CircConv: A Structured Convolution with Low Complexity

Siyu Liao, Bo Yuan
Department of Electrical and Computer Engineering, Rutgers University
siyu.liao@rutgers.edu, bo.yuan@soe.rutgers.edu

Abstract

Deep neural networks (DNNs), especially deep convolutional neural networks (CNNs), have emerged as the powerful technique in various machine learning applications. However, the large model sizes of DNNs yield high demands on computation resource and weight storage, thereby limiting the practical deployment of DNNs. To overcome these limitations, this paper proposes to impose the circulant structure to the construction of convolutional layers, and hence leads to circulant convolutional layers (CircConvs) and circulant CNNs. The circulant structure and models can be either trained from scratch or re-trained from a pre-trained non-circulant model, thereby making it very flexible for different training environments. Through extensive experiments, such strong structure-imposing approach is proved to be able to substantially reduce the number of parameters of convolutional layers and enable significant saving of computational cost by using fast multiplication of the circulant tensor.

Introduction

Large-scale deep neural networks (DNNs), especially deep convolutional neural networks (CNNs), have achieved extraordinary success in various artificial intelligence applications such as image recognition, video analysis, etc. (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Karpathy et al. 2014). However, the large model sizes of DNNs make themselves both computation-intensive and memory-intensive, thereby potentially hindering the expected widespread deployment of DNNs in many latency-sensitive resource-constrained applications.

To address these limitations, many approaches (Han, Mao, and Dally 2015; Gong et al. 2014; Wen et al. 2016; Feng and Darrell 2015) have been proposed to reduce the computational cost and/or memory footprint of DNNs. In general, those existing efforts can be roughly categorized as two types: fully-connected layer-oriented reduction, such as connection pruning (Han, Mao, and Dally 2015)[1], weight clustering (Gong et al. 2014), and convolutional layer-oriented reduction, such as low rank approximation (Jaderberg, Vedaldi, and Zisserman 2014), sparsity regularization (Wen et al. 2016; Feng and Darrell 2015). Nowadays, consider 1) convolutional layers consume most of the computational processing in DNNs and 2) many state-of-the-art DNNs, such as ResNet (He et al. 2016) and Inception (Szegedy et al. 2015), use very few fully-connected layers that only contain a small portion of parameters of the entire models (e.g. less than 5% parameters for fully-connected layers in ResNet-152), the reduction on computational cost and numbers of parameters of convolutional layers become very essential.

Technical preview and advantages. In this paper we propose to impose the circulant structure to the construction of convolutional layers shown in Figure 1, yielding low-computation-complexity, low-space-cost *circulant convolutional layers (CircConvs)* and the corresponding *circulant CNNs*. Different from prior convolutional layer-oriented compression approaches that are based on the unstructured tensors, the model-size reduction in this paper results from the use of *circulant tensors* (Rezghi and Eldén 2011): The weight tensors for convolutional layers, which were original unstructured, are now constructed in the circulant format, thereby leading to substantial reduction in computational cost and numbers of parameters. In short, the proposed approach brings the following advantages:

1) It reduces the space cost of the convolutional layers because of the inherent spatial regularity of circulant tensors, thereby resulting in high compression ratios for the overall network model sizes.

2) It saves the computation of the convolutional layers by leveraging the fast circulant tensor-specific multiplication algorithm, and hence greatly reduces the computational cost of the entire networks.

3) It enables the improved accuracy for the corresponding circulant CNNs as compared with the similar-size non-circulant CNNs. In other words, the benefits of model-size reduction resulting from using circulant convolutional layers can translate to the increase of accuracy.

4) The circulant structure can be imposed by either training from scratch or re-training from a pre-trained non-circulant model, thereby making circulant CNNs very flexible for different training environments.

We conduct extensive experiments on circulant CNNs and the results show that the proposed circulant structure-imposing approach can effectively reduce the model sizes

[1]It can also bring reduction for convolutional layers to some degree. But the most reduction in the number of parameters is achieved on fully-connected layers.

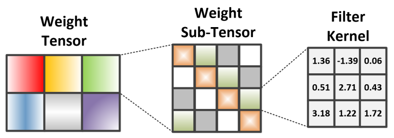

Figure 1: Illustration of a circulant weight tensor. Blocks of the same color in the middle share the same set of kernel weights (on the right). This significantly reduces the total amount of parameters needed to represent this tensor. In addition, the placement of blocks displays a circulant structure, facilitating FFT-based fast algorithms.

and floating point operations (FLOPs) with negligible accuracy drop. In addition, the experiments on wide ResNets show that the proposed approach leads to better accuracy than the non-circulant ResNet models with similar numbers of parameters. Furthermore, we also compare the FLOPs of the proposed circulant CNN models with the non-circulant CNN models, and experimental results show that our proposed method can reduce FLOPs for the inference.

Related Work

Weight pruning/clustering. (Gong et al. 2014) proposes to cluster the weights to reduce the model sizes of DNNs. In that work, various types of weight clustering, including scalar quantization, product quantization, residual quantization, are investigated. (Han, Mao, and Dally 2015) proposes a multi-step compression pipeline comprising of weight pruning, clustering, and quantization to achieve high compression ratios for the entire networks. Notice that as indicated in (Wen et al. 2016), because most parameter reduction in (Gong et al. 2014) and (Han, Mao, and Dally 2015) are achieved on fully-connected layers, the reduction in the computational cost of convolutional layers is not significant. In addition, it is also found that weight parameters in frequency domain can be pruned (Wang et al. 2016) or clustered via a hashing function (Chen et al. 2016).

Low rank approximation (LRA). LRA is an efficient approach to compress DNNs (Jaderberg, Vedaldi, and Zisserman 2014; Sainath et al. 2013; Zhao, Li, and Gong 2016). In (Jaderberg, Vedaldi, and Zisserman 2014), various types of LRA-based solutions are proposed to reduce the numbers of parameters and computational cost of convolutional layers. However, the model-size reduction using LRA usually requires costly reiterations of decomposition and fine-tuning to minimize the approximation error and retain the accuracy.

Sparsity regularization. Increasing the sparsity of network by performing regularization is another popular technique to reduce model sizes of DNNs. (Feng and Darrell 2015), (Girosi, Jones, and Poggio 1995) and (Wen et al. 2016) propose several sparsity-introducing techniques by leveraging different types of regularization, such as L1-norm, group-lasso etc. Though sparsity regularization essentially provides a stable reduction in computational cost, the resulting reduction in model size is not significant.

Structured transform. By using structured matrices, the structured transform can enable very high compression ratios for fully-connected (FC) layers. In (Sindhwani, Sainath, and Kumar 2015; Cheng et al. 2015; Moczulski et al. 2015), weight matrices are constructed in the format of structured matrices to achieve significant reduction in model sizes. (Zhao et al. 2017) further proves that the low displacement rank-based neural networks, which are the generalization of the structured networks can still exhibit universal approximation property. However, the FC layer-specific approaches in (Sindhwani, Sainath, and Kumar 2015; Cheng et al. 2015; Moczulski et al. 2015) cannot be directly applied to the popular and important convolutional layers. Instead, consider FC layer can be viewed as a type of special convolutional layer, our proposed circulant convolution-imposing approach has more generality and is very useful for practical applications. Particularly, compared with (Cheng et al. 2015) and (Sindhwani, Sainath, and Kumar 2015), we generalize the structure from regular 2D weight matrix in fully connected layer to the 4D weight tensor in convolutional layer, where the underlying computation is totally different.

Moreover, this paper is significantly different from two related works (Ding et al. 2017; Wang et al. 2018) in following aspects: 1) we propose an approach on direct operation on 4D circulant tensor with all the algorithm-level details; while prior works require extra processing step to convert tensor to matrix; 2) we perform comprehensive experiments on circulant convolution and present accuracy results and analysis on different datasets; 3) we propose a novel algorithm to convert non-circulant tensor into circulant tensor, thereby making obtaining circulant convolution on pretrained non-circulant models become possible; while prior works can only train the structure from scratch.

Imposing Circulant Structure to Convolutional Layers

Circulant Convolutional Layer

Conventional convolutional layer. In general, a convolutional layer maps a 3-dimensional input tensor $\mathcal{X} \in \mathbb{R}^{W_0 \times H_0 \times C_0}$ into a 3-dimensional output tensor $\mathcal{Y} \in \mathbb{R}^{W_2 \times H_2 \times C_2}$ through convolution with a 4-dimensional kernel tensor $\mathcal{W} \in \mathbb{R}^{W_1 \times H_1 \times C_0 \times C_2}$. Here W_i and H_i for $i = 0, 1, 2$, are the spatial width and height of the input, kernel, and output tensor, respectively; C_0 and C_2 are the number of input channels and output channels. The convolution operation is expressed as follows:

$$\mathcal{Y}(w_2, h_2, c_2) = \sum_{w_1=1}^{W_1} \sum_{h_1=1}^{H_1} \sum_{c_0=1}^{C_0} \Big(\mathcal{X}(w_2 - w_1, h_2 - h_1, c_0) \cdot \mathcal{W}(w_1, h_1, c_0, c_2) \Big). \quad (1)$$

Although stride can be set for convolution, we consider the case of stride that equals to 1 to make a better understanding of circulant convolution. It should be noted that stride wouldn't affect our convolution algorithm and design. Moreover, we can express Eq. 1 in the form of a fiber multiplied by a slice as below:

$$\mathcal{Y}(w_2, h_2, :) = \sum_{w_1=1}^{W_1} \sum_{h_1=1}^{H_1} \Big(\mathcal{X}(w_2 - w_1, h_2 - h_1, :) * \mathcal{W}(w_1, h_1, :, :) \Big), \quad (2)$$

where $*$ and $:$ denote the matrix-vector multiplication and the range of indices, respectively.

Circulant convolutional layer. Different from a conventional convolutional layer, the circulant convolutional layer has a weight tensor $\mathcal{W}$ that exhibits circulant structure. In other words, the $\mathcal{W}$ of a circulant convolution layer is a 4D *circulant tensors* (Rezghi and Eldén 2011). In general, a circulant tensor can exhibit circulant structure along any pair of its dimensions. However, as W_1 and H_1 are usually much smaller than C_0 and C_2 for tensor $\mathcal{W}$, we impose the circulant structure along the input channel and output channel dimensions to achieve high model-size compression ratio. Note that in practice we need to partition the tensor $\mathcal{W}$ into circulant sub-tensors of size $W_1 \times H_1 \times N \times N$. This is necessary because the circulant structure requires that the two corresponding dimension must be equal, while C_0 and C_2 are usually not the same. Larger N means larger compression ratio but it could hurt the model performance to some degree. By adjusting the partition size N we can balance the trade-off between compression ratio and model accuracy.

More specifically, let N be the partition size with $C_0 = R \times N$ and $C_2 = S \times N$ [2], then $\mathcal{W}$ can be defined by a 4-dimensional base tensor $\mathcal{W}' \in \mathbb{R}^{W_1 \times H_1 \times RN \times S}$:

$$\mathcal{W}(w_1, h_1, c_0, c_2) = \mathcal{W}'(w_1, h_1, p, q), \quad (3)$$

where p, q are indices satisfying $\lfloor c_0/N \rfloor = \lfloor p/N \rfloor$, $\lfloor c_2/N \rfloor = q$, and $c_0 - c_2 \equiv p \pmod N$. Fig. 1 illustrates the circulant structure of weight tensor $\mathcal{W}$. From this figure, it can be seen that the circulant structure is imposed to $\mathcal{W}$ along the input/output channel dimensions. The block-circulant weight tensor consists of six circulant weight sub-tensors, where different colors represent different circulant weight sub-tensors. Each circulant weight sub-tensor consists of sixteen kernel filters that are represented in different colors such as green and yellow.

Fast Forward and Backward Propagation Schemes on Circulant Convolutional Layer

Eq. 3 shows that the weight tensor $\mathcal{W}$ of a circulant convolutional layer exhibits the circulant structure and has the reduced number of independent parameters. Besides, according to the tensor theory (Rezghi and Eldén 2011), circulant tensor also has the advantage of fast multiplication. Since multiplication is the kernel computation in neural network training and inference, the existence of fast multiplication of circulant tensor enables the immediate reduction in computational cost. Next, we describe the fast forward and backward propagation schemes by leveraging the fast multiplication of circulant weight tensor.

[2] Zero-padding is needed when N does not divide C_0 or C_2.

Fast forward propagation. We first present the fast forward propagation scheme. Recall that Eq. 2 is the forward propagation scheme for a general convolutional layer. To ease the notation, define $N_k = ((k-1)N + 1, ..., kN)$ for $k = 1, ..., \max(R, S)$, and rewrite Eq. 2 as below:

$$\mathcal{Y}(w_2, h_2, N_i) = \sum_{w_1=1}^{W_1} \sum_{h_1=1}^{H_1} \sum_{j=1}^{R} \Big(\mathcal{X}(w_2 - w_1, h_2 - h_1, N_j) * \mathcal{W}(w_1, h_1, N_j, N_i) \Big), \quad (4)$$

where $i \in \{1, \ldots, S\}$. According to (Rezghi and Eldén 2011; Pan 2012), Fast Fourier Transform (FFT) can be used to accelerate the multiplication of a fiber and a slice of circulant tensor with time complexity reduced from $O(N^2)$ to $O(N \log N)$. Therefore, when $\mathcal{W}$ is a circulant tensor, Eq. 4 can be reformulated using FFT as below:

$$\mathcal{Y}(w_2, h_2, N_i) = \mathrm{ifft}\Big(\sum_{w_1=1}^{W_1} \sum_{h_1=1}^{H_1} \sum_{j=1}^{R} \mathrm{fft}\big(\mathcal{X}(w_2 - w_1, h_2 - h_1, N_j)\big) \circ \mathrm{fft}\big(\mathcal{W}'(w_1, h_1, N_j, N_i)\big) \Big). \quad (5)$$

Here $\circ$ is the element-wise multiplication.

Fast backward propagation. Now consider backward propagation. Given loss function L, it is well known that the goal of backpropgation algorithm (LeCun et al. 1998) is to compute gradients of loss function L with respect to each weight and input. Hence according to the chain rule, the gradient computation for circulant convolutional layer can be derived from Eq. 3 and Eq. 4 as below:

$$\frac{\partial L}{\partial \mathcal{W}'(w_1, h_1, p, q)} = \sum_{w_2=1}^{W_2} \sum_{h_2=1}^{H_2} \sum_{c_2=(q-1)N+1}^{qN} \frac{\partial L}{\partial \mathcal{Y}(w_2, h_2, c_2)} \frac{\partial \mathcal{Y}(w_2, h_2, c_2)}{\partial \mathcal{W}'(w_1, h_1, p, q)}, \quad (6)$$

$$\frac{\partial L}{\partial \mathcal{X}(x, y, c_0)} = \sum_{w_1=1}^{W_1} \sum_{h_1=1}^{H_1} \sum_{c_2 \equiv c_0 (\, mod\, N)} \frac{\partial L}{\partial \mathcal{Y}(w_1 + x, h_1 + y, c_2)} \frac{\partial \mathcal{Y}(w_1 + x, h_1 + y, c_2)}{\partial \mathcal{X}(x, y, c_0)}. \quad (7)$$

Again, according to (Rezghi and Eldén 2011), when $\mathcal{W}$ is a circulant tensor, Eq. 6 and Eq. 7 can also be accelerated by using FFT as below:

$$\frac{\partial L}{\partial \mathcal{W}'(w_1, h_1, N_j, i)} = \mathrm{ifft}(\sum_{w_2=1}^{W_2} \sum_{h_2=1}^{H_2} \mathrm{fft}(\frac{\partial L}{\partial \mathcal{Y}(w_2, h_2, N_i)}) \circ \mathrm{fft}(\mathbf{x}'_\mathbf{j})) \quad (8)$$

$$\frac{\partial L}{\partial \mathcal{X}(x,y,N_j)} = \text{ifft}(\sum_{w_1=1}^{W_1}\sum_{h_1=1}^{H_1}\sum_{i=1}^{S}\text{fft}(\frac{\partial L}{\partial \mathcal{Y}(w_1+x,h_1+y,N_i)}) \circ \text{fft}(\mathbf{w}'_{\mathbf{j},\mathbf{i}})), \quad (9)$$

where $\mathbf{x}'_{\mathbf{j}}$ and $\mathbf{w}'_{\mathbf{j},\mathbf{i}}$ are fibers $\mathcal{X}(w_1,h_1,T)$ and $\mathcal{W}'(x,y,(j-1)N+T,i)$ with $T=(1,N,...,2)$.

Capability of training Circulant CNN from scratch. It should be noted that the gradient computations described in Eq. 8 and Eq. 9 are actually based on $\mathcal{W}'$. Since we can always construct the circulant tensor $\mathcal{W}$ from base tensor $\mathcal{W}'$ using Eq. 3, Eq. 8 and 9 imply that the circulant structure of weight tensor $\mathcal{W}$ is always kept during the training phase. In other words, if we initialize $\mathcal{W}$ as the circulant tensor at the initialization stage of training, then during the training procedure Eq. 8 and 9 can guarantee $\mathcal{W}$ always exhibit circulant structure. Therefore, a circulant CNN can be completely trained from the scratch.

Conversion from Non-circulant Tensor to Circulant Tensor

Forward and backward propagation section indicates that a circulant convolutional layer can be trained from scratch. In this subsection, we also present a conversion technique that can directly convert a non-circulant weight tensor to a circulant one. Such conversion is very useful when a pre-trained model is already available and needs to be imposed with circulant structure.

Specifically, the proposed conversion technique is based on the circulant approximation approach (Chu and Plemmons 2003) used for circulant matrix. In matrix theory, let $\mathbf{Z_1} \in \mathbb{R}^{N\times N}$ denote a permutation matrix as following:

$$\mathbf{Z_1} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & & & & 1 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}. \quad (10)$$

Then a circulant matrix $\mathbf{W}_{circ} \in \mathbb{R}^{N\times N}$ with its first row $\mathbf{w} = (w_0, w_1, \dots, w_{N-1})$ can be represented in the polynomial form of $\mathbf{Z_1}$ as follows:

$$\mathbf{W}_{circ} = \textstyle\sum_{i=0}^{N-1} w_i \mathbf{Z_1}^i. \quad (11)$$

According to (Chu and Plemmons 2003), for a non-circulant matrix $\mathbf{W}_{non-circ} \in \mathbb{R}^{N\times N}$, its nearest circulant matrix $\mathbf{W}_{circ}$ (measured in the Frobenius norm) is given by projection:

$$\begin{gathered} \mathbf{w} = \textit{proj}_N \mathbf{W}_{non-circ}, \\ \forall w_i \in \mathbf{w}, w_i = \frac{1}{N}\langle \mathbf{W}_{non-circ}, \mathbf{Z_1}^i\rangle_{\mathbf{F}}, \end{gathered} \quad (12)$$

where $\langle\cdot,\cdot\rangle_{\mathbf{F}}$ is the Frobenius inner product.

Note that the 4-D weight tensor of a convolutional layer can be viewed as a matrix of size $W_1 \times W_2$ where each entry is a matrix of size $C_0 \times C_2$. Therefore, by using Eq. 12, the conversion from a non-circulant tensor $\mathcal{W}_{non-circ}$ to a circulant tensor $\mathcal{W}_{circ}$ can be achieved by performing the projection as follows:

Table 1: Comparison with (Mathieu, Henaff, and LeCun 2013) in terms of FFT, time and space complexity, where $N = C_0 = C_2$.

Approach	Time Complexity	Space Complexity	FFT Type
Original	$O(W_2H_2W_1H_1N^2)$	$O(W_1H_1N^2)$	N/A
This Work	$O(W_2H_2W_1H_1N\log N)$	$O(W_1H_1RNS)$	1-D
(Mathieu, Henaff, and LeCun 2013)	$O(N^2W_0H_0\log W_0H_0)$	$O(W_1H_1N^2)$	2-D

$$\mathcal{W}'(w_1,h_1,N_j,i) = \textit{proj}_N \mathcal{W}_{non-circ}(w_1,h_1,N_j,N_i), \quad (13)$$

where $\mathcal{W}'$ is the base tensor that defines circulant tensor $\mathcal{W}_{circ}$, and the mapping from $\mathcal{W}'$ to $\mathcal{W}_{circ}$ is given in Eq. 3.

Capability of training Circulant CNN from a pre-trained model. Based on the conversion scheme shown in Eq. 13, any non-circulant convolutional layer of a pre-trained model can be directly converted to a circulant convolutional layer. Typically such direct conversion brings non-negligible accuracy drop incurred by the approximation error. In order to recover the accuracy, further re-training on the converted model is needed by following the backward propagation scheme in Eq. 8 and 9. Consequently, a non-circulant pre-trained model can be imposed with circulant structure by using the proposed circulant conversion and re-training schemes with preserving high accuracy.

Efficiency on Space and Computation

Table 1 summarizes the space and time complexity of the circulant convolutional layers. It can be seen that the proposed circulant structure-imposing approach enables simultaneous improvement on both space efficiency and computation efficiency. Larger N can result in larger FFT size and lower space and time complexity. Also, compared with the 2-D FFT-based fast convolution in (Mathieu, Henaff, and LeCun 2013), our 1-D FFT-based approach has much lower space and time complexity since N is typically much larger than R and S.

Experiments

Dataset, Baseline & Experiment Environment. We evaluate our circulant structure-imposing approaches on two typical image classification datasets: CIFAR-10 (Krizhevsky and Hinton 2009) and ImageNet ILSVRC-2012 (Deng et al. 2009). For each dataset, we take classical network models (ResNet (He et al. 2016) for CIFAR-10 and AlexNet (Krizhevsky, Sutskever, and Hinton 2012) for ImageNet) as the baseline models. The compressed circulant CNN models are generated by replacing convolutional layers of the baseline models with circulant convolutional layers. All models in this paper are trained using NVIDIA GeForce GTX 1080 GPUs and Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz.

Selection of Training Strategy. As presented in Section imposing CircConv, the circulant CNN model can be trained either from scratch or re-trained from a pre-trained non-circulant model. In our experiments we evaluate these two different training strategies on different datasets. Experimental results show that with the same compression configuration setting, the compressed circulant CNN models generated by these two training strategies have very similar test accuracies. Therefore in this paper we only report the results using training-from-scratch strategy.

ResNet on CIFAR-10

In this experiment, ResNet-32 is selected as the baseline model due to its high accuracy and easiness of training. The training data is augmented by following the method in (Simonyan and Zisserman 2014): First pad each side of the image with four pixels and then apply 32×32 sized random crops with horizontal flipping. The compressed ResNet-32 models are trained using stochastic gradient descent (SGD) optimizer with learning rate 0.1, momentum 0.9, batch size 64 and weight decay 0.0001.

Model setting. ResNet-32 consists of 15 *convolutional blocks*, where each convolutional block contains two or three convolutional layers. Considering the number of possible compression configurations on different convolutional layers is very large, we choose to make the layers in the same block have the same compression ratio. In other words, a *block-wise* compression strategy is adopted. Notice that because the first few convolutional layers of ResNet are very sensitive for compression (He et al. 2016), in this experiment we do not impose circulant structure to the convolutional layers in the 1st and 2nd blocks of ResNet-32. Besides, the 6th and 11th blocks are not compressed due to their small weight tensor.

Table 2 shows the detailed compression configurations for different convolutional blocks of ResNet-32. Here we explore 7 different compression configurations and then obtain 7 compressed models. For each compressed model, the compression ratios for its component convolutional blocks are listed in the row direction. Here each number i in a specific compression configuration scheme indicates the compression ratio as i for the convolutional layers in the corresponding convolutional block. When the block is associated with 1, that means the corresponding convolutional block is not compressed. Notice that due to the sensitivity of front blocks, for all the 7 models in Table 2 the compression ratios of the front blocks are typically less than those of the later blocks.

Trade-off between accuracy and model size. Figure 2 shows the test error for 7 compressed models. It can be seen that model 1 even achieves slightly better performance with a smaller model size than the baseline. Moreover, model 2, 3 and 4 achieve around 50% reduction in model size with negligible accuracy drop. With more aggressive compression configurations are selected (such as model 5, 6 and 7), more reduction in model size can be further achieved with slight increase of test error.

Trade-off between accuracy and FLOPs. Our experiment also shows that the use of circulant convolutional layer helps reduce computational cost significantly. As shown in Figure 3, compressed model 1 and 2 achieve fewer FLOPs than baseline with the same or even less test error. For model 3, it can achieve 50% reduction in FLOPs with negligible test error increase. An interesting discovery is that though model 4, 5 and 6 have more aggressive compression configurations than model 3, their corresponding reduction in FLOPs are less than what model 3 achieves. This is because the convolutional layers in the model 3 are mainly compressed with the factor of 2, which corresponds to 2-point FFT computation that only needs real number operations.

Table 2: Compression Configurations. For the convolutional block with compression ratio i, all the convolutional layers in that block has the same compression ratio i.

Partitioned	Model ID						
Block ID	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	1	1	2	2	2	4	4
4	1	1	2	2	2	4	4
5	1	1	2	2	2	4	4
6	1	1	1	1	1	1	1
7	1	1	2	2	4	4	8
8	1	1	2	2	4	4	8
9	1	1	2	2	4	4	8
10	1	1	2	2	4	4	8
11	1	1	1	1	1	1	1
12	1	2	2	4	4	4	16
13	1	2	2	4	4	4	16
14	2	2	2	4	4	4	16
15	2	2	2	4	4	4	16
baseline (He et al. 2016):ResNet-32 without partitioning							

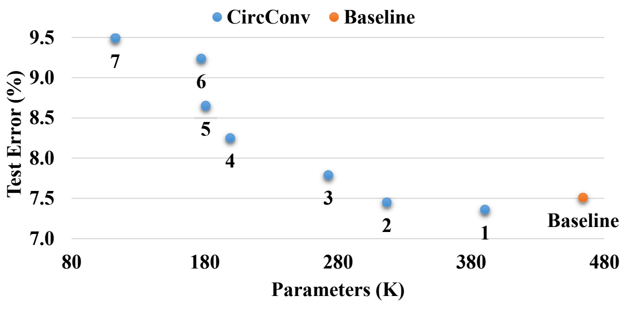

Figure 2: ResNet-32 Test Error and Model Size. Use of circulant convolutional layer can bring half of parameters reduction with negligible test error increase.

Wide ResNet on CIFAR-10

We also conduct the experiment on CIFAR-10 dataset using Wide ResNet (Zagoruyko and Komodakis 2016), which has better performance than conventional ResNet in term of test accuracy. In this experiment, the compressed Wide ResNet models are trained using SGD with learning rate 0.01, momentum 0.9, batch size 64 and weight decay 0.0005.

Model settings. To construct baseline Wide ResNet models, we take the same basic convolutional block structure in

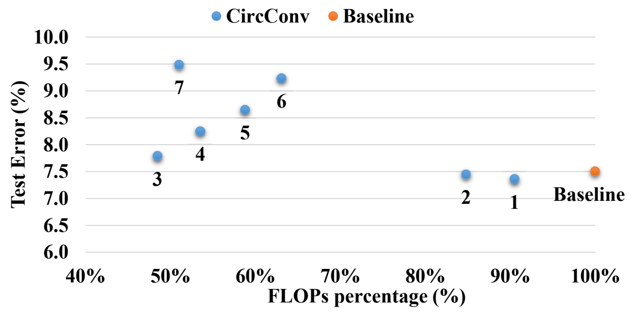

Figure 3: ResNet-32 Test Error and Model Size. Use of circulant convolutional layer can bring half of FLOPs reduction with negligible test error increase.

r

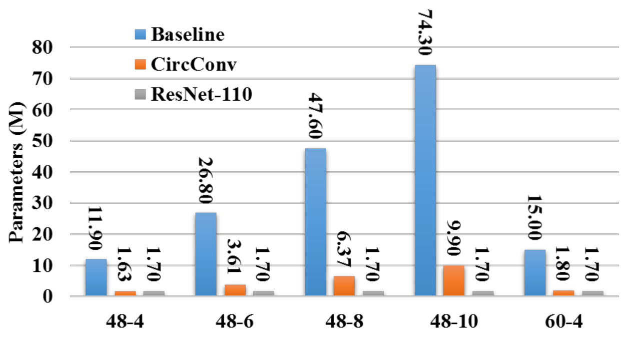

Figure 4: Wide ResNet Model Size Reduction. Compared with baseline models, compressed models achieve similar model size as ResNet-110. Compressed model named like ”48-4” has 48 convolutional layers and widening parameter as 4.

(Zagoruyko and Komodakis 2016) and set different numbers of convolutional blocks and widening parameters for different models. To achieve better performance, we add two more blocks to the convolutional blocks that are wider than $16 \times k$, where k is the inherent widening parameter of each block. Different from the experiment in ResNet experiment section, this experiment on Wide ResNet adopts very aggressive compression strategy: For one convolutional layer, if the numbers of input channels (C_0) and output channels (C_2) are the same, then the compression ratio for that layer is $i = C_0 = C_2$; otherwise the convolutional layer is not compressed. We apply this compression strategy to five different Wide ResNet baselines and obtain five compressed Wide ResNet models. For each compressed model, it is labeled with two numbers (”d-k”), where d and k denote the number of convolutional layers (”depth”) and widening parameter (”width”), respectively. These compressed models are compared with their corresponding baseline models as well as ResNet-110, which achieves the best performance on CIFAR-10 in (He et al. 2016).

Model size reduction. Figure 4 shows the number of parameters of Wide ResNet baselines and the corresponding compressed models after imposing circulant structure. It can be seen that the compressed models greatly reduce the model size. In particular, model ”60-4” can achieve 8.35 times reduction in the model size. Also, it can be seen that the numbers of parameters of Wide ResNet models are similar to the size of ResNet-110 after applying circulant convolutional layer. For instance, Model ”48-4” has around 1.6M parameters which is less than 1.7M for ResNet-110.

Test error analysis. Figure 5 shows test errors of baseline Wide ResNet models and the corresponding compressed models using circulant convolutional layer. It can be seen that all compressed models have slightly test error increase less than 1%. In addition, compared with the state-of-the-art ResNet-110, all of the compressed models have around 1% test error decrease.

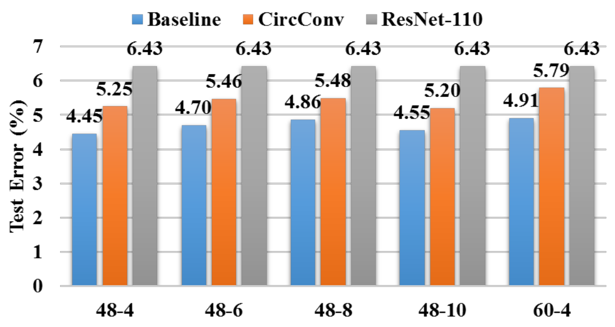

Figure 5: Wide ResNet Test Error. Baseline models are different original Wide ResNets and they are compared with the corresponding compressed models and ResNet-110.

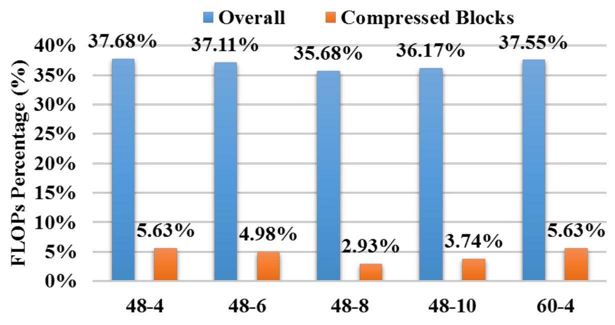

Figure 6: Wide ResNet FLOPs. The overall FLOPs measure the FLOPs percentage of compressed models over corresponding baselines. We also list FLOPs percentage of compressed convolutional blocks over original blocks.

Comparison with ResNet-110. From Figure 5 we can see that all compressed Wide ResNet models have less test error than ResNet-110. Meanwhile, Figure 4 shows these compressed models have similar numbers of parameters as compared with ResNet-110, and model ”48-4” has even fewer parameters. These results demonstrate that circulant structure-imposing approach can be useful in reducing model redundancy and holding less test error while maintaining similar model sizes.

Table 3: Comparison among AlexNet models.

AlexNet Model	Compression Configuration	Test Error (%)	Parameters (%)	FLOPs(%)
Original (Baseline)	N/A	42.9	100	100
CircConv	1-2-2-2-2	**42.75**	50.36	**31.3**
CircConv	1-2-2-4-2	42.99	40.01	**31.3**
CircConv	1-2-4-2-2	43.13	45.19	**31.3**
(Wen et al. 2016)	N/A	42.75	51.20	39.0
(Wen et al. 2016)	N/A	43.00	44.40	43.0
(Wen et al. 2016)	N/A	43.25	42.30	45.0

FLOPs reduction. As shown in Figure 6, we measure the overall FLOPs reduction of Wide ResNet. It is found that the compressed Wide ResNet models can achieve significant reduction in FLOPs: all of them only require around 36% FLOPs as compared to the corresponding baseline models. In addition, the FLOPs reduction for the compressed blocks are very significant. From Figure 6 it can be seen that the FLOPs in the compressed blocks of all compressed models are only less than 6% of the corresponding uncompressed blocks in the original Wide ResNet baseline models.

AlexNet on ImageNet

To test the effectiveness of the proposed circulant-imposing approach on large-scale datasets, we evaluate the performance of circulant CNNs on ImageNet (ILSVRC2012). Here the baseline model is AlexNet (Krizhevsky, Sutskever, and Hinton 2012). All training images are randomly distorted as suggested in (Szegedy et al. 2016). We train our AlexNet models using RMSprop (Tieleman and Hinton 2012) with learning rate 0.01, momentum 0.9, batch size 32 and decay 0.9.

Model settings. We explore three different compression configurations for the five convolutional layers in AlexNet. Table 3 listed the detailed compression configuration schemes by using notation ”a-b-c-d-e”. For instance, ”1-2-2-2-2” means the first convolutional layer is not compressed, and the rest four layers are compressed with the factor of 2. By using these configurations, three compressed AlexNet models are generated and compared with original AlexNet baseline model. Also, since SSL in (Wen et al. 2016) is the state-of-the-art work that explores the relationship between accuracy and compressed model size for AlexNet, we also compare our circulant convolutional layer-based compressed AlexNet models with three SSL regularization-based compressed AlexNet models in (Wen et al. 2016).

Test error analysis. Table 3 shows the test errors of compressed AlexNet models by using circulant structure-imposing and SSL approaches. It can be seen that both these two approaches can render the compressed models with the similar test errors to the original AlexNet model. Among them, the circulant model with ”1-2-2-2-2” compression configuration achieves the least test error.

Model size reduction. Table 3 shows the percentage of number of parameters of each model over original AlexNet model. It can be seen that circulant convolution-based models have similar numbers of parameters to SSL-based models. Among them the circulant convolution-based model with ”1-2-2-4-2” compression configuration has the least number of parameters.

FLOPs reduction. Table 3 shows the percentage of FLOPs of each model over the original AlexNet model. It can be found that all circulant convolution-based models require fewer FLOPs than the SSL regulation-based models. All the circulant convolution-based models have around 31% FLOPs of original uncompressed AlexNet baseline.

Overall comparison. As shown in Table 3 , circulant convolution-based models have similar accuracy to the state-of-the-art SSL models while maintaining similar number of parameters. Meanwhile, Table 3 shows that circulant convolution-based models requires less FLOPs than the SSL models when targeting to the similar accuracy. Therefore, imposing circulant structure to convolutional layer is a very promising accuracy-retained approach to reduce both the space and computational costs.

Acknowledgement

This paper is the collaborative work with Zhe Li, Liang Zhao, Qinru Qiu and Yanzhi Wang. Their names do not appear in the author list due to the mistake during submission. The authors would like to appreciate anonymous reviewers' valuable comments and suggestions. This work is supported by the National Science Foundation Awards CCF-1854742.

Conclusion

In this paper, we propose to impose the circulant structure to convolutional neural network. This structure-imposing approach leads to significant reduction in model size, FLOPs with negligible accuracy drop. Complexity analysis and experiments on different datasets and different network models demonstrate the effectiveness of the proposed approach.

References

Chen, W.; Wilson, J.; Tyree, S.; Weinberger, K. Q.; and Chen, Y. 2016. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1475–1484. ACM.

Cheng, Y.; Yu, F. X.; Feris, R. S.; Kumar, S.; Choudhary, A.; and Chang, S.-F. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, 2857–2865.

Chu, M. T., and Plemmons, R. J. 2003. Real-valued low rank circulant approximation. *SIAM Journal on Matrix Analysis and Applications* 24(3):645–659.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. IEEE.

Ding, C.; Liao, S.; Wang, Y.; Li, Z.; Liu, N.; Zhuo, Y.; Wang, C.; Qian, X.; Bai, Y.; Yuan, G.; et al. 2017. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual*

IEEE/ACM International Symposium on Microarchitecture, 395–408. ACM.

Feng, J., and Darrell, T. 2015. Learning the structure of deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2749–2757.

Girosi, F.; Jones, M.; and Poggio, T. 1995. Regularization theory and neural networks architectures. *Neural computation* 7(2):219–269.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Mathieu, M.; Henaff, M.; and LeCun, Y. 2013. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*.

Moczulski, M.; Denil, M.; Appleyard, J.; and de Freitas, N. 2015. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*.

Pan, V. 2012. *Structured matrices and polynomials: unified superfast algorithms*. Springer Science & Business Media.

Rezghi, M., and Eldén, L. 2011. Diagonalization of tensors with circulant structure. *Linear Algebra and its Applications* 435(3):422–447.

Sainath, T. N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; and Ramabhadran, B. 2013. 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*, 6655–6659. IEEE.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sindhwani, V.; Sainath, T.; and Kumar, S. 2015. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, 3088–3096.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Wang, Y.; Xu, C.; You, S.; Tao, D.; and Xu, C. 2016. Cnnpack: packing convolutional neural networks in the frequency domain. In *Advances in neural information processing systems*, 253–261.

Wang, Y.; Ding, C.; Li, Z.; Yuan, G.; Liao, S.; Ma, X.; Yuan, B.; Qian, X.; Tang, J.; Qiu, Q.; et al. 2018. Towards ultra-high performance and energy efficiency of deep learning systems: an algorithm-hardware co-optimization framework. In *Association for the Advancement of Artificial Intelligence*.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2074–2082.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhao, L.; Liao, S.; Wang, Y.; Tang, J.; and Yuan, B. 2017. Theoretical properties for neural networks with weight matrices of low displacement rank. *arXiv preprint arXiv:1703.00144*.

Zhao, Y.; Li, J.; and Gong, Y. 2016. Low-rank plus diagonal adaptation for deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, 5005–5009. IEEE.