et al. Face model compression by distilling knowledge from neurons. In *AAAI*, pages 3560–3566, 2016.

[49] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[50] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[51] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[52] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. 2016.

[53] Sajid Anwar and Wonyong Sung. Compact deep convolutional neural networks with coarse pruning. *arXiv preprint arXiv:1610.09639*, 2016.

[54] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017.

[55] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 3, page 7, 2017.

[56] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.