An Exploration of Parameter Redundancy in Deep Networks with Circulant Projections

Yu Cheng[1,3]* Felix X. Yu[2,4]* Rogerio S. Feris[1] Sanjiv Kumar[2] Alok Choudhary[3] Shih-Fu Chang[4]

[1]IBM Research [2]Google Research [3]Northwestern University [4]Columbia University

Abstract

We explore the redundancy of parameters in deep neural networks by replacing the conventional linear projection in fully-connected layers with the circulant projection. The circulant structure substantially reduces memory footprint and enables the use of the Fast Fourier Transform to speed up the computation. Considering a fully-connected neural network layer with d input nodes, and d output nodes, this method improves the time complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$ and space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. The space savings are particularly important for modern deep convolutional neural network architectures, where fully-connected layers typically contain more than 90% of the network parameters. We further show that the gradient computation and optimization of the circulant projections can be performed very efficiently. Our experiments on three standard datasets show that the proposed approach achieves this significant gain in storage and efficiency with minimal increase in error rate compared to neural networks with unstructured projections.

1. Introduction

Deep neural network-based methods have recently achieved dramatic accuracy improvements in many areas of computer vision, including image classification [23, 45, 25], object detection [11, 33], face recognition [38, 36], and text recognition [2, 20].

These high-performing methods rely on deep networks containing millions or even billions of parameters. For example, the work by Krizhevsky *et al.* [23] achieved breakthrough results on the 2012 ImageNet challenge using a network containing 60 million parameters with five convolutional layers and three fully-connected layers. 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 [38, 35]. In architectures that rely only on fully-connected layers, the number of parameters can grow to billions [5].

As larger neural networks are considered, with more layers and also more nodes in each layer, reducing their storage and computational costs becomes critical to meeting the requirements of practical applications. Current efforts towards this goal focus mostly on the optimization of convolutional layers [21, 26, 9], which 1 consume the bulk of computational processing in modern convolutional architectures. We 2 instead explore the redundancy of parameters in fully-connected layers, which are often the bottleneck in terms of memory consumption. Our solution relies on a simple and efficient approach based on *circulant projections* to significantly reduce the storage and computational costs of fully-connected neural network layers, while mantaining competitive error rates. Our work brings the following advantages:

- Fully-connected layers in modern convolutional architectures typically contain over 90% of the network parameters. Our approach significantly reduces the cost to store these networks in memory, which is crucial for GPUs or embedded systems with tight memory constraints.
- The proposed method enables FFT-based computation, which speeds up evaluation of fully connected layers. This is especially useful for neural networks with many fully connected layers, or consisting exclusively of fully connected layers [34, 5].
- With much fewer parameters, our method is empirically shown to require less training data.

1.1. Overview of the proposed approach

A basic computation in a fully-connected neural network layer is

$$h(\mathbf{x}) = \phi(\mathbf{R}\mathbf{x}), \qquad (1)$$

where $\mathbf{R} \in \mathbb{R}^{k \times d}$, and $\phi(\cdot)$ is a element-wise nonlinear activation function. The above operation connects a layer

*indicates equal contributions.

with d nodes, and a layer with k nodes. In convolutional neural networks, the fully connected layers are often used before the final softmax output layer, in order to capture global properties of the image. The computational complexity and space complexity of this linear projection are $\mathcal{O}(dk)$. In practice, k is usually comparable or even larger than d. This leads to computation and space complexity at least $\mathcal{O}(d^2)$, creating a bottleneck for many neural network architectures.

In this work, we propose to impose a *circulant structure* on the projection matrix $\mathbf{R}$ in (1)[1].

$$h(\mathbf{x}) = \phi(\mathbf{R}\mathbf{x}), \quad \mathbf{R} \text{ is a circulant matrix.} \tag{2}$$

This special structure dramatically reduces the number of parameters. It also allows us to use the Fast Fourier Transform (FFT) to speed up the computation. Considering a neural network layer with d input nodes, and d output nodes, the proposed method reduces the space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, and the time complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$. Table 1 compares the time and space complexity of the proposed approach with the conventional method.

Surprisingly, although the circulant matrix is highly structured with a very small number of parameters ($\mathcal{O}(d)$), it captures the global information well, and does not impact the final performance much. We show empirically that our method can provide significant reduction of storage and computational costs while achieving very competitive error rates.

1.2. Organization

Our work is organized as follows. We propose imposing the circulant structure on the linear projection matrix of fully-connected layers of neural networks to speed up computations and reduce storage costs in Section 3. We show a method which can efficiently optimize the neural network while keeping the circulant structure in Section 4. We demonstrate with experiments on visual data that the proposed method can speed up the computation and reduce memory needs while maintaining competitive error rates in Section 5. We begin by reviewing related work in the following section.

2. Related Work

2.1. Deep Learning

In the past few years, deep neural network methods have achieved impressive results in many visual recognition tasks [23, 38, 11, 33]. Recent advances on learning these models include the use of drop-out [17] to prevent overfitting, more effective non-linear activation functions such as rectified linear units [12] or max-out [13], and richer modeling through Network in Network (NiN) [25]. In particular, training high-dimensional networks with large quantities of training data is key to obtaining good results, but at the same time incurs increased computation and storage costs.

[1] A sign flipping operation is applied before the circulant projection matrix. We will present the formal framework in Section 3.1

Method	Time	Space	Time (Learning)
Conventional NN	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(ntd^2)$
Circulant NN	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(ntd \log d)$

Table 1. Comparison of the proposed method with neural networks based on unstructured projections. We assume a fully-connected layer, and the number of input nodes and number of output nodes are both d. t is the number of gradient steps in optimizing the neural network.

2.2. Compressing Neural Networks

The work of Collins and Kohli [3] addresses the problem of memory usage in deep networks by applying sparsity-inducing regularizers during training to encourage zero-weight connections in the convolutional and fully-connected layers. Memory consumption is reduced only at test time, whereas our method cuts down storage costs at both training and testing times. Other approaches exploit low-rank matrix factorization [32, 6] to reduce the number of neural network parameters. In contrast, our approach exploits the redundancy in the parametrization of deep architectures by imposing a circulant structure on the projection matrix, reducing its storage to a single column vector, while allowing the use of FFT for faster computation.

Techniques based on *knowledge distillation* [18] aim to compress the knowledge of a network with a large set of parameters into a compact and fast-to-execute network model. This can be achieved by training a compact model to imitate the soft outputs of a larger model. Romero et al [31] further show that the intermediate representations learned by the large model serve as hints to improve the training process and final performance of the compact model. In contrast, our work does not require the training of an auxiliary model.

Network in Network [25] has been recently proposed as a tool for richer local patch modeling in convolutional networks, where linear convolutions in each layer are replaced by convolving the input with a *micro-network* filter defined, for example, by a multi-layer perceptron. The inception architecture [37] extends this work by using these micro-networks as dimensionality reduction modules to remove computational bottlenecks and reduce storage costs. A key differentiating aspect is that we focus on modeling global dependencies and reducing the cost of fully connected layers, which usually contain the large majority of parameters in standard configurations. Therefore, our work is complementary to these methods. Although [25] suggests that fully-connected layers could be replaced by average pooling without hurting performance for general image classification, other works in computer vision [38] and speech recognition [34] highlight the importance of these layers to capture global dependencies and achieve state-of-the-art results.

2.3. Speeding up Neural Networks

Several recent methods have been proposed to speed-up the computation of neural networks, with focus on convolutional architectures [21, 26, 9, 7]. Related to our work, Mathieu *et al.* [26] use the Fast Fourier Transform to accelerate the computation of convolutional layers, through the well-known convolution theorem. In contrast, our work is focused on the optimization of fully-connected layers by imposing circulant structure on the weight matrix to speed up the computation in both training and testing stages.

In the context of object detection, many techniques such as detector cascades or segmentation-based selective search [39, 9] have been proposed to reduce the number of candidate object locations in which a deep neural network is applied. Our proposed approach is complementary to these techniques.

Other approaches for speeding up neural networks rely on hardware-specific optimizations. For example, fast neural network implementations have been proposed for GPUs [8], CPUs [40], FPGAs [10], and on-chip implementations [30].

Our method is also related to the recent efforts around“shallow” neural networks, which show that sometimes shallow structures can match the performance of deep structures [28, 27, 19, 44].

2.4. Linear Projection with Structured Matrices

Structured matrices have been used in improving the space and computational complexities for different learning paradigms. For example, circulant matrices have been used in dimensionality reduction [41], binary embedding [43] and kernel approximation [44]. It has been shown that circulant structure can be used to save space and computation costs without performance degradation. The properties of circulant matrices have also been exploited to avoid expensive rounds of hard negative mining in training of object detectors [14] and for real-time tracking [15].

One could in principle use other structured matrices such as Hadamard matrices along with a sparse random Gaussian matrix to achieve fast projection as was done in the fast Johnson-Lindenstrauss transform [1, 4], but they are slower than the circulant projection and need more space. We note that very recently, this idea has been studied in [42].

3. Circulant Neural Network Model

In this section, we present the general framework of the circulant neural network model, showing the advantages of this model in achieving more efficient computational processing and storage cost savings.

3.1. Framework

A circulant matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a matrix defined by a vector $\mathbf{r} = (r_0, r_1, \cdots, r_{d-1})$:

$$\mathbf{R} = \mathrm{circ}(\mathbf{r}) := \begin{bmatrix} r_0 & r_{d-1} & \dots & r_2 & r_1 \\ r_1 & r_0 & r_{d-1} & & r_2 \\ \vdots & r_1 & r_0 & \ddots & \vdots \\ r_{d-2} & & \ddots & \ddots & r_{d-1} \\ r_{d-1} & r_{d-2} & \dots & r_1 & r_0 \end{bmatrix}. \quad (3)$$

Let $\mathbf{D}$ be a diagonal matrix with each diagonal entry being a Bernoulli variable (± 1 with probability 1/2). For $\mathbf{x} \in \mathbb{R}^d$, its d-dimensional output is:

$$h(\mathbf{x}) = \phi(\mathbf{RDx}), \quad \mathbf{R} = \mathrm{circ}(\mathbf{r}). \quad (4)$$

The projection with the matrix $\mathbf{D}$ corresponds to a random sign flipping step on the data.

3.2. Motivation

The idea of replacing unstructured projection with circulant projection is motivated by using circulant projections in dimensionality reduction [41], binary embedding [43], and kernel approximation [44]. From the efficiency point of view, this structure creates great advantages in both space and computation (detailed in Section 3.4), and enables efficient optimization procedures (Section 4).

It is shown in previous works [16, 41, 43, 44] that when the parameters of the circulant projection matrix are generated *iid.* from the standard normal distribution, the circulant projection (with the random sign flipping matrix) mimics an unstructured randomized projection. In other words, randomized circulant projections can also be used in different frameworks to preserve pairwise ℓ_2 distance [16, 41], angle [43], and shift-invariant kernels [44]. It is then reasonable to conjecture that randomized circulant projections can also achieve good performance in neural networks (compared to using unstructured randomized matrices). This is indeed true, as we will demonstrate empirically. And similar to binary embedding and kernel approximation, by optimizing the parameters of the projection matrix, we can significantly improve the performance.

3.3. The Need for Matrix $\mathbf{D}$

This matrix is required in prior work [16, 41, 43, 44]. By adding the random sign flipping matrix, the resulting projections are less correlated [16, 41]. In practice, the performance of a circulant neural network drops when the random sign flipping is not performed, which we demonstrate in Section 5.5. To simplify the notation, we omit the matrix $\mathbf{D}$ in the following sections.

3.4. Space and Time Efficiency

The two main advantages of the circulant binary embedding are superior space and time efficiency.

Space Efficiency. Typically, over 90% of the storage cost of convolutional neural networks is due to the fully connected layers. So "compressing" such layers is a very important task. As the proposed circulant projection contains only $2d$ parameters (d floats for $\mathbf{R}$, and d booleans for $\mathbf{D}$)[2], the space complexity is $\mathcal{O}(d)$. This is a significant advantage compared to the conventional fully connected layer which requires $\mathcal{O}(d^2)$ parameters. For example, when $d = 4096$, as in the "AlexNet" [23], the proposed method can decrease memory requirements by a factor of thousands, making the most space consuming component (the fully connected layer) negligible in memory cost. We will show in Section 5 that surprisingly, the circulant neural network can have very competitive performance with such dramatic space savings. The small number of parameters also makes the neural network perform better with limited amount of training data. In addition, we can further improve the performance (yet maintain significant space savings) by adding more nodes ("fatter" layers) and more layers.

Time Efficiency. The structure enables the use of Fast Fourier Transform (FFT) to speed up the computation. For d-dimensional data, the 1-layer circulant neural network has time complexity $\mathcal{O}(d \log d)$. Next we explain how we achieve this time complexity. Given a data point $\mathbf{x}$, $h(\mathbf{x})$ can be efficiently computed as follows. Denote by $\circledast$ the operator of a circulant convolution. Based on the definition of a circulant matrix,

$$\mathbf{R}\mathbf{x} = \mathbf{r} \circledast \mathbf{x}. \tag{5}$$

The convolution above can be computed more efficiently in the Fourier domain, using the Discrete Fourier Transform (DFT), for which a fast algorithm (FFT) is available.

$$h(\mathbf{x}) = \phi\left(\mathcal{F}^{-1}(\mathcal{F}(\mathbf{r}) \circ \mathcal{F}(\mathbf{x}))\right), \tag{6}$$

where $\mathcal{F}(\cdot)$ is the DFT operator, and $\mathcal{F}^{-1}(\cdot)$ is the inverse DFT (IDFT) operator. As DFT and IDFT can be efficiently computed in $\mathcal{O}(d \log d)$ with FFT [29], the proposed approach has time complexity $\mathcal{O}(d \log d)$.

3.5. When $k \neq d$

We have so far assumed the number of nodes in the input layer d to be equal to the number of nodes in the output layer.[3] In this section, we provide extensions to handle the case where $k \neq d$.

When $k < d$, the fully connected layer performs a "compression" of the signal. In this case, we still use the circulant matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ with d parameters, but the output is set to be the first k elements in (4). The circulant neural network is not computationally more efficient in this situation compared to when $k = d$.

When $k > d$, the fully connected layer performs an "expansion" of the signal. In this case, the simplest solution is to use multiple circulant projections, and concatenate their output. This has computational complexity $\mathcal{O}(k \log d)$, and space complexity $\mathcal{O}(k)$. Note that the DFT of the feature vector can be reused in this case. An alternative approach is to extend every feature vector to a k-dimensional vector, by appending $k - d$ zeros. This returns the problem to the previous setting described in section 3.1 with d replaced by k. This gives space complexity $\mathcal{O}(k)$, and computational complexity $\mathcal{O}(k \log k)$. In practice, k is usually at most a few times larger than d. Empirically the two approaches take similar computational time. Our experimental results are based on the second approach.

4. Training Circulant Neural Networks

In this section, we propose a highly efficient way of training circulant neural networks. We also discuss a special type of circulant neural network, where the parameters of the circulant matrix are randomized instead of optimized.

4.1. Gradient Computation

The most critical step for optimizing a neural network given a training set is to compute the gradient of the error function with respect to the network weights. Let us consider the conventional neural network with two layers, where the first layer computes a linear transformation followed by a nonlinear activation function:

$$h(\mathbf{x}) = \phi(\mathbf{R}\mathbf{x}), \tag{7}$$

where $\mathbf{R}$ is an unstructured matrix. We assume the second layer is a linear classifier with weights $\mathbf{w}$. Therefore the output of the two-layer neural network is

$$J(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{R}\mathbf{x}). \tag{8}$$

When training the neural network, computing the gradient of the error function involves computing the gradient of $J(\mathbf{x})$ with respect to each entry of $\mathbf{R}$. It is easy to show that

$$\frac{\partial J(\mathbf{x})}{\partial R_{ij}} = w_i \phi'(\mathbf{R}_{i\cdot}\mathbf{x}) x_j, \tag{9}$$

$$i = 0, \cdots, d-1. \quad j = 0, \cdots, d-1. \tag{10}$$

[2]Note that the circulant matrix is never explicitly computed or stored.

[3]Such a setting is commonly used in fully connected layers of recent convolutional neural network architectures.

where $\phi'(\cdot)$ is the derivative of $\phi(\cdot)$.

Note that (9) suffices for the gradient-based optimization of neural networks, as the gradient *w.r.t.* networks with more layers can simply be computed with the chain rule, leading to the well-known "back-propagation" scheme.

In the circulant case, we need to compute the gradient of the following objective function:

$$J(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{R}\mathbf{x}) = \sum_{i=0}^{d-1} w_i \phi\left(\mathbf{R}_{i\cdot}\mathbf{x}\right), \quad \mathbf{R} = \text{circ}(\mathbf{r}). \tag{11}$$

It is easy to show that

$$\frac{\partial \mathbf{w}^T\phi(\mathbf{R}\mathbf{x})}{\partial r_i} = \mathbf{w}^T\left(\phi'(\mathbf{R}\mathbf{x}) \circ s_{\rightarrow i}(\mathbf{x})\right) \tag{12}$$

$$= s_{\rightarrow i}(\mathbf{x})^T(\mathbf{w} \circ \phi'(\mathbf{R}\mathbf{x})), \tag{13}$$

where $s_{\rightarrow i}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$, right (downwards for a column vector) circularly shifts the vector by one element. Therefore,

$$\begin{aligned}
&\nabla_{\mathbf{r}} J(\mathbf{x}) \qquad (14)\\
=&[s_{\rightarrow 0}(\mathbf{x}), s_{\rightarrow 1}(\mathbf{x}), \cdots, s_{\rightarrow (d-1)}(\mathbf{x})]^T(\mathbf{w} \circ \phi'(\mathbf{R}\mathbf{x}))\\
=&\,\text{circ}(s_{\rightarrow 1}(\text{rev}(\mathbf{x})))(\mathbf{w} \circ \phi'(\mathbf{R}\mathbf{x}))\\
=&s_{\rightarrow 1}(\text{rev}(\mathbf{x})) \circledast (\mathbf{w} \circ \phi'(\mathbf{r} \circledast \mathbf{x})),
\end{aligned}$$

where,

$$\text{rev}(\mathbf{x}) = (x_{d-1}, x_{d-2}, \ldots, x_0),$$

$$s_{\rightarrow 1}(\text{rev}(\mathbf{x})) = (x_0, x_{d-1}, x_{d-2}, \cdots, x_1).$$

The above uses the same trick of converting the circulant matrix multiplication to circulant convolution. Therefore, computing the gradient takes only $\mathcal{O}(d \log d)$ time with FFT. Training a multi-layer neural network is nothing more than applying (13) in each layer with the chain rule.

Note that when $k < d$, we can simply set the last $d - k$ entries of $\mathbf{w}$ in (8) to be zero. And when $k > d$, the above derivations can be applied with minimal changes.

4.2. Randomized Circulant Neural Networks

We also consider the case where the elements of $\mathbf{r}$ in (3) are generated independently from a standard normal distribution $\mathcal{N}(0, 1)$. We refer to these models as randomized circulant neural networks. In this case, the parameters of the circulant projections are defined by random weights, without optimization. In other words, in the optimization process, only the parameters of convolutional layers and the softmax classification layer are optimized. This setting is interesting to study as it provides insight into the "capacity" of the model, independent of specific optimization mechanisms.

We will show empirically that compared to unstructured randomized neural networks, the circulant neural network is faster with the same number of nodes, while achieving similar performance. This surprising result is in line with recent theoretical/empirical discoveries around using circulant projections on dimensionality reduction [41], and binary embedding [43]. It has been shown that the circulant projection behaves similarly to fully randomized projections in terms of the distance preserving properties. In other words, the randomized circulant projection can be seen as a simulation of the unstructured randomized projection, both of which can capture global properties of the data.

In addition, we will show that with the optimizations described in Section 4.1, the error rate of the neural networks improves significantly over the randomized version, meaning that the circulant structure is flexible and powerful enough to be used in a data-dependent fashion.

5. Experiments

We apply our model to three standard datasets in our experiments: MNIST, CIFAR-10, and ImageNet. We note that it is not our goal to obtain state-of-the-art results on these datasets, but rather to provide a fair analysis of the effectiveness of circulant projections in the context of deep neural networks, compared to unstructured projections.

Next we describe our implementation and analysis of accuracy and storage costs on these three datasets, followed by an experiment on reduced training set size.

5.1. Experiments on MNIST

The MNIST digit dataset contains 60,000 training and 10,000 test images of ten handwritten digits (0 to 9), with 28×28 pixels. We use the LeNet network [24] as our basic CNN model, which is known to work well on digit classification tasks. LeNet consists of a convolutional layer followed by a pooling layer, another convolution layer followed by a pooling layer, and then two fully connected layers similar to conventional multilayer perceptrons. We used a slightly different version from the original LeNet implementation, where the sigmoid activations are replaced by Rectified Linear Unit (ReLU) activations for the neurons.

Our implementation extends Caffe [22], by replacing the weight matrix of the proposed circulant projections with the same dimensionality. The results are compared and shown in Table 2. Our fast circulant neural network achieves an error rate of 0.95% on the full MNIST test set, which is very competitive with the 0.92% error rate from the conventional neural network. At the same time, the circulant LeNet is 5.7x more space efficient and 1.43x more time efficient than LeNet.

5.2. Experiments on CIFAR

CIFAR-10 is a dataset of natural 32x32 RGB images covering 10-classes with 50,000 images for training and

Method	Train Error	Test Error	Memory (MB)	Testing Time (sec.)
LeNet	0.35%	0.92%	1.56	3.06
Circulant LeNet	0.47%	0.95%	0.27	2.14

Table 2. Experimental results on MNIST.

Method	Train Error	Test Error	Memory (MB)	Testing Time (sec.)
CIFAR-10 CNN	4.45%	15.60%	0.45	4.56
Circulant CIFAR-10 CNN	6.57%	16.71%	0.12	3.92

Table 3. Experimental results on CIFAR-10.

10,000 for testing. Images in CIFAR-10 vary significantly not only in object position and object scale within each class, but also in object colors and textures.

The CIFAR10-CNN network [17] used in our test consists of 3 convolutional layers, 1 fully-connected layer and 1 softmax layer. Rectified linear units (ReLU) are used as the activation units. The circulant CIFAR10-CNN is implemented by using a circulant weight matrix to replace the fully connected layer. Images are cropped to 24x24 and augmented with horizontal flips, rotation, and scaling transformations. We use an initial learning rate of 0.001 and train for 700-300-50 epochs with their default weight decay.

A comparison of the error rates obtained by circulant and unstructured projections is shown in Table 3. Our efficient approach based on circulant networks obtains test error of 16.71% on this dataset, compared to 15.60% obtained by the conventional model. At the same time, the circulant network is 4x more space efficient and 1.2x more time efficient than the conventional CNN.

5.3. Experiments on ImageNet (ILSVRC-2010)

ImageNet is a dataset containing over 15 million labeled high-resolution images belonging to roughly 22,000 categories. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. A subset of ImageNet with roughly 1000 images in each of 1000 considered categories is used in this challenge. Our experiments were performed on ILSVRC-2010.

We use a standard CNN network – “AlexNet” [23] as the building block. The AlexNet consists of 5 convolutional layers, 2 fully-connected layers and 1 final softmax layer. Rectified linear units (ReLU) are used as the activation units. Pooling layers and response normalization layers are also used between convolutional layers. Our circulant network version involves three components: 1) a feature extractor, 2) fully circulant layers, and 3) a softmax classification layer. For 1 and 3 we utilize the Caffe package [22]. For 2, we implement it with Cuda FFT.

All models are trained using mini-batch stochastic gradient descent (SGD) with momentum on batches of 128 images with the momentum parameter fixed at 0.9. We set the initial learning rate to 0.01, and manually decrease the learning rate if the network stops improving as in [23] according to a schedule determined on a validation set. Dataset augmentation is also exploited.

Table 4 shows the error rate of various models. We have used two types of structures for the proposed method. Circulant CNN 1 replaces the fully connected layers of AlexNet with circulant layers. Circulant CNN 2 uses “fatter” circulant layers compared to Circulant CNN 1: d of Circulant CNN 2 is set to be 2^{14}. In “Reduced AlexNet”, we reduce the parameter size on the fully-connected layer of the original AlexNet to a size similar to our Circulant CNN by cutting d. We have the following observations.

- The performance of Randomized Circulant CNN 1[4] is very competitive with Randomized AlexNet. This is expected as the circulant projection closely simulates a fully randomized projection (Section 3.1).
- Optimization significantly improves the performance for both unstructured projections and circulant projections. The performance of Circulant CNN 1 is very competitive with AlexNet, yet with fraction of the space cost.
- By tweaking the structure to include more parameters, Circulant CNN 2 further drops the error rate to 17.8%, yet it takes only a marginally larger amount of space compared to Circulant CNN 1, an 18x space saving compared to AlexNet.
- With the same memory cost, the Reduced AlexNet performs much worse than Circulant CNN 1.

In addition, one interesting finding is that “dropout”, which is widely used in training CNNs, does not improve the performance of circulant neural networks. In fact it increases the error rate from 19.4% (without dropout) to 20.3% (not shown in the figure). This indicates that the proposed method is more immune to over-fitting. We also show the training time (per image) on the standard and circulant

[4]This is Circulant CNN 1 with randomized circulant projections. In other words, only the convolutional layer is optimized.

Method	Top-5 Error	Top-1 Error	Memory (MB)
Randomized AlexNet	33.5%	61.7%	233.2
Randomized Circulant CNN 1	35.2%	62.8%	20.5
AlexNet	17.1 %	42.8%	233.2
Circulant CNN 1	19.4 %	44.1%	20.5
Circulant CNN 2	17.8 %	43.2%	20.7
Reduced-AlexNet	37.2 %	65.3%	20.7

Table 4. Classification error rate and memory cost on ILSVRC-2010.

d	Full projection	Circulant projection	Speedup	Space Saving (in a fully connected layer)
2^{10}	2.97	2.52	1.18x	1,000x
2^{12}	3.84	2.79	1.38x	4,000x
2^{14}	19.5	5.43	3.60x	30,000x

Table 5. Comparison of training time (ms per image) and space of full projection and circulant projection. Speedup is defined as the time of circulant projection divided by the time of unstructured projection. Space saving is defined as the space of storing the circulant model by the space of storing the unstructured matrix. The unstructured projection matrix in conventional neural networks takes more than 90% of the space cost. In AlexNet, d is 2^{12}.

versions of AlexNet. We vary the number of hidden nodes d in the fully connected layers and compare the training time until the model converges (ms/per image). Table 5 shows the result. Our method provides dramatic space savings and significant speedup compared to the conventional approach.

5.4. Reduced Training Set Size

Compared to the neural network model with unstructured projections, the circulant neural network has fewer parameters. Intuitively, this may bring the benefit of better model generalization. In other words, the circulant neural network might be less data hungry compared to conventional neural networks. To verify our assumption, we report the performance of each model when trained with different training set sizes on the MINST, CIFAR-10, and ILSVRC-2010 datasets. Figure 1 shows test error rate when training on a random subset of the training data. On MNIST and ILSVRC-2010, to achieve a fixed error rate, the circulant models need less data. On CIFAR-10, this improvement is limited as the circulant layer only occupies a small part of the model.

5.5. Results Without $\mathbf{D}$

As noted in Section 3.3, the sign flipping matrix $\mathbf{D}$ of (4) is important in our formulation. We provide some empirical results in this section. On the MNIST dataset, the test error rate increases from 0.95% to 2.45% by dropping $\mathbf{D}$. On the CIFAR-10 dataset, the test error rate increases from 16.71% to 21.33% by dropping $\mathbf{D}$.

6. Discussion

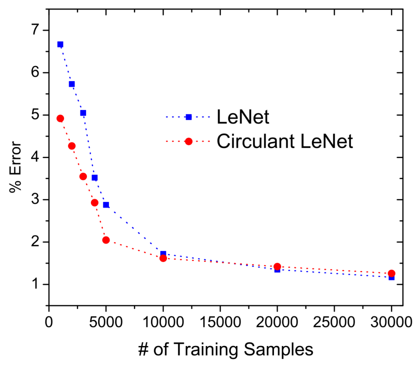
6.1. Fully Connected Layer *vs*. Convolution Layer

The goal of the method developed in this paper is to improve the efficiency of the fully connected layers of neural networks. In convolutional architectures, the fully connected layers are often the bottleneck in terms of the space cost. For example, in "Alexnet", the fully connected layers take 95% of the storage. Remarkably, the proposed method enables dramatic space savings in the fully connected layer (4000x as shown in Table 5), making it negligible in memory cost compared to the convolutional layers. Our discovery resonates with recent work showing that the fully connected layers can be compressed, or even completely removed [25, 37].

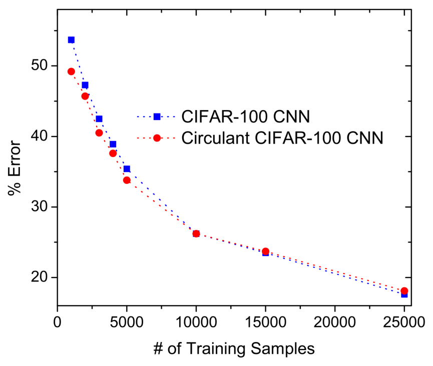
In addition, the fully connected layer costs roughly 20% – 30% of the computation time based on our implementation. The FFT-based implementation can further improve the time cost, though not to the same degree as the realized space savings, if the majority of layers are convolutional. Our method is complementary to the work on improving the time and space cost of convolutional layers [21, 26, 9, 7].

6.2. Circulant Projection *vs*. 2D Convolution

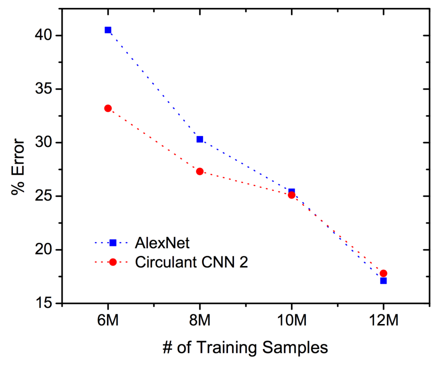
One may notice that, although our approach leverages convolutions for speeding-up computations, it is fundamentally different from the convolutions performed in CNNs. The convolution filters in CNNs are all small 2D filters aiming at capturing local information of the images, whereas the proposed method is used to replace the fully connected layers, which are often "big" layers capturing global information. The operation involved is large 1D convolution rather than small 2D convolution. The circulant projection



(a) MNIST



(b) CIFAR-10



(c) ILSVRC-2010

Figure 1. Test error when training with reduced dataset sizes of circulant CNN and conventional CNN.

can be understood as "simulating" an unstructured projection, with much less cost. Note that one can also apply FFT to compute the convolutions on the 2D convolutional layers, but due to the computational overhead, the speed improvement is generally limited on small-scale problems. In contrast, our method can be used to dramatically speed up and scale the processing in fully connected layers. For instance, when the number of input nodes and output nodes are both 1 million, conventional linear projection is essentially impossible, as it requires TBs of memory. On the other hand, doing a convolution of two 1 million dimensional vectors requires only MBs of memory and tens of milliseconds.

6.3. Towards Larger Neural Networks

Currently, deep neural network models usually contain hundreds of millions of parameters. In real-world applications, there exist problems which involve an increasing amount of data. We may need larger and deeper networks to learn better representations from large amounts of data. Compared to unstructured projections, the circulant projection significantly reduces computation and storage costs. Therefore, with the same amount of resources, circulant neural networks can use deeper as well as larger fully-connected networks. We have conducted preliminary experiments showing that the circulant model can be extended at least 10x deeper than conventional neural networks with the same scale of computational resources.

7. Conclusions

We proposed to use circulant projections to replace the unstructured projections in order to optimize fully connected layers of neural networks. This dramatically improves the computational complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$ and space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. An efficient approach was proposed for optimizing the parameters of the circulant projections. We demonstrated empirically that this optimization can lead to much faster convergence and training time compared to conventional neural networks. Our experimental analysis was carried out on three standard datasets, showing the effectiveness of the proposed approach. We also reported experiments on randomized circulant projections, achieving performance similar to that of unstructured randomized projections. Our ongoing work is to explore different matrix structures to compress and speed up neural networks.

Acknowledgement. Felix X. Yu was supported in part by the IBM PhD Fellowship Award when working on this project at Columbia University. We would like to thank Dan Holtmann-Rice for proofreading the paper.

References

[1] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *ACM Symposium on Theory of Computing*, 2006.

[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013.

[3] A. Collins and P. Kohli. Memory bounded deep convolutional networks. In *http://arxiv.org/abs/1412.1442*, 2014.

[4] A. Dasgupta, R. Kumar, and T. Sarlós. Fast locality-sensitive hashing. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

[5] 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.

[6] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. Freitas. Predicting Parameters in Deep Learning. In *NIPS*, 2013.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, pages 1269–1277, 2014.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[9] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014.

[10] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 257–260. IEEE, 2010.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[14] J. Henriques, J. Carreira, R. Caseiro, and J. Batista. Beyond hard negative mining: Efficient detector learning via block-circulant decomposition. In *ICCV*, 2013.

[15] J. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

[16] A. Hinrichs and J. Vybíral. Johnson-Lindenstrauss lemma for circulant matrices. *Random Structures & Algorithms*, 39(3):391–398, 2011.

[17] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. In *ArXiv e-prints*, 2012.

[18] G. Hinton, O. Vinyals, and J. Dean. Distilling knowledge in a neural network. In *Deep Learning and Representation Learning Workshop, NIPS*, 2014.

[19] P.-S. Huang, H. Avron, T. N. Sainath, V. Sindhwani, and B. Ramabhadran. Kernel methods match deep neural networks on timit. In *ICASSP*. IEEE, 2014.

[20] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*, 2014.

[21] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

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

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

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

[26] M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through FFTs. *arXiv preprint arXiv:1312.5851*, 2013.

[27] M. D. McDonnell, M. D. Tissera, A. van Schaik, and J. Tapson. Fast, simple and accurate handwritten digit classification using extreme learning machines with shaped input-weights. *arXiv preprint arXiv:1412.8307*, 2014.

[28] M. D. McDonnell and T. Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. *arXiv preprint arXiv:1503.04596*, 2015.

[29] A. V. Oppenheim, R. W. Schafer, J. R. Buck, et al. *Discrete-time signal processing*, volume 5. Prentice Hall Upper Saddle River, 1999.

[30] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. K. Likharev, and D. B. Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550):61–64, 2015.

[31] A. Romero, N. Ballas, S. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for thin deep nets. In *ICLR*, 2015.

[32] T. 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 *ICASSP*, 2013.

[33] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[34] H. Soltau, G. Saon, and T. Sainath. Joint training of convoutional and non-convoutional neural networks. In *ICASSP*, 2014.

[35] Y. Sun, X. Wang, , and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *CVPR*, 2015.

[36] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *CVPR*, 2014.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *arXiv preprint arXiv:1409.4842*, 2014.

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

[39] K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011.

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

[41] J. Vybíral. A variant of the johnson–lindenstrauss lemma for circulant matrices. *Journal of Functional Analysis*, 260(4):1096–1105, 2011.

[42] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. *arXiv preprint arXiv:1412.7149*, 2014.

[43] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *International Conference on Machine Learning*, 2014.

[44] F. X. Yu, S. Kumar, H. Rowley, and S.-F. Chang. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*, 2015.

[45] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.