Lipschitz Properties of General Convolutional Neural Networks

Radu Balan

Department of Mathematics, AMSC, CSCAMM and NWC
University of Maryland, College Park, MD

Joint work with Dongmian Zou (UMD), Maneesh Singh (Verisk)

March 22, 2017
Image Analysis Seminar
University of Houston, Houston TX
"This material is based upon work supported by the National Science Foundation under Grant No. DMS-1413249. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.” The author has been partially supported by ARO under grant W911NF1610008.
Table of Contents:

1 Problem Formulation
2 Deep Convolutional Neural Networks
3 Lipschitz Analysis
4 Numerical Results
Consider a nonlinear function between two metric spaces,

\[ \mathcal{F} : (X, d_X) \rightarrow (Y, d_Y). \]
Problem Formulation

Lipschitz analysis of nonlinear systems

\[ \mathcal{F} : (X, d_X) \rightarrow (Y, d_Y) \]

\( \mathcal{F} \) is called \textit{Lipschitz} with constant \( C \) if for any \( f, \tilde{f} \in X \),

\[ d_Y(\mathcal{F}(f), \mathcal{F}(\tilde{f})) \leq C \ d_X(f, \tilde{f}) \]

The optimal (i.e. smallest) Lipschitz constant is denoted \( \text{Lip}(\mathcal{F}) \). The square \( C^2 \) is called Lipschitz bound (similar to the Bessel bound).

\( \mathcal{F} \) is called \textit{bi-Lipschitz} with constants \( C_1, C_2 > 0 \) if for any \( f, \tilde{f} \in X \),

\[ C_1 \ d_X(f, \tilde{f}) \leq d_Y(\mathcal{F}(f), \mathcal{F}(\tilde{f})) \leq C_2 \ d_X(f, \tilde{f}) \]

The square \( C_1^2, C_2^2 \) are called \textit{Lipschitz bounds} (similar to frame bounds).
Consider the typical neural network as a feature extractor component in a classification system:

$$g = \mathcal{F}(f) = \mathcal{F}_M(...\mathcal{F}_1(f; W_1, \varphi_1); ...; W_M, \varphi_M)$$

$$\mathcal{F}_m(f; W_m, \varphi_m) = \varphi_m(W_m f)$$

$W_m$ is a linear operator (matrix); $\varphi_m$ is a Lip(1) scalar nonlinearity (e.g. Rectified Linear Unit).
Problem Formulation
Motivating Example: AlexNet

Example from [SZSBEGF13] ('Intriguing properties ...'). ImageNet dataset [DDSLLF09] (10,184 categories; 8.9 mil.img.); AlexNet architecture [KSH12]:

Figure: From Krizhevsky et al. 2012 [KSH12]: AlexNet: 5 convolutive layers + 3 dense layers. Input size: 224x224x3 pixels. Output size: 1000.
Problem Formulation
Motivating Example: AlexNet

The authors of [SZSBEGF13] ('Intriguing properties ...') found small variations of the input, almost imperceptible, that produced completely different classification decisions:

**Figure:** From Szegedy et al 2013 [SZSBEGF13]: AlexNet: 6 different classes: original image, difference, and adversarial example – all classified as 'ostrich'
Szegedy et al. 2013 [SZSBEGF13] computed the Lipschitz constants of each layer.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
<th>Sing. Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv. 1</td>
<td>$3 \times 11 \times 11 \times 96$</td>
<td>20</td>
</tr>
<tr>
<td>Conv. 2</td>
<td>$96 \times 5 \times 5 \times 256$</td>
<td>10</td>
</tr>
<tr>
<td>Conv. 3</td>
<td>$256 \times 3 \times 3 \times 384$</td>
<td>7</td>
</tr>
<tr>
<td>Conv. 4</td>
<td>$384 \times 3 \times 3 \times 384$</td>
<td>7.3</td>
</tr>
<tr>
<td>Conv. 5</td>
<td>$384 \times 3 \times 3 \times 256$</td>
<td>11</td>
</tr>
<tr>
<td>Fully Conn.1</td>
<td>$9216(43264) \times 4096$</td>
<td>3.12</td>
</tr>
<tr>
<td>Fully Conn.2</td>
<td>$4096 \times 4096$</td>
<td>4</td>
</tr>
<tr>
<td>Fully Conn.3</td>
<td>$4096 \times 1000$</td>
<td>4</td>
</tr>
</tbody>
</table>

Overall Lipschitz constant:

$$Lip \leq 20 \times 10 \times 7 \times 7.3 \times 11 \times 3.12 \times 4 \times 4 = 5,612,006$$
Motivating Example: Scattering Network

Example of Scattering Network; definition and properties: [Mallat12]; this example from [BSZ17]:

Input: $f$; Outputs: $y = (y_{l,k})$. 
Problem Formulation
Motivating Example: Scattering Network

Remarks:
- Outputs from each layer

Lipschitz Analysis of CNN
Problem Formulation

Motivating Example: Scattering Network

Remarks:
- Outputs from each layer
- Tree-like topology
Problem Formulation
Motivating Example: Scattering Network

Remarks:
- Outputs from each layer
- Tree-like topology
- Backpropagation/Chain rule: Lipschitz bound 40.
Problem Formulation
Motivating Example: Scattering Network

Remarks:
- Outputs from each layer
- Tree-like topology
- Backpropagation/Chain rule: Lipschitz bound 40.
- Mallat’s result predicts $\text{Lip} = 1$. 
Problem Formulation

Problem 1

Given a deep network:

Estimate the Lipschitz constant, or bound:

\[ \text{Lip} = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|_2}{\|f - \tilde{f}\|_2}, \quad \text{Bound} = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|_2^2}{\|f - \tilde{f}\|_2^2}. \]
Problem Formulation

Problem 1

Given a deep network:

Estimate the Lipschitz constant, or bound:

\[
Lip = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|_2}{\|f - \tilde{f}\|_2}, \quad Bound = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|^2}{\|f - \tilde{f}\|^2}.
\]

Methods (Approaches):

1. Standard Method: Backpropagation, or chain-rule
2. New Method: Storage function based approach (dissipative systems)
3. Numerical Method: Simulations
Given a deep network:

Estimate the stability of the output to specific variations of the input:

1. Invariance to deformations: \( \tilde{f}(x) = f(x - \tau(x)) \), for some smooth \( \tau \).
2. Covariance to such deformations \( \tilde{f}(x) = f(x - \tau(x)) \), for smooth \( \tau \) and bandlimited signals \( f \);
3. Tail bounds when \( f \) has a known statistical distribution (e.g. normal with known spectral power)
A deep convolution network is composed of multiple layers:

```
Input nodes

Layer 1

Output nodes for Layer 1

Layer 2

Output nodes for Layer 2

Layer M

Output nodes for Layer (M-1)

Output nodes for Layer M
```

Radu Balan (UMD)
ConvNet
One Layer

Each layer is composed of two or three sublayers: convolution, downsampling, detection/pooling/merge.

```
Pooling filters -> Outputs (Feature)

Input nodes -> Convolutional filters -> Downsample / Dilation

Detection & Pooling / Merge -> Output nodes
```
ConvNet: Sublayers
Linear Filters: Convolution and Pooling-to-Output Sublayer

\[ f^{(2)} = g \ast f^{(1)}, \quad f^{(2)}(x) = \int g(x - \xi) f^{(1)}(\xi) d\xi \]

where \( g \in \mathcal{B} = \{ g \in S', \hat{g} \in L^\infty(\mathbb{R}^d) \} \).

\( (\mathcal{B}, \ast) \) is a Banach algebra with norm \( \| g \|_{\mathcal{B}} = \| \hat{g} \|_{\infty} \).
Notation: \( g \) for regular convolution filters, and \( \Phi \) for pooling-to-output filters.
ConvNet: Sublayers

Downsampling Sublayer

\[ f^{(1)} \xrightarrow{\downarrow D} f^{(2)} \]

\[ f^{(2)}(x) = f^{(1)}(Dx) \]

For \( f^{(1)} \in L^2(\mathbb{R}^d) \) and \( D = D_0 \cdot I \), \( f^{(2)} \in L^2(\mathbb{R}^d) \) and

\[
\| f^{(2)} \|_2^2 = \int_{\mathbb{R}^d} |f^{(2)}(x)|^2 \, dx = \frac{1}{|\det(D)|} \int_{\mathbb{R}^d} |f^{(1)}(x)|^2 \, dx = \frac{1}{D_0^d} \| f^{(1)} \|_2^2
\]
ConvNet: Sublayers
Detection and Pooling Sublayer

We consider three types of detection/pooling/merge sublayers:

- **Type I, \( \tau_1 \):** Componentwise Addition: \( z = \sum_{j=1}^{k} \sigma_j(y_j) \)
- **Type II, \( \tau_2 \):** \( p \)-norm aggregation: \( z = \left( \sum_{j=1}^{k} |\sigma_j(y_j)|^p \right)^{1/p} \)
- **Type III, \( \tau_3 \):** Componentwise Multiplication: \( z = \prod_{j=1}^{k} \sigma_j(y_j) \)

**Assumptions:**
1. \( \sigma_j \) are scalar Lipschitz functions with \( Lip(\sigma_j) \leq 1 \);
2. If \( \sigma_j \) is connected to a multiplication block then \( \|\sigma_j\|_\infty \leq 1 \).
MaxPooling can be implemented as follows:

\[ y \xrightarrow{g} \text{max pooling} \quad \iff \quad y \xrightarrow{\mathcal{T}_{V_n} g} \downarrow D \cdot \cdots \cdot \downarrow D \xrightarrow{\text{aggregation}} \]
ConvNet: Sublayers
MaxPooling and AveragePooling

MaxPooling can be implemented as follows:

AveragePooling can be implemented as follows:
ConvNet: Layer $m$

Components of the $m^{th}$ layer

\[ \phi_{m,1} \quad \downarrow D_{m,1;1} \quad \sigma_{m,1;1} \]
\[ g_{m,1;1} \]
\[ g_{m,1;2} \quad \downarrow D_{m,1;2} \quad \sigma_{m,1;2} \]
\[ \ldots \]
\[ g_{m,1;k_{m,1}} \quad \downarrow D_{m,1;k_{m,1}} \quad \sigma_{m,1;k_{m,1}} \]

Detection/Pool:

\[ N'_{m,1} \]
\[ N'_{m,2} \]
\[ \ldots \]

\[ \phi_{m,n_m} \quad \downarrow D_{m,n_m;1} \quad \sigma_{m,n_m;1} \]
\[ g_{m,n_m;1} \]
\[ g_{m,n_m;2} \quad \downarrow D_{m,n_m;2} \quad \sigma_{m,n_m;2} \]
\[ \ldots \]
\[ g_{m,n_m;k_{m,n_m}} \quad \downarrow D_{m,n_m;k_{m,n_m}} \quad \sigma_{m,n_m;k_{m,n_m}} \]

\[ N'_{m,n_m} \]
ConvNet: Layer $m$

Topology coding of the $m^{th}$ layer

$n_m$ denotes the number of input nodes in the $m$-th layer:

$I_m = \{N_m,1, N_m,2, \cdots, N_m,n_m\}$.

Filters:

1. pooling filter: $\phi_{m,n}$ for node $n$, in layer $m$;
2. convolution filter: $g_{m,n,k}$ for input node $n$ to output node $k$, in layer $m$;

For node $n$: $G_{m,n} = \{g_{m,n;1}, \cdots g_{m,n;k_m,n}\}$.

The set of all convolution filters in layer $m$: $G_m = \bigcup_{n=1}^{n_m} G_{m,n}$.
ConvNet: Layer \( m \)

Topology coding of the \( m^{th} \) layer

\( n_m \) denotes the number of input nodes in the \( m \)-th layer:
\[
\mathcal{I}_m = \{ N_{m,1}, N_{m,2}, \ldots, N_{m,n_m} \}.
\]

Filters:

1. pooling filter: \( \phi_{m,n} \) for node \( n \), in layer \( m \);
2. convolution filter: \( g_{m,n,k} \) for input node \( n \) to output node \( k \), in layer \( m \);

For node \( n \):
\[
G_{m,n} = \{ g_{m,n;1}, \ldots, g_{m,n; k_m,n} \}.
\]

The set of all convolution filters in layer \( m \):
\[
G_m = \bigcup_{n=1}^{n_m} G_{m,n}.
\]

\( \mathcal{O}_m = \{ N'_{m,1}, N'_{m,2}, \ldots, N'_{m,n'_m} \} \) the set of output nodes of the \( m \)-th layer.

Note that \( n'_m = n_{m+1} \) and there is a one-one correspondence between \( \mathcal{O}_m \) and \( \mathcal{I}_{m+1} \).

The output nodes automatically partitions \( G_m \) into \( n'_m \) disjoint subsets
\[
G_m = \bigcup_{n'=1}^{n'_m} G'_{m,n'}, \text{ where } G'_{m,n'} \text{ is the set of filters merged into } N'_{m,n'}.
\]
ConvNet: Layer $m$
Topology coding of the $m^{th}$ layer

For each filter $g_{m,n;k}$, we define an associated multiplier $l_{m,n;k}$ in the following way: suppose $g_{m,n;k} \in G'_{m,n'}$, let $K = |G'_{m,n'}|$ denote the cardinality of $G'_{m,n'}$. Then

$$l_{m,n;k} = \begin{cases} K & \text{, if } g_{m,n;k} \in \tau_1 \cup \tau_3 \\ K^{\max\{0,2/p-1\}} & \text{, if } g_{m,n;k} \in \tau_2 \end{cases}$$

(2.1)
Layer Analysis
Bessel Bounds

In each layer $m$ and for each input node $n$ we define three types of Bessel bounds:

- 1st type Bessel bound:

$$B_{m,n}^{(1)} = \| \phi_{m,n} \|^2 + \sum_{k=1}^{k_{m,n}} l_{m,n;k} D_{m,n;k}^{-d} |\hat{g}_{m,n;k}|^2 \|_{\infty}$$  \hspace{1cm} (3.2)

- 2nd type Bessel bound:

$$B_{m,n}^{(2)} = \| \sum_{k=1}^{k_{m,n}} l_{m,n;k} D_{m,n;k}^{-d} |\hat{g}_{m,n;k}|^2 \|_{\infty}$$ \hspace{1cm} (3.3)

- 3rd type (or generating) bound:

$$B_{m,n}^{(3)} = \| \phi_{m,n} \|^2_{\infty}.$$ \hspace{1cm} (3.4)
Layer Analysis
Bessel Bounds

Next we define the layer $m$ Bessel bounds:

1st type Bessel bound $B_m^{(1)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(1)}$ \hspace{1cm} (3.5)

2nd type Bessel bound $B_m^{(2)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(2)}$ \hspace{1cm} (3.6)

3rd type (generating) Bessel bound $B_m^{(3)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(3)}$. \hspace{1cm} (3.7)

Remark. These bounds characterize semi-discrete Bessel systems.
Theorem

[BSZ17] Consider a Convolutional Neural Network with $M$ layers as described before, where all scalar nonlinear functions are Lipschitz with $\text{Lip}(\varphi_{m,n,n'}) \leq 1$. Additionally, those $\varphi_{m,n,n'}$ that aggregate into a multiplicative block satisfy $\|\varphi_{m,n,n'}\|_\infty \leq 1$. Let the $m$-th layer 1st type Bessel bound be

$$B_m^{(1)} = \max_{1 \leq n \leq n_m} \|\hat{\phi}_{m,n}\|^2 + \sum_{k=1}^{l_{m,n;k}} I_{m,n;k} D_{m,n;k}^{-d} \|\hat{g}_{m,n;k}\|^2 \|_{\infty}.$$

Then the Lipschitz bound of the entire CNN is upper bounded by $\prod_{m=1}^{M} \max(1, B_m^{(1)})$. Specifically, for any $f, \tilde{f} \in L^2(\mathbb{R}^d)$:

$$\|F(f) - F(\tilde{f})\|_2^2 \leq \left( \prod_{m=1}^{M} \max(1, B_m^{(1)}) \right) \|f - \tilde{f}\|_2^2.$$
Lipschitz Analysis
Second Result

Theorem

Consider a Convolutional Neural Network with $M$ layers as described before, where all scalar nonlinearities satisfy the same conditions as in the previous result. For layer $m$, let $B_{m}^{(1)}$, $B_{m}^{(2)}$, and $B_{m}^{(3)}$ denote the three Bessel bounds defined earlier. Denote by $L$ the optimal solution of the following linear program:

$$
\Gamma = \max_{y_1, \ldots, y_M, z_1, \ldots, z_M \geq 0} \sum_{m=1}^{M} z_m \\
\text{s.t. } y_0 = 1 \quad \text{(3.8)}
$$

$$
\begin{align*}
y_m + z_m & \leq B_{m}^{(1)} y_{m-1}, & 1 \leq m \leq M \\
y_m & \leq B_{m}^{(2)} y_{m-1}, & 1 \leq m \leq M \\
z_m & \leq B_{m}^{(3)} y_{m-1}, & 1 \leq m \leq M
\end{align*}
$$
Lipschitz Analysis
Second Result - cont’d

Theorem

Then the Lipschitz bound satisfies $\text{Lip}(F)^2 \leq \Gamma$. Specifically, for any $f, \tilde{f} \in L^2(\mathbb{R}^d)$:

$$\|F(f) - F(\tilde{f})\|^2_2 \leq \Gamma \|f - \tilde{f}\|^2_2,$$
Example 1: Scattering Network

The Lipschitz constant:

- Backpropagation/Chain rule: Lipschitz bound 40 (hence \( Lip \leq 6.3 \)).
Example 1: Scattering Network

The Lipschitz constant:

- Backpropagation/Chain rule: Lipschitz bound 40 (hence $Lip \leq 6.3$).
- Using our main theorem, $Lip \leq 1$, but Mallat’s result: $Lip = 1$.

Filters have been chosen as in a dyadic wavelet decomposition. Thus $B_m^{(1)} = B_m^{(2)} = B_m^{(3)} = 1, 1 \leq m \leq 4$. 

Radu Balan (UMD)
Example 2: A General Convolutive Neural Network
Example 2: A General Convolutive Neural Network

Set $p = 2$ and:

$$F(\omega) = \exp\left(\frac{4\omega^2 + 4\omega + 1}{4\omega^2 + 4\omega}\right)\chi(-1,-1/2)(\omega) + \chi(-1/2,1/2)(\omega) + \exp\left(\frac{4\omega^2 - 4\omega + 1}{4\omega^2 - 4\omega}\right)\chi(1/2,1)(\omega).$$

\[
\begin{align*}
\hat{\phi}_1(\omega) &= F(\omega) \\
\hat{g}_{1,j}(\omega) &= F(\omega + 2j - 1/2) + F(\omega - 2j + 1/2) , \ j = 1, 2, 3, 4 \\
\hat{\phi}_2(\omega) &= \exp\left(\frac{4\omega^2 + 12\omega + 9}{4\omega^2 + 12\omega + 8}\right)\chi(-2,-3/2)(\omega) + \\
&\quad \chi(-3/2,3/2)(\omega) + \exp\left(\frac{4\omega^2 - 12\omega + 9}{4\omega^2 - 12\omega + 8}\right)\chi(3/2,2)(\omega) \\
\hat{g}_{2,j}(\omega) &= F(\omega + 2j) + F(\omega - 2j) , \ j = 1, 2, 3 \\
\hat{g}_{2,4}(\omega) &= F(\omega + 2) + F(\omega - 2) \\
\hat{g}_{2,5}(\omega) &= F(\omega + 5) + F(\omega - 5) \\
\hat{\phi}_3(\omega) &= \exp\left(\frac{4\omega^2 + 20\omega + 25}{4\omega^2 + 20\omega + 24}\right)\chi(-3,-5/2)(\omega) + \\
&\quad \chi(-5/2,5/2)(\omega) + \exp\left(\frac{4\omega^2 - 20\omega + 25}{4\omega^2 - 20\omega + 25}\right)\chi(5/2,3)(\omega).
\end{align*}
\]
Example 2: A General Convolutive Neural Network

Bessel Bounds: \( B_m^{(1)} = 2e^{-1/3} = 1.43 \), \( B_m^{(2)} = B_m^{(3)} = 1 \).

The Lipschitz bound:

- Using backpropagation/chain-rule: \( \text{Lip}^2 \leq 5 \).
- Using Theorem 1: \( \text{Lip}^2 \leq 2.9430 \).
- Using Theorem 2 (linear program): \( \text{Lip}^2 \leq 2.2992 \).
References


[LSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir, On the computational efficiency of training neural networks, Advances in


