The Singular Values of Convolutional Layers

Hanie Sedghi
Google Brain
Mountain View, CA 94043
hsedghi@google.com

Vineet Gupta
Google Brain
Mountain View, CA 94043
vineet@google.com

Philip M. Long
Google Brain
Mountain View, CA 94043
plong@google.com

May 25, 2018

Abstract

We characterize the singular values of the linear transformation associated with a convolution applied to a two-dimensional feature map with multiple channels. Our characterization enables efficient computation of the singular values of convolutional layers used in popular deep neural network architectures. It also leads to an algorithm for projecting a convolutional layer onto the set of layers obeying a bound on the operator norm of the layer. We show that this is an effective regularizer; periodically applying these projections during training improves the test error of a residual network on CIFAR-10 from 6.2% to 5.3%.

1 Introduction

Exploding and vanishing gradients [Hochreiter, 1991, Hochreiter et al., 2001, Goodfellow et al., 2016] are fundamental obstacles to effective training of deep neural networks. Many deep networks used in practice are layered. We can think of such networks as the composition of a number of feature transformations, followed by a linear classifier on the final layer of features. The singular values of the Jacobian of a layer bound the factor by which it increases or decreases the norm of the backpropagated signal. If these singular values are all close to 1, then gradients neither explode nor vanish. These singular values also bound these factors in the forward direction, which affects the stability of computations, including whether the network produces the dreaded "Nan". These considerations have led authors to regularize networks by driving down the largest singular value of a layer, otherwise known as the operator norm [Yoshida and Miyato, 2017, Miyato et al., 2018]. Orthogonal initialization [Saxe et al., 2013, Pennington et al., 2017] is motivated by similar considerations.

Convolutional layers [LeCun et al., 1998] are key components of modern deep networks. They compute linear transformations of their inputs. The Jacobian of a linear transformation is always equal to the linear transformation itself. Because of the central importance of convolutional layers to the practice of deep learning, and the fact that the singular values of the linear transformation computed by a convolutional layer are the key to its contribution to exploding and vanishing gradients, we study these singular values.

We consider the convolutional layers commonly applied to image analysis tasks. The input to a typical layer is a feature map, with multiple channels for each position in an $n \times n$ field. If there are m channels, then the input as a whole is a $m \times n \times n$ tensor. The output is also an $n \times n$ field with multiple channels per position[1]. Each channel of the output is obtained by taking a linear combination of the values of the features in all channels in a local neighborhood centered at the corresponding position in the input feature map. Crucially, the same linear combination is used for all positions in the feature map. The coefficients are compiled in the

[1]Here, to keep things simple, we are concentrating on the case that the stride is 1.

kernel of the convolution. If the neighborhood is a $k \times k$ region, a kernel K is a $m \times m \times k \times k$ tensor. The projection $K_{c,:,:,:}$ gives the coefficients that determine the cth channel of a output, in terms of the values found in all of the channels of all positions in it neighborhood; $K_{c,d,:,:}$ gives the coefficients to apply to the dth input channel, and $K_{c,d,p,q}$ are the coefficients to apply to this input at in the position in the field offset horizontally by p and vertically by q. For ease of exposition, we assume that feature maps and local neighborhoods are square and that the number of channels in the output is equal to the number of channels in the input - the extension to the general case is completely straightforward.

To handle edge cases in which the offsets call for inputs that are off the feature maps, practical convolutional layers either do not compute outputs (reducing the size of the feature map), or pad the input with zeros. The behavior of these layers can be approximated by a layer that treats the input as if it were a torus; when the offset calls for a pixel that is off the right end of the image, the layer "wraps around" to take it from the left edge, and similarly for the other edges. The quality of this approximation has been heavily analyzed in the case of one-dimensional signals [Gray, 2006]. Consequently, theoretical analysis of convolutions that wrap around has been become standard. This is the case analyzed in this paper.

Summary of Results: Our main result is a characterization of the singular values of a convolutional layer in terms of the kernel tensor K. Our characterization enables these singular values to be computed in a simple and practically fast way, using $O(n^2m^2(m + \log n))$ time. For comparison, the brute force solution that performs SVD on the matrix that encodes the convolutional layer's linear transformation would take $O((n^2m)^3) = O(n^6m^3)$ time, and is impractical for commonly used network sizes. As another point of comparison, simply to compute the convolution takes $O(n^2m^2k^2)$ time. We prove that the following two lines of NumPy correctly compute the singular values.

```
def SingularValues(kernel, input_shape):
  transform_coefficients = np.fft.fft2(kernel, input_shape, axes=[0, 1])
  return np.linalg.svd(transform_coefficients, compute_uv=False)
```

Here kernel is any 4D tensor, and input_shape is the shape of the feature map to be convolved. A TensorFlow implementation is similarly simple.

Timing tests, reported in Section 4.1, confirm that this characterization speeds up the computation of singular values by multiple orders of magnitude – making it usable in practice. We used our code to compute the singular values of the convolutional layers of the official ResNet-v2 model released with TensorFlow [He et al., 2016]. The results are described in Section 4.3. The algorithm first performs m^2 FFTs, and then it performs n^2 SVDs. The FFTs, and then the SVDs, may be executed in parallel. Our TensorFlow implementation runs a lot faster than the NumPy implementation (see Figure 1); we think that this parallelism is the cause.

Exposing the singular values of a convolutional layer opens the door to a variety of regularizers for these layers, including operator-norm regularizers [Yoshida and Miyato, 2017, Miyato et al., 2018][2] and path-norm regularizers [Neyshabur et al., 2015]. (For another regularizer for convolutional layers with similar aims to the operator-norm regularizer, see [Gouk et al., 2018a].) For example, we may project a convolutional layer onto the set of layers with a bounded operator norm by clipping its singular values [Lefkimmiatis et al., 2013]. Since the filters in the projected network may no longer use a local, $k \times k$ neighborhood, we further project the filter coefficients that correspond to the clipped SVD, by setting to zero the coefficients for offsets other than the original $k \times k$ neighborhood. Repeating the two projections alternately gives us the required projection to the intersection.

[2]Yoshida and Miyato [2017] used the operator norm of a reshaping of K in place of the operator norm for the linear transformation associated with K in their experiments. The two can be seen to be different by considering the $1 \times 1 \times 2 \times 2$ kernel with every entry equal to 1.

In Section 4.2, we evaluate an algorithm that periodically projects each convolutional layer onto a operator-norm ball. Using the projections improves the test error from 6.2% to 5.3% on CIFAR-10.

Gouk et al. [2018b] propose regularizing using a per-mini-batch approximation to the operator norm. They find the largest ratio between the input and output of a layer in the minibatch, and then scale down the transformation (thereby scaling down all of the singular values, not just the maximizers) so that the new value of this ratio obeys a constraint. Having access to the whole spectrum, we can clip the singular values and compute a projection using the exact value of the operator norm which leads to a larger improvement on a stronger baseline.

Overview of the Analysis: If the signal is 1D and there is a single input and output channel, then the linear transformation associated with a convolution is encoded by a *circulant matrix*, i.e., a matrix whose rows are circular shifts of a single row [Gray, 2006]. For example, for a row $a = (a_1, a_2, a_3)$, the circulant matrix $\mathrm{circ}(a)$ generated by a is $\begin{pmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{pmatrix}$. In the special case of a 2D signal with a single input channel and single output channel, the linear transformation is *doubly block circulant* (see [Goodfellow et al., 2016]). Such a matrix is made up of a circulant matrix of blocks, each of which in turn is itself circulant. Finally, when there are m input channels and m output channels, there are three levels to the hierarchy: there is a $m \times m$ matrix of blocks, each of which is doubly block circulant. Our analysis extends tools from the literature built for circulant [Horn and Johnson, 2012] and doubly circulant [Chao, 1974] matrices to analyze the matrices with a third level in the hierarchy arising from the convolutional layers used in deep learning. One key point is that the eigenvectors of a circulant matrix are Fourier basis vectors: in the 2D, one-channel case, the matrix whose columns are the eigenvectors is $F \otimes F$, for the matrix F whose columns form the Fourier basis. Multiplying by this matrix is a 2D Fourier transform. In the multi-channel case, we show that the singular values can be computed by (a) finding the eigenvalues of each of the m^2 doubly circulant matrices (of dimensions $n^2 \times n^2$) using a 2D Fourier transform, (b) by forming multiple $m \times m$ matrices, one for each eigenvalue, by picking out the i-th eigenvalue of each of the $n^2 \times n^2$ blocks, for $i \in [1..n^2]$. The union of all of the singular values of all of those $m \times m$ matrices is the multiset of singular values of the layer.

Notation: We use upper case letters for matrices, lower case for vectors. For matrix M, $M_{i,:}$ represents the i-th row and $M_{:,j}$ represents the j-th column; we will also use the analogous notation for higher-order tensors. The operator norm of M is denoted by $||M||_2$. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{0, 1, \ldots, n-1\}$ (instead of usual $\{1, \ldots, n\}$). We will index the rows and columns of matrices using elements of $[n]$, i.e. numbering from 0. Addition of row and column indices will be done mod n unless otherwise indicated. (Tensors will be treated analogously.) Let $\sigma(\cdot)$ be the mapping from a matrix to (the multiset of) its singular values. [3]

Let $\omega = \exp(2\pi i/n)$, where $i = \sqrt{-1}$. (Because we need a lot of indices in this paper, our use of i to define ω is the only place in the paper where we will use i to denote $\sqrt{-1}$.)

Let F be the $n \times n$ matrix that represents the discrete Fourier transform: $F_{ij} = \omega^{ij}$. We use I_n to denote the identity matrix of size $n \times n$. We use $e_i, i \in [n]$ to represent the basis vectors in $\mathbb{R}^n$. We use $\otimes$ to represent the Kronecker product between two matrices (which also refers to the outer product of two vectors).

[3] For two multisets $\mathcal{S}$ and $\mathcal{T}$, we use $\mathcal{S} \cup \mathcal{T}$ to denote the multiset obtained by summing the multiplicities of members of $\mathcal{S}$ and $\mathcal{T}$.

2 Analysis

2.1 One filter

As a warmup, we focus on the case that the number m of input channels and output channels is 1. In this case, the filter coefficients are simply a $k \times k$ matrix. It will simplify notation, however, if we embed this $k \times k$ matrix in an $n \times n$ matrix, by padding with zeroes (which corresponds to the fact that the offsets with those indices are not used). Let us refer to this $n \times n$ matrix also as K.

An $n^2 \times n^2$ matrix A is *doubly block circulant* if A is a circulant matrix of $n \times n$ blocks that are in turn circulant.

For a matrix X, let $\mathrm{vec}(X)$ be the vector obtained by stacking the columns of X.

Lemma 1 (see [Jain, 1989, Goodfellow et al., 2016]) *For any filter coefficients K, the linear transform for the convolution by K is represented by the following doubly block circulant matrix:*

$$A = \begin{bmatrix} \mathrm{circ}(K_{0,:}) & \mathrm{circ}(K_{1,:}) & \dots & \mathrm{circ}(K_{n-1,:}) \\ \mathrm{circ}(K_{n-1,:}) & \mathrm{circ}(K_{0,:}) & \dots & \mathrm{circ}(K_{n-2,:}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{circ}(K_{1,:}) & \mathrm{circ}(K_{2,:}) & \dots & \mathrm{circ}(K_{0,:}) \end{bmatrix}. \tag{1}$$

That is, if X is an $n \times n$ matrix, and

$$\forall ij,\ Y_{ij} = \sum_{p \in [n]} \sum_{q \in [n]} X_{i+p,j+q} K_{p,q} \tag{2}$$

then $\mathrm{vec}(Y) = A\ \mathrm{vec}(X)$.

So now we want to determine the singular values of a doubly block circulant matrix.

We will make use of the characterization of the eigenvalues and eigenvectors of doubly block circulant matrices, which uses the following definition: $Q \stackrel{\text{def}}{=} \frac{1}{n}(F \otimes F)$.

Theorem 2 ([Jain, 1989]) *For any $n^2 \times n^2$ doubly block circulant matrix A, the eigenvectors of A are the columns of Q.*

To get singular values in addition to eigenvalues, we need the following two lemmas.

Lemma 3 ([Jain, 1989]) *Q is unitary.*

Using Theorem 2 and Lemma 3, we can get the eigenvalues through the diagonal elements of $Q^* A Q$.

Lemma 4 *The matrix A defined in* (1) *is normal, i.e., $A^T A = A A^T$.*

Proof:

$$AA^T = AA^* = Q^* D Q Q^* D^* Q = Q^* D D^* Q = Q^* D^* D Q = Q^* D^* Q Q^* D Q = A^* A = A^T A.$$

□

The following theorem characterizes the singular values of A as a simple function of K. As we will see, a characterization of the eigenvalues plays a major role. Chao [1974] provided a more technical characterization of the eigenvalues which may be regarded as making partial progress toward Theorem 5. However, we provide a proof from first principles, since it is the cleanest way we know to prove the theorem.

Theorem 5 *For the matrix A defined in* (1)*, the eigenvalues of A are the entries of $F^T K F$, and its singular values are their magnitudes. That is, the singular values of A are*

$$\left\{ \left| (F^T K F)_{u,v} \right| \ : \ u, v \in [n] \right\}. \tag{3}$$

Proof: By Theorems 2 and Lemma 3, the eigenvalues of A are the diagonal elements of $Q^* A Q = \frac{1}{n^2}(F^* \otimes F^*) A (F \otimes F)$. If we view $(F^* \otimes F^*) A (F \otimes F)$ as a compound $n \times n$ matrix of $n \times n$ blocks, for $u, v \in [n]$, the $(un + v)$th diagonal element is the vth element of the uth diagonal block. Let us first evaluate the uth diagonal block. Using i, j to index blocks, we have

$$(Q^* A Q)_{uu} = \frac{1}{n^2} \sum_{i,j \in [n]} (F^* \otimes F^*)_{ui} A_{ij} (F \otimes F)_{ju} = \frac{1}{n^2} \sum_{i,j \in [n]} \omega^{-ui} F^* \mathrm{circ}(K_{j-i,:}) \omega^{ju} F$$

$$= \frac{1}{n^2} \sum_{i,j \in [n]} \omega^{u(j-i)} F^* \mathrm{circ}(K_{j-i,:}) F. \tag{4}$$

To get the vth element of the diagonal of (4), we may sum the vth elements of the diagonals of each of its terms. Toward this end, we have

$$(F^* \mathrm{circ}(K_{j-i,:}) F)_{vv} = \sum_{r,s \in [n]} \omega^{-vr} \mathrm{circ}(K_{j-i,:})_{rs} \omega^{sv} = \sum_{r,s \in [n]} \omega^{v(s-r)} K_{j-i,s-r}.$$

Substituting into (4), we get $\frac{1}{n^2} \sum_{i,j,r,s \in [n]} \omega^{u(j-i)} \omega^{v(s-r)} K_{j-i,s-r}$. Collecting terms where $j - i = p$ and $s - r = q$, this is $\sum_{p,q \in [n]} \omega^{up} \omega^{vq} K_{p,q} = (F^T K F)_{uv}$.

Since the singular values of any normal matrix are the magnitudes of its eigenvalues [Horn and Johnson, 2012], applying Lemma 4 completes the proof. □

Note that $F^T K F$ is the 2D Fourier transform of K, and recall that $||A||_2$ is the largest singular value of A.

One consequence of Theorem 5 is an upper bound on the operator norm of A. (In the single-channel case, this is also a consequence of Young's convolution inequality [Young, 1912].)

Proposition 6 *For all real K, we have $||A||_2 \leq \sum_{p,q \in [n]} |K_{p,q}|$.*

Proof: For any u and v, we have

$$\left| (F^T K F)_{u,v} \right| = \left| \sum_{p,q \in [n]} \omega^{up} \omega^{vq} K_{p,q} \right| \leq \sum_{p,q \in [n]} |\omega^{up} \omega^{vq} K_{p,q}| = \sum_{p,q \in [n]} |K_{p,q}|$$

since $|\omega| = 1$. Since this holds for any u and v, we have $||A||_2 \leq \sum_{p,q \in [n]} |K_{pq}|$. □

When the coefficients of K are non-negative, the bound of Proposition 6 is tight, giving the exact norm.

Proposition 7 *If $K_{jl} \geq 0$ for all $j, l \in [n]$, then $||A||_2 = \sum_{p,q \in [n]} K_{p,q}$.*

Proof: Applying Theorem 5 with $u = v = 0$, we get $||A||_2 \geq \sum_{p,q \in [n]} K_{p,q}$ and combining with Proposition 6 completes the proof. □

2.2 Multi-channel convolution

Now, we consider case where the number m of channels may be more than one. We follow the notation of [Goodfellow et al., 2016]. Assume we have a 4D kernel tensor K with element $K_{c,d,p,q}$ giving the connection strength between a unit in channel d of the input and a unit in channel c of the output, with an offset of p rows and q columns between the input unit and the output unit. The input $X \in \mathbb{R}^{m\times n\times n}$; element $X_{d,i,j}$ is the value of the input unit within channel d at row i and column j. The output $Y \in \mathbb{R}^{m\times n\times n}$ has the same format as X, and is produced by

$$Y_{crs} = \sum_{d\in[m]} \sum_{p\in[n]} \sum_{q\in[n]} X_{d,r+p,s+q} K_{c,d,p,q}. \tag{5}$$

By inspection, $\mathrm{vec}(Y) = M\mathrm{vec}(X)$, where M is as follows

$$M = \begin{bmatrix} B_{00} & B_{01} & \cdots & B_{0(m-1)} \\ B_{10} & B_{11} & \cdots & B_{1(m-1)} \\ \vdots & \vdots & \cdots & \vdots \\ B_{(m-1)0} & B_{(m-1)1} & \cdots & B_{(m-1)(m-1)} \end{bmatrix}, \tag{6}$$

and each B_{cd} is a doubly block circulant matrix from Lemma 1 corresponding to the portion $K_{c,d,:,:}$ of K that concerns the effect of the d-th input channel on the c-th output channel. (We can think of each output in the multichannel case as being a sum of single channel filters parameterized by one of the $K_{c,d,:,:}$'s.)

The following is our main result.

Theorem 8 *For any $K \in \mathbf{R}^{m\times m\times n\times n}$, if M is the matrix encoding the linear transformation computed by a convolutional layer parameterized by K, defined as in (6), then*

$$\sigma(M) = \bigcup_{u\in[n],v\in[n]} \sigma\left(\left((F^T K_{c,d,:,:} F)_{u,v}\right)_{cd}\right). \tag{7}$$

Before proving Theorem 8, let us take a moment to parse the RHS of (7). For each $u \in [n], v \in [n]$, we use $\left((F^T K_{c,d,:,:} F)_{u,v}\right)_{cd}$ to denote the $m \times m$ matrix whose cd entry is $(F^T K_{c,d,:,:} F)_{u,v}$. Then $\sigma(M)$ is the union of the singular values of all of these matrices.

The rest of this section is devoted to proving Theorem 8 through a series of lemmas.

The analysis of Section 2.1 implies that for all $c, d \in [m]$, $D_{cd} \stackrel{\text{def}}{=} Q^* B_{cd} Q$ is diagonal. Define

$$L \stackrel{\text{def}}{=} \begin{bmatrix} D_{00} & D_{01} & \cdots & D_{0(m-1)} \\ D_{10} & D_{11} & \cdots & D_{1(m-1)} \\ \vdots & \vdots & \cdots & \vdots \\ D_{(m-1)0} & D_{(m-1)1} & \cdots & D_{(m-1)(m-1)} \end{bmatrix}. \tag{8}$$

Lemma 9 *M and L have the same singular values.*

Proof: We have

$$M = \begin{bmatrix} B_{00} & \cdots & B_{0(m-1)} \\ \vdots & \vdots & \vdots \\ B_{(m-1)0} & \cdots & B_{(m-1)(m-1)} \end{bmatrix} = \begin{bmatrix} QD_{00}Q^* & \cdots & QD_{0(m-1)}Q^* \\ \vdots & \vdots & \vdots \\ QD_{(m-1)0}Q^* & \cdots & QD_{(m-1)(m-1)}Q^* \end{bmatrix}$$
$$= R \begin{bmatrix} D_{00} & \cdots & D_{0(m-1)} \\ \vdots & \vdots & \vdots \\ D_{(m-1)0} & \cdots & D_{(m-1)(m-1)} \end{bmatrix} R^* = RLR^*,$$

where $R \stackrel{\text{def}}{=} I_m \otimes Q$. Note that R is unitary because

$$RR^* = (I_m \otimes Q)(I_m \otimes Q^*) = (I_m I_m) \otimes (QQ^*) = I_{mn^2};$$

this implies that M and L have the same singular values. □

So now we have as a subproblem characterizing the singular values of a block matrix whose blocks are diagonal. To express the the characterization, it helps to reshape the nonzero elements of L into a $m \times m \times n^2$ tensor G as follows: $G_{cdw} = (D_{cd})_{ww}$.

Theorem 10 $\sigma(L) = \bigcup_{w \in [n^2]} \sigma\left(G_{:,:,w}\right).$

Proof: Choose an arbitrary $w \in [n^2]$, and a (scalar) singular value σ of $G_{:,:,w}$ whose left singular vector is x and whose right singular vector is y, so that $G_{:,:,w}y = \sigma x$. Recall that $e_w \in \mathbb{R}^{n^2}$ is a standard basis vector.

We claim that $L(y \otimes e_w) = \sigma(x \otimes e_w)$. Since D_{cd} is diagonal, $D_{cd}e_w = (D_{cd})_{ww}e_w = G_{cdw}e_w$. Thus we have $(L(y \otimes e_w))_c = \sum_{d \in [m]} D_{cd}y_d e_w = (\sum_{d \in [m]} G_{cdw}y_d)e_w = (G_{:,:,w}y)_c e_w = \sigma x_c e_w$, which shows that

$$L(y \otimes e_w) = \begin{bmatrix} D_{00} & \cdots & D_{0(m-1)} \\ D_{10} & \cdots & D_{1(m-1)} \\ \vdots & \cdots & \vdots \\ D_{(m-1)0} & \cdots & D_{(m-1)(m-1)} \end{bmatrix} \begin{bmatrix} y_0 e_w \\ \vdots \\ y_{m-1} e_w \end{bmatrix} = \begin{bmatrix} \sigma x_0 e_w \\ \vdots \\ \sigma x_{m-1} e_w \end{bmatrix} = \sigma(x \otimes e_w).$$

If $\tilde{\sigma}$ is another singular value of $G_{:,:,w}$ with a left singular vector $\tilde{x}$ and a right singular vector $\tilde{y}$, then $\langle (x \otimes e_w), (\tilde{x} \otimes e_w) \rangle = \langle x, \tilde{x} \rangle = 0$ and, similarly $\langle (y \otimes e_w), (\tilde{y} \otimes e_w) = 0$. Also, $\langle (x \otimes e_w), x \otimes e_w) \rangle = 1$ and $\langle (y \otimes e_w), y \otimes e_w) \rangle = 1$.

For any x and $\tilde{x}$, whether they are equal or not, if $w \neq \tilde{w}$, then $\langle (x \otimes e_w), (\tilde{x} \otimes e_{\tilde{w}}) \rangle = 0$, simply because their non-zero components do not overlap.

Thus, by taking the Kronecker product of each singular vector of $G_{:,:,w}$ with e_w and assembling the results for various w, we may form a singular value decomposition of L whose singular values are $\cup_{w \in [n^2]} \sigma(G_{:,:,w})$. This completes the proof. □

Using Lemmas 9 and Theorem 10, we are now ready to prove Theorem 8.

Proof (of Theorem 8). Recall that, for each input channel c and output channel d, the diagonal elements of $D_{c,d}$ are the eigenvalues of $B_{c,d}$. By Theorem 5, this means that the diagonal elements of $D_{c,d}$ are

$$\{(F^T K_{c,d,:,:} F)_{u,v} : u, v \in [n]\}. \tag{9}$$

The elements of (9) map to the diagonal elements of $D_{c,d}$ as follows:

$$G_{cdw} = (D_{c,d})_{ww} = (F^T K_{c,d,:,:} F)_{\lfloor w/m \rfloor, w \mod m}$$

and thus

$$G_{:,:,w} = \left((F^T K_{c,d,:,:} F)_{\lfloor w/m \rfloor, w \mod m}\right)_{cd}$$

which in turn implies

$$\bigcup_{w \in [n^2]} \sigma(G_{:,:,w}) = \cup_{u \in [n], v \in [n]} \sigma\left(\left((F^T K_{c,d,:,:} F)_{u,v}\right)_{cd}\right).$$

Applying Lemmas 9 and 10 completes the proof. □

We also have the following generalization of Proposition 6 to the case of multiple filters.

Proposition 11

$$||M||_2 \leq \sqrt{\sum_{c\in[m]} \max_{d\in[m]} \left(\sum_{p\in[n],q\in[n]} |K_{cdpq}| \right)^2} \leq \sum_{c,d\in[m],p,q\in[n]} |K_{cdpq}|.$$

Proof:

For an arbitrary input X, let $\tilde{X}$ be the $m \times n^2$ matrix obtained by flattening each $X_{d,:,:}$ to form $\tilde{X}_{d,:}$. Recalling that

$$M = \begin{bmatrix} B_{00} & B_{01} & \dots & B_{0(m-1)} \\ B_{10} & B_{11} & \dots & B_{1(m-1)} \\ \vdots & \vdots & \dots & \vdots \\ B_{(m-1)0} & B_{(m-1)1} & \dots & B_{(m-1)(m-1)} \end{bmatrix},$$

we have

$$\begin{aligned}
\frac{||M\text{vec}(X)||_2^2}{||\text{vec}(X)||_2^2} &= \frac{\sum_{c\in[m]} \left|\left|\sum_{d\in[m]} B_{cd}\tilde{X}_{d,:}^\top\right|\right|_2^2}{\sum_{d\in[m]} ||\tilde{X}_{d,:}||_2^2} \\
&\leq \frac{\sum_{c\in[m]} \sum_{d\in[m]} ||B_{cd}\tilde{X}_{d,:}^\top||_2^2}{\sum_{d\in[m]} ||\tilde{X}_{d,:}||_2^2} \\
&\leq \frac{\sum_{c\in[m]} \sum_{d\in[m]} ||B_{cd}||_2^2 ||\tilde{X}_{d,:}||_2^2}{\sum_{d\in[m]} ||\tilde{X}_{d,:}||_2^2} \\
&\leq \frac{\sum_{c\in[m]} (\max_{d\in[m]} ||B_{cd}||_2^2) \sum_{d\in[m]} ||\tilde{X}_{d,:}||_2^2}{\sum_{d\in[m]} ||\tilde{X}_{d,:}||_2^2} \\
&\leq \sum_{c\in[m]} \max_{d\in[m]} ||B_{cd}||_2^2.
\end{aligned}$$

Applying Proposition 6,

$$\begin{aligned}
\frac{||M\text{vec}(X)||_2}{||\text{vec}(X)||_2} &\leq \sqrt{\sum_{c\in[m]} \max_{d\in[m]} \left(\sum_{p\in[n],q\in[n]} |K_{c,d,p,q}| \right)^2} \\
&\leq \sum_{c,d\in[m],p,q\in[n]} |K_{c,d,p,q}|,
\end{aligned}$$

completing the proof. □

3 Regularization

We now show how to use the spectrum computed above to project a convolution onto the set of convolutions with bounded operator norm. The following proposition characterizes the projection of any matrix onto a ball with respect to the operator norm.

Proposition 12 ([Lefkimmiatis et al., 2013]) *Let $A \in \mathbf{R}^{n \times n}$, and let $A = UDV^\top$ be its singular value decomposition. Let $\tilde{A} = U\tilde{D}V^\top$, where, for all $i \in [n]$, $\tilde{D}_{ii} = \min(D_{ii}, c)$ and $\mathcal{B} = \{X \mid ||X||_2 < c\}$. Then $\tilde{A}$ is the projection of A onto $\mathcal{B}$; i.e. $\tilde{A} = \arg\min_{X \in \mathcal{B}} ||A - X||_F$.*

Thus we can use the spectrum computed above to project the transformation matrix for any convolutional layer onto the set of transformations with operator norms bounded by a given constant. Note that the eigenvectors remained the same in the proposition, hence the projected matrix is still generated by a convolution. However, after the projection, the resulting convolution neighborhood may become as large as $n \times n$. We now project this convolution onto the set of convolutions with $k \times k$ neighborhoods, by zeroing out all other coefficients.

We note that the two sets we are projecting onto — (1) the set of convolutions with operator norms bounded by a given constant and (2) the set of convolutions with $k \times k$ neighborhoods, are both convex sets. Thus repeating these projections alternately gets us to the closest convolution in their intersection [Cheney and Goldstein, 1959, Boyd and Dattorro, 2003]. In practice, we run the two projections once every few steps, thus letting the projection alternate with the training.

Here is the NumPy code for operator norm projection. The TensorFlow code is similar.

```
def Clip_OperatorNorm(kernel, input_shape, clip_to):
  transform_coefficients = np.fft.fft2(kernel, input_shape, axes=[0, 1])
  U, D, V = np.linalg.svd(transform_coefficients, compute_uv=True, full_matrices=False)
  D_clipped = np.minimum(D, clip_to)
  if kernel.shape[2] > kernel.shape[3]:
    clipped_transform_coefficients = np.matmul(U, D_clipped[..., None] * V)
  else:
    clipped_transform_coefficients = np.matmul(U * D_clipped[..., None, :], V)
  clipped_kernel = np.fft.ifft2(clipped_transform_coefficients, axes=[0, 1]).real
  return clipped_kernel[np.ix_(*[range(d) for d in kernel.shape])]
```

4 Experiments

First, we validated Theorem 8 with unit tests in which the output of the code given in the introduction is compared with evaluating the singular values by constructing the full matrix encoding the linear transformation corresponding to the convolutional layer and computing its SVD.

4.1 Timing

We generated 4D tensors of various shapes with random standard normal values, and computed their singular values using the full matrix method, the NumPy code given above and the equivalent TensorFlow code. For small tensors, the NumPy code was faster than TensorFlow, but for larger tensors, the TensorFlow code was able to exploit the parallelism in the algorithm and run much faster on a GPU. The timing results are shown in Figure 1.

4.2 Regularization

We next explored the effect of regularizing the convolutional layers by clipping their operator norms as described in Section 3. We ran the CIFAR-10 benchmark with a standard 32 layer residual network with 2.4M

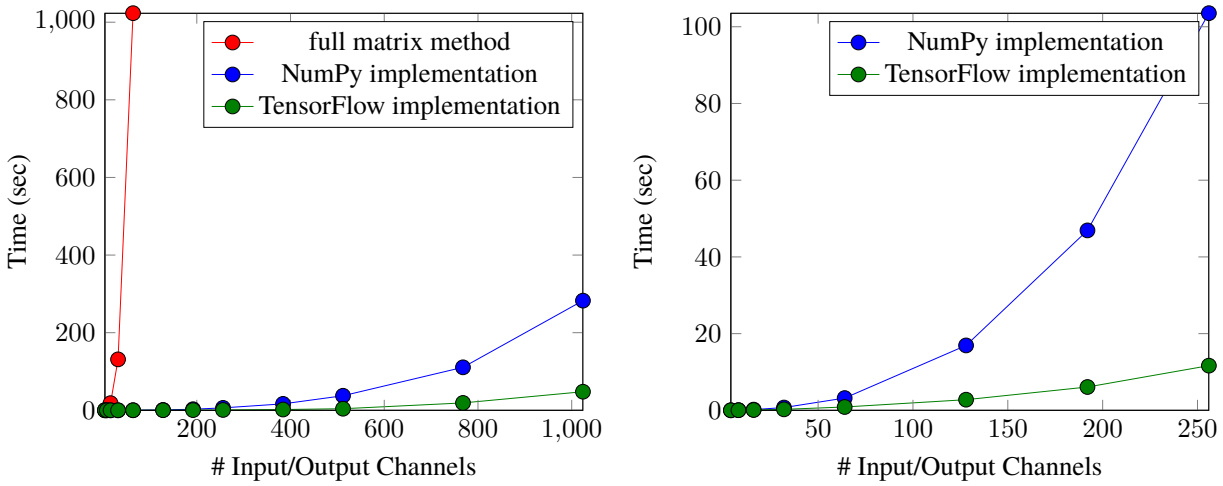

Figure 1: Time used to compute singular values. The left graph is for a 3×3 convolution on a 16×16 image with the number of input/output channels on the x-axis. The right graph is for a 11×11 convolution on a 64×64 image (no curve for full matrix method is shown as this method could not complete in a reasonable time for these inputs).

training parameters; [He et al., 2016]. This network reached a test error rate of 6.2% after 250 epochs, using a learning rate schedule determined by a grid search (shown by the gray plot in Figure 2). We then evaluated an algorithm that, every 100 steps, clipped the norms of the convolutional layers to various different values between 0.1 and 3.0. As expected, clipping to 2.5 and 3.0 had little impact on the performance, since the norms of the convolutional layers were between 2.5 and 2.8. Clipping to 0.1 yielded a surprising 6.7% test error, whereas clipping to 0.5 and 1.0 yielded test errors of 5.3% and 5.5% respectively (shown in Figure 2). A plot of test error against training time is provided in Figure 3, showing that the projections did not slow down the training very much.

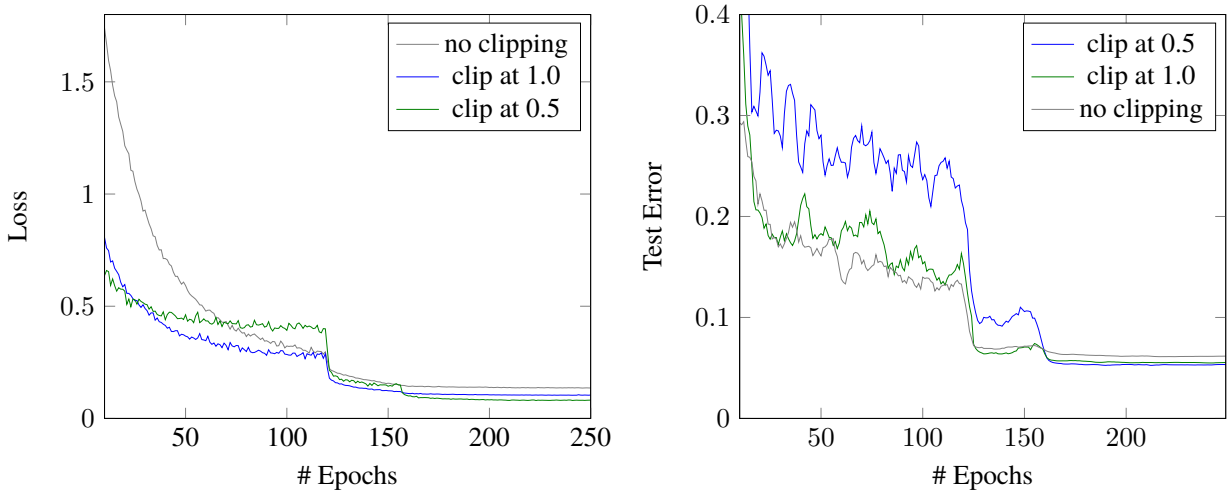

Figure 2: Training loss and test error for ResNet model [He et al., 2016] for CIFAR-10.

Figure 3 shows the plots of test error vs. training time in our CIFAR-10 experiment.

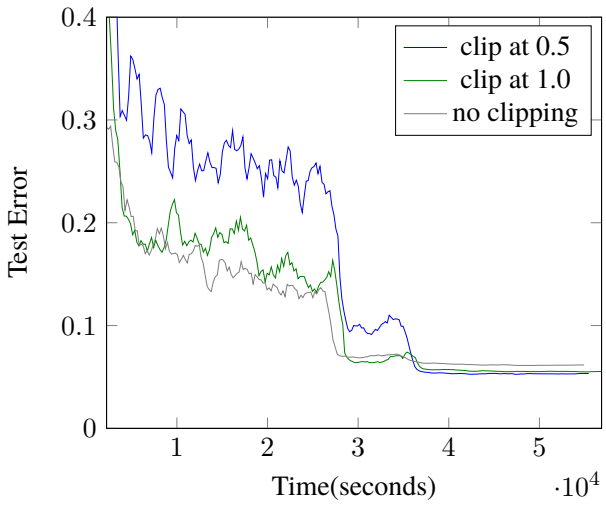

Figure 3: Test error vs. training time for ResNet model [He et al., 2016] for CIFAR-10.

We also evaluated the use of a regularizer, motivated by Proposition 11, that adds a penalty proportional to $\sum_{c,d,p,q} |K_{c,d,p,q}|$ to the loss. This did not yield as much of an improvement as our method based on clipping the operator norm; we also saw that Proposition 11 is sometimes a very loose upper bound.

4.3 The official pre-trained ResNet model

We computed the singular values of the convolutional layers from the official "Resnet V2" pre-trained model [He et al., 2016]. These are plotted in Figure 4.

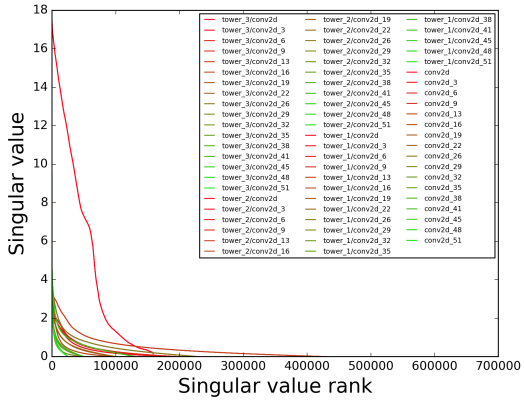

Figure 4: Plot of the singular values of the linear operators associated with the convolutional layers of the pretrained "ResNet V2" from the TensorFlow website. The singular values are ordered by value. Only layers with kernels larger than 1×1 are plotted. The curves are plotted with a mixture of red and green; layers closer to the input are plotted with colors with a greater share of red.

The transformations with the largest operator norms are closest to the inputs. As the data has undergone

more rounds of processing, the number of non-negligible singular values increases for a while, but at the end, it tapers off. One possible interpretation of this finding is as follows. Near the inputs, there are a few strong, but superficial, directions in the feature maps, such as edge detectors. After some processing, the feature maps are richer representations, so that variation in a wider variety of directions is meaningful. The final layers perform a few fine-grained adjustments to the features. The effective rank of the later layers of the network is getting small, which suggests that they could be compressed to obtain smaller models.

5 Conclusion

We characterized singular values corresponding to a two-dimensional multi-channel convolutional layer and provided an efficient and practical method for computing the singular values for deep convolutional networks. This characterization opens a door to various regularizers. We showed that we can effectively project each layer to a set of bounded operator norm and experiments demonstrated significant improvement in test error on CIFAR-10 on TensorFlow ResNet model. Our method is easily extendable to 3D models for analyzing videos. Our analysis can help improve the performance on any deep convolutional model. This is especially valuable in the domains where regularization is challenging such as improving the performance of state-of-the-art GAN models.

6 Acknowledgements

We thank Tomer Koren, Nishal Shah and Yoram Singer for valuable conversations.

References

Stephen Boyd and Jon Dattorro. Alternating projections, 2003. `https://web.stanford.edu/class/ee392o/alt_proj.pdf`.

Chong-Yun Chao. A note on block circulant matrices. *Kyungpook Mathematical Journal*, 14:97–100, 1974.

Ward Cheney and Allen A. Goldstein. Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, 10(3):448–450, 1959. ISSN 00029939, 10886826. URL `http://www.jstor.org/stable/2032864`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018a.

Henry Gouk, Bernhard Pfahringer, Eibe Frank, and Michael Cree. MaxGain: Regularisation of neural networks by constraining activation magnitudes. *arXiv preprint arXiv:1804.05965*, 2018b.

Robert M Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. `http://download.tensorflow.org/models/official/resnet_v2_imagenet_checkpoint.tar.gz`; downloaded on on 5/1/18.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1, 1991.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 2nd edition, 2012. ISBN 0521548233, 9780521548236.

Anil K Jain. *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall,, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Stamatios Lefkimmiatis, John Paul Ward, and Michael Unser. Hessian schatten-norm regularization for linear inverse problems. *IEEE transactions on image processing*, 22(5):1873–1888, 2013.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018.

Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pages 4788–4798, 2017.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

William Henry Young. On the multiplication of successions of fourier constants. *Proc. R. Soc. Lond. A*, 87(596): 331–339, 1912.