Lecture 1: Theoretical Foundations of Deep Learning

Gitta Kutyniok

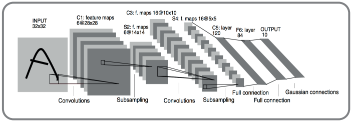
(Ludwig-Maximilians-Universität München and University of Tromsø)

Arnold Sommerfeld School "Physics meets Artificial Intelligence"
LMU Munich, September 12 – 16, 2022

The Dawn of Artificial Intelligence in Public Life

Self-Driving Cars

Telecommunication/
Speech Recognition



Legal Issues

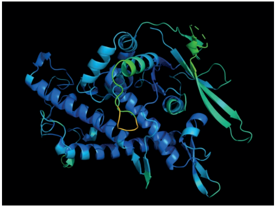
Health Care

NEWS · 30 NOVEMBER 2020

‘It will change everything’: DeepMind’s AI makes gigantic leap in solving protein structures

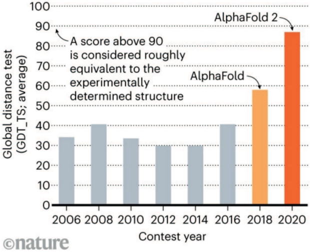
Google’s deep-learning program for determining the 3D shapes of proteins stands to transform biology, say scientists.

Nature **588**, 203-204 (2020)



STRUCTURE SOLVER

DeepMind’s AlphaFold 2 algorithm significantly outperformed other teams at the CASP14 protein-folding contest — and its previous version’s performance at the last CASP.



Impact on Mathematical/Physical Problem Settings

Some Examples:

- Inverse Probleme/Imaging Science *(2012–)*
  - $\rightsquigarrow$ *Denoising*
  - $\rightsquigarrow$ *Edge Detection*
  - $\rightsquigarrow$ *Inpainting*
  - $\rightsquigarrow$ *Classification*
  - $\rightsquigarrow$ *Superresolution*
  - $\rightsquigarrow$ *Limited-Angle Computed Tomography*
  - $\rightsquigarrow$ ...

Impact on Mathematical/Physical Problem Settings

Some Examples:

- Inverse Probleme/Imaging Science *(2012–)*
  - ⇝ *Denoising*
  - ⇝ *Edge Detection*
  - ⇝ *Inpainting*
  - ⇝ *Classification*
  - ⇝ *Superresolution*
  - ⇝ *Limited-Angle Computed Tomography*
  - ⇝ ...

- Numerical Analysis of Partial Differential Equations *(2017–)*
  - ⇝ *Black-Scholes PDE*
  - ⇝ *Allen-Cahn PDE*
  - ⇝ *Parametric PDEs*
  - ⇝ ...

Impact on Mathematical/Physical Problem Settings

Some Examples:

- Inverse Probleme/Imaging Science *(2012–)*
  - $\leadsto$ *Denoising*
  - $\leadsto$ *Edge Detection*
  - $\leadsto$ *Inpainting*
  - $\leadsto$ *Classification*
  - $\leadsto$ *Superresolution*
  - $\leadsto$ *Limited-Angle Computed Tomography*
  - $\leadsto$ ...

- Numerical Analysis of Partial Differential Equations *(2017–)*
  - $\leadsto$ *Black-Scholes PDE*
  - $\leadsto$ *Allen-Cahn PDE*
  - $\leadsto$ *Parametric PDEs*
  - $\leadsto$ ...

- Modelling *(2018–)*
  - $\leadsto$ *Learning physical laws from data*

Artificial Intelligence = Alchemy?

AAAS | Science

AI researchers allege that machine learning is alchemy

By **Matthew Hutson** | May. 3, 2018 , 11:15 AM

Ali Rahimi, a researcher in artificial intelligence (AI) at Google in San Francisco, California, took a swipe at his field last December—and received a 40-second ovation for it. Speaking at an AI conference, Rahimi charged that machine learning algorithms, in which computers learn through trial and error, **have become a form of "alchemy."** Researchers, he said, do not know why some algorithms work and others don't, nor do they have rigorous criteria for choosing one AI architecture over another. Now, in a paper presented on 30 April at the International Conference on Learning Representations in Vancouver, Canada, Rahimi and his collaborators **document examples** of what they see as the alchemy problem and offer prescriptions for bolstering AI's rigor.

Problem with Reliability

By Linda Geddes 5th December 2018

Computers can be made to see a sea turtle as a gun or hear a concerto as someone's voice, which is raising concerns about using artificial intelligence in the real world.

MACHINE MINDS | ARTIFICIAL INTELLIGENCE BBC

Role of Theory

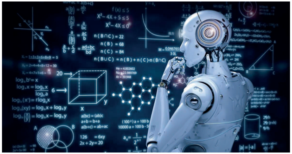
Two Key Challenges:

Mathematics for Artificial Intelligence!

- Can we derive a deep theoretical understanding of deep learning?
- How can we make deep learning more robust?
- ...

Artificial Intelligence for Mathematical/Physical Problem Settings!

- How can we use deep learning to improve imaging science?
- Can we develop superior PDE solvers via deep learning?
- ...



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Delving Deeper into Artificial Intelligence...

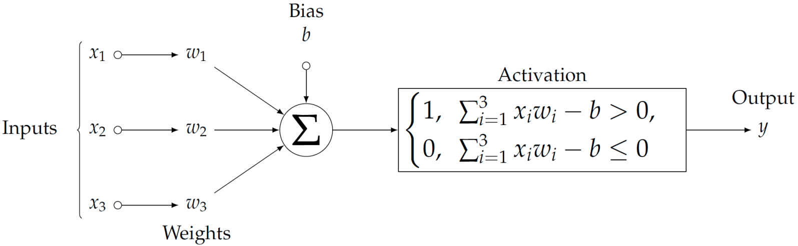
First Appearance of Artificial Intelligence

Key Task of McCulloch and Pitts (1943):

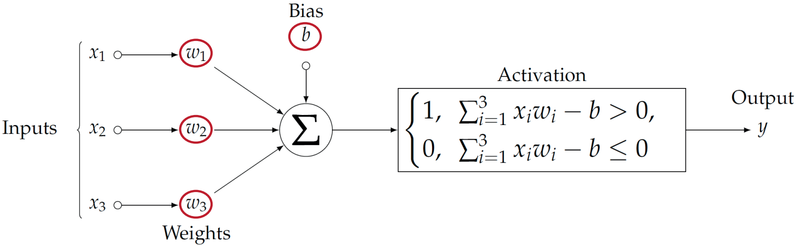
- Develop an algorithmic approach to learning.
- Mimicking the functionality of the human brain.

Goal: Artificial Intelligence!

Artificial Neurons



Artificial Neurons



Artificial Neurons

Definition: An *artificial neuron* with *weights* $w_1, ..., w_n \in \mathbb{R}$, *bias* $b \in \mathbb{R}$ and *activation function* $\varrho : \mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x_1, ..., x_n) = \varrho\left(\sum_{i=1}^{n} x_i w_i - b\right) = \varrho(\langle x, w\rangle - b),$$

where $w = (w_1, ..., w_n)$ and $x = (x_1, ..., x_n)$.

Artificial Neurons

Definition: An *artificial neuron* with *weights* $w_1, \ldots, w_n \in \mathbb{R}$, *bias* $b \in \mathbb{R}$ and *activation function* $\varrho : \mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x_1, \ldots, x_n) = \varrho\left(\sum_{i=1}^{n} x_i w_i - b\right) = \varrho(\langle x, w\rangle - b),$$

where $w = (w_1, \ldots, w_n)$ and $x = (x_1, \ldots, x_n)$.

Examples of Activation Functions:

- Heaviside function $\varrho(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0. \end{cases}$
- Sigmoid function $\varrho(x) = \frac{1}{1+e^{-x}}$.
- *Rectifiable Linear Unit (ReLU)* $\varrho(x) = \max\{0, x\}$.

Affine Linear Maps and Weights

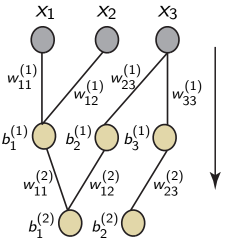
Remark: Concatenating artificial neurons leads to *compositions of affine linear maps and activation functions*.

Example: The following part of a neural network is given by

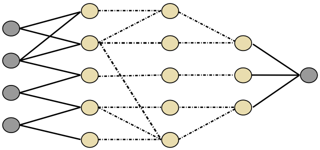
$$\Phi : \mathbb{R}^3 \to \mathbb{R}^2, \quad \Phi(x) = W^{(2)} \varrho(W^{(1)}x + b^{(1)}) + b^{(2)}.$$

$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & 0 \\ 0 & 0 & w_{23}^{(1)} \\ 0 & 0 & w_{33}^{(1)} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} & 0 \\ 0 & 0 & w_{23}^{(2)} \end{pmatrix}$$



Definition of a Deep Neural Network



Definition:

Assume the following notions:

- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- $\varrho : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$, where $T_\ell x = W^{(\ell)} x + b^{(\ell)}$

Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x))), \quad x \in \mathbb{R}^d,$$

is called *(deep) neural network (DNN)*.

Second Appearance of Neural Networks

Key Observations by Y. LeCun et al. (around 2000):

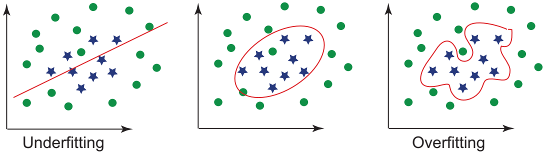
- Drastic improvement of computing power.
  $\rightsquigarrow$ *Networks with hundreds of layers can be trained.*
  $\rightsquigarrow$ *Deep Neural Networks!*
- Age of Data starts.
  $\rightsquigarrow$ *Vast amounts of training data is available.*

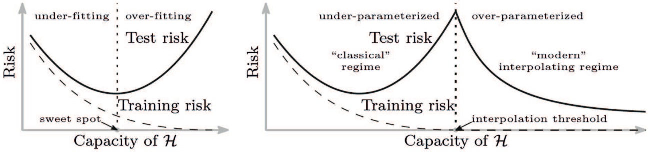
Second Appearance of Neural Networks

Key Observations by Y. LeCun et al. (around 2000):

- Drastic improvement of computing power.
  $\rightsquigarrow$ *Networks with hundreds of layers can be trained.*
  $\rightsquigarrow$ *Deep Neural Networks!*
- Age of Data starts.
  $\rightsquigarrow$ *Vast amounts of training data is available.*

Surprising Phenomenon:



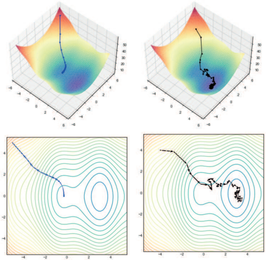


(Source: Belkin, Hsu, Ma, Mandal; 2019)

Second Appearance of Neural Networks

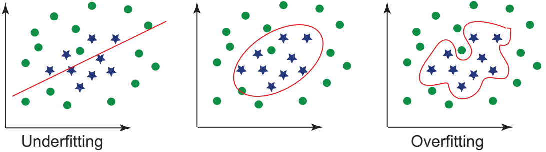
Key Observations by Y. LeCun et al. (around 2000):

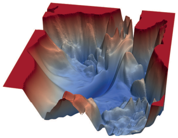
- Drastic improvement of computing power.
  - $\rightsquigarrow$ *Networks with hundreds of layers can be trained.*
  - $\rightsquigarrow$ *Deep Neural Networks!*
- Age of Data starts.
  - $\rightsquigarrow$ *Vast amounts of training data is available.*

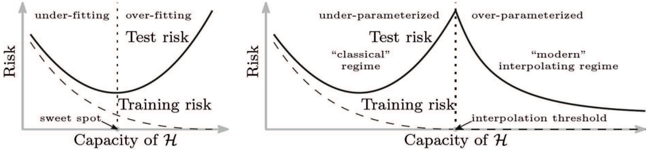


(Source: Berner, Grohs, Kutyniok, Petersen; 2

Surprising Phenomenon:







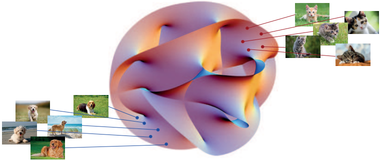
(Source: Belkin, Hsu, Ma, Mandal; 2019)

Training of Deep Neural Networks

High-Level Set Up:

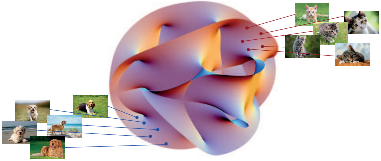
- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.

  $\rightsquigarrow$ *Training- and test data set.*

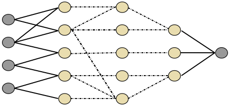


Training of Deep Neural Networks

High-Level Set Up:



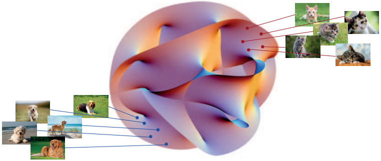
- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.
  $\rightsquigarrow$ *Training- and test data set.*



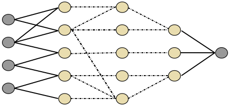
- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and ϱ.
  Sometimes selected entries of the matrices $(W^{(\ell)})_{\ell=1}^L$, i.e., weights, are set to zero at this point.

Training of Deep Neural Networks

High-Level Set Up:



- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.
  $\rightsquigarrow$ *Training- and test data set.*



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and ϱ.
  Sometimes selected entries of the matrices $(W^{(\ell)})_{\ell=1}^L$, i.e., weights, are set to zero at this point.
- Learn the affine-linear functions $(T_\ell)_{\ell=1}^L = (W^{(\ell)} \cdot + b^{(\ell)})_{\ell=1}^L$ by

$$\min_{(W^{(\ell)}, b^{(\ell)})_\ell} \sum_{i=1}^m \mathcal{L}(\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell}(x_i), f(x_i)) + \lambda \mathcal{R}((W^{(\ell)}, b^{(\ell)})_\ell)$$

  yielding the network $\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell} : \mathbb{R}^d \to \mathbb{R}^{N_L}$,

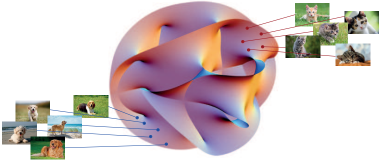
$$\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell}(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x)))).$$

  This is often done by stochastic gradient descent.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

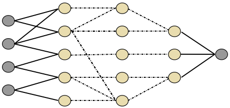
Training of Deep Neural Networks

High-Level Set Up:



- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.

  $\rightsquigarrow$ *Training- and test data set.*



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and ϱ.

  Sometimes selected entries of the matrices $(W^{(\ell)})_{\ell=1}^L$, i.e., weights, are set to zero at this point.

- Learn the affine-linear functions $(T_\ell)_{\ell=1}^L = (W^{(\ell)} \cdot + b^{(\ell)})_{\ell=1}^L$ by

$$\min_{(W^{(\ell)}, b^{(\ell)})_\ell} \sum_{i=1}^m \mathcal{L}(\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell}(x_i), f(x_i)) + \lambda \mathcal{R}((W^{(\ell)}, b^{(\ell)})_\ell)$$

  yielding the network $\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell} : \mathbb{R}^d \to \mathbb{R}^{N_L}$,

$$\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell}(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x)))).$$

  This is often done by stochastic gradient descent.

Goal: $\Phi_{(W^{(\ell)}, b^{(\ell)})_\ell}(x_i) \approx f(x_i)$ for the test data!

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

- **Expressivity:**
  - Which *aspects of a neural network architecture* affect the performance of deep learning?

$\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*

- **Expressivity:**
  - Which *aspects of a neural network architecture* affect the performance of deep learning?

$\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*

- **Learning:**
  - Why does *stochastic gradient descent* converge to good local minima despite the non-convexity of the problem?

$\rightsquigarrow$ *Algebraic/Differential Geometry, Optimal Control, Optimization, ...*

- **Expressivity:**
  - Which *aspects of a neural network architecture* affect the performance of deep learning?

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- **Learning:**
  - Why does *stochastic gradient descent* converge to good local minima despite the non-convexity of the problem?

  $\rightsquigarrow$ *Algebraic/Differential Geometry, Optimal Control, Optimization, ...*
- **Generalization:**
  - Can we derive overall *success guarantees* (on the test data set)?

  $\rightsquigarrow$ *Learning Theory, Probability Theory, Statistics, ...*

Mathematics for Artificial Intelligence

- **Expressivity:**
  - Which *aspects of a neural network architecture* affect the performance of deep learning?

$\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*

- **Learning:**
  - Why does *stochastic gradient descent* converge to good local minima despite the non-convexity of the problem?

$\rightsquigarrow$ *Algebraic/Differential Geometry, Optimal Control, Optimization, ...*

- **Generalization:**
  - Can we derive overall *success guarantees* (on the test data set)?

$\rightsquigarrow$ *Learning Theory, Probability Theory, Statistics, ...*

- **Explainability:**
  - Why did a trained deep neural network *reach a certain decision*?

$\rightsquigarrow$ *Information Theory, Uncertainty Quantification, ...*

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Artificial Intelligence for Mathematical/Physical Problem Settings

- **Inverse Problems:**
  - How do we *optimally combine* deep learning with model-based approaches?
  - Are neural networks capable of *replacing highly specialized numerical algorithms* in natural sciences?

$\rightsquigarrow$ *Imaging Science, Inverse Problems, Microlocal Analysis, ...*

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Artificial Intelligence for Mathematical/Physical Problem Settings

- **Inverse Problems:**
  - How do we *optimally combine* deep learning with model-based approaches?
  - Are neural networks capable of *replacing highly specialized numerical algorithms* in natural sciences?

$\rightsquigarrow$ *Imaging Science, Inverse Problems, Microlocal Analysis, ...*

- **Partial Differential Equations:**
  - Why do neural networks perform well in *very high-dimensional environments*?
  - Are neural networks capable of *replacing highly specialized numerical algorithms* in natural sciences?

$\rightsquigarrow$ *Numerical Mathematics, Partial Differential Equations, ...*

Plan for the 2 Lectures

Are Deep Neural Networks at Least as Good as All Previous Mathematical Methods?

- *Expressivity*

Plan for the 2 Lectures

Are Deep Neural Networks at Least as Good as All Previous Mathematical Methods?

- *Expressivity*

Are Deep Neural Networks Really Better Than Classical Methods?
Solving...

- *...Inverse Problems:* Optimally combining deep learning with classical methods!
- *...Partial Differential Equations:* Breaking the curse of dimensionality!

Plan for the 2 Lectures

Are Deep Neural Networks at Least as Good as All Previous Mathematical Methods?

- *Expressivity*

Are Deep Neural Networks Really Better Than Classical Methods?
Solving...

- *...Inverse Problems:* Optimally combining deep learning with classical methods!
- *...Partial Differential Equations:* Breaking the curse of dimensionality!

Is Artificial Intelligence Reliable?

- *Generalization*
- *Explainability*
- *Limitations*

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Are Deep Neural Networks at Least as Good as All Previous Mathematical Methods?

Expressivity

One major ingredient of mathematical methods is typically a *suitable representation/approximation* of the function/data:

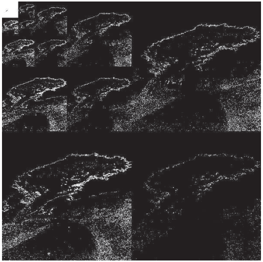
Deep neural networks are universal!

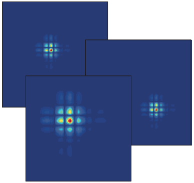
Some Key Questions in Expressivity:

- What is the expressive power of a *given architecture*?
- What effect has the *depth* of a neural network in this respect?
- What is the *complexity* of the approximating neural network?
- What are *suitable function spaces* to consider?

Revisiting Approximation Theory

The World is Compressible!





Wavelet Transform (JPEG2000):

$$f \mapsto (\langle f, \psi_{j,m} \rangle)_{j,m}.$$

Definition: For a wavelet $\psi \in L^2(\mathbb{R}^2)$, a *wavelet system* is defined by

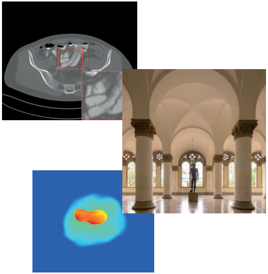
$$\{\psi_{j,m} : j \in \mathbb{Z}, m \in \mathbb{Z}^2\}, \quad \text{where } \psi_{j,m}(x) := 2^j \psi(2^j x - m).$$

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

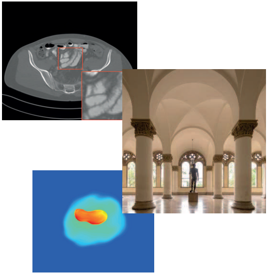
Modeling Multivariate Data/Functions

Key Observation:

Directional structures are often crucial!



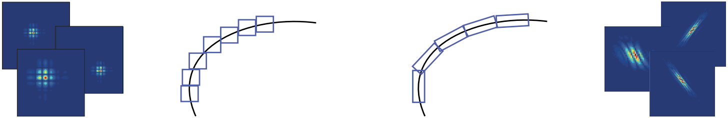
Modeling Multivariate Data/Functions



Key Observation:

> Directional structures are often crucial!

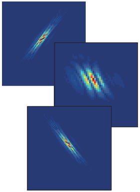
Problem with Wavelets:



Shearlets

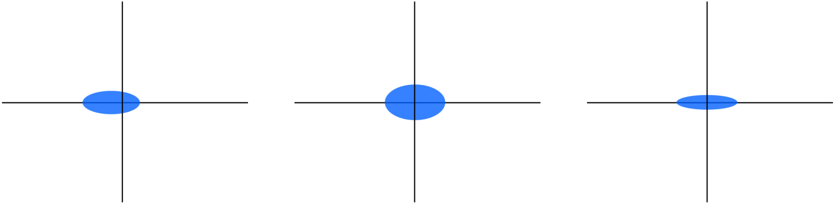
Shearlets (Kutyniok, Labate; 2006):

$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

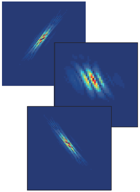
$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$



Shearlets

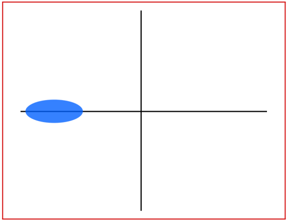
Shearlets (Kutyniok, Labate; 2006):

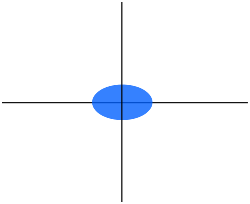
$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j,k \in \mathbb{Z}.$$

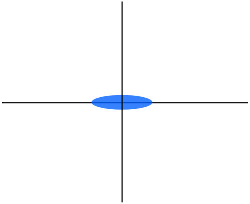


Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$



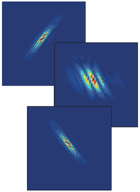




Shearlets

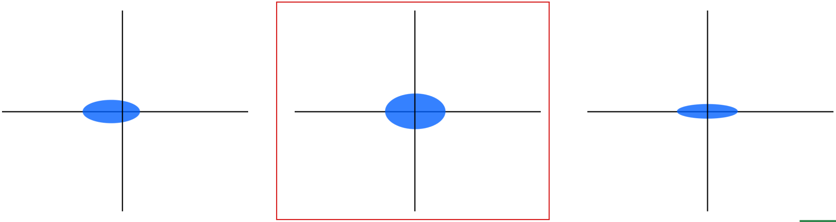
Shearlets (Kutyniok, Labate; 2006):

$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

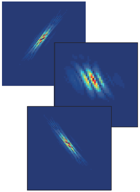
$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$



Shearlets

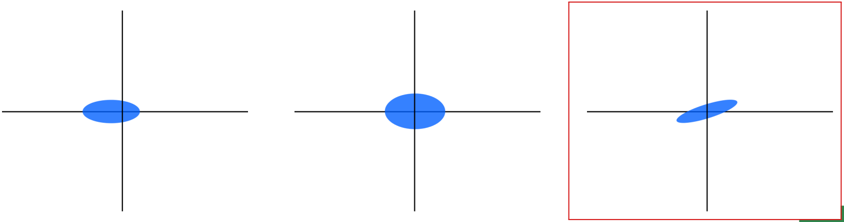
Shearlets (Kutyniok, Labate; 2006):

$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j,k \in \mathbb{Z}.$$



Then

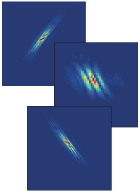
$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot -m).$$



Shearlets

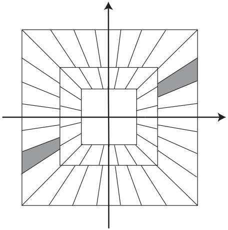
Shearlets (Kutyniok, Labate; 2006):

$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j,k \in \mathbb{Z}.$$



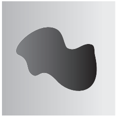
Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$



Shearlets are Optimal

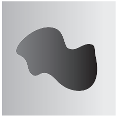
Model of Images (Donoho; 2001):
"*Cartoon-functions* are functions governed by a discontinuity curve."



Theorem (Kutyniok, Lim; 2011):
"Shearlets fulfill the *optimal compression rate* for cartoon-functions."

Shearlets are Optimal

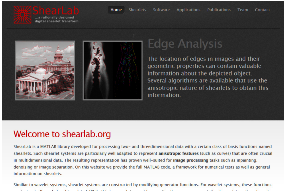
Model of Images (Donoho; 2001):
"*Cartoon-functions* are functions governed by a discontinuity curve."



> **Theorem (Kutyniok, Lim; 2011):**
> "Shearlets fulfill the *optimal compression rate* for cartoon-functions."

2D&3D (parallelized) Fast Shearlet Transform (`www.ShearLab.org`):

- Matlab *(Kutyniok, Lim, Reisenhofer; 2013)*
- Julia *(Loarca; 2017)*
- Python *(Look; 2018)*
- Tensorflow *(Loarca; 2019)*



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Function Approximation in a Nutshell

Goal: Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and $(\varphi_i)_{i\in I} \subseteq L^2(\mathbb{R}^d)$. Measure the suitability of $(\varphi_i)_{i\in I}$ for uniformly approximating functions from $\mathcal{C}$.

Definition: The *error of best N-term approximation* of some $f \in \mathcal{C}$ is given by

$$\|f - f_N\|_2 := \inf_{I_N \subset I, \# I_N = N, (c_i)_{i\in I_N}} \|f - \sum_{i \in I_N} c_i \varphi_i\|_2.$$

The largest $\gamma > 0$ such that

$$\sup_{f\in\mathcal{C}} \|f - f_N\|_2 = O(N^{-\gamma}) \qquad \text{as } N \to \infty$$

determines the *optimal (sparse) approximation rate* of $\mathcal{C}$ by $(\varphi_i)_{i\in I}$.

Function Approximation in a Nutshell

Goal: Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and $(\varphi_i)_{i\in I} \subseteq L^2(\mathbb{R}^d)$. Measure the suitability of $(\varphi_i)_{i\in I}$ for uniformly approximating functions from $\mathcal{C}$.

Definition: The *error of best N-term approximation* of some $f \in \mathcal{C}$ is given by

$$\|f - f_N\|_2 := \inf_{I_N \subset I, \#I_N = N, (c_i)_{i\in I_N}} \|f - \sum_{i\in I_N} c_i \varphi_i\|_2.$$

The largest $\gamma > 0$ such that

$$\sup_{f\in\mathcal{C}} \|f - f_N\|_2 = O(N^{-\gamma}) \qquad \text{as } N \to \infty$$

determines the *optimal (sparse) approximation rate* of $\mathcal{C}$ by $(\varphi_i)_{i\in I}$.

Approximation accuracy $\leftrightarrow$ *Complexity of approximating system in terms of sparsity*

Universality of Deep Neural Networks

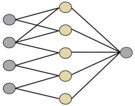
Remark: Assume ϱ is a polynomial of degree q. Then $\varrho(Wx + b)$ is also a polynomial of degree q, hence Φ is also a polynomial of degree $\leq L \cdot q$. Hence in this case $C(\mathbb{R}^d)$ cannot be well approximated.

Universality of Shallow Neural Networks

Remark: Assume ϱ is a polynomial of degree q. Then $\varrho(Wx + b)$ is also a polynomial of degree q, hence Φ is also a polynomial of degree $\leq L \cdot q$. Hence in this case $C(\mathbb{R}^d)$ cannot be well approximated.

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\epsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ with

$$\|f - \sum_{k=1}^{N} a_k \varrho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \epsilon.$$



Every continuous function on a compact set can be arbitrarily well approximated with a neural network with one single hidden layer.

Idea of Proof

- For $d \geq 1, \varrho$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ TFAE:
  - (i) $\mathrm{span}\{\varrho(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  - (ii) ϱ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth activation function ϱ.

Idea of Proof

- For $d \geq 1, \varrho$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ TFAE:
  - (i) $\mathrm{span}\{\varrho(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  - (ii) ϱ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth activation function ϱ.
- Since ϱ is not a polynomial, there exists one $x_0 \in \mathbb{R}$ with

$$\varrho^{(k)}(-x_0) \neq 0 \text{ for all } k.$$

Idea of Proof

- For $d \geq 1, \varrho$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ TFAE:
  (i) $\mathrm{span}\{\varrho(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  (ii) ϱ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth activation function ϱ.
- Since ϱ is not a polynomial, there exists one $x_0 \in \mathbb{R}$ with

$$\varrho^{(k)}(-x_0) \neq 0 \text{ for all } k.$$

- Constant functions can be arbitrarily well approximated:

$$\varrho(hx - x_0) \to \varrho(-x_0) \text{ as } h \to 0.$$

Idea of Proof

- For $d \geq 1, \varrho$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ TFAE:
  (i) $\mathrm{span}\{\varrho(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  (ii) ϱ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth activation function ϱ.
- Since ϱ is not a polynomial, there exists one $x_0 \in \mathbb{R}$ with

$$\varrho^{(k)}(-x_0) \neq 0 \text{ for all } k.$$

- Constant functions can be arbitrarily well approximated:

$$\varrho(hx - x_0) \to \varrho(-x_0) \text{ as } h \to 0.$$

- Linear functions can be arbitrarily well approximated:

$$\underbrace{\frac{\varrho((\lambda + h)x - x_0) - \varrho(x - x_0)}{h}}_{\to x\varrho'(\lambda x - x_0) \text{ for } h \to 0} \to x \cdot \varrho'(-x_0), \qquad \text{as } h, \lambda \to 0.$$

Idea of Proof

- For $d \geq 1, \varrho$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ TFAE:
  - (i) $\mathrm{span}\{\varrho(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  - (ii) ϱ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth activation function ϱ.
- Since ϱ is not a polynomial, there exists one $x_0 \in \mathbb{R}$ with

$$\varrho^{(k)}(-x_0) \neq 0 \text{ for all } k.$$

- Constant functions can be arbitrarily well approximated:

$$\varrho(hx - x_0) \to \varrho(-x_0) \text{ as } h \to 0.$$

- Linear functions can be arbitrarily well approximated:

$$\underbrace{\frac{\varrho((\lambda + h)x - x_0) - \varrho(x - x_0)}{h}}_{\to x\varrho'(\lambda x - x_0) \text{ for } h \to 0} \to x \cdot \varrho'(-x_0), \qquad \text{as } h, \lambda \to 0.$$

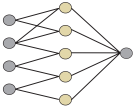
$\rightsquigarrow$ Any polynomial can be well approximated, then use Stone-Weierstraß

$\rightsquigarrow$ Finally, extend to d arbitrary.

Universality of Shallow Neural Networks

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\epsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ with

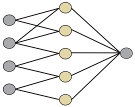
$$\|f - \sum_{k=1}^{N} a_k \varrho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \epsilon.$$



Universality of Shallow Neural Networks

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\epsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ with

$$\|f - \sum_{k=1}^{N} a_k \varrho(\langle w_k, \cdot\rangle - b_k)\|_\infty \leq \epsilon.$$

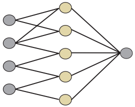


Approximation accuracy $\leftrightarrow$ Complexity of approximating network?

Universality of Shallow Neural Networks

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\varrho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\epsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ with

$$\|f - \sum_{k=1}^{N} a_k \varrho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \epsilon.$$

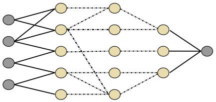


Approximation accuracy $\leftrightarrow$ Complexity of approximating network?

What about even optimality?

Complexity of a Deep Neural Network

Recall:



- L: Number of layers.
- $\varrho : \mathbb{R} \to \mathbb{R}$: *Activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$, where $T_\ell x = W^{(\ell)} x + b^{(\ell)}$

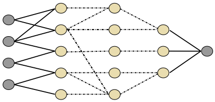
Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x)))), \quad x \in \mathbb{R}^d,$$

is called *(deep) neural network (DNN)*.

Complexity of a Deep Neural Network

Recall:



- L: Number of layers.
- $\varrho : \mathbb{R} \to \mathbb{R}$: *Activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$, where $T_\ell x = W^{(\ell)} x + b^{(\ell)}$

Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x))), \quad x \in \mathbb{R}^d,$$

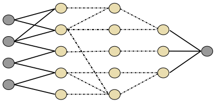
is called *(deep) neural network (DNN)*.

Measure for Complexity: The *complexity $C(\Phi)$* is defined by

$$C(\Phi) := \sum_{\ell=1}^{L} \left(\|W^{(\ell)}\|_0 + \|b^{(\ell)}\|_0 \right).$$

Complexity of a Deep Neural Network

Recall:



- L: Number of layers.
- $\varrho : \mathbb{R} \to \mathbb{R}$: *Activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$, where $T_\ell x = W^{(\ell)} x + b^{(\ell)}$

Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \varrho(T_{L-1} \varrho(\ldots \varrho(T_1(x))), \quad x \in \mathbb{R}^d,$$

is called *(deep) neural network (DNN)*.

Measure for Complexity: The *complexity $C(\Phi)$* is defined by

$$C(\Phi) := \sum_{\ell=1}^{L} \left(\|W^{(\ell)}\|_0 + \|b^{(\ell)}\|_0 \right).$$

Key Challenge:

Approximation accuracy $\leftrightarrow$ Complexity of approximating network in terms of memory efficiency!

Lower Bounds for Approximation

Classical Approach:

- VC Dimension

Lower Bounds for Approximation

Classical Approach:

- VC Dimension

Towards Optimal Complexity:

- How well can functions be approximated by neural networks with few non-zero weights?
  - Can we derive a lower bound on the necessary number of weights?
  - Can we construct neural networks which attain this bound?
- Are neural networks as good approximators as wavelets and shearlets?

A Fundamental Lower Bound

Complexity of a Function Class:

The *optimal exponent* $\gamma^*(\mathcal{C})$ measures the complexity of $\mathcal{C} \subset L^2(\mathbb{R}^d)$.

A Fundamental Lower Bound

Complexity of a Function Class:
The *optimal exponent* $\gamma^*(\mathcal{C})$ measures the complexity of $\mathcal{C} \subset L^2(\mathbb{R}^d)$.

Theorem (Bölcskei, Grohs, Kutyniok, and Petersen; 2019):
Let $d \in \mathbb{N}$, $\varrho : \mathbb{R} \to \mathbb{R}$, and let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. Further, let

$$\textbf{Learn} : (0,1) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,\varrho}$$

satisfy that, for each $f \in \mathcal{C}$ and $0 < \epsilon < 1$,

$$\sup_{f \in \mathcal{C}} \|f - \textbf{Learn}(\epsilon, f)\|_2 \leq \epsilon.$$

Then, for all $\gamma < \gamma^*(\mathcal{C})$,

$$\epsilon^\gamma \sup_{f \in \mathcal{C}} C(\textbf{Learn}(\epsilon, f)) \to \infty, \qquad \text{as } \epsilon \to 0.$$

Conceptual bound independent on the learning algorithm!

A Fundamental Lower Bound

Complexity of a Function Class:

The *optimal exponent* $\gamma^*(\mathcal{C})$ measures the complexity of $\mathcal{C} \subset L^2(\mathbb{R}^d)$.

Theorem (Bölcskei, Grohs, Kutyniok, and Petersen; 2019):

Let $d \in \mathbb{N}$, $\varrho : \mathbb{R} \to \mathbb{R}$, and let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. Further, let

$$\textbf{Learn} : (0,1) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,\varrho}$$

satisfy that, for each $f \in \mathcal{C}$ and $0 < \epsilon < 1$,

$$\sup_{f \in \mathcal{C}} \|f - \textbf{Learn}(\epsilon, f)\|_2 \leq \epsilon.$$

Then, for all $\gamma < \gamma^*(\mathcal{C})$,

$$\epsilon^\gamma \sup_{f \in \mathcal{C}} C(\textbf{Learn}(\epsilon, f)) \to \infty, \qquad \text{as } \epsilon \to 0.$$

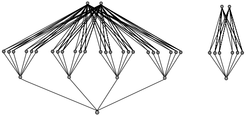
Conceptual bound independent on the learning algorithm!

$\rightsquigarrow$ *What happens for $\gamma = \gamma^*(\mathcal{C})$?*

Optimal Approximation

Key Ideas for a Specific Function Class:

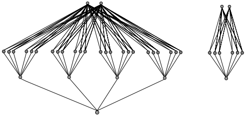
- Consider a representation system with an optimal approximation rate.
- Realize each element of a representation system by a neural network.
- Mimic best N-term approximation by networks.



Optimal Approximation

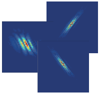
Key Ideas for a Specific Function Class:

- Consider a representation system with an optimal approximation rate.
- Realize each element of a representation system by a neural network.
- Mimic best N-term approximation by networks.



Choice for our Result:

Use the affine system of *shearlets*.



Theorem (Bölcskei, Grohs, Kutyniok, and Petersen; 2019):

Let ϱ be a suitably chosen, and let $\epsilon > 0$. For all $f \in \mathcal{E}^2(\mathbb{R}^2)$ and $N \in \mathbb{N}$, there exists a neural network Φ with 3 layers and $C(\Phi) = O(N)$ satisfying

$$\|f - \Phi\|_2 \lesssim N^{-1+\epsilon} \to 0 \quad \text{as } N \to \infty.$$

This is the optimal rate; hence the first bound is sharp!

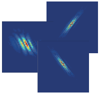
Optimal Approximation

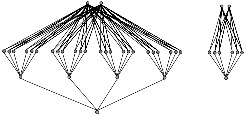
Key Ideas for a Specific Function Class:

- Consider a representation system with an optimal approximation rate.
- Realize each element of a representation system by a neural network.
- Mimic best N-term approximation by networks.

Choice for our Result:

Use the affine system of *shearlets*.





Theorem (Bölcskei, Grohs, Kutyniok, and Petersen; 2019):

Let ϱ be a suitably chosen, and let $\epsilon > 0$. For all $f \in \mathcal{E}^2(\mathbb{R}^2)$ and $N \in \mathbb{N}$, there exists a neural network Φ with 3 layers and $C(\Phi) = O(N)$ satisfying

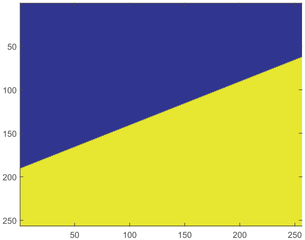
$$\|f - \Phi\|_2 \lesssim N^{-1+\epsilon} \to 0 \quad \text{as } N \to \infty.$$

This is the optimal rate; hence the first bound is sharp!

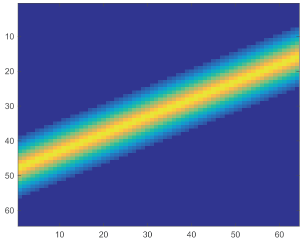
Deep neural networks achieve optimal approximation properties of all affine systems combined!

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Numerical Experiments (with ReLUs & Backpropagation)

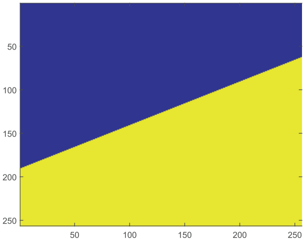


Linear Singularity

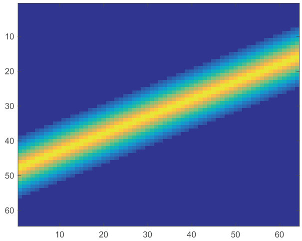


Subnetworks: $\approx$ *Ridgelets!*

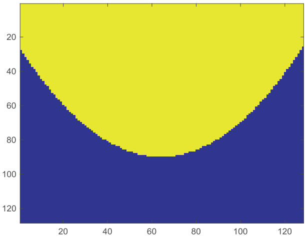
Numerical Experiments (with ReLUs & Backpropagation)



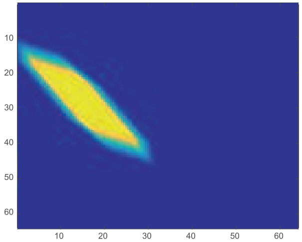
Linear Singularity



Subnetworks: $\approx$ *Ridgelets!*



Curvilinear Singularity



Subnetworks: $\approx$ *Shearlets!*

Are Deep Neural Networks Really Better

Than Classical Methods?

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

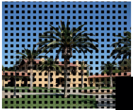
Inverse Problems

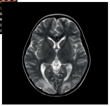
Recovering the original data from a transformed version!

Inverse Problems

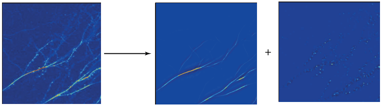
Recovering the original data from a transformed version!

Some Examples from Imaging:





- Inpainting.
  $\leadsto$ *Recovery from incomplete data.*
- Magnetic Resonance Imaging.
  $\leadsto$ *Recovery from point samples of the Fourier transform.*
- Feature Extraction.
  $\leadsto$ *Separating the image into different features.*



Ill-Posed Inverse Problems

General Setting:

Given $K : X \to Y$ and $y \in Y$, compute $x \in X$ with $Kx = y$.

Well-Posedness Conditions (Hadamard):

- *Existence:* For each $y \in Y$, there exists some $x \in X$ with $Kx = y$.
- *Uniqueness:* Such an $x \in X$ is unique.
- *Stability:* $\lim_{n\to\infty} Kx_n \to Kx$ implies $\lim_{n\to\infty} x_n \to x$.

Ill-Posed Inverse Problems:

Need for regularization!

Tikhonov Regularization

Standard Tikhonov Regularization:

Given an *ill-posed inverse problems* $Kx = y$, where $K : X \to Y$, an approximate solution $x^\alpha \in X$, $\alpha > 0$, can be determined by minimizing

$$J_\alpha(x) := \underbrace{\|Kx - y\|^2}_{\text{Data fidelity term}} + \alpha \cdot \underbrace{\|x\|^2}_{\text{Regularization Term}}, \quad x \in X.$$

Tikhonov Regularization

Standard Tikhonov Regularization:

Given an *ill-posed inverse problems* $Kx = y$, where $K : X \to Y$, an approximate solution $x^\alpha \in X$, $\alpha > 0$, can be determined by minimizing

$$J_\alpha(x) := \underbrace{\|Kx - y\|^2}_{\text{Data fidelity term}} + \alpha \cdot \underbrace{\|x\|^2}_{\text{Regularization Term}}, \quad x \in X.$$

Generalization:

$$J_\alpha(x) := \underbrace{\|Kx - y\|^2}_{\text{Data fidelity term}} + \alpha \cdot \underbrace{\mathcal{R}(x)}_{\text{Regularization Term}}, \quad x \in X.$$

The *Regularization Term* $\mathcal{R}$

- ensures continuous dependence on the data,
- incorporates properties of the solution.

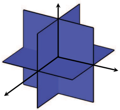
Paradigm for Data Processing: Sparsity!

Sparse Signals:

A signal $x \in \mathbb{R}^n$ is *k-sparse*, if

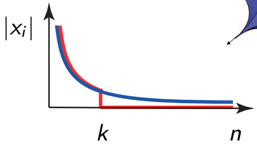
$$\|x\|_0 = \#\text{non-zero coefficients} \leq k.$$

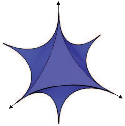
$\rightsquigarrow$ Model Σ_k: Union of k-dimensional subspaces



Compressible Signals:

A signal $x \in \mathbb{R}^n$ is *compressible*, if the sorted coefficients have rapid (power law) decay.

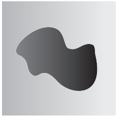




$\rightsquigarrow$ Model: ℓ_p ball with $p \leq 1$

Recall: Shearlets as Sparsifying System

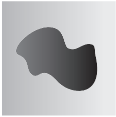
Model of Images (Donoho; 2001):
"*Cartoon-functions* are functions governed by a discontinuity curve."



Theorem (Kutyniok, Lim; 2011):
"Shearlets fulfill the *optimal compression rate* for cartoon-functions."

Recall: Shearlets as Sparsifying System

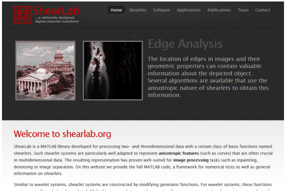
Model of Images (Donoho; 2001):
"*Cartoon-functions* are functions governed by a discontinuity curve."



> **Theorem (Kutyniok, Lim; 2011):**
> "Shearlets fulfill the *optimal compression rate* for cartoon-functions."

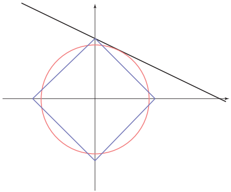
2D&3D (parallelized) Fast Shearlet Transform (`www.ShearLab.org`):

- Matlab *(Kutyniok, Lim, Reisenhofer; 2013)*
- Julia *(Loarca; 2017)*
- Python *(Look; 2018)*
- Tensorflow *(Loarca; 2019)*



How to Penalize Non-Sparsity?

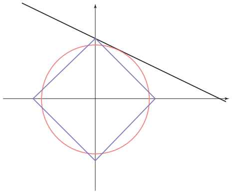
Intuition:



$\rightsquigarrow$ *Use the ℓ_1 norm!*

How to Penalize Non-Sparsity?

Intuition:



$\leadsto$ *Use the ℓ_1 norm!*

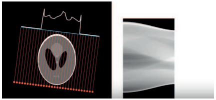
Sparse Regularization:

Solve an *ill-posed inverse problem* $Kf = g$ by

$$f^\alpha := \operatorname*{argmin}_f \Big[\underbrace{\|Kf - g\|^2}_{\text{Data fidelity term}} + \alpha \cdot \underbrace{\|(\langle f, \psi_{j,m} \rangle)_{j,m}\|_1}_{\text{Penalty term}} \Big].$$

Problem with Classical Approaches

(Limited Angle-) Computed Tomography



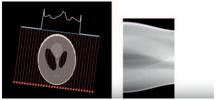
A CT scanner samples the *Radon transform*

$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x)dS(x),$$

for $L(\phi, s) = \{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

(Limited Angle-) Computed Tomography

A CT scanner samples the *Radon transform*



$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x) dS(x),$$

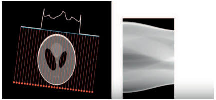
for $L(\phi, s) = \{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

Challenging inverse problem if $\mathcal{R}f(\cdot, s)$ is only sampled on $[-\phi, \phi] \subset [-\pi/2, \pi/2)$.

Applications: Dental CT, electron tomography,...

(Limited Angle-) Computed Tomography

A CT scanner samples the *Radon transform*



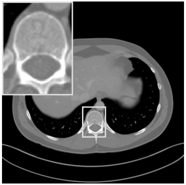
$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x)dS(x),$$

for $L(\phi, s) = \{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

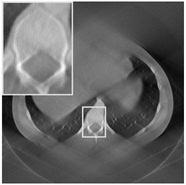
Challenging inverse problem if $\mathcal{R}f(\cdot, s)$ is only sampled on $[-\phi, \phi] \subset [-\pi/2, \pi/2)$.

Applications: Dental CT, electron tomography,...

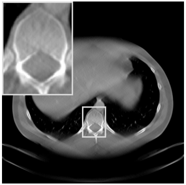
Model-Based Approaches Fail (60° Missing Angle):



Original Image



Filtered Backprojection



Sparse Regularization with Shearlets

Deep Learning Enters the Stage

Overview

Different Forms of Hybrid Approaches:

- *Supervised approaches:*
  - Train a neural network end-to-end.
  - Incorporate information about the operator K into the neural network.
  - Combine neural networks with classical model-based approaches (Plug-and-play, etc.)

Overview

Different Forms of Hybrid Approaches:

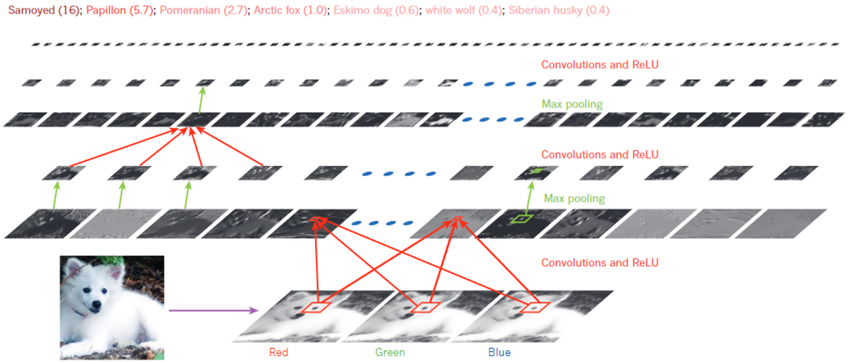
- *Supervised approaches:*
  - Train a neural network end-to-end.
  - Incorporate information about the operator K into the neural network.
  - Combine neural networks with classical model-based approaches (Plug-and-play, etc.)
- *Semi-supervised approaches:*
  - Encode the regularization by a neural network (Adversarial regularizers, etc.)
  - The learning algorithm only requires a set of labels as well as a method to assess how hard the inverse problem for this label would be.

Different Forms of Hybrid Approaches:

- *Supervised approaches:*
  - Train a neural network end-to-end.
  - Incorporate information about the operator K into the neural network.
  - Combine neural networks with classical model-based approaches (Plug-and-play, etc.)
- *Semi-supervised approaches:*
  - Encode the regularization by a neural network (Adversarial regularizers, etc.)
  - The learning algorithm only requires a set of labels as well as a method to assess how hard the inverse problem for this label would be.
- *Unsupervised approaches:*
  - Parametrize the solutions as the output of a neural network (Deep image priors, etc.)

Convolutional Neural Networks (CNNs)

Schematic Illustration:

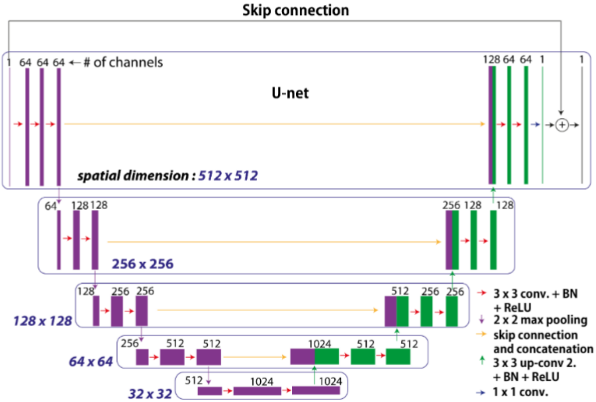


Operation in each Layer:

Input $\rightarrow$ Convolution $\rightarrow$ Activation $\rightarrow$ Pooling $\rightarrow$ Output

CNN Architecture for Inverse Problems

- U-Net architecture (Ronneberger et al.; 2015)
- Encoder-Decoder CNN with skip-connections



[Unser et al.,2017]

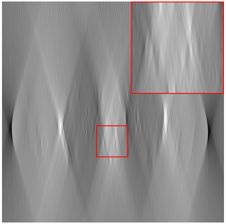
Models and Data

**How to take the best out of both worlds:
Models and Data?**

General Strategy:

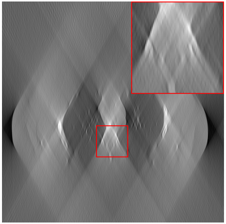
- Employ *model-based approaches* as far as they are reliable.
- Apply *deep learning* only when it is necessary.

Zooming in on the Limited-Angle CT Problem



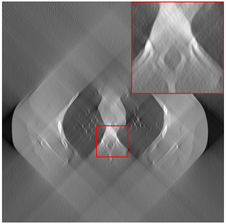
$\phi = 15°$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



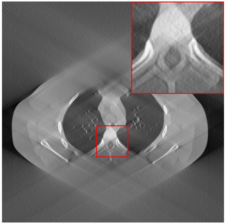
$\phi = 30°$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



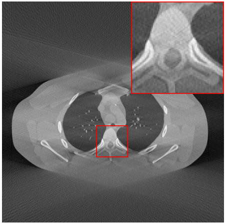
$\phi = 45^\circ$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



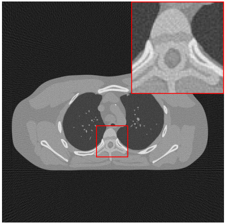
$\phi = 60^\circ$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



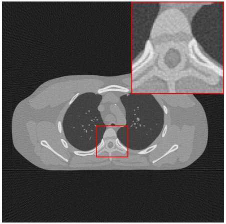
$\phi = 75^\circ$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



$\phi = 90°$, filtered backprojection (FBP)

Zooming in on the Limited-Angle CT Problem



$\phi = 90^\circ$, filtered backprojection (FBP)

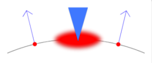
Illustration of Theorem [Quinto, 1993]:

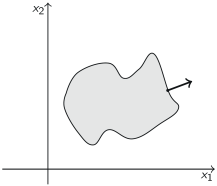
‘visible”: singularities tangent to sampled lines

“invisible”: singularities not tangent to sampled lines

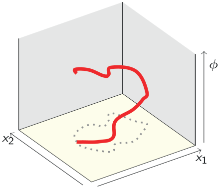
Shearlets can Help

Key Idea: Filling the missing angle is an inpainting problem of the wavefront set!



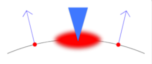


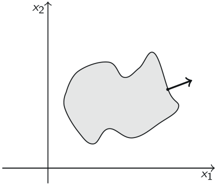
$f = 1_D$ for a set $D \subseteq \mathbb{R}^2$
with smooth boundary



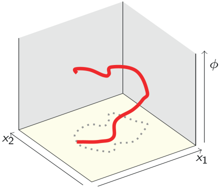
Shearlets can Help

Key Idea: Filling the missing angle is an inpainting problem of the wavefront set!





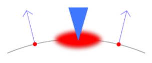
$f = 1_D$ for a set $D \subseteq \mathbb{R}^2$
with smooth boundary

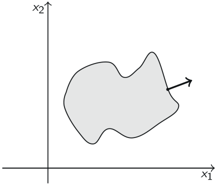


Theorem (Kutyniok, Labate; 2006):
"Shearlets can identify the wavefront set at fine scales."

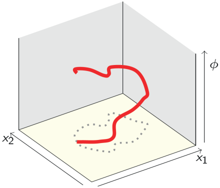
Shearlets can Help

Key Idea: Filling the missing angle is an inpainting problem of the wavefront set!



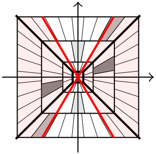


$f = 1_D$ for a set $D \subseteq \mathbb{R}^2$
with smooth boundary



Theorem (Kutyniok, Labate; 2006):
"Shearlets can identify the wavefront set at fine scales."

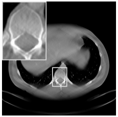
Shearlets can Separate the Visible and Invisible Part:



Our Approach "Learn the Invisible (LtI)"

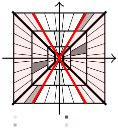
(Bubba, Kutyniok, Lassas, März, Samek, Siltanen, Srinivan; 2019)

Step 1: *Reconstruct the visible*



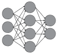
$$f^* := \operatorname*{argmin}_{f \geq 0} \| \mathcal{R}_\phi f - g\|_2^2 + \| \mathrm{SH}_\psi(f)\|_{1,w}$$

- Best available classical solution (little artifacts, denoised)
- Access "wavefront set" via sparsity prior on shearlets:
  - For $(j,k,l) \in \mathcal{I}_{\mathtt{inv}}$: $\mathrm{SH}_\psi(f^*)_{(j,k,l)} \approx 0$
  - For $(j,k,l) \in \mathcal{I}_{\mathtt{vis}}$: $\mathrm{SH}_\psi(f^*)_{(j,k,l)}$ reliable and near perfect



Step 2: *Learn the invisible*

$$\mathcal{NN}_\theta : \ \mathrm{SH}_\psi(f^*)_{\mathcal{I}_{\mathtt{vis}}} \longrightarrow F \ \left(\stackrel{!}{\approx} \mathrm{SH}_\psi(f_{\mathtt{gt}})_{\mathcal{I}_{\mathtt{inv}}} \right)$$

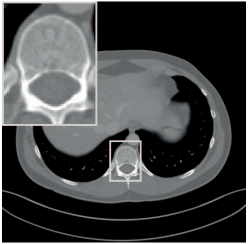


U-Net

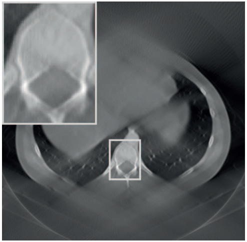
Step 3: *Combine*

$$f_{\mathtt{LtI}} = \mathrm{SH}_\psi^T \left(\mathrm{SH}_\psi(f^*)_{\mathcal{I}_{\mathtt{vis}}} + F\right)$$

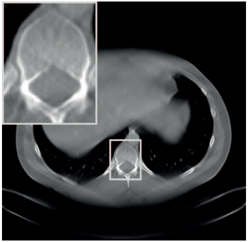
Numerical Results



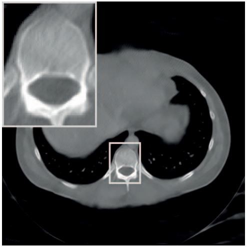
Original



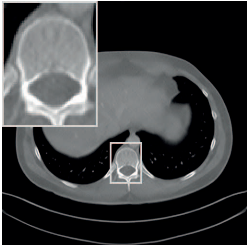
Filtered Backprojection



Sparse Regularization with Shearlets

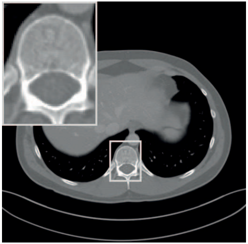


[Gu & Ye, 2017]

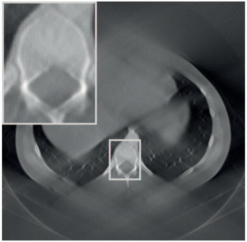


Learn the Invisible (LtI)

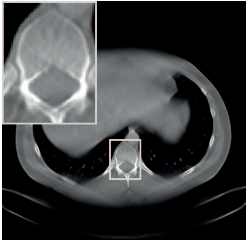
Numerical Results



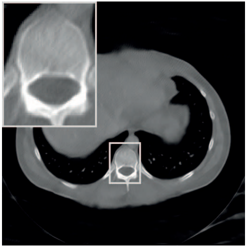
Original



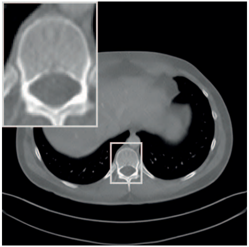
Filtered Backprojection



Sparse Regularization with Shearlets



[Gu & Ye, 2017]



Learn the Invisible (LtI)

Deep neural networks can outperform classical methods by far!

Deep Network Shearlet Edge Extractor (DeNSE)

(Andrade-Loarca, Kutyniok, Öktem, Petersen; 2019)

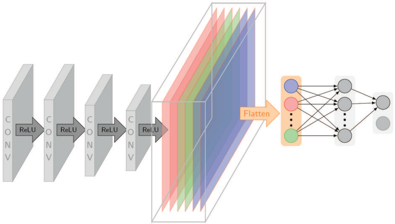
Key Steps:

(1) Apply the shearlet transform to an image.
  $\rightsquigarrow$ *Extract the correct features.*
  $\rightsquigarrow$ *Derive a good data representation.*

(2) Consider patches of shearlet coefficients.
  $\rightsquigarrow$ *Localize to each position.*

(3) Apply a convolutional neural network.
  $\rightsquigarrow$ *Predict the direction (180 directions) in each patch.*

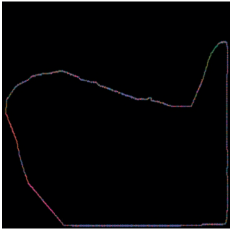
Network Architecture:



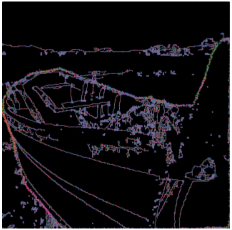
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Numerical Results

Original

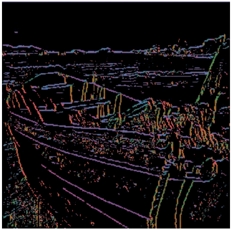


Human Annotation



SEAL [Yu et al; 2018]

CoShREM [Reisenhofer et al.; 2015]



DeNSE

Theoretically Analyzing the Effectiveness of Deep Neural Networks: Solving PDEs!

Another Mystery

Recall from Expressivity:

Deep neural networks match the performance of the best classical approximation tool in virtually every task!

Another Mystery

Recall from Expressivity:

Deep neural networks match the performance of the best classical approximation tool in virtually every task!

Surprise from Practise of Neural Networks:

- Perform incredibly well in approximating high-dimensional functions.
- Often outperform classical, non-specialized approximation methods.

Another Mystery

Recall from Expressivity:

Deep neural networks match the performance of the best classical approximation tool in virtually every task!

Surprise from Practise of Neural Networks:

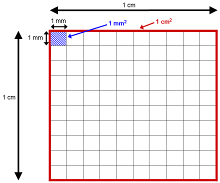
- Perform incredibly well in approximating high-dimensional functions.
- Often outperform classical, non-specialized approximation methods.

The Curse of Dimensionality:

Every approximation method deteriorates exponentially fast with increasing dimension!

Key Problem: The Curse!

"Introduction": Bellman; 1961



Curse of Dimensionality: 10^2 evenly spaced points suffice to sample a uni interval with no more than 10^{-2} distance between points. *But* an equivalent sampling of a 10-dimensional unit hypercube with a lattice of the same spacing would require $10^{20} = (10^2)^{10}$ sample points.
$\rightsquigarrow$ *Exponential growth*.

Examples:

- Combinatorics
- Function approximation
- Machine learning
- Numerical integration

Partial Differential Equations

Some Facts about PDE Solvers:

- Precise physical models exist.
- The discretization process is very well understood.
- Often optimal solvers are available.
- A rich bouquet of highly sophisticated solvers are developed:
  - Finite-element methods
  - Wavelet-based approaches
  - ...

Partial Differential Equations

Some Facts about PDE Solvers:

- Precise physical models exist.
- The discretization process is very well understood.
- Often optimal solvers are available.
- A rich bouquet of highly sophisticated solvers are developed:
  - Finite-element methods
  - Wavelet-based approaches
  - ...

Why do we need deep neural networks?

Partial Differential Equations

Some Facts about PDE Solvers:

- Precise physical models exist.
- The discretization process is very well understood.
- Often optimal solvers are available.
- A rich bouquet of highly sophisticated solvers are developed:
  - Finite-element methods
  - Wavelet-based approaches
  - ...

Why do we need deep neural networks?

$\rightsquigarrow$ Deep neural networks can *beat the curse of dimensionality in high dimensional problems*!

Deep Learning Approaches to PDEs

Common Approach to Solve PDEs with Neural Networks:

Approximate the solution u of a PDE

$$\mathcal{L}(u) = f$$

by a neural network Φ, i.e., determine

$$\mathcal{L}(\Phi) \approx f.$$

Key Ideas:

- Sampling of points in the spatial domain
- Incorporate PDE in the loss functions

Incomplete List of Contributions:

[Lagaris, Likas, Fotiadis; 1998], [E, Yu; 2017], [Czarnecki, Osindero, Jaderberg, Swirszcz, Pascanu; 2017], [Sirignano, Spiliopoulos; 2017], [Han, Jentzen, E; 2017], [Raissi, Perdikaris, Karniadakis; 2020], [Grohs, Herrmann; 2021], . . .

Let's Now Enter the World of Parametric PDEs

Why Parametric PDEs?

Parameter dependent families of PDEs arise in basically any branch of science and engineering.

Some Exemplary Problem Classes:

- Complex design problems
- Inverse problems
- Optimization tasks
- Uncertainty quantification
- ...

The number of parameters can be

- finite (physical properties such as domain geometry, ...)
- infinite (modeling of random stochastic diffusion field, ...)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

The Parametric Map

Example of Parametric Diffusion Equation:

The following parametric diffusion equation has the form

$$-\nabla \cdot (a_y(\mathbf{x}) \cdot \nabla u_y(\mathbf{x})) = f(\mathbf{x}), \qquad \text{on } \Omega = (0,1)^2, \quad u_y|_{\partial\Omega} = 0,$$

where $f \in L^2(\Omega)$ and $a_y \in L^\infty(\Omega)$ is a diffusion coefficient depending on a parameter $y \in \mathcal{Y}$.

Parametric Map:

Consider the map $\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto u_y$, where $p \in \mathbb{N}$, for various choices of parametrizations

$$\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto a_y.$$

The Parametric Map

Example of Parametric Diffusion Equation:
The following parametric diffusion equation has the form

$$-\nabla \cdot (a_y(\mathbf{x}) \cdot \nabla u_y(\mathbf{x})) = f(\mathbf{x}), \quad \text{on } \Omega = (0,1)^2, \quad u_y|_{\partial\Omega} = 0,$$

where $f \in L^2(\Omega)$ and $a_y \in L^\infty(\Omega)$ is a diffusion coefficient depending on a parameter $y \in \mathcal{Y}$.

Parametric Map:
Consider the map $\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto u_y$, where $p \in \mathbb{N}$, for various choices of parametrizations

$$\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto a_y.$$

General Form:

$$\mathcal{Y} \ni y \;\mapsto\; u_y \in \mathcal{H} \quad \text{such that} \quad \mathcal{L}(u_y, y) = f_y.$$

Curse of Dimensionality: Computational cost too high!

What can Deep Neural Networks do?

Parametric Map:

$$\mathbb{R}^p \supseteq \mathcal{Y} \ni y \mapsto \mathbf{u}_y^{\mathrm{h}} \in \mathbb{R}^D \quad \text{such that} \quad b_y\left(u_y^h, v\right) = f_y(v) \text{ for all } v.$$

Can a neural network approximate the parametric map?

What can Deep Neural Networks do?

Parametric Map:

$$\mathbb{R}^p \supseteq \mathcal{Y} \ni y \mapsto \mathbf{u}_y^{\mathrm{h}} \in \mathbb{R}^D \quad \text{such that} \quad b_y\left(u_y^h, v\right) = f_y(v) \text{ for all } v.$$

Can a neural network approximate the parametric map?

Advantages:

- After training, extremely rapid computation of the map.
- Flexible, universal approach.

Questions: Let $\epsilon > 0$.

(1) Does *there exist a neural network* Φ such that

$$\|\Phi(y) - \mathbf{u}_y^{\mathrm{h}}\| \leq \epsilon \qquad \text{for all } y \in \mathcal{Y}?$$

(2) How does the *complexity of* Φ depend on p and D?

(3) How do neural networks *perform numerically* on this task?

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Theoretical Results

Theorem (Kutyniok, Petersen, Raslan, Schneider; 2021):

- There exists a neural network Φ which approximates the parametric map:
$$\|\Phi(y) - \mathbf{u}_y^{\mathrm{h}}\| \leq \epsilon \qquad \text{for all } y \in \mathcal{Y}.$$
- The dependence of $C(\Phi)$ on p and D can be (polynomially) controlled.

Theoretical Results

Theorem (Kutyniok, Petersen, Raslan, Schneider; 2021):

- There exists a neural network Φ which approximates the parametric map:
$$\|\Phi(y) - \mathbf{u}_y^{\mathrm{h}}\| \leq \epsilon \qquad \text{for all } y \in \mathcal{Y}.$$
- The dependence of $C(\Phi)$ on p and D can be (polynomially) controlled.

Proof:

- Consider the reduced basis method.
- Approximate the solution derived now by a neural network.
- This requires approximating multiplication and inversion of matrices.
- Monitor the complexity of this network.

Theoretical Results

Theorem (Kutyniok, Petersen, Raslan, Schneider; 2021):

- There exists a neural network Φ which approximates the parametric map:
$$\|\Phi(y) - \mathbf{u}_y^{\mathrm{h}}\| \leq \epsilon \qquad \text{for all } y \in \mathcal{Y}.$$
- The dependence of $C(\Phi)$ on p and D can be (polynomially) controlled.

Proof:

- Consider the reduced basis method.
- Approximate the solution derived now by a neural network.
- This requires approximating multiplication and inversion of matrices.
- Monitor the complexity of this network.

Do neural networks also beat the curse when trained?

Test Set-Up for Numerical Experiments

Parametric Diffusion Equation:

We will consider the following parametric diffusion equation:

$$-\nabla \cdot (a_y(\mathbf{x}) \cdot \nabla u_y(\mathbf{x})) = f(\mathbf{x}), \quad \text{on } \Omega = (0,1)^2, \quad u_y|_{\partial\Omega} = 0,$$

where $f \in L^2(\Omega)$ and $a_y \in L^\infty(\Omega)$ is a diffusion coefficient depending on a parameter $y \in \mathcal{Y}$.

Parametric Map:

We learn a discretization of the map $\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto u_y$, where $p \in \mathbb{N}$, for various choices of parametrizations

$$\mathbb{R}^p \supset \mathcal{Y} \ni y \mapsto a_y.$$

What We Vary...

- Type of parametrization
- Dimension of parameter space
- Complexity of hyper-parameters

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Parametric Diffusion Equation

Parametric Diffusion Equation:

$$-\nabla \cdot (a(\mathbf{x}) \cdot \nabla u_a(\mathbf{x})) = f(\mathbf{x}), \quad \text{on } \Omega = (0,1)^2, \quad u|_{\partial\Omega} = 0,$$

where

$$a \in \mathcal{A} = \{a_y : \ y \in \mathcal{Y}\} \subset L^\infty(\Omega) \quad \text{and} \quad f(x) = 20 + 10x_1 - 5x_2.$$

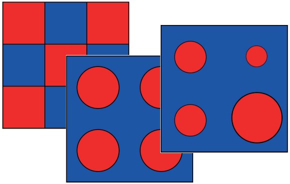
Affine Parametrization: For fixed functions $(a_i)_{i=0}^p \subset L^\infty(\Omega)$,

$$\mathcal{A} = \left\{ a_y = a_0 + \sum_{i=1}^p y_i a_i : y = (y_i)_{i=1}^p \in \mathcal{Y} \right\}.$$

- Trigonometric polynomials
- Chessboard partition
- Cookies with fixed radii

Non-Affine Parametrization:

- Cookies with variable radii
- Clipped polynomials



Further Set-Up

Finite Element Space:

- $\Omega = [0,1]^2$ with 101×101 equidistant grid points

Fixed Neural Network:

- $(p, 300, \ldots, 300, 10201)$ with $L = 11$ layers
- Activation function: 0.2-LReLU.

Fixed Training Procedure:

- Training set: 20000 i.i.d. parameter samples
- Neural network: Initialized according to a normal distribution with mean 0 and standard deviation 0.1
- Loss function: Relative error on the finite-element discretization of $\mathcal{H}$
- Optimization: Batch gradient descent

Dimension:

- Various dimensions of the parameter set up to 91.

Numerical Experiments, I

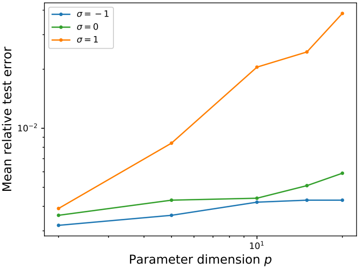
Trigonometric Polynomials:

$$\mathcal{A}^{\mathrm{tp}}(p,\sigma) := \left\{ \mu + \sum_{i=1}^{p} y_i \cdot i^{\sigma} \cdot (1 + a_i) : \ y \in \mathcal{Y} = [0,1]^p \right\},$$

for some fixed shift $\mu > 0$, scaling coefficient $\sigma \in \mathbb{R}$, and

$$a_i(\mathbf{x}) = \sin\left(\left\lfloor \frac{i+2}{2} \right\rfloor \pi x_1 \right) \sin\left(\left\lceil \frac{i+2}{2} \right\rceil \pi x_2 \right), \quad \text{for } i = 1, \dots, p.$$

Numerical Results:



Source: Geist, Petersen, Raslan, Schneider, Kutyniok. Numerical Solution of the Parametric Diffusion Equation by Deep Neural Networks. J. Sci. Comput., to appear.

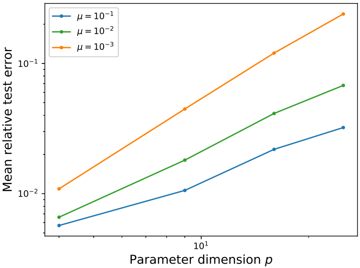
Numerical Experiments, II

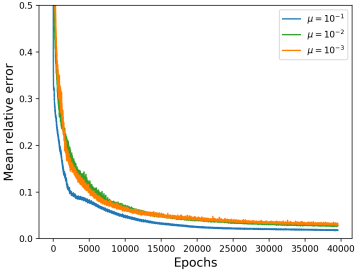
Chessboard Partition: Let $p = s^2$ for some $s \in \mathbb{N}$. Then

$$\mathcal{A}^{\mathrm{cb}}(p,\mu) := \left\{ \mu + \sum_{i=1}^{p} y_i \mathcal{X}_{\Omega_i} : \; y \in \mathcal{Y} = [0,1]^p \right\},$$

where $(\Omega_i)_{i=1}^p$ forms a $s \times s$ chessboard partition of $(0,1)^2$ and $\mu > 0$ is a fixed shift.

Numerical Results:





$p = 25$

Source: Geist, Petersen, Raslan, Schneider, Kutyniok. Numerical Solution of the Parametric Diffusion Equation by Deep Neural Networks. J. Sci. Comput., to appear.

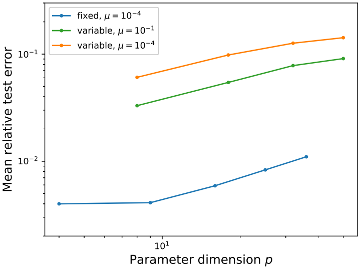
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

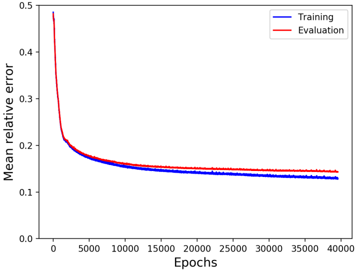
Numerical Experiments, III

Cookies with Variable Radii: For $s \in \mathbb{N}$ and every $i = 1, \ldots, s$, we are given disks $\Omega_{i,y_{i+s^2}}$ with centers $((2k+1)/(2s), (2\ell-1)/(2s))$, where $i = ks + \ell$ for uniquely determined $k \in \{0, \ldots s-1\}$ and $\ell \in \{1, \ldots, s\}$ and radius $y_{i+s^2}/(2s)$:

$$\mathcal{A}^{\mathrm{cvr}}(p, \mu) := \left\{ \mu + \sum_{i=1}^{p} y_i \mathcal{X}_{\Omega_{i,y_{i+s^2}}} \;:\; y \in \mathcal{Y} = [0,1]^p \times [0.5, 0.9]^p \right\}.$$

Numerical Results:





$p = 50$ and $\mu = 10^{-4}$

Source: Geist, Petersen, Raslan, Schneider, Kutyniok. Numerical Solution of the Parametric Diffusion Equation by Deep Neural Networks. J. Sci. Comput., to appear.

Interpretation

Hypotheses and Results:

- *The performance does not suffer from the curse of dimensionality.*
  - *True*, we never observed an exponential scaling.

Interpretation

Hypotheses and Results:

- *The performance does not suffer from the curse of dimensionality.*
  - *True*, we never observed an exponential scaling.
- *The performance is very sensitive to parametrization.*
  - *True*, there are strong differences in the performance.
  - More complex parametrized sets yield higher errors, whereas simpler sets or spaces with intuitively lower intrinsic dimensionality yield smaller errors.

$\rightsquigarrow$ *The approximation theoretical intrinsic dimension of the parametric problem is a main factor in determining the hardness!*

Interpretation

Hypotheses and Results:

- *The performance does not suffer from the curse of dimensionality.*
  - *True*, we never observed an exponential scaling.
- *The performance is very sensitive to parametrization.*
  - *True*, there are strong differences in the performance.
  - More complex parametrized sets yield higher errors, whereas simpler sets or spaces with intuitively lower intrinsic dimensionality yield smaller errors.

$\rightsquigarrow$ *The approximation theoretical intrinsic dimension of the parametric problem is a main factor in determining the hardness!*

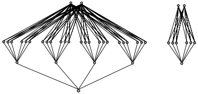
- *Learning is efficient also for non-affinely parametrized problems.*
  - *True*, there is no fundamental difference of the performance for non-affinely parametrized problems.

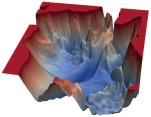
Some Final Thoughts...

Conclusions

Artificial Intelligence:

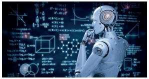
- *Impressive performance* in real-world applications!
- A *theoretical foundation* of is largely missing!





Mathematics for Deep Learning:

- *Expressivity*: Optimal architectures?
- *Learning*: Controllable, efficient algorithms?
- *Generalization*: Performance on test data sets?
- *Explainability*: Explaining network decisions?



Deep Learning for Mathematical/Physical Problem Settings:

- Significantly better solvers of *inverse problems*.
- Beating the curse of dimensionality for *partial differential equations*.

THANK YOU!

References available at:

`www.ai.math.lmu.de/kutyniok`

Survey Paper (arXiv:2105.04026):

Berner, Grohs, Kutyniok, Petersen, *The Modern Mathematics of Deep Learning*.

Check related information on Twitter at:

`@GittaKutyniok`

Upcoming Book:

- P. Grohs and G. Kutyniok, eds.
  Mathematical Aspects of Deep Learning
  Cambridge University Press, 2022.