AIMS Mathematics, 8(11): 26188–26198.
DOI: 10.3934/math.20231335
Received: 28 June 2023
Revised: 21 August 2023
Accepted: 31 August 2023
Published: 12 September 2023

http://www.aimspress.com/journal/Math

Research article

New regularization methods for convolutional kernel tensors

Pei-Chang Guo*

School of Science, China University of Geosciences, Beijing 100083, China

* **Correspondence:** Email: gpeichang@126.com.

Abstract: Convolution is a very basic and important operation for convolutional neural networks. For neural network training, how to bound the convolutional layers is a currently popular research topic. Each convolutional layer is represented by a tensor, which corresponds to a structured transformation matrix. The objective is to ensure that the singular values of each transformation matrix are bounded around 1 by changing the entries of the tensor. We propose three new regularization terms for a convolutional kernel tensor and derive the gradient descent algorithm for each penalty function. Numerical examples are presented to demonstrate the effectiveness of the algorithms.

Keywords: regularization; singular values; doubly blocked banded Toeplitz matrices; convolutional kernel tensor
Mathematics Subject Classification: 15B05, 65F15

1. Introduction

Convolutional neural networks (CNNs) are an important class of deep learning models and they have been applied successfully in image understanding in recent years. The use of CNNs is now the dominant approach for almost all recognition and detection tasks [8]. Despite the great success, the training of deep convolutional networks remains to be difficult both theoretically and practically. It has been shown that exploiting the orthogonality to regularize convolutional layers can improve the stability and performance of CNNs and alleviate the issue of unstable gradients [2, 4, 9, 16, 17, 21, 24]. In this paper, we propose three new regularization terms for convolutional layers and derive the gradient descent algorithm for each penalty function.

First we introduce some necessary notations used in this paper. The notation $*$ denotes the convolution arithmetic in neural networks. $vec(X)$ denotes the vectorization of X. When X is a matrix, with the columns of X stacked on top of one another, $vec(X)$ denotes the resulting column vector. When X is a tensor, $vec(X)$ denotes the column vector obtained by stacking the columns of the flattening of X along the first index (please see [7] on the flattening of a tensor). The notation $\ulcorner\cdot\urcorner$

means to round a number to the nearest integer greater than or equal to the number. For a matrix A, $\sigma_{max}(A)$ and $\sigma_{min}(A)$ denote the largest and smallest singular values, respectively.

The tensor is an important concept in many disciplines [5, 15]. Tensors can represent multi-relational data or nonlinear relationships. In CNNs, the convolution is a basic and important operation, which is represented by a tensor. Each convolution arithmetic is associated with a linear structured transformation matrix. Given a convolutional kernel tensor K, $Y = K * X$ is mathematically equivalent to

$$vec(Y) = Mvec(X), \tag{1.1}$$

where M is the structured transformation matrix.

In the field of deep learning, there exist different forms of convolution arithmetic because of different choices about strides and padding patterns [6]. In this paper, without losing generality, the same convolution with unit strides is used to introduce our method. For the one-channel case, a convolutional kernel is represented by a matrix $K \in \mathbb{R}^{k\times k}$ and the input is a matrix $X \in \mathbb{R}^{N\times N}$; then, the output $Y \in \mathbb{R}^{N\times N}$ is computed by

$$Y_{r,s} = (K * X)_{r,s} = \sum_{p\in\{1,\cdots,k\}} \sum_{q\in\{1,\cdots,k\}} X_{r-m+p,s-m+q} K_{p,q},$$

where $m = \lceil k/2 \rceil$ and $X_{i,j} = 0$ if $i \le 0$ or $i > N$, or if $j \le 0$ or $j > N$.

In deep convolutional networks, multi-channel convolutions are more common and a convolutional kernel is represented by a 4 dimensional tensor. For a kernel tensor $K \in \mathbb{R}^{k\times k\times g\times h}$ and the input represented by a 3 dimensional tensor $X \in \mathbb{R}^{N\times N\times g}$, the output $Y = K * X$, $Y \in \mathbb{R}^{N\times N\times h}$ is given by

$$Y_{r,s,c} = (K * X)_{r,s,c} = \sum_{d\in\{1,\cdots,g\}} \sum_{p\in\{1,\cdots,k\}} \sum_{q\in\{1,\cdots,k\}} X_{r-m+p,s-m+q,d} K_{p,q,d,c},$$

where $m = \lceil k/2 \rceil$ and $X_{i,j,d} = 0$ if $i \le 0$ or $i > N$, or if $j \le 0$ or $j > N$.

Deep neural networks are usually layered. The singular values of the Jacobian of a layer bound the factors by which the norms of forward-propagated and backpropagated signals change. In the backward direction, if the singular values of the layers are all close to zero or all significantly larger than 1, gradient exploding or gradient vanishing will occur, which are fundamental obstacles for training deep networks [8, 11, 17]. In the forward direction, if the the singular values of the layers are all bounded, the computations will be more stable, the generalization error can be bounded and the robustness to adversarial examples can be improved [1, 4, 13, 19, 20, 25]. Therefore, it is desirable to constrain the operator norms of network layers. The stability and Hopf bifurcation of some delayed neural networks have been investigated [14, 22, 23]. Convolutional layers are important components of CNNs. In this paper we will give three new regularization terms for the singular values of convolutional layers and develop the gradient descent algorithms; thus, we can modify the singular values of M in (1.1) as desired by changing the entries of K.

In the field of deep learning, there have been many research papers studying how to enforce the orthogonality or spectral norm regularization on the weights of a neural network [2, 4, 16, 24]. Unlike the preceding papers including [2, 4, 16, 24] and the references therein, in this paper we handle convolutions differently. They get the $h \times (gkk)$ matrix by reshaping the kernel $K \in \mathbb{R}^{k\times k\times g\times h}$; they then enforce the constraint directly on the $h \times (gkk)$ matrix. We enforce the constraint on the

transformation matrix M associated with the convolutional kernel tensor K. In [17], the authors project a convolutional layer onto an operator-norm ball and confirm that this is an effective regularizer by conducting numerical experiments. Although the projection method in [17] can effectively prevent the singular values of the transformation matrix from being large, it can't prevent the singular values from being too small. In [10, 21], regularization methods are given to ensure that the transformation matrix is near orthogonal, where the largest and smallest singular values are modified simultaneously.

In this paper we present new regularization methods for the convolutional kernel tensor K. We have two main contributions. First, the new proposed regularization terms can decrease the largest singular value and increase the smallest singular value of convolutional layers independently. Thus the regularization will be more flexible and targeted, depending on the practical need during the training process. Existing methods have no clear impact on the singular values of the transformation matrix M, or they cannot effectively prevent the singular values from being smaller, or can only simultaneously decrease the largest singular value and increase the smallest singular value [10, 16, 17, 21, 24]. Second, we give the formulae for partial derivatives of the proposed penalty functions versus the convolutional kernel tensor, which are first order perturbation results, revealing how each entry of a convolutional kernel tensor affects the singular values of the associated structured transformation matrix.

The rest of the paper is organized as follows: In Section 2, as a warm-up, we handle the one-channel case in which the kernel K is a $k \times k$ matrix. We propose the penalty functions and give the formulas for computing partial derivatives. In Section 3, we handle the multi-channel case, where the kernel is represented by a tensor $K \in \mathbb{R}^{k\times k\times g\times h}$. We also propose the penalty functions and give the gradient descent algorithms. In Section 4, we present numerical results to show that the proposed methods are effective. In Section 5, some conclusions and discussions are given.

2. One-channel convolution

For the one-channel case, the convolutional kernel is a $k\times k$ matrix, and there exist one input channel and one output channel. Suppose that the convolutional kernel K is a 3×3 matrix and the input data matrix is $N \times N$; we show the form of the associated structured transformation matrix. Here,

$$K = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}.$$

For $Y = K * X$, the linear transformation matrix M satisfies the equation $vec(Y) = Mvec(X)$, so we can get the linear transformation matrix M as

$$M = \begin{pmatrix} A_0 & A_{-1} & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & A_{-1} & \ddots & \ddots & \vdots \\ 0 & A_1 & A_0 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & A_{-1} & 0 \\ \vdots & \ddots & \ddots & A_1 & A_0 & A_{-1} \\ 0 & \cdots & 0 & 0 & A_1 & A_0 \end{pmatrix}, \tag{2.1}$$

where

$$A_0 = \begin{pmatrix} k_{22} & k_{32} & 0 & 0 & \cdots & 0 \\ k_{12} & k_{22} & k_{32} & \ddots & \ddots & \vdots \\ 0 & k_{12} & k_{22} & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & k_{32} & 0 \\ \vdots & \ddots & \ddots & k_{12} & k_{22} & k_{32} \\ 0 & \cdots & 0 & 0 & k_{12} & k_{22} \end{pmatrix}, \quad A_{-1} = \begin{pmatrix} k_{23} & k_{33} & 0 & 0 & \cdots & 0 \\ k_{13} & k_{23} & k_{33} & \ddots & \ddots & \vdots \\ 0 & k_{13} & k_{23} & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & k_{33} & 0 \\ \vdots & \ddots & \ddots & k_{13} & k_{23} & k_{33} \\ 0 & \cdots & 0 & 0 & k_{13} & k_{23} \end{pmatrix},$$

$$A_1 = \begin{pmatrix} k_{21} & k_{31} & 0 & 0 & \cdots & 0 \\ k_{11} & k_{21} & k_{31} & \ddots & \ddots & \vdots \\ 0 & k_{11} & k_{21} & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & k_{31} & 0 \\ \vdots & \ddots & \ddots & k_{11} & k_{21} & k_{31} \\ 0 & \cdots & 0 & 0 & k_{11} & k_{21} \end{pmatrix}.$$

For this case, the $N^2 \times N^2$ matrix M is a doubly blocked banded Toeplitz matrix, i.e., a banded block Toeplitz matrix with its blocks represented as banded Toeplitz matrices. For the details about Toeplitz matrices, we recommend the references [3, 12]. We use $\mathcal{T}$ to represent the set of all matrices with the same structure as M in (2.1), i.e., doubly blocked banded Toeplitz matrices with a fixed bandwidth.

From the structure of M, we see that one entry of K corresponds to more than one entry of M. The value of $K_{p,q}$ will appear at different (i, j) indexes of the matrix M. In this section, We use $\mathbb{S}$ to denote this index set, to which each (i, j) index corresponding to $K_{p,q}$ belongs. That is to say, we have that $m_{ij} = K_{p,q}$ for each $(i, j) \in \mathbb{S}$ and $m_{ij} \neq K_{p,q}$ for each (i, j) that does not satisfy $(i, j) \in \mathbb{S}$.

2.1. Regularization 1 to obtain a smaller Frobenius norm of M

Given a matrix M, the square of the Frobenius norm of M, $\|M\|_F^2$, is the sum of squares of all the entries of M. Meanwhile, it is equal to the sum of squares of all the singular values of M [7]. We will use $\frac{1}{2}\|M\|_F^2$ as the regularization term for the convolutional kernel K to prevent the singular values from being too large, and we derive the formula $\partial \frac{1}{2}\|M\|_F^2 / \partial K_{p,q}$. We give the following simple lemma, which will be useful in the following derivation.

Lemma 2.1. *For $A \in \mathbb{R}^{n\times n}$, in terms of the partial derivative of the square of its Frobenius norm with respect to entries a_{ij}, it holds that $\partial\|A\|_F^2/\partial a_{ij} = 2A$.*

Proof. Combining

$$\partial \Sigma a_{ij}^2 / \partial a_{ij} = 2a_{ij} \text{ with } \|A\|_F^2 = \Sigma a_{ij}^2,$$

we can get that

$$\partial\|A\|_F^2/\partial a_{ij} = 2A.$$

□

As we see, one entry of K corresponds to more than one entry of M. For the entry $K_{p,q}$, the (i, j) index set $\mathbb{S}$ represents the locations in M. According to the chain rule formula about the derivative, in order to get $\partial\|M\|_F^2/\partial K_{p,q}$, we need to compute $\partial\|M\|_F^2/\partial m_{ij}$ for all $(i, j) \in \mathbb{S}$ and take the sum. Then we can organize the above analysis result as the following theorem.

Theorem 2.1. *Let $M \in \mathbb{R}^{n\times n}$ be the structured transformation matrix associated with the kernel $K \in \mathbb{R}^{k\times k}$. Given (p, q), if $\mathbb{S}$ denotes the set of all indices (i, j) such that $m_{ij} = k_{p,q}$, it holds that*

$$\frac{1}{2}\frac{\partial\|M\|_F^2}{\partial K_{p,q}} = \sum_{(i,j)\in\mathbb{S}} m_{ij}. \tag{2.2}$$

Proof. As we see from the structure of M, each entry of K corresponds to more than one entry of M. Given (p, q), as $\mathbb{S}$ denotes the set of all indices (i, j) such that $m_{ij} = k_{p,q}$, combining Lemma 2.1 with the chain rule formula about the derivative, we get

$$\frac{1}{2}\frac{\partial\|M\|_F^2}{\partial K_{p,q}} = \frac{1}{2}\sum_{(i,j)\in\mathbb{S}}\frac{\partial\|M\|_F^2}{\partial m_{ij}} = \sum_{(i,j)\in\mathbb{S}} m_{ij}.$$

□

As we know, the Frobenius norm of a matrix equals the sum of squares of the singular values. Formula (2.2) could be used to implement the gradient descent algorithm for $\|M\|_F^2$. So, we can change the entries of a convolutional kernel K to let singular values of M be smaller.

2.2. Regularization 2 to obtain a larger $\sigma_{min}(M)$

In this subsection, we show how to increase the smallest singular value of M by modifying the entries of K. To compute $\partial\sigma_{min}(M)/\partial K_{p,q}$, we need the following lemma, which is the perturbation analysis result for a simple singular value of a matrix; see [18] for the details.

Lemma 2.2. *For a matrix $A = [a_{ij}] \in \mathbb{R}^{m\times m}$, if σ is a simple singular value, and u and v are respectively the normalized left and right singular vectors associated with σ. Then $\partial\sigma/\partial a_{ij}$ is uv^T.*

The value of $K_{p,q}$ will appear at different (i, j) indexes of the matrix M, where $\mathbb{S}$ is the index set. Therefore we can use the chain rule and Lemma 2.2 to get the next theorem.

Theorem 2.2. *For the one-channel convolutional kernel $K \in \mathbb{R}^{k\times k}$, let $M \in \mathbb{R}^{n\times n}$ be the structured transformation matrix. Assume that $\sigma_{min}(M)$ is simple and $\sigma_{min}(M) > 0$, and that u, v are the normalized left and right singular vectors associated with $\sigma_{min}(M)$. Given (p, q), if $\mathbb{S}$ denotes the set of all indices (i, j) such that $m_{ij} = k_{p,q}$, we have*

$$\partial\sigma_{min}(M)/\partial K_{p,q} = \sum_{(i,j)\in\mathbb{S}} u(i)v(j). \tag{2.3}$$

Proof. Each entry of K corresponds to more than one entry of M. Given (p, q), $\mathbb{S}$ denotes the set of all indices (i, j) such that $m_{ij} = k_{p,q}$. Combining Lemma 2.2 with the chain rule formula for the derivative, we get (2.3). □

Formula (2.3) could be used to implement the gradient descent for the penalty function $\sigma_{min}(M)$. Then we can modify the entries of K to increase $\sigma_{min}(M)$.

2.3. Regularization 3 to ensure that the singular values of M are neither large nor small

Now we can combine Theorems 2.1 and 2.2 to ensure that the singular values of M are neither large nor small. As we know, $\|M\|_F^2$ is the squared sum of all singular values of M. If M is $n \times n$, $\|M\|_F^2$ is the squared sum of n singular values. We may choose $\frac{1}{2}\|M\|_F^2 - n\sigma_{min}(M)$ as the regularization term to ensure that the singular values of M are neither large nor small. This leads to the next theorem.

Theorem 2.3. *Let $M \in \mathbb{R}^{n\times n}$ be the structured transformation matrix corresponding to the one channel convolutional kernel $K \in \mathbb{R}^{k\times k}$. Assume that $\sigma_{min}(M) > 0$ and that $\sigma_{min}(M)$ is simple, and u, v are the normalized left and right singular vectors of M that are respectively associated with $\sigma_{min}(M)$. Given (p, q), if $\mathbb{S}$ denotes the set of all indices (i, j) such that $m_{ij} = k_{p,q}$, we have*

$$\partial(\frac{1}{2}\|M\|_F^2 - n\sigma_{min}(M))/\partial K_{p,q} = \sum_{(i,j)\in\mathbb{S}} (m_{ij} - nu(i)v(j)). \tag{2.4}$$

Proof. Combining (2.2) with (2.3), we can get (2.4). □

3. Multi-channel convolution

For the case of multi-channel convolution, the convolutional kernel is represented by a tensor $K \in \mathbb{R}^{k\times k\times g\times h}$. The tensor $X \in \mathbb{R}^{N\times N\times g}$ denotes the input, where element $X_{i,j,d}$ is the value of the input unit within channel d at row i and column j. Entries of $Y = K * X$, $Y \in \mathbb{R}^{N\times N\times h}$ are computed according to

$$Y_{r,s,c} = (K * X)_{r,s,c} = \sum_{d\in\{1,\cdots,g\}} \sum_{p\in\{1,\cdots,k\}} \sum_{q\in\{1,\cdots,k\}} X_{r-m+p,s-m+q,d} K_{p,q,d,c},$$

where $X_{i,j,d} = 0$ if $i \le 0$ or $i > N$, or if $j \le 0$ or $j > N$. Through calculation, the structured transformation matrix M such that $vec(Y) = Mvec(X)$ is as follows

$$M = \begin{pmatrix} M_{(1)(1)} & M_{(1)(2)} & \cdots & M_{(1)(g)} \\ M_{(2)(1)} & M_{(2)(2)} & \cdots & M_{(2)(g)} \\ \vdots & \vdots & \cdots & \vdots \\ M_{(h)(1)} & M_{(h)(2)} & \cdots & M_{(h)(g)} \end{pmatrix}, \tag{3.1}$$

where $M_{(c)(d)} \in \mathbb{S}$, i.e., $M_{(c)(d)}$ is a $N^2 \times N^2$ doubly blocked banded Toeplitz matrix. $M_{(c)(d)}$ corresponds to the portion $K_{:,:,d,c}$ that is convolved with the d-th input channel to get the c-th output channel.

In this section, we use $\Omega_{p,q,z,y}$ to denote the set of all indexes (i, j) satisfying that $m_{ij} = K_{p,q,z,y}$. That is to say, we have that $m_{ij} = K_{p,q,z,y}$ for each $(i, j) \in \Omega_{p,q,z,y}$ and $m_{ij} \neq K_{p,q,z,y}$ for each (i, j) that does not satisfy $(i, j) \in \Omega_{p,q,z,y}$.

We can generalize the results for one-channel convolution to the multi-channel case, which are summarized as the following two theorems.

Theorem 3.1. *For the convolutional kernel $K \in \mathbb{R}^{k\times k\times g\times h}$, let M be the associated structured transformation matrix as defined in (3.1). Given (p, q, z, y), if $\Omega_{p,q,z,y}$ is the set of all indices (i, j) such that $m_{ij} = k_{p,q,z,y}$, it holds that*

$$\frac{1}{2}\frac{\partial\|M\|_F^2}{\partial K_{p,q,z,y}} = \sum_{(i,j)\in\Omega_{p,q,z,y}} m_{ij}. \tag{3.2}$$

The proof of Theorem 3.1 follows from Lemma 2.1 as in Theorem 2.1; it is omitted here.

Theorem 3.2. *For the convolutional kernel $K \in \mathbb{R}^{k\times k\times g\times h}$, let M be the associated structured transformation matrix as defined in (3.1). Given (p, q, z, y), if $\Omega_{p,q,z,y}$ is the set of all indices (i, j) such that $m_{ij} = k_{p,q,z,y}$, it holds that*

$$\partial\sigma_{min}(M)/\partial K_{p,q,z,y} = \sum_{(i,j)\in\Omega_{p,q,z,y}} u(i)v(j). \tag{3.3}$$

The proof of Theorem 3.2 follows from Lemma 2.2 as in Theorem 2.2; it is omitted here.

Theorem 3.3. *For the convolutional kernel $K \in \mathbb{R}^{k\times k\times g\times h}$, let M be the associated structured transformation matrix as defined in (3.1). Given (p, q, z, y), if $\Omega_{p,q,z,y}$ is the set of all indices (i, j) such that $m_{ij} = k_{p,q,z,y}$, it holds that*

$$\partial(\frac{1}{2}\|M\|_F^2 - min(g,h)N^2\sigma_{min}(M))/\partial K_{p,q,z,y} = \sum_{(i,j)\in\Omega_{p,q,z,y}} (m_{ij} - min(g,h)u(i)v(j)). \tag{3.4}$$

Here, $min(g, h)$ denotes the smaller value of g and h.

Proof. Combining (3.2) with (3.3), we can get (3.4). □

Then we present the detailed gradient descent algorithm for the three different penalty functions, where in Algorithm 3.3, $min(g, h)$ denotes the smaller value of g and h.

Algorithm 3.1. *Gradient descent algorithm for $\mathcal{R}_\alpha(K) = \frac{1}{2}\|M\|_F^2$*

(1) Input: a convolutional kernel tensor $K \in \mathbb{R}^{k\times k\times g\times h}$, step size λ, and input size $N \times N \times g$.
(2) If $\sigma_{max}(M)$ is large:
(3) Compute $G = [\frac{\partial\frac{1}{2}\|M\|_F^2}{\partial k_{p,q,z,y}}]_{p,q,z,y=1}^{k,k,g,h}$ by (3.2);
(4) Update $K = K - \lambda G$;
(5) End

Algorithm 3.2. *Gradient descent algorithm for $\mathcal{R}_\alpha(K) = -\sigma_{min}(M)$*

(1) Input: a convolutional kernel tensor $K \in \mathbb{R}^{k\times k\times g\times h}$, step size λ, and input size $N \times N \times g$.
(2) If $\sigma_{min}(M)$ is small:
(3) Compute $G = [\frac{-\partial\sigma_{min}(M)}{\partial k_{p,q,z,y}}]_{p,q,z,y=1}^{k,k,g,h}$ by (3.3);
(4) Update $K = K - \lambda G$;
(5) End

Algorithm 3.3. *Gradient descent algorithm for $\mathcal{R}_\alpha(K) = \frac{1}{2}\|M\|_F^2 - min(g,h)N^2\sigma_{min}(M)$*

(1) Input: a convolutional kernel tensor $K \in \mathbb{R}^{k\times k\times g\times h}$, step size λ, and input size $N \times N \times g$.
(2) While not converged:
(3) Compute $G = [\frac{\partial(\frac{1}{2}\|M\|_F^2 - min(g,h)N^2\sigma_{min}(M))}{\partial k_{p,q,z,y}}]_{p,q,z,y=1}^{k,k,g,h}$ by (3.4);
(4) Update $K = K - \lambda G$;
(5) End

4. Numerical experiments

We do numerical experiments by using MATLAB R2016b on a laptop. The laptop had specifications of 3.0 GHz and 16GB of memory. M denotes the transformation matrix corresponding to the convolutional kernel tensor. $\sigma_{max}(M)$ and $\sigma_{min}(M)$, the iteration steps (denoted as "iter"), are used to show the effectiveness of the proposed algorithms. We randomly generated multi-channel convolutional kernels using the following command

rand('state',1),
$K = rand(k, k, g, h)$.

We considered $K \in \mathbb{R}^{3\times3\times g\times h}$ with different values of g, h, i.e., kernels of different sizes with 3×3 filters. For each kernel, we used $20 \times 20 \times g$ as the size of the input data matrix. We then minimized the three different penalty functions by using Algorithms 3.1–3.3 respectively.

Regarding the choice of the step size λ, although we have no theoretical result, we have a good rule of thumb. According to our numerical experimental results, the step size $\lambda = 1e - 5$ is suitable for Algorithms 3.1 and 3.3 and the step size $\lambda = 1e - 4$ is suitable for Algorithm 3.2. Regarding the process to obtain $\Omega_{p,q,z,y}$, i.e., the set of all indexes (i, j) such that $m_{ij} = k_{p,q,z,y}$, we first generated a structured matrix A with the entry $k_{p,q,z,y}$ and then we used the MATLAB command "find(A)" to get the row and column subscripts of each nonzero element in A. Besides, at each iteration step we used MATLAB commands "norm(M)" and "cond(M)" to compute the largest and smallest singular value of the new transformation matrix.

We present the results for $3\times3\times3\times1$ and $3\times3\times1\times3$ kernels in the following figures. In Figure 1, we demonstrate the changes of the largest singular value of M by using Algorithm 3.1.

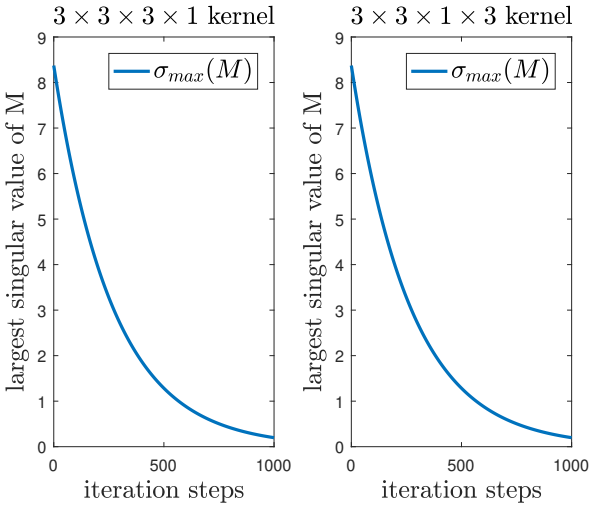

Figure 1. Changes of $\sigma_{max}(M)$ for different kernel sizes.

As the number of iterations increases, $\sigma_{max}(M)$ is reduced. In Figure 2, we demonstrate the changes of the smallest singular value of M by using Algorithm 3.2. As the number of iterations increases, $\sigma_{min}(M)$ increases.

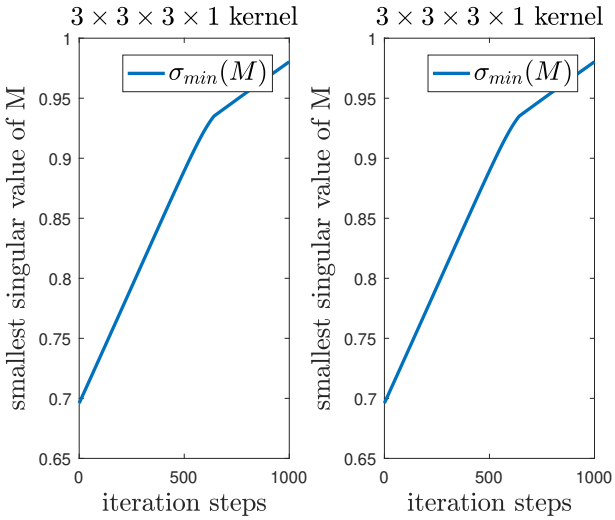

Figure 2. Changes of $\sigma_{min}(M)$ for different kernel sizes.

In Figure 3, the changes of $\sigma_{max}(M)$ and $\sigma_{min}(M)$ are shown, respectively as a result of using Algorithm 3.3. As the number of iterations increases, $\sigma_{max}(M)$ on the left axis scale is reduced and meanwhile $\sigma_{min}(M)$ on the right axis scale increases. The changes of $\sigma_{max}(M)$ and $\sigma_{min}(M)$ in the figures confirm that the three proposed algorithms are effective. In the training of deep neural networks, the practitioners decide which algorithm should be used based on the knowledge about the specific neural network architecture.

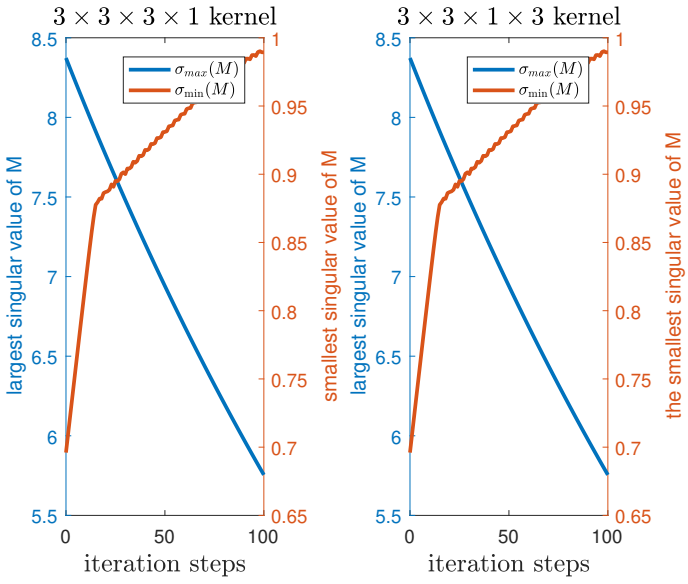

Figure 3. Changes of $\sigma_{max}(M)$ and $\sigma_{min}(M)$ for different kernel sizes.

Numerical experiments were performed by using other randomly generated examples, which include random kernels with each entry uniformly distributed on $[0, 1]$. The figures illustrating the

convergence of $\sigma_{max}(M)$ and $\sigma_{min}(M)$ were similar to the figures presented in the paper.

5. Conclusions

In this paper, we provide new methods to modify the singular values of the convolutional kernel tensors. From the perspective of linear algebra, each convolution operation corresponds to a structured transformation matrix. We have applied the knowledge about linear algebra in combination with the chain rule formula for computing derivatives to get the new regularization methods. New regularization terms for convolutional kernels have been proposed and the gradient decent algorithms for the regularization terms have been provided. The methods are shown to be effective in modifying the singular values of convolutional kernel tensors.

Use of AI tools declaration

The author declares he has not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 12001504) and the Fundamental Research Funds for the Central Universities (Grant No. 2652019320).

Conflict of interest

The author declares no conflict of interest.

References

1. P. L. Bartlett, D. J. Foster, M. Telgarsky, Spectrally-normalized margin bounds for neural networks, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, 6241–6250.
2. A. Brock, T. Lim, J. M. Ritchie, N. Weston, Neural photo editing with introspective adversarial networks, *ArXiv*, 2017. https://doi.org/10.48550/arXiv.1609.07093
3. R. H. F. Chan, X. Jin, *An introduction to iterative toeplitz solvers*, SIAM Press, 2007.
4. M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: improving robustness to adversarial examples, *Proceedings of the 34th International Conference on Machine Learning*, **70** (2017), 854–863.
5. W. Ding, Y. Wei, *Theory and computation of tensors: multi-dimensional arrays*, Academic Press, 2016. https://doi.org/10.1016/C2014-0-04764-8
6. V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, *ArXiv*, 2018. https://doi.org/10.48550/arXiv.1603.07285
7. G. H. Golub, C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, 2013. https://doi.org/10.56021/9781421407944

8. I. J. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016.
9. I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, *ArXiv*, 2015. https://doi.org/10.48550/arXiv.1412.6572
10. P. C. Guo, Q. Ye, On the regularization of convolutional kernels in neural networks, *Linear Multilinear Algebra*, **70** (2022), 2318–2330. https://doi.org/10.1080/03081087.2020.1795058
11. J. F. Kolen, S. C. Kremer, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, Wiley-IEEE Press, 2001. https://doi.org/10.1109/9780470544037.ch14
12. X. Q. Jin, *Developments and applications of block Toeplitz iterative solvers*, Springer Science & Business Media, 2003.
13. J. Kovačević, A. Chebira, *An introduction to frames*, Now Publishers Inc., 2008.
14. P. Li, Y. Lu, C. Xu, J. Ren, Insight into Hopf bifurcation and control methods in fractional order BAM neural networks incorporating symmetric structure and delay, *Cognit. Comput.*, 2023. https://doi.org/10.1007/s12559-023-10155-2
15. L. H. Lim, Tensors in computations, *Acta Numer.*, **30** (2021), 555–764. https://doi.org/10.1017/S0962492921000076
16. T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, *ArXiv*, 2018. https://doi.org/10.48550/arXiv.1802.05957
17. H. Sedghi, V. Gupta, P. M. Long, The singular values of convolutional layers, *ArXiv*, 2018. https://doi.org/10.48550/arXiv.1805.10408
18. G. W. Stewart. *Matrix algorithms*, SIAM Publications Library, 2001. https://doi.org/10.1137/1.9780898718058
19. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, et al., Intriguing properties of neural networks, *ArXiv*, 2013. https://doi.org/10.48550/arXiv.1312.6199
20. Y. Tsuzuku, I. Sato, M. Sugiyama, Lipschitz-Margin training: scalable certification of perturbation invariance for deep neural networks, *Adv. Neural Inf. Process.*, **31** (2018), 6542–6551.
21. J. Wang, Y. Chen, R. Chakraborty, S. X. Yu, Orthogonal convolutional neural networks, *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. https://doi.org/10.1109/CVPR42600.2020.01152
22. C. Xu, Z. Liu, P. Li, J. Yan, L. Yao, Bifurcation mechanism for fractional-order three-triangle multi-delayed neural networks, *Neural Process. Lett.*, 2022. https://doi.org/10.1007/s11063-022-11130-y
23. C. Xu, W. Zhang, Z. Liu, L. Yao, Delay-induced periodic oscillation for fractional-order neural networks with mixed delays, *Neurocomputing*, **488** (2022), 681–693. https://doi.org/10.1016/j.neucom.2021.11.079
24. Y. Yoshida, T. Miyato, Spectral norm regularization for improving the generalizability of deep learning, *ArXiv*, 2017. https://doi.org/10.48550/arXiv.1705.10941
25. C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning (still) requires rethinking generalization, *Commun. ACM*, **64** (2021), 107–115. https://doi.org/10.1145/3446776