When Harmonic Analysis Meets Machine Learning: Lipschitz Analysis of Deep Convolution 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)

October 10, 2017
IEEE Computational Intelligence Society and Signal Processing Society
University of Maryland, College Park, MD

"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 and LTS under grant H9823013D00560049.

Table of Contents:

Machine Learning

According to Wikipedia (attributed to Arthur Samuel 1959), "Machine Learning [...] gives computers the ability to learn without being explicitly programmed."
While it has been first coined in 1959, today's machine learning, as a field, evolved from and overlaps with a number of other fields: computational statistics, mathematical optimizations, theory of linear and nonlinear systems.

Machine Learning

According to Wikipedia (attributed to Arthur Samuel 1959), "Machine Learning [...] gives computers the ability to learn without being explicitly programmed."

While it has been first coined in 1959, today's machine learning, as a field, evolved from and overlaps with a number of other fields: computational statistics, mathematical optimizations, theory of linear and nonlinear systems.

Types of problems (tasks) in machine learning:

1. Supervised Learning: The machine (computer) is given pairs of inputs and desired outputs and is left to learn the general association rule.
2. Unsupervised Learning: The machine is given only input data, and is left to discover structures (patterns) in data.
3. Reinforcement Learning: The machine operates in a dynamic environment and had to adapt (learn) continuously as it navigates the problem space (e.g. autonomous vehicle).

Example 1: The AlexNet

The ImageNet Dataset

Dataset: ImageNet dataset [DDSLLF09]. Currently (2017): 14.2 mil.images; 21841 categories; image-net.org

Task: Classify an input image, i.e. place it into one category.

Figure: The "ostrich" category "Struthio Camelus" 1393 pictures. From image-net.org

Example 1: The AlexNet

The Supervised Machine Learning

The AlexNet is 8 layer network, 5 convolutive layers plus 3 dense layers. Introduced by (Alex) Krizhevsky, Sutskever and Hinton in 2012 [KSH12]. Trained on a subset of the ImageNet: Part of the ImageNet Large Scale Visual Recognition Challenge 2010-2012: 1000 object classes and 1,431,167 images.

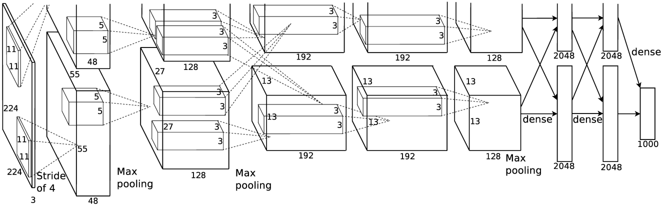


Figure: From Krizhevsky et all 2012 [KSH12]: AlexNet: 5 convolutive layers + 3 dense layers. Input size: 224x224x3 pixels. Output size: 1000.

Example 1: The AlexNet
Adversarial Perturbations

The authors of [SZSBEGF13] (Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus, 'Intriguing properties ...') found small variations of the input, almost imperceptible, that produced completely different classification decisions:

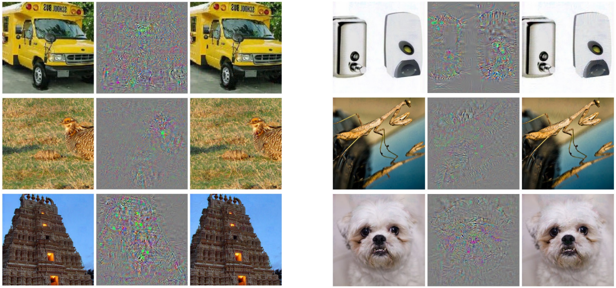


Figure: From Szegedy et all 2013 [SZSBEGF13]: AlexNet: 6 different classes: original image, difference, and adversarial example – all classified as 'ostrich'

Example 1: The AlexNet
Lipschitz Analysis

Szegedy et all 2013 [SZSBEGF13] computed the Lipschitz constants of each layer.

Layer	Size	Sing.Val
Conv. 1	$3 \times 11 \times 11 \times 96$	20
Conv. 2	$96 \times 5 \times 5 \times 256$	10
Conv. 3	$256 \times 3 \times 3 \times 384$	7
Conv. 4	$384 \times 3 \times 3 \times 384$	7.3
Conv. 5	$384 \times 3 \times 3 \times 256$	11
Fully Conn.1	$9216(43264) \times 4096$	3.12
Fully Conn.2	4096×4096	4
Fully Conn.3	4096×1000	4

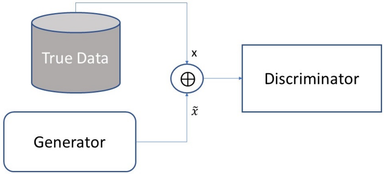
Overall Lipschitz constant:

$$Lip \leq 20 * 10 * 7 * 7.3 * 11 * 3.12 * 4 * 4 = 5,612,006$$

Example 2: Generative Adversarial Networks

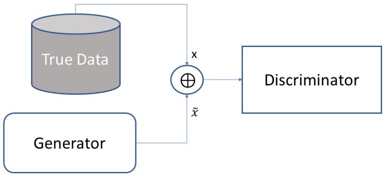
The GAN Problem

Two systems are involved: a *generator* network producing synthetic data; a *discriminator* network that has to decide if its input is synthetic data or real-world (true) data:



Example 2: Generative Adversarial Networks
The GAN Problem

Two systems are involved: a *generator* network producing synthetic data; a *discriminator* network that has to decide if its input is synthetic data or real-world (true) data:



Introduced by Goodfellow et al [GPMXWOCB14] in 2014, GANs solve a minimax optimization problem:

$$\min_G \max_D \mathbb{E}_{x\sim P_r}\left[log(D(x))\right] + \mathbb{E}_{\tilde{x}\sim P_g}\left[log(1 - D(\tilde{x}))\right]$$

where P_r is the distribution of true data, P_g is the generator distribution, and $D : x \mapsto D(x) \in [0, 1]$ is the discriminator map (1 for likely true data; 0 for likely synthetic data).

Example 2: Generative Adversarial Networks

The Wasserstein Optimization Problem

In practice, the training algorithms do not behave well ("saddle point effect").
The Wasserstein GAN (Arjovsky et al [ACB17]) replaces the Jensen-Shannon divergence by the Wasserstein-1 distance:

$$\min_{G} \max_{D \in Lip(1)} \mathbb{E}_{x \sim P_r}\left[D(x)\right] - \mathbb{E}_{\tilde{x} \sim P_g}\left[D(\tilde{x})\right]$$

where $Lip(1)$ denotes the set of Lipschitz functions with constant 1, enforced by weight clipping.

Example 2: Generative Adversarial Networks

The Wasserstein Optimization Problem

In practice, the training algorithms do not behave well ("saddle point effect").
The Wasserstein GAN (Arjovsky et al [ACB17]) replaces the Jensen-Shannon divergence by the Wasserstein-1 distance:

$$\min_{G} \max_{D \in Lip(1)} \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{\tilde{x} \sim P_g}[D(\tilde{x})]$$

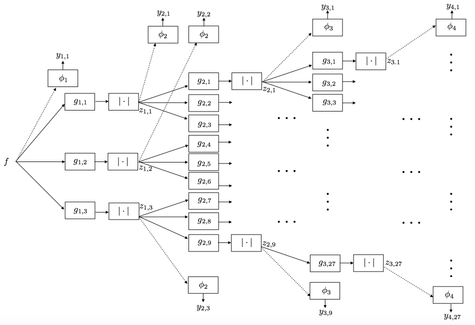
where $Lip(1)$ denotes the set of Lipschitz functions with constant 1, enforced by weight clipping.
Gulrajani et al in [GAADC17] propose to incorporate the Lip(1) condition into the optimization criterion using a soft Lagrange multiplier technique for minimization of:

$$L = \mathbb{E}_{\tilde{x} \sim P_g}[D(x)] - \mathbb{E}_{x \sim P_r}[D(x)] + \lambda \, \mathbb{E}_{\hat{x} \sim P_{\hat{x}}}\left[\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]$$

where $\hat{x}$ is sampled uniformly between $x \sim P_r$ and $\tilde{x} \sim P_g$.

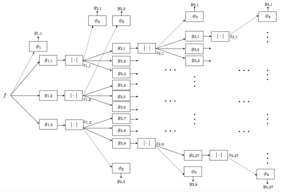
Example 3: The Scattering Network
Topology

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



Input: f; Outputs: $y = (y_{l,k})$.

Example 3: Scattering Network
Lipschitz Analysis

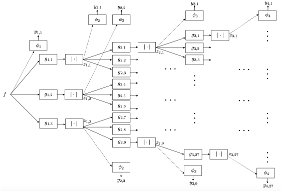


Remarks:

- Outputs from each layer

Example 3: Scattering Network

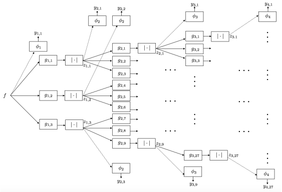
Lipschitz Analysis



Remarks:

- Outputs from each layer
- Tree-like topology

Example 3: Scattering Network
Lipschitz Analysis

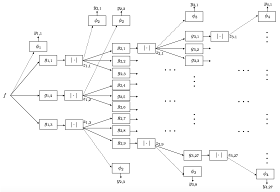


Remarks:

- Outputs from each layer
- Tree-like topology
- Backpropagation/Chain rule: Lipschitz bound 40.

Example 3: Scattering Network

Lipschitz Analysis



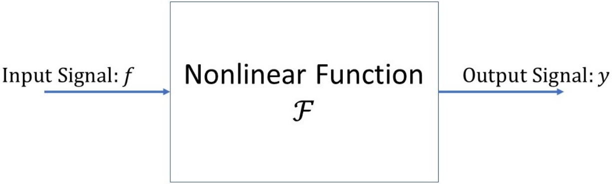
Remarks:

- Outputs from each layer
- Tree-like topology
- Backpropagation/Chain rule: Lipschitz bound 40.
- Mallat's result predicts $Lip = 1$.

Problem Formulation
Nonlinear Maps

Consider a nonlinear function between two metric spaces,

$$\mathcal{F} : (X, d_X) \to (Y, d_Y).$$



Problem Formulation

Lipschitz analysis of nonlinear systems

$$\mathcal{F} : (X, d_X) \to (Y, d_Y)$$

$\mathcal{F}$ is called *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 $Lip(\mathcal{F})$. The square C^2 is called Lipschitz bound (similar to the Bessel bound).

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

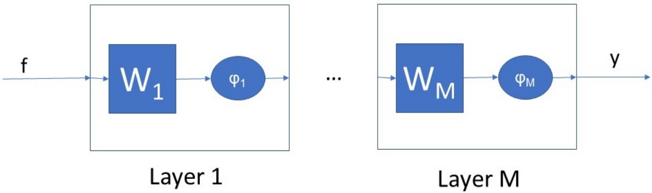
$$C_1 \; d_X(f, \tilde{f}) \leq d_Y(\mathcal{F}(f), \mathcal{F}(\tilde{f})) \leq C_2 \; d_X(f, \tilde{f})$$

The square C_1^2, C_2^2 are called *Lipschitz bounds* (similar to frame bounds).

Problem Formulation

Motivating Examples

Consider the typical neural network as a feature extractor component in a classification system:



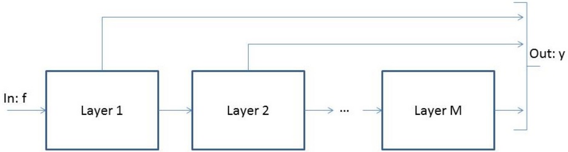
$$g = \mathcal{F}(f) = \mathcal{F}_M(...\mathcal{F}_1(f; W_1, \varphi_1); ...; W_M, \varphi_M)$$

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

W_m is a linear operator (matrix); φ_m is a Lip(1) scalar nonlinearity (e.g. Rectified Linear Unit).

Problem Formulation
Problem 1

Given a deep network:



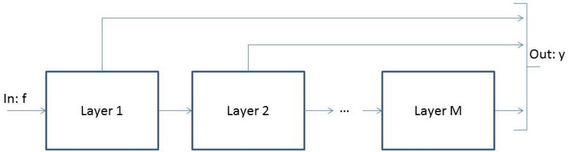
Estimate the Lipschitz constant, or bound:

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

Problem Formulation

Problem 1

Given a deep network:



Estimate the Lipschitz constant, or bound:

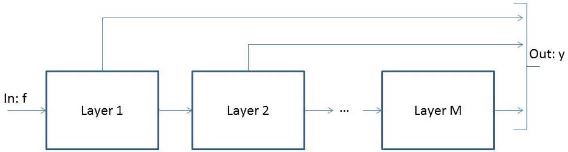
$$Lip = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|_2}{\|f - \tilde{f}\|_2} \quad , \quad Bound = \sup_{f \neq \tilde{f} \in L^2} \frac{\|y - \tilde{y}\|_2^2}{\|f - \tilde{f}\|_2^2}.$$

Methods (Approaches):

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

Problem Formulation
Problem 2

Given a deep network:

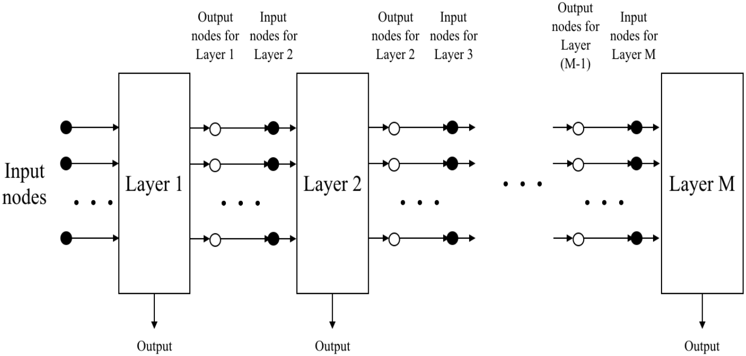


Estimate the stability of the output to specific variations of the input:

1. Invariance to deformations: $\tilde{f}(x) = f(x - \tau(x))$, for some smooth τ.
2. Covariance to such deformations $\tilde{f}(x) = f(x - \tau(x))$, for smooth τ and bandlimited signals f;
3. Tail bounds when f has a known statistical distribution (e.g. normal with known spectral power)

ConvNet
Topology

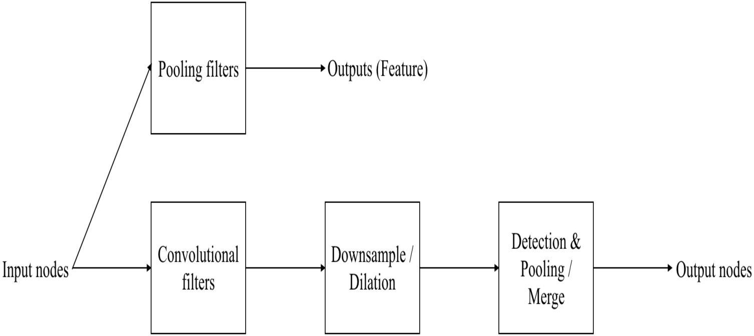
A deep convolution network is composed of multiple layers:



ConvNet

One Layer

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



ConvNet: Sublayers

Linear Filters: Convolution and Pooling-to-Output Sublayer

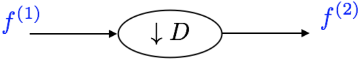
$$f^{(2)} = g * f^{(1)} \quad , \quad f^{(2)}(x) = \int g(x-\xi) f^{(1)}(\xi) d\xi$$

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

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

ConvNet: Sublayers

Downsampling Sublayer



$$f^{(2)}(x) = f^{(1)}(Dx)$$

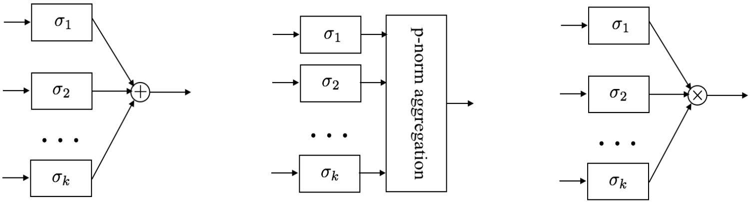
For $f^{(1)} \in L^2(\mathbb{R}^d)$ and $D = D_0 \cdot I$, $f^{(2)} \in L^2(\mathbb{R}^d)$ and

$$\|f^{(2)}\|_2^2 = \int_{\mathbb{R}^d} |f^{(2)}(x)|^2 dx = \frac{1}{|det(D)|} \int_{\mathbb{R}^d} |f^{(1)}(x)|^2 dx = \frac{1}{D_0^d} \|f^{(1)}\|_2^2$$

ConvNet: Sublayers
Detection and Pooling Sublayer

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

- Type I, τ_1: Componentwise Addition: $z = \sum_{j=1}^{k} \sigma_j(y_j)$
- Type II, τ_2: p-norm aggregation: $z = \left(\sum_{j=1}^{k} |\sigma_j(y_j)|^p\right)^{1/p}$
- Type III, τ_3: Componentwise Multiplication: $z = \prod_{j=1}^{k} \sigma_j(y_j)$

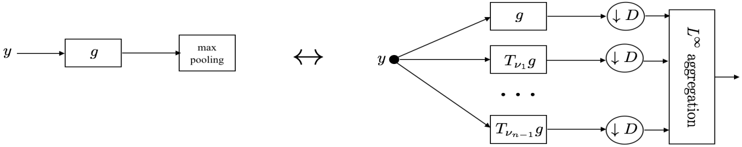


Assumptions: (1) σ_j are scalar Lipschitz functions with $Lip(\sigma_j) \leq 1$; (2) If σ_j is connected to a multiplication block then $\|\sigma_j\|_\infty \leq 1$.

ConvNet: Sublayers

MaxPooling and AveragePooling

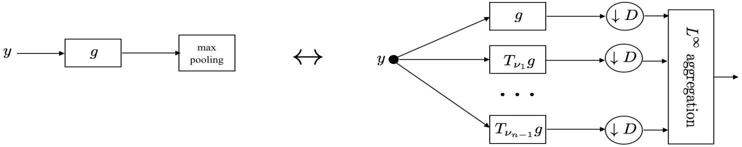
MaxPooling can be implemented as follows:



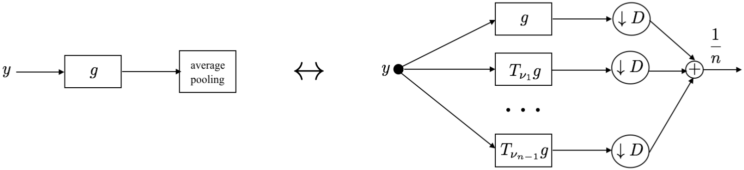
ConvNet: Sublayers

MaxPooling and AveragePooling

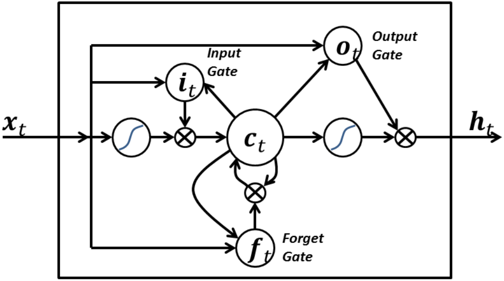
MaxPooling can be implemented as follows:



AveragePooling can be implemented as follows:



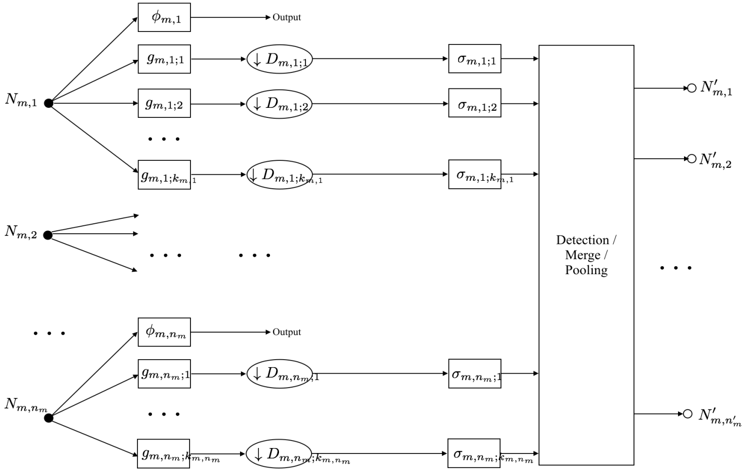
ConvNet: Sublayers
Long Short-Term Memory



Long Short-Term Memory (LSTM) networks [HS97, GSKSS15].
By BiObserver - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=43992484

ConvNet: Layer m

Components of the m^{th} layer



ConvNet: Layer m

Topology coding of the m^{th} layer

n_m denotes the number of input nodes in the m-th layer:
$\mathcal{I}_m = \{N_{m,1}, N_{m,2}, \cdots, N_{m,n_m}\}$.
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: $G_m = \cup_{n=1}^{n_m} G_{m,n}$.

ConvNet: Layer m

Topology coding of the m^{th} layer

n_m denotes the number of input nodes in the m-th layer:
$\mathcal{I}_m = \{N_{m,1}, N_{m,2}, \cdots, N_{m,n_m}\}$.
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: $G_m = \cup_{n=1}^{n_m} G_{m,n}$.
$\mathcal{O}_m = \{N'_{m,1}, N'_{m,2}, \cdots, N'_{m,n'_m}\}$ 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 $G_m = \cup_{n'=1}^{n'_m} G'_{m,n'}$, where $G'_{m,n'}$ is the set of filters merged into $N'_{m,n'}$.

ConvNet: Layer m

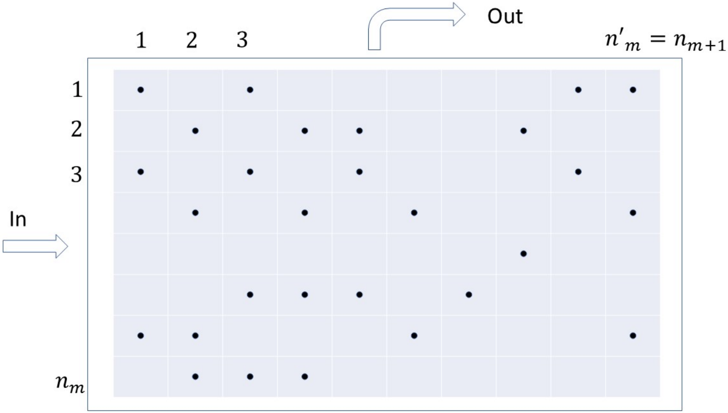
Topology coding of the m^{th} layer

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

$$l_{m,n;k} = \begin{cases} K & , \text{ if } g_{m,n;k} \in \mathcal{T}_1 \cup \mathcal{T}_3 \\ K^{\max\{0,2/p-1\}} & , \text{ if } g_{m,n;k} \in \mathcal{T}_2 \end{cases} \quad (3.1)$$

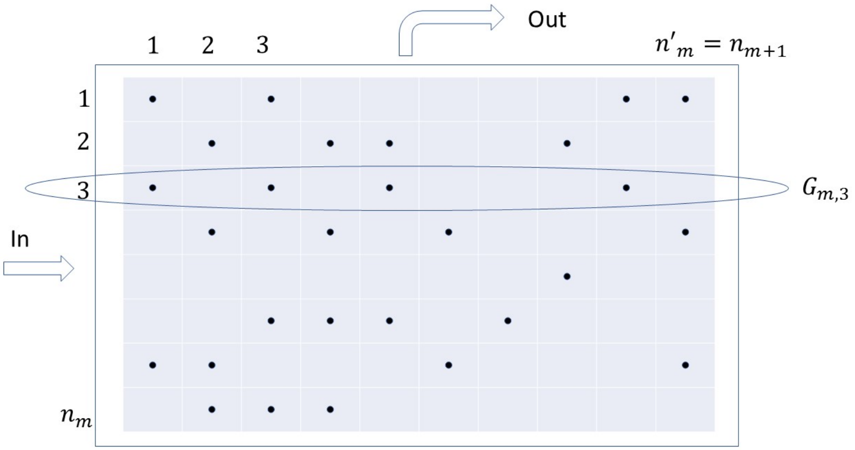
ConvNet: Layer m

Topology coding of the m^{th} layer



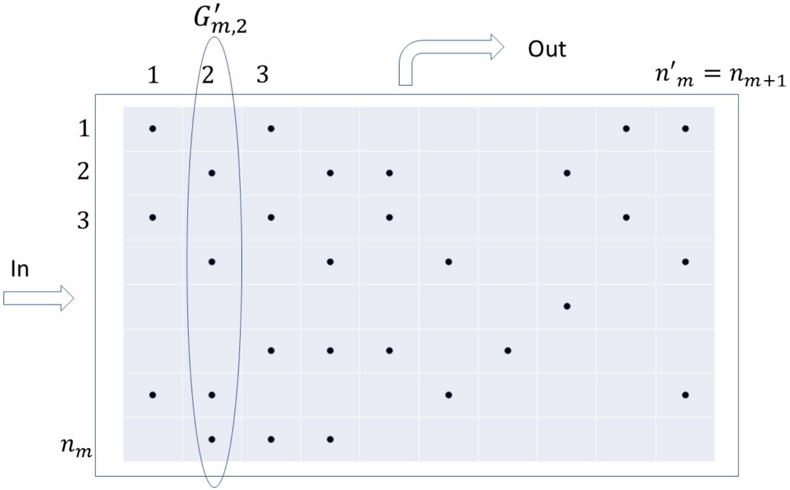
ConvNet: Layer m

Topology coding of the m^{th} layer



ConvNet: Layer m

Topology coding of the m^{th} layer



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:

$$B_{m,n}^{(1)} = \left\| \left|\hat{\phi}_{m,n}\right|^2 + \sum_{g_{m,n;k} \in G_{m,n}} l_{m,n;k} D_{m,n;k}^{-d} \left|\hat{g}_{m,n;k}\right|^2 \right\|_{\infty} \tag{4.2}$$

- 2nd type Bessel bound:

$$B_{m,n}^{(2)} = \left\| \sum_{g_{m,n;k} \in G_{m,n}} l_{m,n;k} D_{m,n;k}^{-d} \left|\hat{g}_{m,n;k}\right|^2 \right\|_{\infty} \tag{4.3}$$

- 3rd type (or generating) bound:

$$B_{m,n}^{(3)} = \left\|\hat{\phi}_{m,n}\right\|_{\infty}^2 . \tag{4.4}$$

Layer Analysis
Bessel Bounds

Next we define the layer m Bessel bounds:

$$\text{1}^{\text{st}} \text{ type Bessel bound} \quad B_m^{(1)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(1)} \tag{4.5}$$

$$\text{2}^{\text{nd}} \text{ type Bessel bound} \quad B_m^{(2)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(2)} \tag{4.6}$$

$$\text{3}^{\text{rd}} \text{ type (generating) Bessel bound} \quad B_m^{(3)} = \max_{1 \leq n \leq n_m} B_{m,n}^{(3)}. \tag{4.7}$$

Remark. These bounds characterize semi-discrete Bessel systems.

Lipschitz Analysis

First Result

Theorem

[BSZ17] Consider a Convolutional Neural Network with M layers as described before, where all scalar nonlinear functions are Lipschitz with $Lip(\varphi_{m,n,n'}) \leq 1$. *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*

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

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

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

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:

$$\begin{aligned}
\Gamma = \max_{y_1,\ldots,y_M,z_1,\ldots,z_M\geq 0} \quad & \sum_{m=1}^{M} z_m \\
s.t. \quad & y_0 = 1 \\
& y_m + z_m \leq B_m^{(1)} y_{m-1}, \quad 1 \leq m \leq M \\
& y_m \leq B_m^{(2)} y_{m-1}, \quad 1 \leq m \leq M \\
& z_m \leq B_m^{(3)} y_{m-1}, \quad 1 \leq m \leq M
\end{aligned} \tag{4.8}$$

Lipschitz Analysis

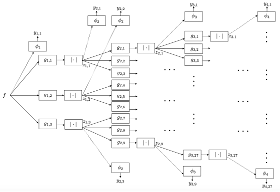
Second Result - cont'd

Theorem

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

$$\|\mathcal{F}(f) - \mathcal{F}(\tilde{f})\|_2^2 \leq \Gamma \|f - \tilde{f}\|_2^2,$$

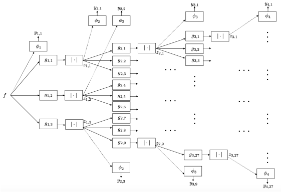
Example 1: Scattering Network



The Lipschitz constant:

- Backpropagation/Chain rule: Lipschitz bound 40 (hence $Lip \leq 6.3$).

Example 1: Scattering Network

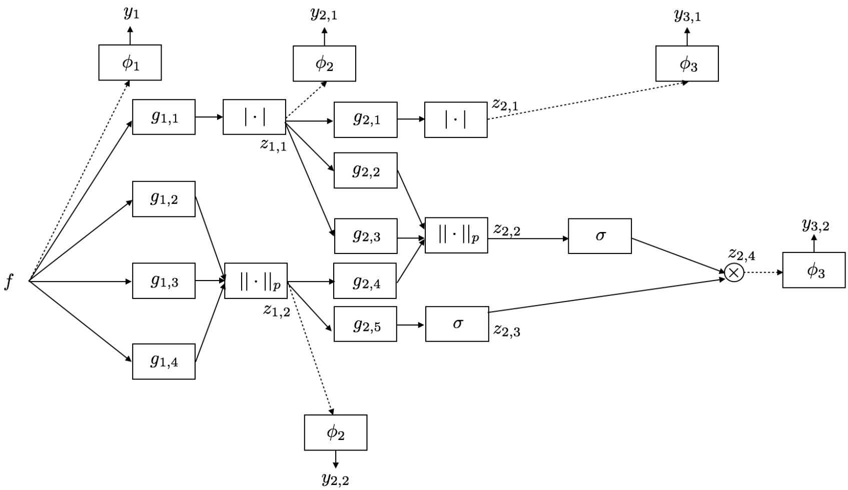


The Lipschitz constant:

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

Example 2: A General Convolutive Neural Network



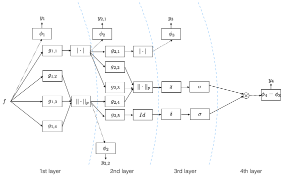
Example 2: A General Convolutive Neural Network

Set $p=2$ and:

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

$$\begin{aligned}
\hat{\phi}_1(\omega) &= F(\omega)\\
\hat{g}_{1,j}(\omega) &= F(\omega+2j-1/2)+F(\omega-2j+1/2)\ ,\ j=1,2,3,4\\
\hat{\phi}_2(\omega) &= \exp(\frac{4\omega^2+12\omega+9}{4\omega^2+12\omega+8})\chi_{(-2,-3/2)}(\omega)+\\
&\quad \chi_{(-3/2,3/2)}(\omega)+\exp(\frac{4\omega^2-12\omega+9}{4\omega^2-12\omega+8})\chi_{(3/2,2)}(\omega)\\
\hat{g}_{2,j}(\omega) &= F(\omega+2j)+F(\omega-2j)\ ,\ j=1,2,3\\
\hat{g}_{2,4}(\omega) &= F(\omega+2)+F(\omega-2)\\
\hat{g}_{2,5}(\omega) &= F(\omega+5)+F(\omega-5)\\
\hat{\phi}_3(\omega) &= \exp(\frac{4\omega^2+20\omega+25}{4\omega^2+20\omega+24})\chi_{(-3,-5/2)}(\omega)+\\
&\quad \chi_{(-5/2,5/2)}(\omega)+\exp(\frac{4\omega^2-20\omega+25}{4\omega^2-20\omega+25})\chi_{(5/2,3)}(\omega).
\end{aligned}$$

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: $Lip^2 \leq 5$.
- Using Theorem 1: $Lip^2 \leq 2.9430$.
- Using Theorem 2 (linear program): $Lip^2 \leq 2.2992$.

Example 3: Lipschitz constant as an optimization criterion

Nonlinear Discriminant Analysis

In Linear Discriminant Analysis (LDA), the objective is to maximize the "separation" between two classes, while controlling the variances within class.

A similar nonlinear *discriminant* can be defined:

$$S = \frac{\|\mathbb{E}[\mathcal{F}(f)|f \in C_1] - \mathbb{E}[\mathcal{F}(f)|f \in C_2]\|^2}{\|Cov(\mathcal{F}(f)|f \in C_1)\|_F + \|Cov(\mathcal{F}(f)|f \in C_2)\|_F}.$$

Example 3: Lipschitz constant as an optimization criterion
Nonlinear Discriminant Analysis

In Linear Discriminant Analysis (LDA), the objective is to maximize the "separation" between two classes, while controlling the variances within class.
A similar nonlinear *discriminant* can be defined:

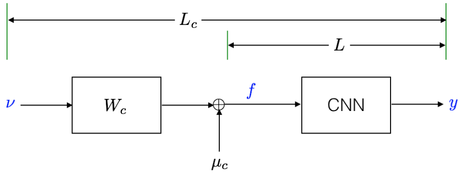
$$S = \frac{\|\mathbb{E}[\mathcal{F}(f)|f \in C_1] - \mathbb{E}[\mathcal{F}(f)|f \in C_2]\|^2}{\|Cov(\mathcal{F}(f)|f \in C_1)\|_F + \|Cov(\mathcal{F}(f)|f \in C_2)\|_F}.$$

Replace the statistics $\|Cov\|_F$ by Lipschitz bounds:
Lipschitz bound based separation:

$$\tilde{S} = \frac{\|\mathbb{E}[\mathcal{F}(f)|f \in C_1] - \mathbb{E}[\mathcal{F}(f)|f \in C_2]\|^2}{Lip_1^2 + Lip_2^2}.$$

Example 3: Lipschitz constant as an optimization criterion
Nonlinear Discriminant Analysis

The Lipschitz bounds Lip_1^2, Lip_2^2 are computed using Gaussian generative models for the two classes: $(\mu_c, W_c W_c^T)$, where W_c represents the whitening filter for class $c \in \{1,2\}$.



Example 3: Lipschitz constant as an optimization criterion

Numerical Results

Dataset: MNIST database; input images: 28×28 pixels. Two classes: "3" and "8"

Classifier: 3 layer and 4 layer random CNN, followed by a trained SVM.

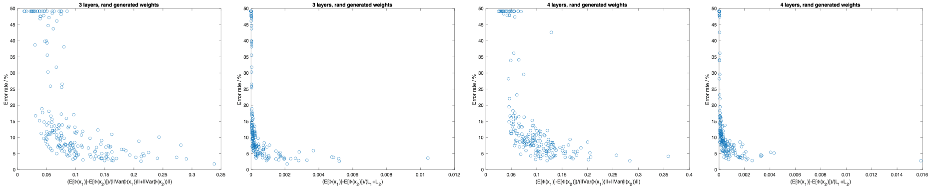


Figure: Results for uniformly distributed random weights

Conclusion: The error rate decreases as the Lipschitz bound separation increases. The discriminant spread is wider.

Example 3: Lipschitz constant as an optimization criterion

Numerical Results

Dataset: MNIST database; input images: 28×28 pixels. Two classes: "3" and "8"
Classifier: 3 layer and 4 layer random CNN, followed by a trained SVM.

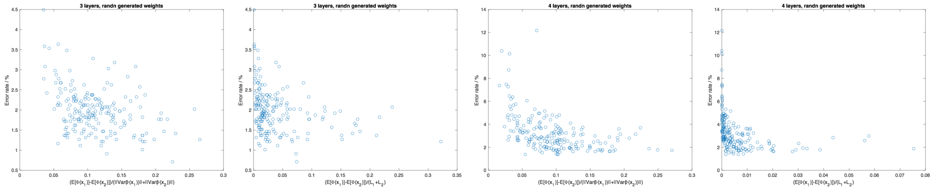


Figure: Results for normaly distributed random weights

References

- [ACB17] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein GAN*, online arXive:1701.07875, 2017.
- [BSZ17] Radu Balan, Manish Singh, Dongmian Zou, *Lipschitz Properties for Deep Convolutional Networks*, available online arXiv 1701.01527 [cs.LG], 18 Jan 2017.
- [BZ15] Radu Balan, Dongmian Zou, *On Lipschitz Analysis and Lipschitz Synthesis for the Phase Retrieval Problem*, available online arXiv 1506.02092v1 [mathFA], 6 June 2015; Lin. Alg. and Appl. 496 (2016), 152-181.
- [BGC16] Yoshua Bengio, Ian Goodfellow and Aaron Courville, *Deep learning*, MIT Press, 2016.
- [BM13] Joan Bruna and Stephane Mallat, *Invariant scattering convolution networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 8, 1872–1886.

[BCLPS15] Joan Bruna, Soumith Chintala, Yann LeCun, Serkan Piantino, Arthur Szlam, and Mark Tygert, *A theoretical argument for complex-valued convolutional networks*, CoRR **abs/1503.03438** (2015).

[DDSLLF09] Jia Deng,Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei *Imagenet: A large-scale hierarchical image database*, IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, 248–255. 2009.

[GPMXWOCB14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, W. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative Adverserial Nets*, Advances in Neural Information Processing Systems, 2672–2680, 2014.

[GSKSS15] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, Jurgen Schmidhuber, *LSTM: A Search Space Odyssey*, arXiv:1503.04069v1 [cs.NE], 13 Mar. 2015.

- [GAADC17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, *Improved Training of Wasserstein GANs*, online arXiv:1704.00028 [cs.LG], 29 May 2017.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural Comput. **9** (1997), no. 8, 1735–1780.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems 25, 1106–1114, 2012.
- [LBH15] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.
- [LSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir, *On the computational efficiency of training neural networks*, Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, eds.), Curran Associates, Inc., 2014, pp. 855–863.

[Mallat12] Stephane Mallat, *Group invariant scattering*, Communications on Pure and Applied Mathematics **65** (2012), no. 10, 1331–1398.

[SVS15] Tara N. Sainath, Oriol Vinyals, Andrew W. Senior, and Hasim Sak, *Convolutional, long short-term memory, fully connected deep neural networks*, 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015, 2015, pp. 4580–4584.

[SLJSRAEVR15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Angelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, CVPR 2015, 2015.

[SZSBEGF13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, CoRR **abs/1312.6199** (2013).

[WB15a] Thomas Wiatowski and Helmut Bölcskei, *Deep convolutional neural networks based on semi-discrete frames*, Proc. of IEEE International Symposium on Information Theory (ISIT), June 2015, pp. 1212–1216.

[WB15b]Thomas Wiatowski and Helmut Bölcskei, *A mathematical theory of deep convolutional neural networks for feature extraction*, IEEE Transactions on Information Theory (2015).