A Frobenius Norm Regularization Method for Convolutional Kernel Tensors in Neural Networks

Pei-Chang Guo

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

Correspondence should be addressed to Pei-Chang Guo; gpeichang@126.com

Received 31 January 2022; Revised 28 May 2022; Accepted 25 July 2022; Published 25 August 2022

1. Introduction

In recent years, deep convolutional neural networks have been applied successfully in many fields, such as face recognition, self-driving cars, natural language understanding, and speech recognition [1]. As we know, in the field of deep learning, convolution without the flip is very important arithmetic [2]. From the viewpoint of linear algebra, each convolutional kernel corresponds to a linear transformation matrix. Given the input \( X \) and a kernel \( K \), the output of convolution \( Y = K \ast X \) can be reshaped from the matrix-vector multiplication of a corresponding transformation matrix \( M \) with the reshaped \( X \). We use \( \text{vec}(X) \) to denote the vectorization of \( X \). If \( X \) is a matrix, \( \text{vec}(X) \) is the column vector got by stacking the columns of \( X \) on top of one another. If \( X \) is a tensor, \( \text{vec}(X) \) is the column vector got by stacking the columns of the flattening of \( X \) along the first index (see [3] for more on flattening of a tensor). Thus, given a kernel \( K \), assume \( M \) is the linear transformation matrix corresponding to the kernel \( K \), we have

\[
\text{vec}(Y) = M \text{vec}(X).
\] (1)

Training a neural network can be seen as an optimization problem, which is seeking the optimal weights (parameters) by reaching the minimum of loss function on the training data. Exploding and vanishing gradients are fundamental obstacles to effectively training a deep neural network [4]. The singular values of the Jacobian of a layer bound the factor by which it changes the norm of the backpropagated signal. On the other hand, the generalizability of a model can be improved by reducing the sensitivity of a loss function against the input data perturbation [5–7]. Therefore, when training deep convolutional neural networks, if the singular values of the Jacobian of each layer are close to 1, it can help to avoid the exploding/vanishing gradient problem [4, 8–10] and improve the generalizability of a neural network [7, 11, 12]. This is to let the singular values of each linear transformation matrix \( M \) be bounded around 1. Our contribution in this paper is to give a Frobenius norm penalty function for a convolutional kernel tensor to let the singular values of the corresponding transformation matrix be bounded around 1. We show how to carry out the gradient-type methods. This provides a potentially useful regularization method for the weights of convolutional layers.
a number to the nearest integer greater than or equal to that number. If a convolutional kernel is a matrix \( K \in \mathbb{R}^{k \times k} \) and the input is a matrix \( X \in \mathbb{R}^{N \times N} \), each entry of the output \( Y \in \mathbb{R}^{N \times N} \) is produced by

\[
Y_{r,s} = (K \ast X)_{r,s} = \sum_{p=1}^{k} \sum_{q=1}^{k} X_{r-m+p,s-m+q}K_{p,q},
\]

where \( m = k/2 \) and \( X_{i,j} = 0 \) if \( i \leq 0 \) or \( i > N \), or \( j \leq 0 \) or \( j > N \).

In convolutional neural networks, usually there are multichannels and a convolutional kernel is represented by a 4-dimensional tensor. If a convolutional kernel is a 4-dimensional tensor \( K \in \mathbb{R}^{k \times k \times g \times h} \) and the input is 3-dimensional tensor \( X \in \mathbb{R}^{N \times N \times g} \), each entry of the output \( Y \in \mathbb{R}^{N \times N \times h} \) is produced by

\[
Y_{r,s} = (K \ast X)_{r,s} = \sum_{d=1}^{g} \sum_{c=1}^{h} X_{r-m+p,s-m+q,d}K_{p,q,d,c},
\]

where \( m = k/2 \) and \( X_{i,j,d} = 0 \) if \( i \leq 0 \) or \( i > N \), or \( j \leq 0 \) or \( j > N \). When we refer to the convolution arithmetic in deep learning, only element-wise multiplication and addition are performed and there is no reverse for the convolutional kernel in deep learning.

Given a general kernel \( K \) whose size is \( k \times k \times g \times h \) and the input data matrix size \( N \times N \times g \), and assume \( M \) is the linear transformation matrix corresponding to the kernel \( K \), we will give a regularization term to let the following term toward a smaller value.

\[
\max \left[ |\sigma_{\max}(M) - 1|, |\sigma_{\min}(M) - 1| \right].
\]

But the above term (4) is hard to minimize directly. In this paper, we will use \( \|M^T M - I\|_F^2 \) as the penalty function, where \( \| \cdot \|_F \) denotes the Frobenius norm of a matrix and \( I \) is the identity matrix, to let the singular values of \( M \) be bounded around 1.

In this paper, we will show how to modify the entries of the kernel \( K \) to minimize \( \|M^T M - I\|_F^2 \) from the viewpoint of linear algebra. The knowledge in the field of matrix analysis plays a key role in this paper. As we know, given a matrix, the singular values/eigenvalues are continuous functions depending on the entries of the matrix [13]. We can calculate the partial derivatives of a singular value with respect to the entries, and let the singular values of a matrix be changed by modifying the entries. Here the transformation matrix \( M \) corresponding to a convolutional kernel is structured, i.e., \( M \) has a special matrix structure. Our goal is to regularize the singular values of \( M \) through modifying entries of the kernel \( K \). The modification of \( K \) is actually to carry out a modification of \( M \) on a special matrix manifold.

The contribution is that we derive a mathematical formula for the gradient of \( \|M^T M - I\|_F^2 \) with respect to the kernel \( K \), i.e., \( \partial \|M^T M - I\|_F^2 / \partial K_{p,q,d,c} \). Then gradient-based algorithms can be applied to effectively let the singular values of convolutional layers be bounded. Compared with the 2 norm, the Frobenius norm of a matrix is less sensitive to the perturbations of matrix entries. This Frobenius norm-based formula will be numerically stable.

There are many techniques from different perspectives to improve the performance of a neural network model. In [14], a semisupervised deep model, which is robust over imbalanced and small training data sets, is proposed for human activity recognition from multimodal wearable sensory data. A semisupervised feature selection method, which shows superiority in video semantic recognition-related tasks, is proposed from the perspective that the instances with similar labels should have a larger probability of being neighbors [15]. Two deep learning-based frameworks are proposed, which make sense of spatio-temporal preserving representations for electroencephalography-based human intension recognition [16]. A modeling method based on neural networks describes the hysteresis nonlinearity better by adding a nonlinear function in the input layer [17].

As we know, batch normalization and dropout are two popular regularization methods [18–21]. Recently, for the weights of a neural network, there have been many papers devoted to enforcing the orthogonality or spectral norm regularization [8, 9, 12, 22]. The difference between our paper and papers including [8, 9, 12, 22] and the references therein is about how to handle convolutions. They enforce the constraint directly on the \( h \times (gkk) \) matrix reshaped from the kernel \( K \in \mathbb{R}^{k \times k \times g \times h} \), while we enforce the constraint on the transformation matrix \( M \) corresponding to the convolution kernel \( K \). In [10], a convolutional layer is projected onto the set of layers obeying a bound on the operator norm of the layer and this is shown to be an effective regularizer. A drawback of the method in [10] is that projection can prevent the singular values of the transformation matrix from being large but cannot avoid the singular values to be too small. In [23], a 2-norm regularization method is proposed for convolutional kernels, but it is not a stable algorithm because the largest singular value may be overtaken by the second or the third largest singular value after one updating. In this paper, we propose a Frobenius norm regularization method for convolutional kernels.

As we have mentioned, the input channels and the output channels may be more than one, so the kernel is usually represented by a tensor \( K \in \mathbb{R}^{k \times k \times g \times h} \). The rest of the paper is organized as follows: In Section 2, we first consider the case that the numbers of input channels and the output channels are both 1. We propose the penalty function, calculate the partial derivatives, and propose the gradient descent algorithm for this case. In Section 3, we propose the penalty function and calculate the partial derivatives for the case of multichannel convolution. In Section 4, we present numerical results to show the method is feasible and effective. In Section 5, we will give some conclusions and point out some interesting work that could be done in the future.

2. Penalty Function for One-Channel Convolution

As a warm-up, we first focus on the case that the numbers of input channels and the output channels are both 1. In this case, the weights of the kernel are a \( k \times k \) matrix. Without loss of generality, assuming the data matrix is \( N \times N \), we use
a $3 \times 3$ matrix as a convolution kernel to show the associated transformation matrix. Let $K$ be the convolution kernel:

$$K = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}. \quad (5)$$

Then the transformation matrix corresponding with the convolution arithmetic is

$$M = \begin{pmatrix} A_0 & A_{-1} & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & A_{-1} & \ddots & \vdots & \vdots \\ 0 & A_1 & A_0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & A_0 & A_{-1} \\ 0 & \cdots & 0 & A_1 & A_0 & 0 \end{pmatrix}, \quad (6)$$

where

$$A_0 = \begin{pmatrix} k_{22} & k_{32} & 0 & 0 & \cdots & 0 \\ k_{12} & k_{22} & k_{23} & \ddots & \vdots & \vdots \\ 0 & k_{12} & k_{22} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & k_{12} & k_{13} \\ 0 & \cdots & 0 & k_{12} & k_{13} & k_{14} \end{pmatrix},$$

$$A_{-1} = \begin{pmatrix} k_{13} & k_{23} & k_{33} & \ddots & \vdots & \vdots \\ 0 & k_{13} & k_{23} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & k_{23} & k_{24} \\ 0 & \cdots & 0 & k_{23} & k_{24} & k_{25} \end{pmatrix},$$

$$A_1 = \begin{pmatrix} k_{21} & k_{31} & 0 & 0 & \cdots & 0 \\ k_{11} & k_{21} & k_{23} & \ddots & \vdots & \vdots \\ 0 & k_{11} & k_{21} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & k_{11} & k_{13} \\ 0 & \cdots & 0 & k_{11} & k_{13} & k_{14} \end{pmatrix}. \quad (7)$$

In this case, the transformation matrix $M$ corresponding to the convolutional kernel $K$ is a $N^2 \times N^2$ doubly block banded Toeplitz matrix, i.e., a block banded Toeplitz matrix with its blocks are banded Toeplitz matrices. For the details about Toeplitz matrices, please see references [24, 25]. We will let $n = N^2$ and use $\mathcal{T}$ to denote the set of all matrices like $M$ in (6), i.e., doubly block banded Toeplitz matrices with the fixed band.

We will use $\|M^TM - I\|_F^2$ as the penalty function to regularize the convolutional kernel $K$ and calculate $\partial\|M^TM - I\|_F^2/\partial K_{p,q}$, i.e., the partial derivative of Frobenius norm of $M^TM - I$ versus each entry $K_{p,q}$ of the convolution kernel. Our method provides a new method to calculate the gradient of the penalty function of the transformation matrix versus the convolution kernel. People can construct other penalty functions of $M$ and get the gradient descent method when training their convolutional networks. The following lemma is easy but useful in the following derivation.

**Lemma 1.** The partial derivative of square of Frobenius norm of $A \in \mathbb{R}^{n \times n}$ with respect to each entry $a_{ij}$ is $\partial \|A\|_F^2/\partial a_{ij} = 2a_{ij}$.

If an entry $a_{ij}$ of the matrix $A \in \mathbb{R}^{n \times n}$ changes, only the entries belonging to $j$-th row or $j$-th volume of the matrix $A^TA$ are affected. Actually, we have the following lemma.

**Lemma 2.** If we use $(A^TA)_{s,t}$ to denote the $(s,t)$ entry of the matrix $A^TA$, then $\partial(A^TA)_{s,t}/\partial a_{ij}$ is the $(s,t)$ entry of the matrix $D = A^T(e_i e_j^T + (e_i e_j^T)A$, where

$$D = \begin{pmatrix} 0 & \cdots & 0 & a_{ii} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_s & a_{s+1} & \cdots & a_n \\ 0 & \cdots & 0 & a_{t} & a_{t+1} & \cdots & a_{n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{in} & 0 & \cdots & 0 \end{pmatrix}. \quad (8)$$

We have the following formula from Lemma 1 and Lemma 2

$$\frac{1}{2} \frac{\partial \|M^TM - I\|_F^2}{\partial m_{ij}} = \frac{1}{2} \sum_{s,t \in \Omega \cap \Omega} \frac{\partial \|M^TM - I\|_F^2}{\partial (M^TM - I)_{s,t}} \frac{\partial (M^TM - I)_{s,t}}{\partial m_{ij}}$$

$$= \sum_{s \in \Omega} (M^TM - I)_{s} m_{s} + \sum_{s \in \Omega} (M^TM - I)_{s} m_{s}. \quad (9)$$

For a matrix $M \in \mathcal{T}$, the value of $K_{p,q}$ will appear in different $(i, j)$ indexes. We use $\Omega$ to denote this index set, i.e., for each $(i, j) \in \Omega$, we have $m_{ij} = K_{p,q}$. The chain rule formula above the derivative tells us that, if we want to calculate $\partial \|M^TM - I\|_F^2/\partial K_{p,q}$, we should calculate $\partial \|M^TM - I\|_F^2/\partial m_{ij}$ for all $(i, j) \in \Omega$ and take the sum, i.e.,

$$\frac{1}{2} \frac{\partial \|M^TM - I\|_F^2}{\partial K_{p,q}} = \frac{1}{2} \sum_{(i,j) \in \Omega} \frac{\partial \|M^TM - I\|_F^2}{\partial m_{ij}}$$

$$= \sum_{(i,j) \in \Omega} \left( \sum_{s \in \Omega} (M^TM - I)_{s} m_{s} + \sum_{s \in \Omega} (M^TM - I)_{s} m_{s} \right). \quad (10)$$

We summarize the above results as the following theorem. We can use the formula (11) to carry out the gradient descent method for $\|M^TM - I\|_F^2$.

**Theorem 1.** Assume $M \in \mathbb{R}^{n \times n}$ is the doubly block banded Toeplitz matrix corresponding to the one-channel convolution arithmetic.
kernel \( K \in \mathbb{R}^{k \times k} \). If \( \Omega \) is the set of all indexes \((i, j)\) such that \( m_{ij} = K_{p,q} \) we have

\[
\frac{1}{2} \frac{\partial \| M^T M - I \|_F^2}{\partial K_{p,q,y}} = \sum_{(i,j) \in \Omega_{p,q,y}} \left( \sum_{t=1}^{N^3} (M^T M - I)_{j,t} m_{it} + \sum_{s=1}^{N^2} (M^T M - I)_{j,s} m_{is} \right)
\]

(11)

Theorem 1 provides new insight into how to regularize a convolutional kernel \( K \) such that singular values of the corresponding transformation matrix are in a bounded interval. We can use the formula (11) to carry out the gradient-type methods. In the future, we can construct other penalty functions to let the transformation matrix corresponding to a convolutional kernel have some prescribed property and calculate the gradient of the penalty function with respect to the kernel as we have done in this paper.

3. The Penalty Function and the Gradient for Multichannel Convolution

In this section, we consider the case of multichannel convolution. First, we show the transformation matrix corresponding to multichannel convolution. At each convolutional layer, we have a convolution kernel \( K \in \mathbb{R}^{k \times k \times g \times h} \), and the input \( X \in \mathbb{R}^{N \times N \times g} \); element \( X_{i,j,d} \) is the value of the input unit within the channel \( d \) at row \( i \) and column \( j \). Each entry of the output \( Y \in \mathbb{R}^{N \times N \times k \times h} \) is produced by

\[
Y_{r,s,d} = (K \ast X)_{r,s,d} = \sum_{d' \in [1,\ldots,g]} \sum_{p \in [-\Lambda,\ldots,\Lambda]} \sum_{q \in [-\Lambda,\ldots,\Lambda]} X_{r+d'-p,s+d'-q} K_{p,q,d',d'}
\]

(12)

where \( X_{i,j,d} = 0 \) if \( i \leq 0 \) or \( i > N \), or \( j \leq 0 \) or \( j > N \). By inspection, \( \text{vec}(Y) = M \text{vec}(X) \), where \( M \) is as follows:

\[
M = \begin{pmatrix}
M_{(1)(1)} & M_{(1)(2)} & \cdots & M_{(1)(g)} \\
M_{(2)(1)} & M_{(2)(2)} & \cdots & M_{(2)(g)} \\
\vdots & \vdots & \ddots & \vdots \\
M_{(h)(1)} & M_{(h)(2)} & \cdots & M_{(h)(g)}
\end{pmatrix}
\]

(13)

and each \( B_{(j)(d)} \in \mathcal{S} \), i.e., \( B_{(j)(d)} \) is a \( N^2 \times N^2 \) doubly block banded Toeplitz matrix corresponding to the portion \( K_{d\ldots d_2} \) of \( K \) that concerns the effect of the \( d \)-th input channel on the \( c \)-th output channel.

Similar to the proof in Section 2, we have the following theorem.

Theorem 2. Assume \( M \) is the structured matrix corresponding to the multichannel convolution kernel \( K \in \mathbb{R}^{k \times k \times g \times h} \) as defined in (13). Given \((p, q, z, y)\), if \( \Omega_{p,q,z,y} \) is the set of all indexes \((i, j)\) such that \( m_{ij} = K_{p,q,z,y} \), we have

\[
\frac{\partial \| M^T M - I \|_F^2}{\partial K_{p,q,z,y}} = 2 \sum_{(i,j) \in \Omega_{p,q,z,y}} \left( \sum_{t=1}^{N^3} (M^T M - I)_{j,t} m_{it} + \sum_{s=1}^{N^2} (M^T M - I)_{j,s} m_{is} \right)
\]

(14)

Then the gradient descent algorithm for the penalty function \( \| M^T M - I \|_F^2 \) can be devised, where the number of channels may be more than one. We present the detailed gradient descent algorithm for the penalty function \( \| M^T M - I \|_F^2 \) as follows:

4. Numerical Experiments

The numerical tests were performed on a laptop (3.0 GHz and 16G Memory) with MATLAB R2016b.

We use \( M \) to denote the transformation matrix corresponding to the convolutional kernel. The largest singular value and smallest singular value of \( M \) (denoted as \( \sigma_{\text{max}}(M) \) and \( \sigma_{\text{min}}(M) \)), the iteration steps (denoted as “iter”) are demonstrated to show the effectiveness of our method. The efficiency is related to the step size \( \lambda \). According to our experience, the norm of the matrix reshaped from the gradient tensor \( G \in \mathbb{R}^{k \times k \times g \times h} \) in Algorithm 1 decreases as the number of iteration steps becomes larger. Therefore, we can let the step size \( \lambda \) be a small number at first and gradually increase \( \lambda \). In our numerical experiments, for Algorithm 1 we use the following dynamic adjustment of step size:

\[
\text{if } \text{iter} < 10 \Rightarrow \lambda = 1 \times 10^{-5}; \\
\text{elseif } \text{iter} < 20 \Rightarrow \lambda = 1 \times 10^{-4}; \\
\text{else} \Rightarrow \lambda = 1 \times 10^{-3}; \\
\text{end}
\]

Numerical experiments are implemented on extensive test problems. Our method is effective in letting \( \sigma_{\text{max}}(M) \) and \( \sigma_{\text{min}}(M) \) be approximate to 1. We present the numerical results for some random generated multichannel convolution kernels.

We start from a random kernel with each entry normally distributed on \([0,1]\), i.e., in MATLAB, \( K \) is generated by the following command.

\[
\text{rng(1)}; \\
K = \text{randn}(k,k,g,h).
\]

(15)

We consider kernels of different sizes with \( 3 \times 3 \) filters, namely \( K \in \mathbb{R}^{k \times k \times g \times h} \) for various values of \( g, h \). For each kernel, we use the input data matrix of size \( 20 \times 20 \times g \).
(1) Input: an initial kernel $K \in \mathbb{R}^{k \times k \times g \times h}$, input size $N \times N \times g$, and learning rate $\lambda$.
(2) While not converged:
(3) Compute $G = \| \partial_i M^T M - I \|^2_F / \partial_k p_{q,r,z,y}^{k,k,g,h}$, by (14);
(4) Update $K = K - \lambda G$;
(5) End.

**Algorithm 1: Gradient Descent for $\mathcal{R}_g(K) = \| M^T M - I \|^2_F$.**

We then minimize $\mathcal{R}_1(K) = \| M^T M - I \|^2_F$ using Algorithm 1 and we demonstrate the beneficial effect of decreasing $\sigma_{\text{max}}(M)$ while increasing $\sigma_{\text{min}}(M)$. We present in Figure 1 the results of $3 \times 3 \times 3 \times 1$, $3 \times 3 \times 1 \times 3$, $3 \times 3 \times 3 \times 6$, and $3 \times 3 \times 6 \times 3$ kernels. In the figures, we have shown the convergence of $\sigma_{\text{max}}(M)$ (blue line) on the left axis scale and $\sigma_{\text{min}}(M)$ (red line) on the right axis scale. From Figure 1, we see that at the first about 20 steps, $\sigma_{\text{max}}(M)$ and $\sigma_{\text{min}}(M)$ all decrease and then $\sigma_{\text{max}}(M)$ becomes very close to 1. Then in the following steps $\sigma_{\text{max}}(M)$ become almost unchanged while $\sigma_{\text{min}}(M)$ increase from smaller than 1 to be very close to 1.

We would like to point out, we have used $\mathcal{R}_1(K) = \| M^T M - I \|^2_F$ to do numerical experiments on other random generated examples, including random kernels with each entry uniformly distributed on $[0,1]$. The convergence figures of $\sigma_{\text{max}}(M)$ and $\sigma_{\text{min}}(M)$ are similar to the subfigures in Figure 1.

About the running time of Algorithm 1, we have the following remarks. Given a $k \times k \times g \times h$ kernel and the input matrix size $N \times N \times g$, the gradient tensor $G \in \mathbb{R}^{k \times k \times g \times h}$ is computed by (14). From the extensive numerical experiments, it is observed that, for a given kernel, when the value of $N$ changes, the values of the gradient tensor $G$ computed by (14) remain almost unchanged. For example, for the kernel generated by the command "$\text{randn}(1); K = \text{randn}(3,3,6,3)"$, let $G^{(1)}$ denote the gradient tensor computed with $N = 6$, and let $G^{(2)}$ denote the gradient tensor computed with $N = 64$, then each entry $G^{(1)}_{p,q,r,z,y}$ has at least three significant digits of the corresponding entry $G^{(2)}_{p,q,r,z,y}$. So in the practical training of neural networks, where $N$ could be $N = 64$ or even larger for a convolutional layer, we can always compute the gradient tensor $G$ with a smaller value of $N$, for example, with $N = 6$. From the extensive numerical experiments about different kernels, when $N = 64$, $N = 128$, and

![Figure 1: Convergence of $\sigma_{\text{max}}(M)$ and $\sigma_{\text{min}}(M)$ for different kernel sizes.](image-url)
$N = 256$, using the gradient tensor $G$ computed with $N = 6$ to carry out Algorithm 1, the convergence figures of $\sigma_{\text{max}}(M)$ and $\sigma_{\text{min}}(M)$ are similar to the subfigures in Figure 1. For a given $3 \times 3 \times 64 \times 64$ kernel, the time for computing the gradient tensor $G$ once with $N = 6$ is 12.10 seconds on our laptop. For a given $3 \times 3 \times 5 \times 1$ kernel, the time for computing the gradient tensor $G$ once with $N = 6$ is 0.02 seconds on our laptop. The time cost of our regularization method is affordable.

5. Conclusions

In this paper, from the viewpoint of linear algebra, we provide the Frobenius norm method to let the convolution be norm preserving, i.e., orthogonal. We let the singular values of the structured transformation matrix corresponding to a convolutional kernel tensor be close to 1. We give the penalty function and propose the gradient decent algorithm for the convolutional kernel tensor. Numerical experiments confirm that this method is effective to modify the singular values of the convolution operations. This mathematical method may cast new light on the training of very deep convolutional neural networks. However, up to now, we only show the method is mathematically feasible and it may be of potential use in the field of deep learning. In the future, further numerical experiments are needed to show the effectiveness of this method during the training of neural networks.

In the future, we will evaluate this method on state-of-the-art architectures like ResNet and DenseNet [26, 27], considering all aspects like recognition rate and computational efficiency. As we know, training a neural network involves many details. It is not easy to see the performance improved while conducting experiments. This will be left as our future work [28].

Data Availability

Data involved in the paper will be shared upon request. If anybody is interested in the data, please send an email to the corresponding author’s email: peichang@cugb.edu.cn. The corresponding author will send the data through email.

Disclosure

An earlier draft version of this paper was uploaded to arXiv and https://arxiv.org/abs/1907.11235 is the link on arXiv.

Conflicts of Interest

There are no conflicts of interest to declare.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (Grant Nos. 12001504 and Grant No. 41874165) and the Fundamental Research Funds for the Central Universities (Grant Number 2652019320).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[2] D. Vincent and F. Visin, “A Guide to Convolution Arithmetic for Deep Learning,” ArXiv, 2018, https://arxiv.org/abs/1603.07285.
[3] G.-H. Golub and C.-F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 2012.
[4] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” in Field Guide to Dynamical Recurrent NetworksIEEE Press, 2001.
[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” ICLR, 2015, https://arxiv.org/abs/1412.6572.
[6] C. Szegedy, W. Zaremba, I. Sutskever et al., “Intriguing Properties of Neural Networks,” ICLR, 2014, https://arxiv.org/abs/1312.6199.
[7] Y. Tsuzuku, I. Sato, and M. Sugiyama, “Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks,” in Proceedings of the NIPS, 2018, https://papers.nips.cc/paper/2018/hash/485843481a7edacb6ce101eb1e4d2a8-Abstract.html.
[8] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, “Parseval networks: improving robustness to adversarial examples,” in Proceedings of the ICML, Sydney, Australia, August 2017, https://proceedings.mlr.press/v70/cisse17a/cisse17a.pdf.
[9] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral Normalization for Generative Adversarial Networks,” ICLR, 2018, https://arxiv.org/abs/1802.05957.
[10] H. Sedghi, V. Gupta, and M. Philip, “Long. The Singular Values of Convolutional Layers,” ICLR, 2019, https://arxiv.org/abs/1805.10408.
[11] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in Proceedings of the ICLR, New York, NY, February 2021, https://arxiv.org/abs/1611.03530.
[12] Y. Yoshida and T. Miyato, “Spectral Norm Regularization for Improving the Generalizability of Deep Learning,” ArXiv 2017, https://arxiv.org/abs/1705.10941.
[13] G. W. Stewart, Matrix Algorithms, Eigensystems, SIAM Press, Philadelphia, PA, USA, 2001.
[14] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, “A semisupervised recurrent convolutional attention model for human activity recognition,” IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 5, pp. 1747–1756, 2020.
[15] M. Luo, X. Chang, L. Nie, Y. Yang, A. G. Hauptmann, and Q. Zheng, “An adaptive semisupervised feature analysis for video semantic recognition,” IEEE Transactions on Cybernetics, vol. 48, no. 2, pp. 648–660, 2018.
[16] D. Zhang, L. Yao, K. Chen, S. Wang, X. Chang, and Y. Liu, “Making sense of spatio-temporal preserving representations for EEG-based human intention recognition,” IEEE Transactions on Cybernetics, vol. 50, no. 7, pp. 3033–3044, 2020.
[17] M. Zhou and Q. Zhang, “Hysteresis model of magnetically controlled shape memory alloy based on a PID neural network,” IEEE Transactions on Magnetics, vol. 51, no. 11, pp. 1–4, 2015.
[18] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in Proceedings of the ICMII, Lille, France, 2015, https://proceedings.mlr.press/v37/ioffe15.pdf.
[19] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, *How Does Batch Normalization Help Optimization?*, NIPS, December 2018.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] H. Pan, X. Niu, R. Li, S. Shen, and Y. Dou, “DropFilterR: a novel regularization method for learning convolutional neural networks,” *Neural Processing Letters*, vol. 51, no. 2, pp. 1285–1298, 2020.

[22] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Neural Photo Editing with Introspective Adversarial Networks,” ICLR, 2017, https://arxiv.org/abs/1609.07093.

[23] P. Guo and Q. Ye, “On the Regularization of Convolutional Kernel Tensors in Neural Networks,” *Linear and Multilinear Algebra*, vol. 70, 2020.

[24] R. Chan and X. Jin, *An Introduction to Iterative Toeplitz Solvers*, SIAM Press, Philadelphia, PA, USA, 2007.

[25] X. Jin, *Developments and Applications of Block Toeplitz Iterative Solvers*, Science Press, Beijing, 2002.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the CVPR*, Las Vegas, NV, USA, June 2016.

[27] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the CVPR*, Honolulu, HI, USA, July 2018.

[28] Kova, evi, A. Chebira, “An Introduction to Frames,” *Now Publishers Inc*, vol. 2, no. 1, pp. 1–94, Boston, MA, USA, 2008.