Uncertainty Quantification in Neural Networks with Applications to MRI Processing

Radu Balan

Department of Mathematics, AMSC Program and the Norbert Wiener Center for Harmonic Analysis and Applications
University of Maryland, College Park, MD

September 12, 2025

Table of Contents

High-Level Overview

In this talk we discuss a few harmonic analysis techniques and problems applied to machine learning.
1. Neural Networks: A Quick Introduction & Motivating Examples
2. CRLB based Uncertainty Propagation: we use Cramer-Rao Lower Bound to quantify uncertainty in MRI estimation using deep neural networks
3. NN and MRI: Motion Compensation and Image Reconstruction
4. Lipschitz analysis: we provide rationals for studying Lipschitz properties of NNs, and then we perform a Lipschitz analysis of these networks. We focus on two aspects of this analysis: stochastic modelng of local vs. global analysis, and a scattering network inspired Lipschitz analysis of convolutive networks.

Outline

Neural Networks: Architectures and Properties

Neural networks were introduced a long time ago ...

1. 1925: Ising model – first Recurrent Neural Network (RNN)
2. 1940s: Hebbian learning for neuroplasticity – weights are learned dynamically
3. 1958: Rosenblatt introduced the perceptron, a 1-layer NN
4. 1965: Ivakhnenko and Lapa: Multi-Layer Perceptron (MLP)
5. 1967: Amari studied stochastic gradient descent (SGD) for training/learning
6. 1980: Fukushima introduced the convolutional neural network (CNN)
7. 1991-2: Schmidhuber introduced adversarial networks (precursors of GANs - 2014 by Goodfellow), generative models, and the transformers with linearized self-attention

Network Architectures

Deep Neural Networks

- Input layer: $x = (x_1, x_2, \cdots, x_n)^T$
- Output layer: $y = (y_1, y_2, \cdots, y_m)^T$
- Number of Layers: L

$$y = A_{L+1} \cdot \sigma(A_L \cdot \sigma(A_{L-1} \cdots \sigma(A_1 \cdot x + b_1) \cdots) + b_{L-1}) + b_L) + b_{L+1}$$

The scalar *activation function* $\sigma' : \mathbb{R} \to \mathbb{R}$ acts entrywise.

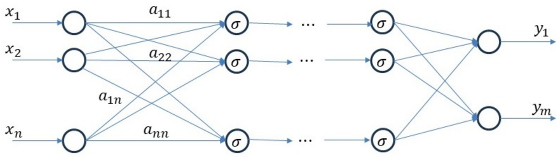


Figure: A general Feed-Forward Network, or a Deep Neural Network (DNN)

Network Architectures

Convolutive Neural Networks (CNN)

A Convolutive Neural Network is a Deep Neural Network with two additional features:

1. Linear operators A_k are convolutive operators, and implemented as convolutions
2. Activation functions are followed by downsampling and (optional) *pooling layers*: either max-pooling or sum-pooling.

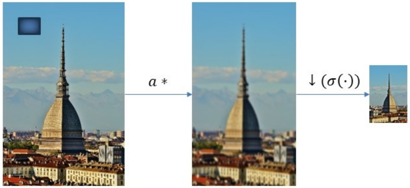


Figure: One layer of a Convolutive Neural Network (picture curtesy of robygarba@pixabay)

Convolutive Neural Networks (CNN)

Alex Net

The AlexNet is 8 layer network, 5 convolutive layers plus 3 dense layers. Introduced by (Alex) Krizhevsky, Sutskever and Hinton in 2012 .

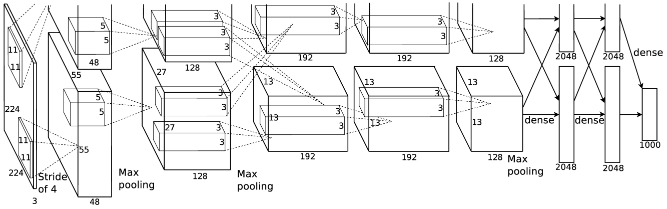


Figure: From Krizhevsky et all 2012 : AlexNet: 5 convolutive layers + 3 dense layers. Input size: 224x224x3 pixels. Output size: 1000.

Universal Approximation Properties of Neural Netwoks

Conventional wisdom says that neural networks can approximate arbitrary well any "reasonable" function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$.
Earliest results showed that even one hidden layer networks approximate target functions equally well. One hidden layer networks are called *perceptrons*. The input-output characterization of a single-layer perceptron $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$, is given by:

$$\Phi(x) = a^T \sigma(Wx + b) + b_0 \quad , \quad x \mapsto \Phi(x) = \sum_{k=1}^{p} a_k \sigma(\sum_{j=1}^{n} W_{k,j} x_j + b_k) + b_0.$$

Universal Approximation Properties of Neural Netwoks

Conventional wisdom says that neural networks can approximate arbitrary well any “reasonable” function $f : \mathbb{R}^n \to \mathbb{R}^m$.
Earliest results showed that even one hidden layer networks approximate target functions equally well. One hidden layer networks are called *perceptrons*. The input-output characterization of a single-layer perceptron $\Phi : \mathbb{R}^n \to \mathbb{R}$, is given by:

$$\Phi(x) = a^T \sigma(Wx + b) + b_0 \quad , \quad x \mapsto \Phi(x) = \sum_{k=1}^{p} a_k \sigma(\sum_{j=1}^{n} W_{k,j} x_j + b_k) + b_0 .$$

Theorem (Cybenko 1989)

Assume $\sigma : \mathbb{R} \to \mathbb{R}$ is a bounded continuous function that satisfies $\lim_{t\to\infty} \sigma(t) = 1$ and $\lim_{t\to-\infty} \sigma(t) = 0$. Then for any continuous $f : [0,1]^n \to \mathbb{R}$ there are (W, a, b, b_0) so that $|f(x) - a^T \sigma(Wx + b) - b_0| < \varepsilon$ for every $x \in [0,1]^n$.

Further Results

Remark

The compact set $[0,1]^n$ can be replaced by any compact set K: scale and translate to bring it inside $[0,1]^n$; then use Tietze extension theorem.

Remark

Recent results extend the density result to various other spaces, such as $C^k(K)$, $W^{k,p}(K)$, etc; they also extend to the case of certain unbounded σ, e.g., the ReLU function, $ReLU(x) = x1_{(0,\infty)}$.

Remark

Cybenko's proof (or several subsequent results) is not constructive. Recent results by other researchers (e.g., Petersen and Voigtlaender; Bolcskei, Grohs, Kutyniok and Petersen) provide explicit architectures (number of layers, number of hidden nodes) and even memory cost (i.e., quantized weights) that achieves a preset approximation accuracy.

Outline

Example 1: The AlexNet

The ImageNet Dataset

Dataset: ImageNet dataset. Currently: 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.

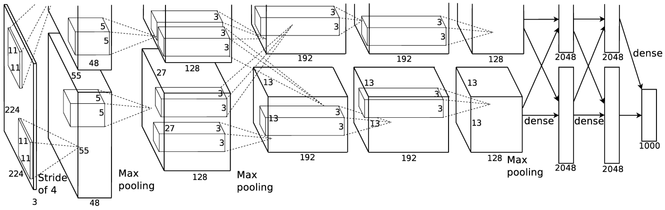


Figure: From Krizhevsky et all 2012: AlexNet: 5 convolutive layers + 3 dense layers. Input size: 224x224x3 pixels. Output size: 1000.

Example 1: The AlexNet

Adversarial Perturbations

The authors of [Szegedy'13] (Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus, 'Intriguing properties ...') found small variations of the input, almost imperceptible, that produced completely different classification decisions:

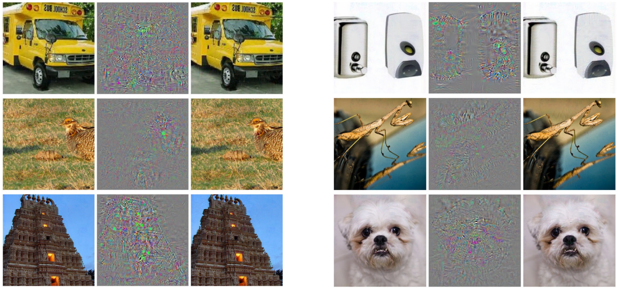


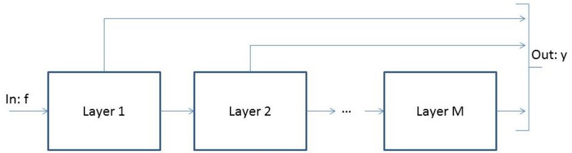
Figure: From Szegedy et all 2013: AlexNet: 6 different classes: original image, difference, and adversarial example – all classified as 'ostrich'

Other Examples

1. Generative Adversarial Networks: The Wasserstein distance based GANs
2. Uncertainty Propagation through DNN: This example is based on a project with Prof. Thomas Ernst, UMB, School of Medicine, Baltimore.
3. The Scattering Networks: Naive vs. Exact analysis

Lipschitz Analysis

Given a deep network:



Estimate the Lipschitz constant, or bound:

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

This yields:

$$\|y-\tilde{y}\|_2 \leq Lip\|f-\tilde{f}\|_2,$$

for any two inputs $f, \tilde{f}'$, arbitrary close or far apart.

The goal of this study is to estimate the Lipschitz constants and bounds, and understand how perturbations propagate through deep networks.

Outline

Uncertainty Quantification and Propagation through DNN

Collaborators:
UMD: Danial Ludwig, Michael Rawson
UMB:Thomas Ernst, Bo Li, Xiaoke Wang, Ze Wang
Joint Work:
ISMRM 2022: Estimating Noise Propagation of Neural Network based Image Reconstruction using Automatic Differentiation

MRI Model

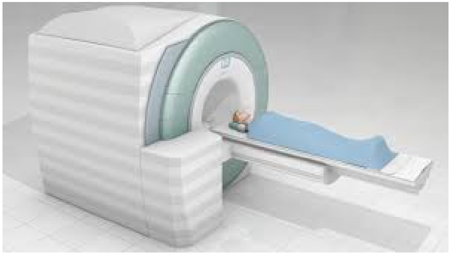


Figure: Credits: hopkinsmedicine.org

The measurement model. For coil $k \in [N_c]$,

$$x_k = \mathcal{F}(S_k z) + \nu_k$$

where $\mathcal{F}$ is the Fourier acquisition matrix, S_k is the diagonal matrix with the coil k sensitivity map, ν_k is measurement noise, and z is the brain signal.

Knowns: $\mathcal{F}$, $x_1, ..., x_{N_c}$. Unknowns: $S_1, ..., S_{N_c}, \nu_1, ..., \nu_{N_c}, z$. Target: z.

Lots of research, lots of Nobel prizes, lots of companies (Siemens, GE, Philips), lots of techniques (compressive sampling, GRAPPA, SENSE, ...) to solve the inverse problem: $z = G(measurements)$.
More recent: Use of Deep Neural Networks.

The MRI Inverse Problem

At an abstract level, the forward model, $z \mapsto x$ and the reconstruction (inverse) model, $x \mapsto \hat{z}$ are:

$$x = F(z) + \nu \quad , \quad \hat{z} = G(x).$$

To fix notations: the target (brain) signal $z \in \mathbb{R}^d$, the measured (acquired) signal $x \in \mathbb{R}^n$.
The DNN approach proposes to implement G using certain Neural Network architectures. Out of many architectures out there, we focused on a specific network, namely the *end-to-end variational neural network (E2E-VarNet)* introduced by Sriram,et al, at MICCAI 2020.

Our problem: Given a trained network that implements a reconstruction algorithm G, quantify the level of uncertainty per reconstructed pixel.

Assumption: We assume the network has been trained well enough so that $G(F(z)) = z$, i.e., perfect reconstruction in the absence of noise.

CRLB and FIM

An often used approach of quantifying uncertainty is through the Cramer-Rao Lower Bound (CRLB). The CRLB has been used many times for experimental design in Medical Imaging and elsewhere.

Fisher Information Matrix (FIM) $I(z)$ and $CRLB$:

$$I(z) = \mathbb{E}\left[(\nabla_z log(p(x;z)))(\nabla_z log(p(x;z)))^T\right] \quad , \quad CRLB = (I(z))^{-1}$$

Interpretation: Covariance of any *unbiased* estimator of z is lower bounded $CRLB$. Assume further, the noise is AWGN with variance σ^2. A simple computation yields:

$$CRLB = \sigma^2 \left(J_F^T J_F\right)^{-1} \quad , \quad J_F = \left[\frac{\partial F_k}{\partial z_j}\right]_{(j,k)\in[n]\times[d]} \in \mathbb{R}^{n\times d}$$

where J_F denotes the Jacobian matrix of the forward model F.

Goal: Determine $CRLB$ and use it to measure the confidence in the reconstructed image $\hat{z}$.

Challenge: The exact form of F is unknown!

The CRLB and the Jacobian of the NN

Our main theoretical result is to connect $CRLB = (I(z))^{-1}$ to the Jacobian[1] J_G of G.

Lemma

Assume $A \in \mathbb{R}^{n \times d}$ is full rank with $n \geq d$.

1. *For any $B \in \mathbb{R}^{d \times n}$ such that $BA = I_d$ (i.e., a left inverse), $BB^T \geq (A^T A)^{-1}$.*
2. *If $B_0 = (A^T A)^{-1} A^T$ is the pseudo-inverse of A then, $B_0 B_0^T = (A^T A)^{-1}$.*

[1]The importance of Jacobians has been shown by (Antun et al, 2020), “On instabilities of deep learning in image reconstruction ...”

The CRLB and the Jacobian of the NN

Our main theoretical result is to connect $CRLB = (I(z))^{-1}$ to the Jacobian[1] J_G of G.

Lemma

Assume $A \in \mathbb{R}^{n \times d}$ is full rank with $n \geq d$.

1. *For any $B \in \mathbb{R}^{d \times n}$ such that $BA = I_d$ (i.e., a left inverse), $BB^T \geq (A^T A)^{-1}$.*
2. *If $B_0 = (A^T A)^{-1} A^T$ is the pseudo-inverse of A then, $B_0 B_0^T = (A^T A)^{-1}$.*

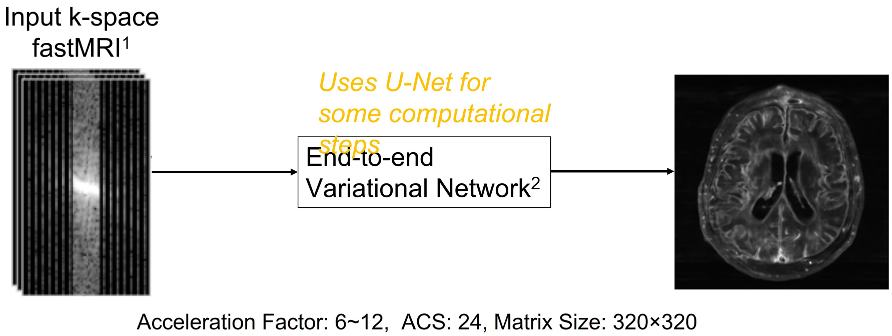
Consequence:

$$\boxed{CRLB = \sigma^2 J_{G_0} J_{G_0}^T \quad , \quad G_0 = argmin_{G:G(F(z))=z} trace(J_G J_G^T)}$$

Use $trace(J_G J_G^T)$ as an additional term in the NN training loss function.

[1] The importance of Jacobians has been shown by (Antun et al, 2020), "On instabilities of deep learning in image reconstruction ..."

Architecture


Input k-space
fastMRI[1]
Uses U-Net for some computational steps
End-to-end Variational Network[2]
Acceleration Factor: 6~12, ACS: 24, Matrix Size: 320×320

1. *Zbontar J, Knoll F, Sriram A, et al. fastMRI: An Open Dataset and Benchmarks for Accelerated MRI. Published online November 21, 2018. Accessed November 10, 2021. https://arxiv.org/abs/1811.08839v2*
2. *Sriram, Anuroop, et al. "End-to-end variational networks for accelerated MRI reconstruction." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2020. (Facebook AI Research and NYU)*

ACS: Auto-Calibration Signal used by GRAPPA, 24 lines out of 320.

Reconstruction

Measurement model: for each coil k,

$$x_k = \mathcal{F}(S_k z) + \nu_k$$

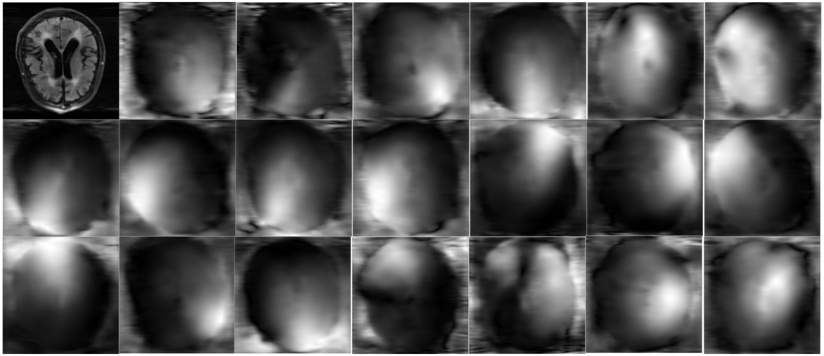
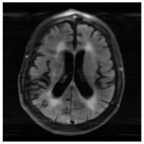


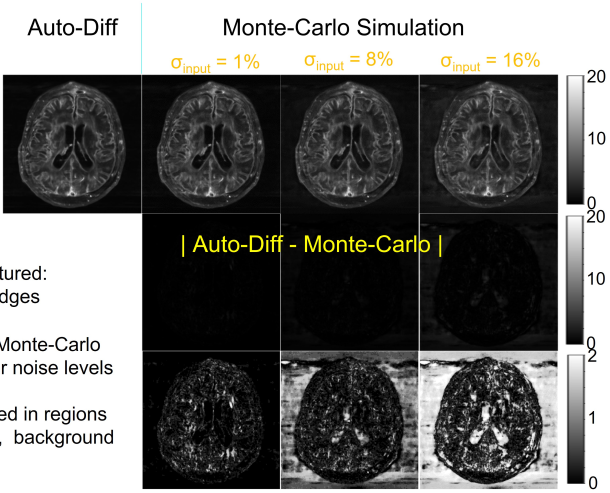
Figure: Reconstructed image and sensitivity maps (S_k) by E2E-VARNET. The reduction factor of the k-space under-sampling is 11.

Results (1)

Results



$$\frac{\text{RMSE}}{\sigma_{\text{input}}}$$

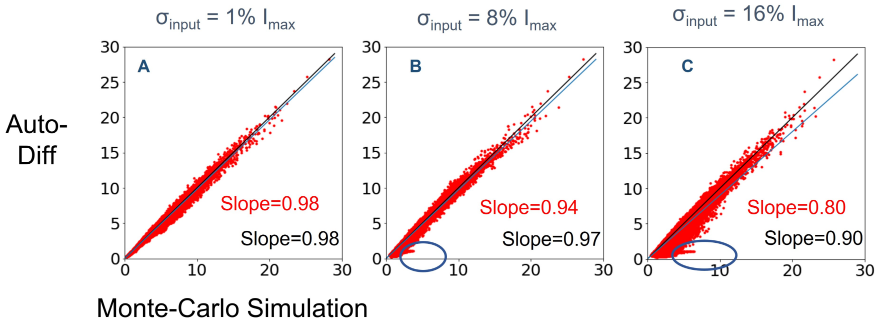


- Noise amplification is structured: generally higher at sharp edges

- Auto-Diff agrees well with Monte-Carlo
- Agreement poorer at higher noise levels → non-linearity of NN?
- Deviations more pronounced in regions of low signal intensity (e.g., background and in ventricles)

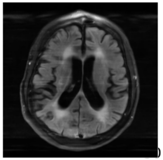
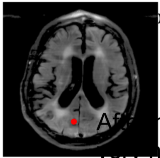
Results (2)

Pixel-by-pixel $\frac{\text{RMSE}}{\sigma_{\text{input}}}$: Auto-Diff versus MC

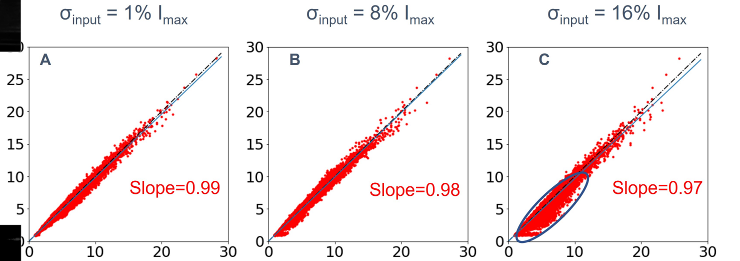


- Auto-Diff (Linear Model) agrees well with Monte-Carlo Simulation
- Agreement is less strong with higher noise level
- The outlying voxels are mostly in background and ventricles

Results (3)

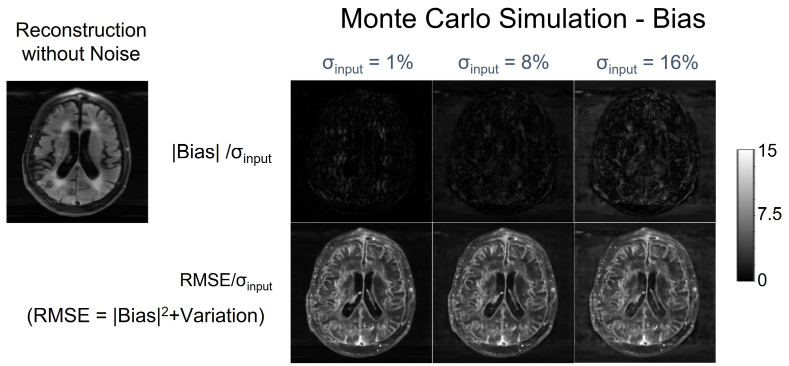
$\frac{\text{RMSE}}{\sigma_{\text{input}}}$ with masking: Auto-Diff versus MC


$\sigma_{input} = 1\%\ I_{max}$
$\sigma_{input} = 8\%\ I_{max}$
$\sigma_{input} = 16\%\ I_{max}$
A
Slope=0.99
B
Slope=0.98
C
Slope=0.97
0
5
10
15
20
25
30

Monte-Carlo Simulation

- After masking, very good agreement between MC and the Auto-Diff even at very high noise levels
- However, Auto-Diff still tends to underestimate noise

Results (4)



- As the standard deviation of noise increases, so does the bias.
- This may also have contributed to the divergence between the auto-diff and Monte-Carlo simulation
- However, even at the highest noise level, the bias was lower than the RMSE

Outline

Based on this article:
Lei Zhang, X. Wang, M. Rawson, R.B., E. Herskovits, E.R. Melhem, L. Chang, Z. Wang, T. Ernst, *Motion Correction for Brain MRI Using Deep Learning and a Novel Hybrid Loss Function*, Algorithms 17, 215, (2024), https://doi.org/10.3390/a17050215

Motion Correction Problem

MRI is relatively slow and can take several minutes for a typical 3D volume scan. Motion by human subjects is unavoidable and can be caused by respiration, cardiac motion, and unintended patient movements.

Motion Correction is needed!

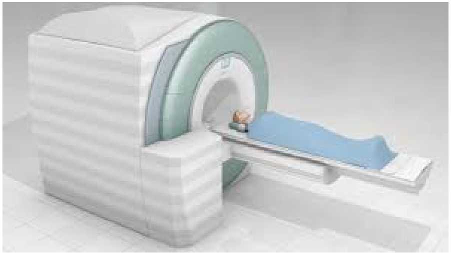


Figure: Credits: hopkinsmedicine.org

The measurement model. For coil k,

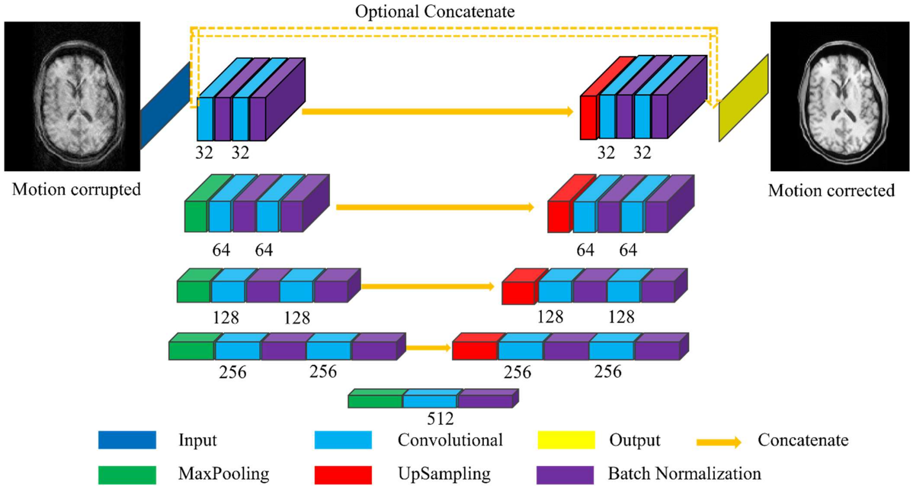
$$x_k = \mathcal{F}(S_k T z) + \nu_k$$

where $\mathcal{F}$ is the FULL Fourier acquisition matrix, S_k is the diagonal matrix with the coil k sensitivity map, T denotes the motion operator (translation+rotation), ν_k is measurement noise, and z is the brain signal.

Knowns: $\mathcal{F}, x_1, ..., x_{N_c}$. Unknowns: $S_1, ..., S_{N_c}, \nu_1, ..., \nu_{N_c}, T, z$. Target: z.

MC-Net Architecture

Motion Correction Net (MC-Net) takes a motion-corrupted image as input and outputs a motion-corrected image. It is derived from UNet (2019) architecture.



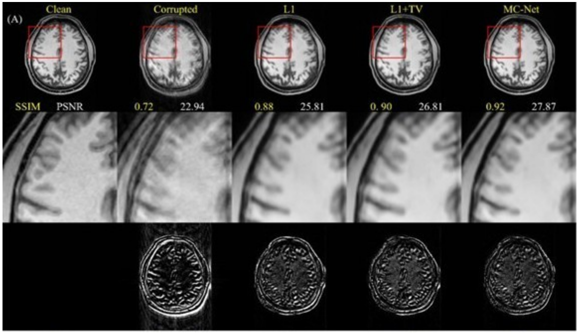
Training Loss

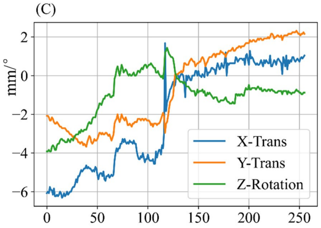
The MC-Net was trained with a two-stage training strategy using a hybrid loss function, L, that combines L1-loss and TV-loss:

$$L_1 = \sum_{i,j} |I(i,j) - I_0(i,j)|, \; TV = \sum_{i,j} ((I(i+1,j) - I(i,j))^2 + (I(i,j+1) - I(i,j))^2)^{1}$$

During the first training stage $L = L_1$ to suppress overall motion-induced artifacts. The pre-trained stage 1 model was then fine-tuned with $L = L_1 + TV$. Yhe hybrid loss function encourages the model to produce output images with low total variation that can have sharp edges and reduced motion artifacts.

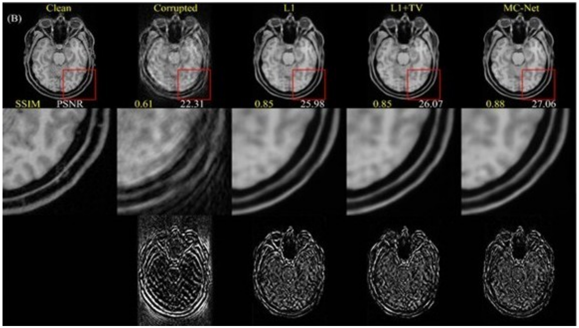
Results (1)

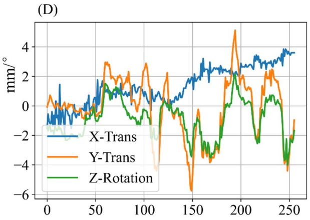




Example of motion artifact removal. first row: the clean reference image, corrupted image, and motion correction results obtained using the L1, L1 + TV, and MC-Net algorithms; second row zooms in on the red rectangle with Structural Similarity Index Measure (SSIM) and Peak-Signal-to-Noise-Ratio (PSNR); third row shows the error maps multiplied by a factor of five; lower plot refers to y-positions in k-space.

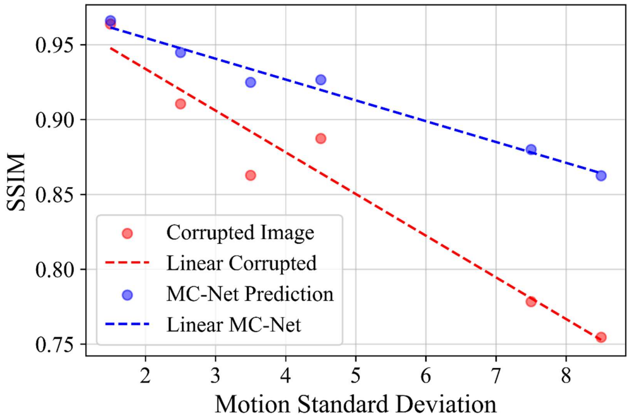
Results (2)





Example of motion artifact removal. first row: the clean reference image, corrupted image, and motion correction results obtained using the L1, L1 + TV, and MC-Net algorithms; second row zooms in on the red rectangle with Structural Similarity Index Measure (SSIM) and Peak-Signal-to-Noise-Ratio (PSNR); third row shows the error maps multiplied by a factor of five; lower plot refers to y-positions in k-space.

SSIM Performance



SSIM values of images corrected with MC-Net (blue dots) relative to those of corrupted images (red dots) are plotted against the standard deviation across 256 time points (in mm or deg). The red line (Y=0.99 - 0.028X) and blue line (Y=0.98-0.014X) show linear regressions.

Outline

More details included in:

- D. Zou, R. Balan, M. Singh, *On Lipschitz Bounds of General Convolutional Neural Networks*, IEEE Trans.on Info.Theory, vol. 66(3), 1738–1759 (2020) doi: 10.1109/TIT.2019.2961812.
- R. Balan, M. Singh, D. Zou, “Lipschitz Properties for Deep Convolutional Networks”, arXiv:1701.05217 [cs.LG], Contemporary Mathematics 706, 129-151 (2018) http://dx.doi.org/10.1090/conm/706/14205.

Example 1: The AlexNet

The ImageNet Dataset

Dataset: ImageNet dataset. Currently: 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.

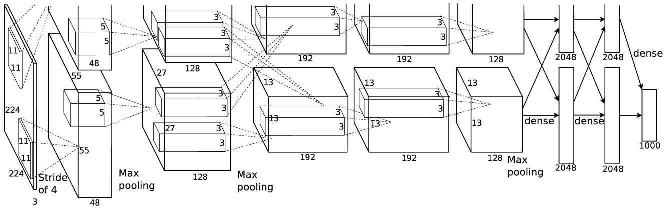
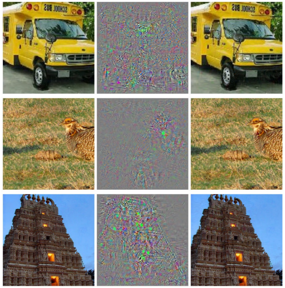


Figure: From Krizhevsky et all 2012: AlexNet: 5 convolutive layers + 3 dense layers. Input size: 224x224x3 pixels. Output size: 1000.

Example 1: The AlexNet

Adversarial Perturbations

The authors of [Szegedy'13] (Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus, 'Intriguing properties ...') found small variations of the input, almost imperceptible, that produced completely different classification decisions:



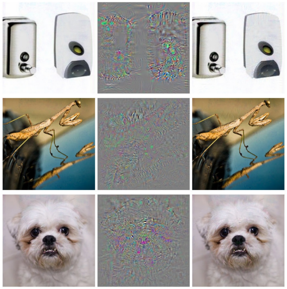


Figure: From Szegedy et all 2013: AlexNet: 6 different classes: original image, difference, and adversarial example – all classified as 'ostrich'

Example 1: The AlexNet

Lipschitz Analysis

Szegedy et all 2013 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

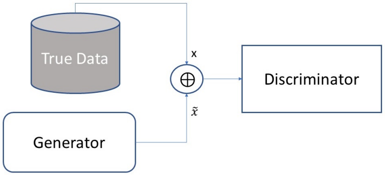
Overall Lipschitz constant:

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

Example 2: Generative Adversarial Networks

The GAN Problem

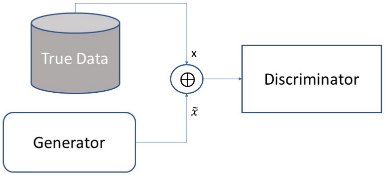
Two systems are involved: a *generator* network producing synthetic data; a *discriminator* network that has to decide if its input is synthetic data or real-world (true) data:



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 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 2017) 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 2017) 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 2017 proposed 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: Uncertainty Propagation through DNN

This example is based on a recent project with Prof. Thomas Ernst, UMB, School of Medicine, Baltimore.
The standard way of quantifying uncertainty is through the Cramer-Rao Lower Bound (CRLB). Fisher Information Matrix $I(z)$ and $CRLB$:

$$I(z) = \mathbb{E}\left[(\nabla_z log(p(x;z)))\,(\nabla_z log(p(x;z)))^T\right] \quad , \quad CRLB = (I(z))^{-1}$$

Interpretation: Covariance of any *unbiased* estimator of z is lower bounded $CRLB$. For AWGN with variance σ^2,

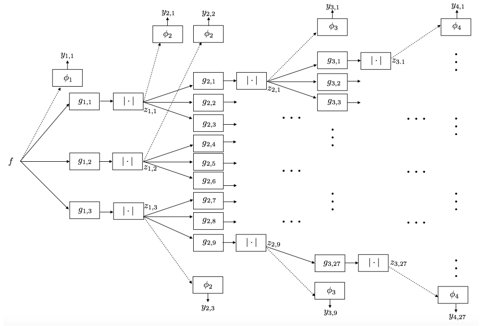
$$CRLB = \sigma^2 \left(J_F^T J_F\right)^{-1} \quad , \quad J_F = \left[\frac{\partial F_k}{\partial z_j}\right]_{(j,k)\in[n]\times[d]} \in \mathbb{R}^{n\times d}$$

where J_F denotes the Jacobian matrix of the forward model.
Goal: Determine $CRLB$ and use it to measure the confidence in the reconstructed image $\hat{z}$.
Challenge: The exact form of F is unknown! But we assume we know a left-inverse (the DNN) G_0. It turns out a good proxy is $CRLB = \sigma^2 J_{G_0} J_{G_0}^T$.

Example 4: The Scattering Network

Topology

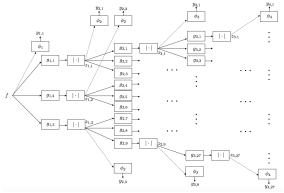
Example of Scattering Network; definition and properties: [Mallat'12]; this example from [B.,Singh,Zou'17]:



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

Example 4: Scattering Network

Lipschitz Analysis

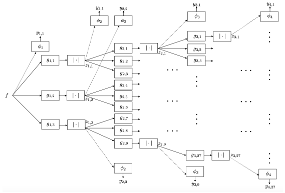


Remarks:

- Outputs from each layer

Example 4: Scattering Network

Lipschitz Analysis

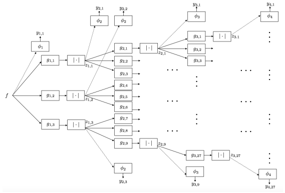


Remarks:

- Outputs from each layer
- Tree-like topology

Example 4: Scattering Network

Lipschitz Analysis

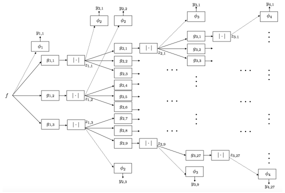


Remarks:

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

Example 4: Scattering Network

Lipschitz Analysis



Remarks:

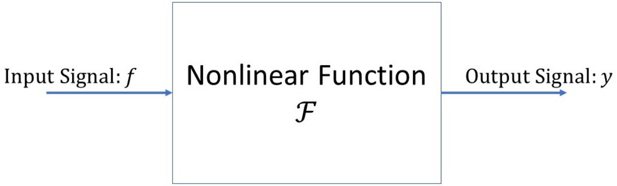
- Outputs from each layer
- Tree-like topology
- Backpropagation/Chain rule: Lipschitz bound 40.
- Mallat's result predicts $Lip = 1$.

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 $f, \tilde{f} \in X$,

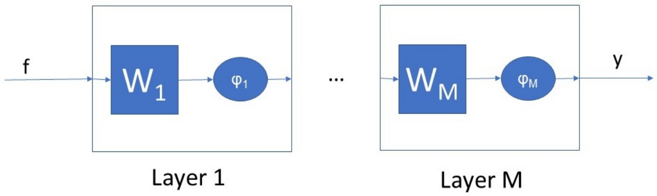
$$C_1\ d_X(f, \tilde{f}) \leq d_Y(\mathcal{F}(f), \mathcal{F}(\tilde{f})) \leq C_2\ d_X(f, \tilde{f})$$

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:



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

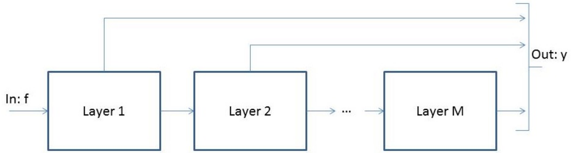
$$\mathcal{F}_m(f; W_m, \varphi_m) = \varphi_m(W_m f)$$

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:

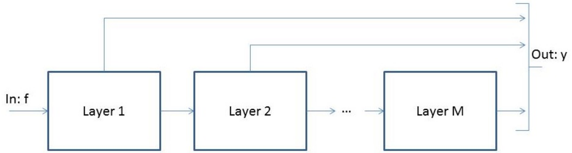


Estimate the Lipschitz constant, or bound:

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

Problem Formulation
Problem 1

Given a deep network:



Estimate the Lipschitz constant, or bound:

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

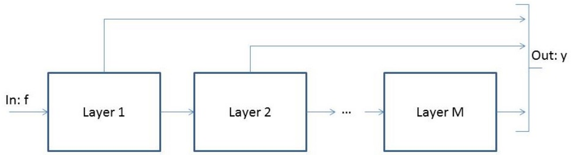
Methods (Approaches):

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:



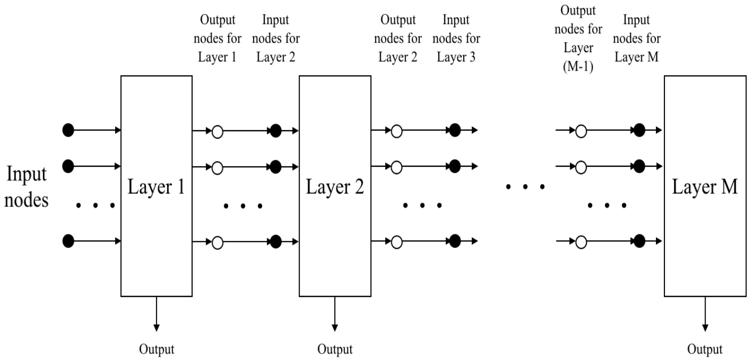
Estimate the stability of the output to specific variations of the input:

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

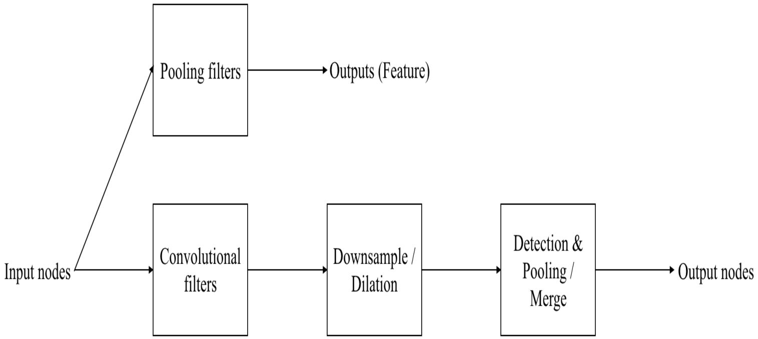
Topology

A deep convolution network is composed of multiple layers:



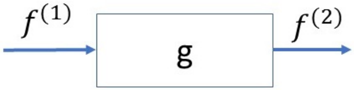
ConvNet
One Layer

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



ConvNet: Sublayers

Linear Filters: Convolution and Pooling-to-Output Sublayer



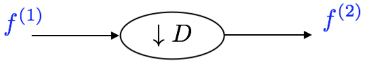
$$f^{(2)} = g * f^{(1)} \quad , \quad g * f^{(1)}(x) = \int g(x-\xi) f^{(1)}(\xi) d\xi$$

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

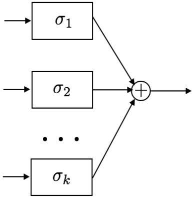
$$\|f^{(2)}\|_2^2 = \int_{\mathbb{R}^d} |f^{(2)}(x)|^2 dx = \frac{1}{|det(D)|} \int_{\mathbb{R}^d} |f^{(1)}(x)|^2 dx = \frac{1}{D_0^d} \|f^{(1)}\|_2^2$$

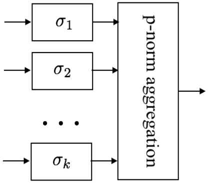
ConvNet: Sublayers

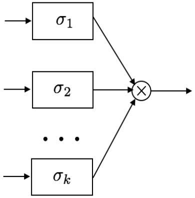
Detection and Pooling Sublayer

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: $z = \prod_{j=1}^{k} \sigma_j(y_j)$





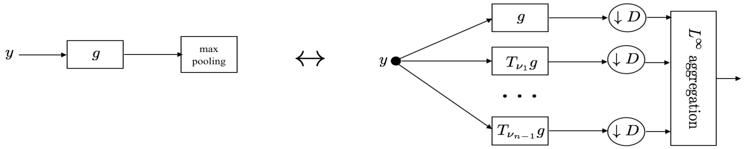


Assumptions: (1) σ_j are scalar Lipschitz functions with $Lip(\sigma_j) \leq 1$; (2) If σ_j is connected to a multiplication block then $\|\sigma_j\|_\infty \leq 1$.

ConvNet: Sublayers

MaxPooling and AveragePooling

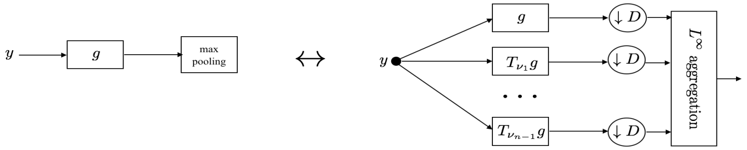
MaxPooling can be implemented as follows:



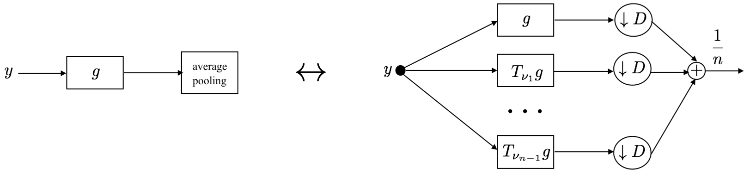
ConvNet: Sublayers

MaxPooling and AveragePooling

MaxPooling can be implemented as follows:

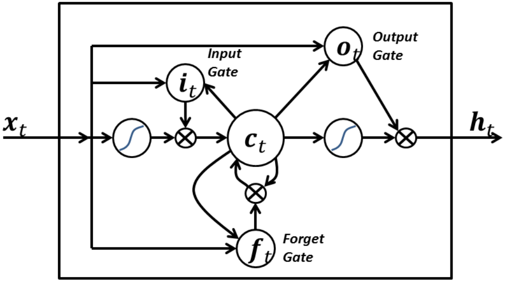


AveragePooling can be implemented as follows:



ConvNet: Sublayers

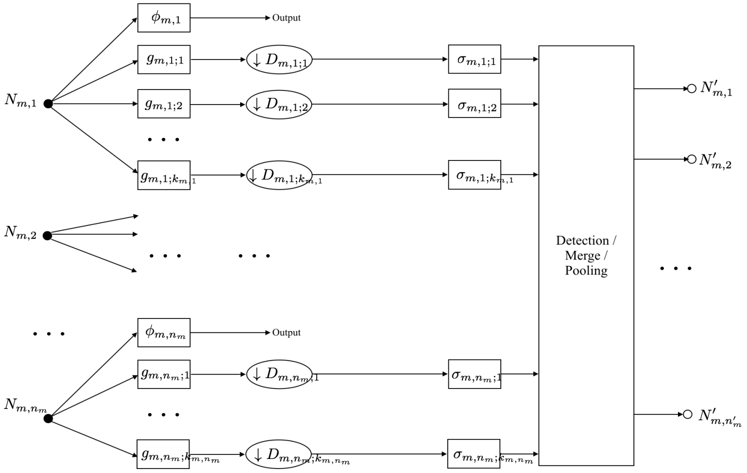
Long Short-Term Memory



Long Short-Term Memory (LSTM) networks [Hochreiter,Schmidhuber.'97],[Greff et.al.'15].
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

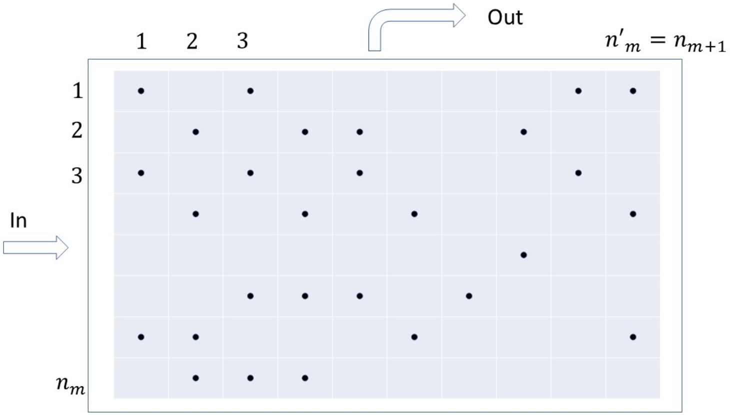
Topology coding of the m^{th} layer

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

$$l_{m,n;k} = \begin{cases} K & \text{, if } g_{m,n;k} \in \tau_1 \cup \tau_3 \\ K^{\max\{0,2/p-1\}} & \text{, if } g_{m,n;k} \in \tau_2 \end{cases} \tag{5.1}$$

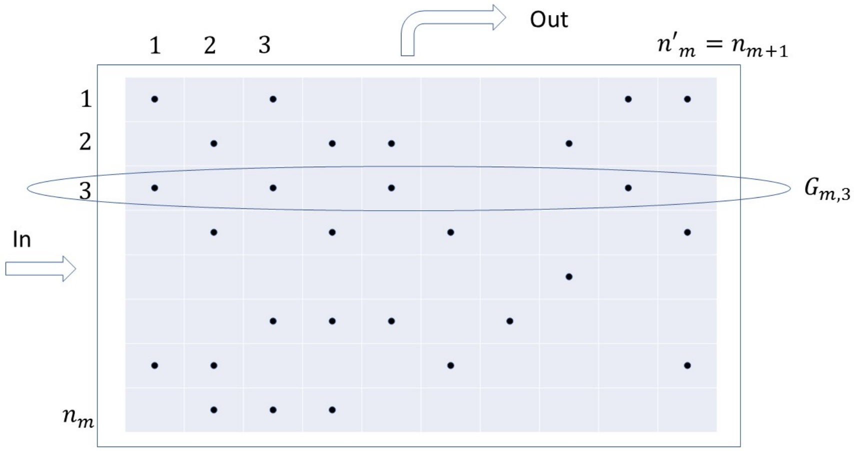
ConvNet: Layer m

Topology coding of the m^{th} layer



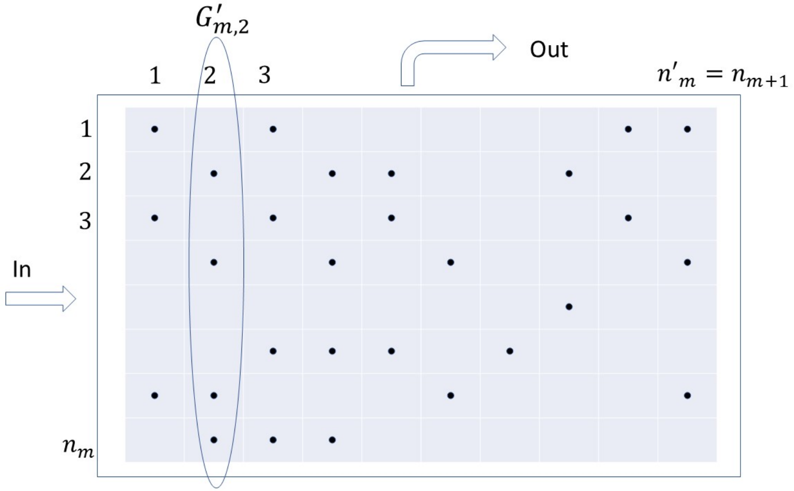
ConvNet: Layer m

Topology coding of the m^{th} layer



ConvNet: Layer m

Topology coding of the m^{th} layer



Semi-discrete Bessel Systems

A countable set of functions $\{g_n\ ,\ n\geq 1\}\subset L^2(S)$ (where S is a LCA group) is called a *semi-discrete Bessel system* in $L^2(S)$ if there is a constant (called a *Bessel bound*) $B\geq 0$ such that, for every $f\in L^2(S)$,

$$\sum_{n\geq 1}\|f*g_n\|_2^2\leq B\|f\|_2^2 \quad , \quad f*g_n(x)=\int_S f(x-y)g_n(y)dy.$$

The Lipschitz constant of a linear operator equals its operator norm. For nonlinear maps, the Lipschitz bound (square of its Lipschitz constant) is a replacement for the Bessel bound (or, the upper frame bound).

Lemma

Assume $\{g_n\ ,\ n\geq 1\}$ is a semi-discrete Bessel system in $L^2(\mathbb{R}^d)$. Then its optimal Bessel bound is given by

$$B=\sup_{\omega\in\mathbb{R}^n}\sum_{n\geq 1}|\widehat{g_n}(\omega)|^2=:\|\sum_{n\geq 1}|\widehat{g_n}|^2\|_\infty.$$

Layer Analysis

Bessel Bounds

In each layer *m* and for each *input* node *n* we define three types of Bessel bounds (one for each type of the detection/pooling/merge sublayer):

- 1st type Bessel bound:

$$B^{(1)}_{m,n} = \left\| \left|\hat{\phi}_{m,n}\right|^2 + \sum_{g_{m,n;k} \in G_{m,n}} l_{m,n;k} D^{-d}_{m,n;k} \left|\hat{g}_{m,n;k}\right|^2 \right\|_\infty \tag{5.2}$$

- 2nd type Bessel bound:

$$B^{(2)}_{m,n} = \left\| \sum_{g_{m,n;k} \in G_{m,n}} l_{m,n;k} D^{-d}_{m,n;k} \left|\hat{g}_{m,n;k}\right|^2 \right\|_\infty \tag{5.3}$$

- 3rd type (or generating) bound:

$$B^{(3)}_{m,n} = \|\hat{\phi}_{m,n}\|^2_\infty \; . \tag{5.4}$$

Layer Analysis

Bessel Bounds

Next we define the layer m Bessel bounds:

$$1^{\text{st}} \text{ type Bessel bound } \; B_m^{(1)} = \max_{1 \le n \le n_m} B_{m,n}^{(1)} \tag{5.5}$$

$$2^{\text{nd}} \text{ type Bessel bound } \; B_m^{(2)} = \max_{1 \le n \le n_m} B_{m,n}^{(2)} \tag{5.6}$$

$$3^{\text{rd}} \text{ type (generating) Bessel bound } \; B_m^{(3)} = \max_{1 \le n \le n_m} B_{m,n}^{(3)}. \tag{5.7}$$

Remark. These bounds characterize Bessel bounds of the associated semi-discrete Bessel systems.

Lipschitz Analysis

First Result

Theorem (1. BSZ'17)

Consider a Convolutional Neural Network $\mathcal{F}$ with M layers as described before, with non-expansive Lipschitz activation functions, $Lip(\varphi_{m,n,n'}) \leq 1$. 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

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

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 (2. BSZ'20)

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:

$$\begin{aligned} \Gamma = \max_{y_1,\dots,y_M,z_1,\dots,z_M\geq 0} \quad & \sum_{m=1}^{M} z_m \\ s.t. \quad & y_0 = 1 \\ & y_m + z_m \leq B_m^{(1)} y_{m-1}, \quad 1 \leq m \leq M \\ & y_m \leq B_m^{(2)} y_{m-1}, \quad 1 \leq m \leq M \\ & z_m \leq B_m^{(3)} y_{m-1}, \quad 1 \leq m \leq M \end{aligned} \tag{5.8}$$

Lipschitz Analysis

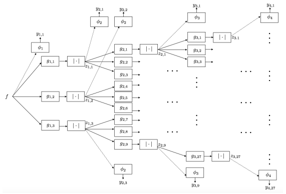
Second Result - cont'd

Theorem (2. BSZ'20)

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

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

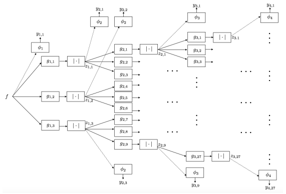
Example 1: Scattering Network



The Lipschitz constant:

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

Example 1: Scattering Network

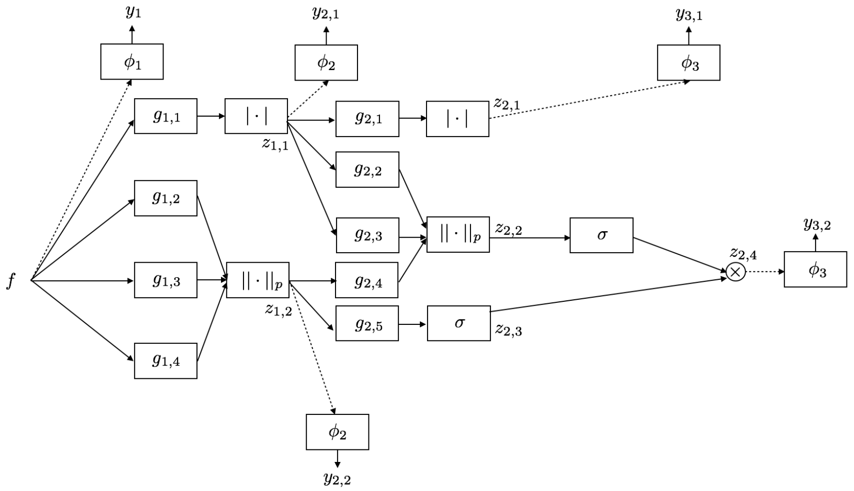


The Lipschitz constant:

- 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



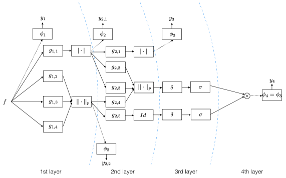
Example 2: A General Convolutive Neural Network

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 based objective functions

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 based objective functions

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:

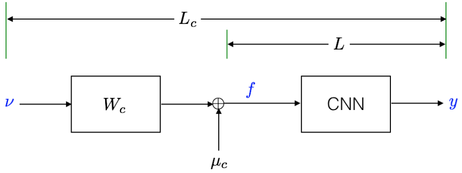
Lipschitz bound based separation:

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

Example 3: Lipschitz constant based objective functions

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 based objective functions

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.

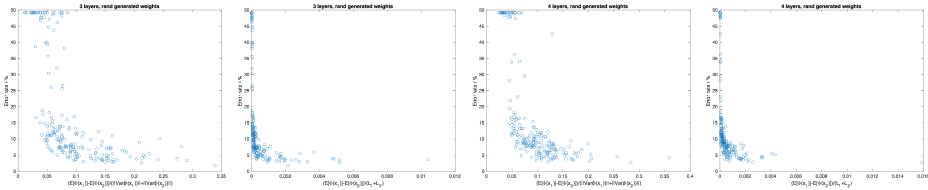


Figure: Results for uniformly distributed random weights

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

Example 3: Lipschitz constant based objective functions

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.

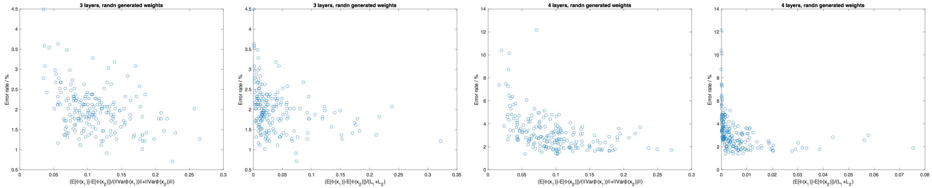


Figure: Results for normaly distributed random weights

Local Analysis

Consider a deep network $\mathcal{F} : (X, \|\cdot\|_2) \to (Y, \|\cdot\|_2)$ between Euclidean finite-dimensional linear spaces with M layers, where the i^{th} layer is characterized by the input-output nonlinear Lipschitz map $\mathcal{F}_i$. Denote by $J_{\mathcal{F}}$, $J_{\mathcal{F}_i}$ the Jacobian matrices of these maps. Then by an application of the Fundamental Theorem of Calculus (plus Lebesgue's differentiation theorem), the optimal Lipschitz constant is

$$Lip(\mathcal{F}) = \sup_{x \in X} \|J_{\mathcal{F}}(x)\|_{Op} = \sup_{x \in X} \|J_{\mathcal{F}_M} \cdots J_{\mathcal{F}_1}(x)\|_{Op}$$

where the Op norm is the largest singular value of the corresponding Jacobian.
In the case of type I or II network (i.e., no multiplicative aggregation), the nonlinear are homogeneous of degree 1, and in each layer the Jacobian factors as a product of 3 matrices:

$$J_{\mathcal{F}}(x) = P_M(x) D_M(x) A_M P_{M-1}(x) D_{M-1}(x) A_{M-1} \cdots P_1(x) D_1(x) A_1,$$

Local Analysis (2)

$$J_{\mathcal{F}}(x) = P_M(x)D_M(x)A_M P_{M-1}(x)D_{M-1}(x)A_{M-1}\cdots P_1(x)D_1(x)A_1,$$

where: A_i is the matrix associated to linear operators (filters), D_i is the diagonal matrix associated to derivative of activation functions (it is a binary matrix composed of 0's and 1's in the case of ReLU activation), and P_i is the matrix associated to the composition of downsampling and pooling sublayers. In the case of sum-pooling, P_i is independent of input x; in the case of max-filter, it has a weak dependency on x. In both cases it is sparse, with binary entries.

Local Analysis (2)

$$J_{\mathcal{F}}(x) = P_M(x)D_M(x)A_M P_{M-1}(x)D_{M-1}(x)A_{M-1}\cdots P_1(x)D_1(x)A_1,$$

where: A_i is the matrix associated to linear operators (filters), D_i is the diagonal matrix associated to derivative of activation functions (it is a binary matrix composed of 0's and 1's in the case of ReLU activation), and P_i is the matrix associated to the composition of downsampling and pooling sublayers. In the case of sum-pooling, P_i is independent of input x; in the case of max-filter, it has a weak dependency on x. In both cases it is sparse, with binary entries.

Results for Alex Net using method:	Lip const
Analytical estimate: based on Theorem 1	2.51×10^3
Empirical bound: quotient from pairs of samples	7.32×10^{-3}
Numerical estimate: maximize the "sandwich" formula	1.44

Local Analysis: Domains of linearity

It is not suprising that the analytic estimate 2.51×10^3 is bigger than the numerical estimate 1.44. The suprising conclusion is the difference between the numerical estimate, 1.44, and the empirical bound 7.32^{-3}.

The "sandwich" formula provides additional information: The upper bound is achieved locally for the principal right-singular vector v at the specific input x where the maximum is achieved. We performed the following numerical expriment: we computed the ratio $R(t) = \frac{1}{t}\|\mathcal{F}(x+tv)-\mathcal{F}(x)\|_2$:

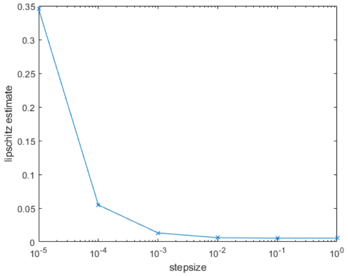


Figure: The ratio $R(t) = ||\mathcal{F}(x + t \cdot v) - \mathcal{F}(x)||/t$ for different t.

Lipschitz Analysis: Stochastic Model

The numerical study of the Alex Net showed that the optimal Lipschitz constant is somewhat theoretical and is achieved by very small perturbations. Notice for two inputs x_1 and x_2:

$$\mathcal{F}(x_1)-\mathcal{F}(x_2) = \int_0^1 J_{\mathcal{F}_M} J_{\mathcal{F}_{M-1}} \cdots J_{\mathcal{F}_1}((1-t)x_1+tx_2)(x_2-x_1)dt = J_* \cdot (x_2-x_1)$$

where the *effective Jacobian* J_* is estimated by

$$J_* \approx (\mathbb{E}[P_M])(\mathbb{E}[D_M])A_M \cdots (\mathbb{E}[P_1])(\mathbb{E}[D_1])A_1$$

where we assume:

1. (ergodicity) x_1 and x_2 are sufficientlly distinct so that the network passes through all linearity domains during the convex combination $x_1 \to (1-t)x_2 + tx_2 \to x_2$, and
2. (independence) the behavior of activation maps and pooling sublayers are independent from layer to layer.