An Information Theoretic Approach to Validate Deep Learning-Based Algorithms

Gitta Kutyniok

(Technische Universität Berlin and University of Tromsø)

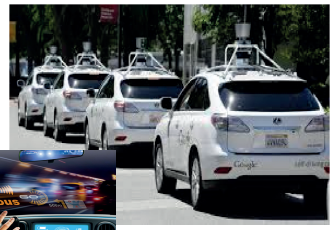
joint with

Stephan Wäldchen, Jan Macdonald, and Sascha Hauch (TU Berlin)

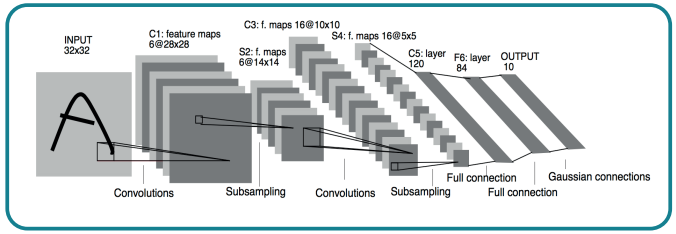
Workshop on Validation and Guarantees in Learning Physical Models
IPAM, Los Angeles, October 28 – November 1, 2019

The Dawn of Deep Learning

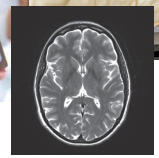
Self-Driving Cars



Surveillance



Legal Issues

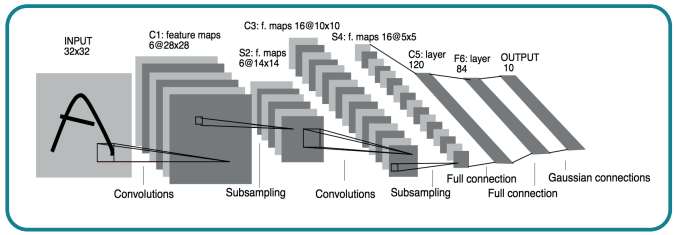


Health Care

The Dawn of Deep Learning

Self-Driving Cars

Surveillance



Legal Issues

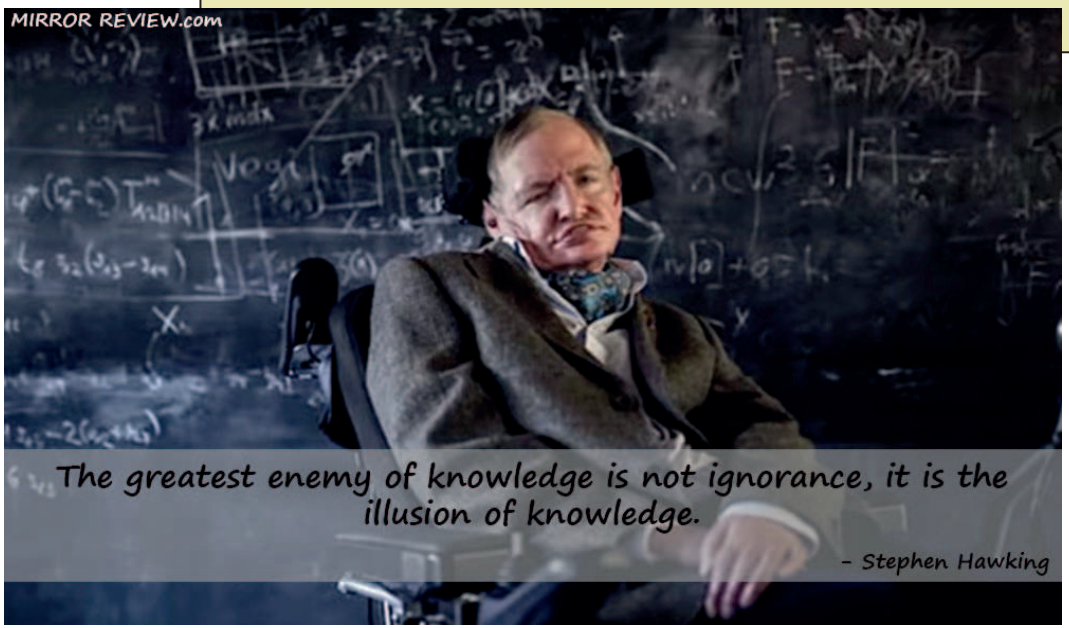
Health Care

Very few theoretical results explaining their success!

Deep Learning = Alchemy?

„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**...."

Science, May 2018

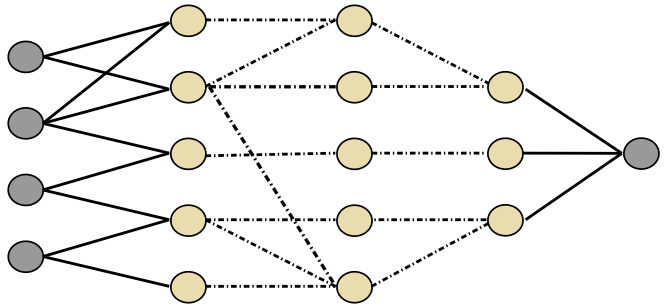


Theoretical Foundations of Deep Learning

The Mathematics of Deep Neural Networks

Definition:

Assume the following notions:



- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- N: Number of neurons.
- $\rho : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$: Affine linear maps.

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

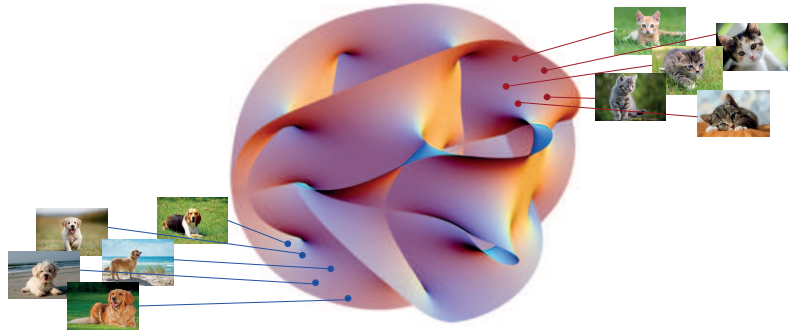
$$\Phi(x) = T_L \rho(T_{L-1} \rho(\ldots \rho(T_1(x))), \quad x \in \mathbb{R}^d,$$

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

Training of Deep Neural Networks

High-Level Set Up:

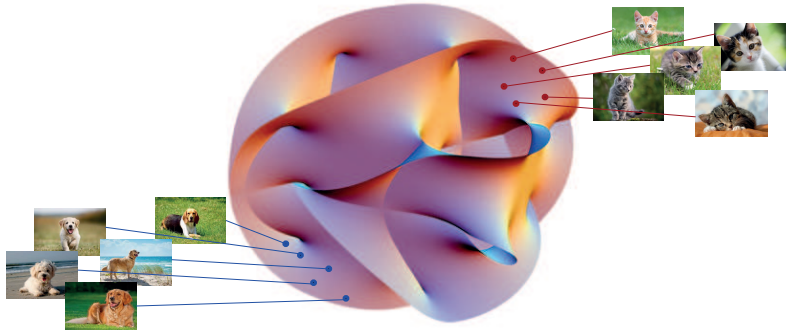
- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.



Training of Deep Neural Networks

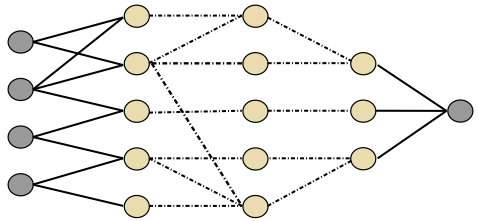
High-Level Set Up:

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



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

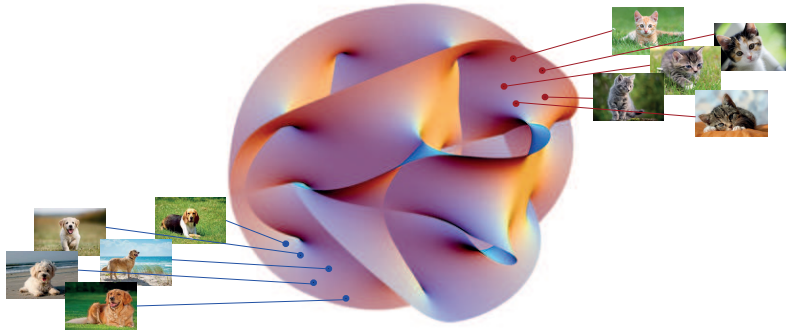
  Sometimes selected entries of the matrices $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



Training of Deep Neural Networks

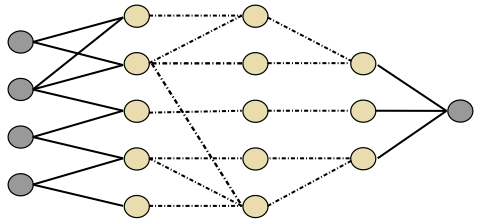
High-Level Set Up:

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



- 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 $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



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

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

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

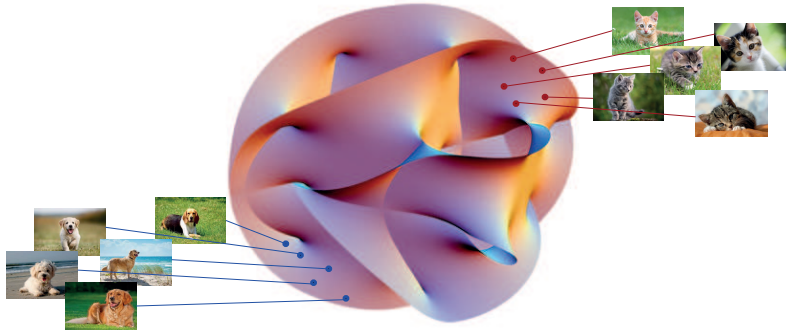
$$\Phi_{(A_\ell, b_\ell)_\ell}(x) = T_L \rho(T_{L-1} \rho(\dots \rho(T_1(x)))).$$

This is often done by stochastic gradient descent.

Training of Deep Neural Networks

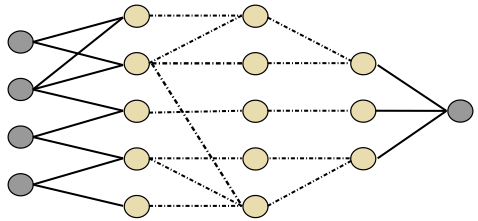
High-Level Set Up:

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



- 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 $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



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

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

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

$$\Phi_{(A_\ell, b_\ell)_\ell}(x) = T_L \rho(T_{L-1} \rho(\dots \rho(T_1(x))).$$

This is often done by stochastic gradient descent.

Goal: $\Phi_{(A_\ell, b_\ell)_\ell} \approx f$

Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

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

Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

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

Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

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

Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Interpretability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

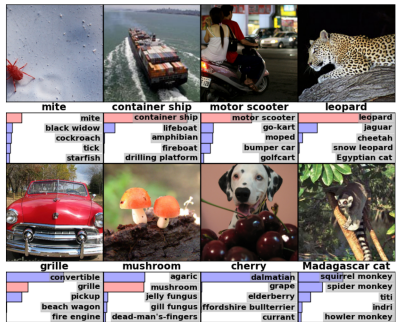
  $\rightsquigarrow$ *Information Theory, Uncertainty Quantification, ...*

Interpretability

Origin of Interpretability

Classification as a Classical Task for Neural Networks:

- Which features are most relevant for the decision?
  - Treat every pixel separately
  - Consider combinations of pixels
  - Incorporate additional knowledge
- How certain is the decision?



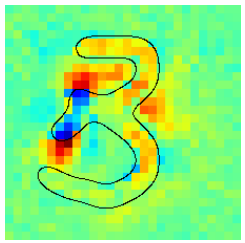
Origin of Interpretability

Classification as a Classical Task for Neural Networks:

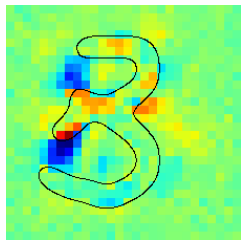
- Which features are most relevant for the decision?
  - Treat every pixel separately
  - Consider combinations of pixels
  - Incorporate additional knowledge
- How certain is the decision?

Illustration of a Relevance Map:

map for digit 3



map for digit 8



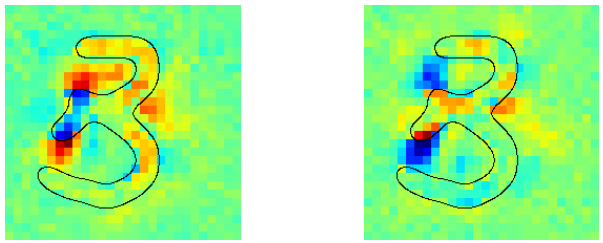
Origin of Interpretability

Classification as a Classical Task for Neural Networks:

- Which features are most relevant for the decision?
  - Treat every pixel separately
  - Consider combinations of pixels
  - Incorporate additional knowledge
- How certain is the decision?

Illustration of a Relevance Map:

map for digit 3 map for digit 8



Does this provide sufficient knowledge about a network decision?

History of the Field

Previous Relevance Mapping Methods:

- Gradient based methods:
  - Sensitivity Analysis (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller, 2010)
  - SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017)

History of the Field

Previous Relevance Mapping Methods:

- Gradient based methods:
  - Sensitivity Analysis (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller, 2010)
  - SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017)
- Backwards propagation based methods:
  - Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015)
  - Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015)
  - Deep Taylor (Montavon, Samek, Müller, 2018)

History of the Field

Previous Relevance Mapping Methods:

- Gradient based methods:
  - Sensitivity Analysis (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller, 2010)
  - SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017)
- Backwards propagation based methods:
  - Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015)
  - Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015)
  - Deep Taylor (Montavon, Samek, Müller, 2018)
- Surrogate model based methods:
  - LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro, Singh, Guestrin, 2016)

History of the Field

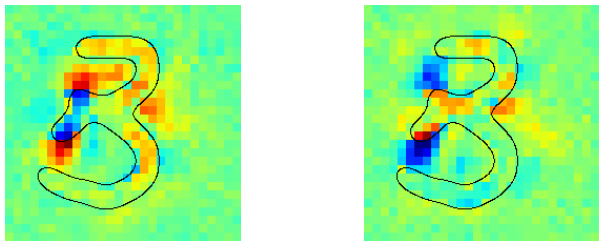
Previous Relevance Mapping Methods:

- Gradient based methods:
  - Sensitivity Analysis (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller, 2010)
  - SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017)
- Backwards propagation based methods:
  - Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015)
  - Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015)
  - Deep Taylor (Montavon, Samek, Müller, 2018)
- Surrogate model based methods:
  - LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro, Singh, Guestrin, 2016)
- Game theoretic methods:
  - Shapley values (Shapley, 1953), (Kononenko, Štrumbelj, 2010)
  - SHAP (Shapley Additive Explanations) (Lundberg, Lee, 2017)

What is Relevance?

Main Goal: We aim to understand decisions of "black-box" predictors!

map for digit 3 map for digit 8



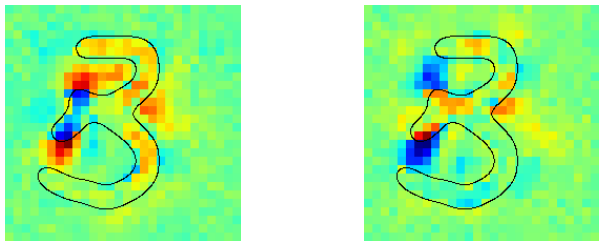
Challenges:

- What exactly is relevance in a mathematical sense?
- What is a good relevance map?
- How to compare different relevance maps?

What is Relevance?

Main Goal: We aim to understand decisions of "black-box" predictors!

map for digit 3 map for digit 8



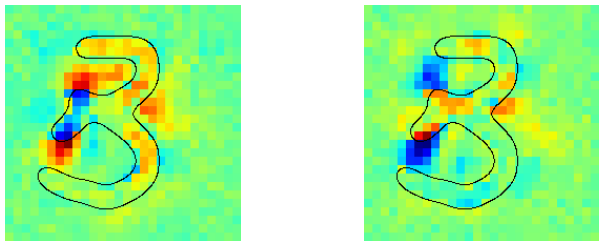
Challenges:

- What exactly is relevance in a mathematical sense?
  $\rightsquigarrow$ *Rigorous definition of relevance by information theory.*
- What is a good relevance map?
- How to compare different relevance maps?

What is Relevance?

Main Goal: We aim to understand decisions of "black-box" predictors!

map for digit 3 map for digit 8



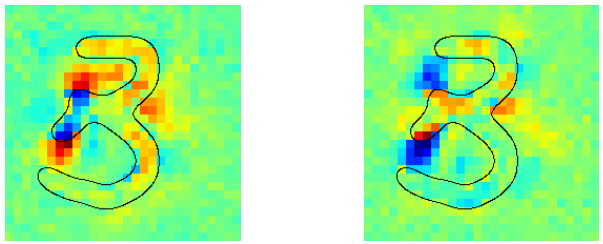
Challenges:

- What exactly is relevance in a mathematical sense?
  $\rightsquigarrow$ *Rigorous definition of relevance by information theory.*
- What is a good relevance map?
  $\rightsquigarrow$ *Formulation of interpretability as optimization problem.*
  $\rightsquigarrow$ *Theoretical analysis of complexity.*
- How to compare different relevance maps?

What is Relevance?

Main Goal: We aim to understand decisions of "black-box" predictors!

map for digit 3 map for digit 8



Challenges:

- What exactly is relevance in a mathematical sense?
  $\rightsquigarrow$ *Rigorous definition of relevance by information theory.*
- What is a good relevance map?
  $\rightsquigarrow$ *Formulation of interpretability as optimization problem.*
  $\rightsquigarrow$ *Theoretical analysis of complexity.*
- How to compare different relevance maps?
  $\rightsquigarrow$ *Canonical framework for comparison.*

The Relevance Mapping Problem

The Relevance Mapping Problem

The Setting: Let

- $\Phi\colon [0,1]^d \to [0,1]$ be a classification function,
- $x \in [0,1]^d$ be an input signal.

$$\xrightarrow{\Phi} \quad \Phi(x) = 0.97 \qquad \text{“Monkey”}$$

$$\xrightarrow{\Phi} \quad \Phi(x) = 0.07 \qquad \text{“Not a monkey”}$$

The Relevance Mapping Problem

The Task:

- Determine the most relevant components of x for the prediction $\Phi(x)$.
- Choose $S \subseteq \{1, \ldots, d\}$ of components that are considered relevant.
- S should be small (usually not everything is relevant).
- S^c is considered non-relevant.

Original image x

Relevant components S

Non-relevant components S^c

Rate-Distortion Viewpoint

$\Phi(x) = 0.97$

"Monkey"

Original image x

Partial image S

Random completion y

$\Phi(y) = 0.91$

"Monkey"

Obfuscation: Let

- $n \sim \mathcal{V}$ be a random noise vector, and
- y be a random vector defined as $y_S = x_S$ and $y_{S^c} = n_{S^c}$.

Rate-Distortion Viewpoint

Recall: Let

- $\Phi\colon [0,1]^d \to [0,1]$ be a classification function,
- $x \in [0,1]^d$ be an input signal,
- $n \sim \mathcal{V}$ be a random noise vector, and
- y be a random vector defined as $y_S = x_S$ and $y_{S^c} = n_{S^c}$.

Expected Distortion:

$$D(S) = D(\Phi, x, S) = \mathbb{E}\left[\frac{1}{2}\left(\Phi(x) - \Phi(y)\right)^2\right]$$

Rate-Distortion Function:

$$R(\epsilon) = \min_{S \subseteq \{1,\dots,d\}} \{|S| \,:\, D(S) \leq \epsilon\}$$

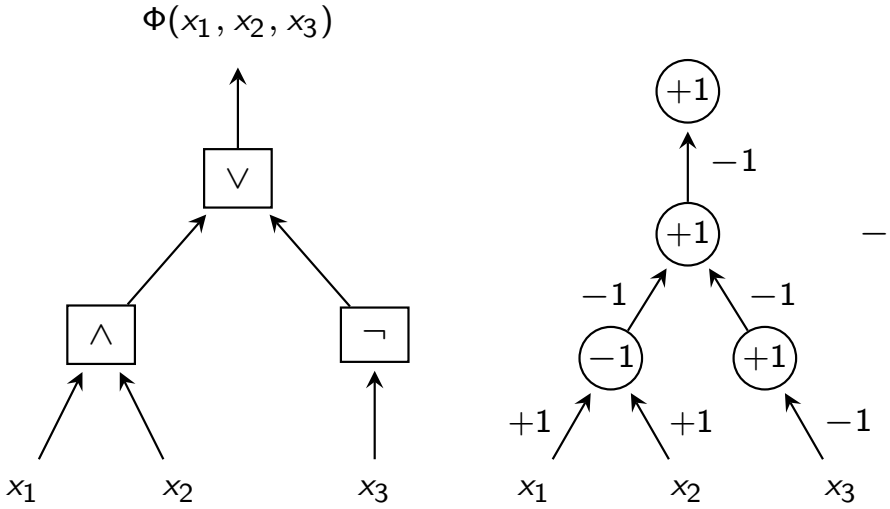
$\rightsquigarrow$ *Use this viewpoint for the definition of a relevance map!*

Finding a minimizer of $R(\epsilon)$

or even approximating it is very hard!

Hardness Results

Boolean Functions as ReLU Neural Networks:

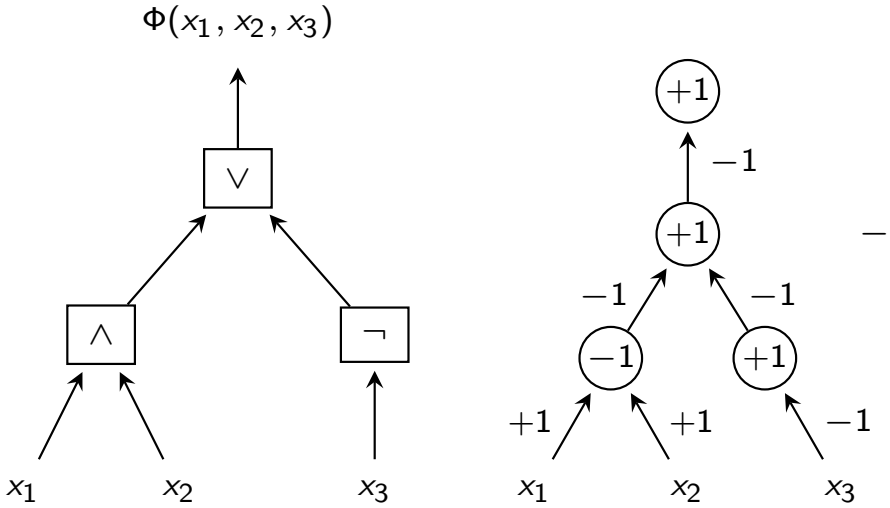


ReLU activation function $\varrho(x) = \max\{0, x\}$

$$-\varrho\left(\begin{bmatrix}-1 & -1\end{bmatrix}\varrho\left(\begin{bmatrix}1 & 1 & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}-1\\ 1\end{bmatrix}\right)+1\right)+1$$

Hardness Results

Boolean Functions as ReLU Neural Networks:



ReLU activation function $\varrho(x) = \max\{0, x\}$

$$-\varrho\left(\begin{bmatrix}-1 & -1\end{bmatrix}\varrho\left(\begin{bmatrix}1 & 1 & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}-1\\ 1\end{bmatrix}\right)+1\right)+1$$

The Binary Setting: Let

- $\Phi\colon \{0,1\}^d \to \{0,1\}$ be classifier functions,
- $x \in \{0,1\}^d$ be signals, and
- $\mathcal{V} = \mathcal{U}(\{0,1\}^d)$ be a uniform distribution.

Hardness Results

We consider the binary case.

Theorem (Wäldchen, Macdonald, Hauch, K, 2019):
Given Φ, x, $k \in \{1, \dots, d\}$, and $\epsilon < \frac{1}{4}$. Deciding whether $R(\epsilon) \leq k$ is $\mathrm{NP}^{\mathrm{PP}}$-complete.

Finding a minimizer of $R(\epsilon)$ is hard!

Theorem (Wäldchen, Macdonald, Hauch, K, 2019):
Given Φ, x, and $\alpha \in (0, 1)$. Approximating $R(\epsilon)$ to within a factor of $d^{1-\alpha}$ is NP-hard.

Even the approximation problem of it is hard!

Preprint: "The Computational Complexity of Understanding Network Decisions", https://arxiv.org/abs/1905.09163

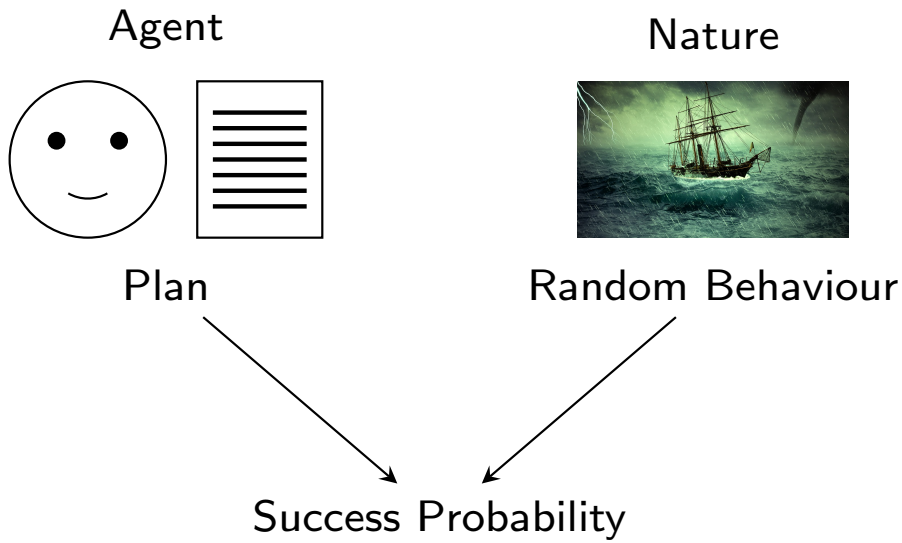
What is NP^{PP}?

The Complexity Class NP^{PP}:

Many important problems in artificial intelligence belong to this class.

Some Examples:

- Planning under uncertainties
- Finding maximum a-posteriori configurations in graphical models
- Maximizing utility functions in Bayesian networks



Our Method:

Rate-Distortion Explanation (RDE)

Problem Relaxation

	Discrete problem	Continuous problem
Relevant set	$S \subseteq \{1, \ldots, d\}$	
Obfuscation	$y_S = x_S$, $y_{S^c} = n_{S^c}$	
Distortion	$D(S)$	
Rate/Size	$\lvert S \rvert$	

Problem Relaxation

	Discrete problem	Continuous problem
Relevant set	$S \subseteq \{1, \dots, d\}$	$s \in [0,1]^d$
Obfuscation	$y_S = x_S,\ y_{S^c} = n_{S^c}$	$y = s \odot x + (1-s) \odot n$
Distortion	$D(S)$	$D(s)$
Rate/Size	$\lvert S \rvert$	$\lVert s \rVert_1$

Problem Relaxation

	Discrete problem	Continuous problem
Relevant set	$S \subseteq \{1, \dots, d\}$	$s \in [0,1]^d$
Obfuscation	$y_S = x_S,\ y_{S^c} = n_{S^c}$	$y = s \odot x + (1-s) \odot n$
Distortion	$D(S)$	$D(s)$
Rate/Size	$\lvert S \rvert$	$\lVert s \rVert_1$

Resulting Minimization Problem:

$$\text{minimize} \quad D(s) + \lambda \|s\|_1 \quad \text{subject to} \quad s \in [0,1]^d$$

Observations

Distortion:

$$D(s) = \mathbb{E}\left[\frac{1}{2}\left(\Phi(x) - \Phi(y)\right)^2\right]$$
$$= \frac{1}{2}\left(\Phi(x) - \mathbb{E}\left[\Phi(y)\right]\right)^2 + \frac{1}{2}\operatorname{cov}\left[\Phi(y)\right]$$

Obfuscation:

$$\mathbb{E}\left[y\right] = s \odot x + (1 - s) \odot \mathbb{E}\left[n\right]$$
$$\operatorname{cov}\left[y\right] = \operatorname{diag}(1 - s)\operatorname{cov}\left[n\right]\operatorname{diag}(1 - s)$$

Observations

$$\mathbb{E}[y], \mathrm{cov}[y] \xrightarrow{\Phi} \mathbb{E}[\Phi(y)], \mathrm{cov}[\Phi(y)]$$

Observations

$$\mathbb{E}[y], \mathrm{cov}[y] \xrightarrow{\Phi} \mathbb{E}[\Phi(y)], \mathrm{cov}[\Phi(y)]$$

Generic Approach:

- Estimate using sample mean and sample covariance
- Possible for any classifier function Φ
- Might require large number of samples

Observations

$$\mathbb{E}[y], \operatorname{cov}[y] \xrightarrow{\Phi} \mathbb{E}[\Phi(y)], \operatorname{cov}[\Phi(y)]$$

Generic Approach:

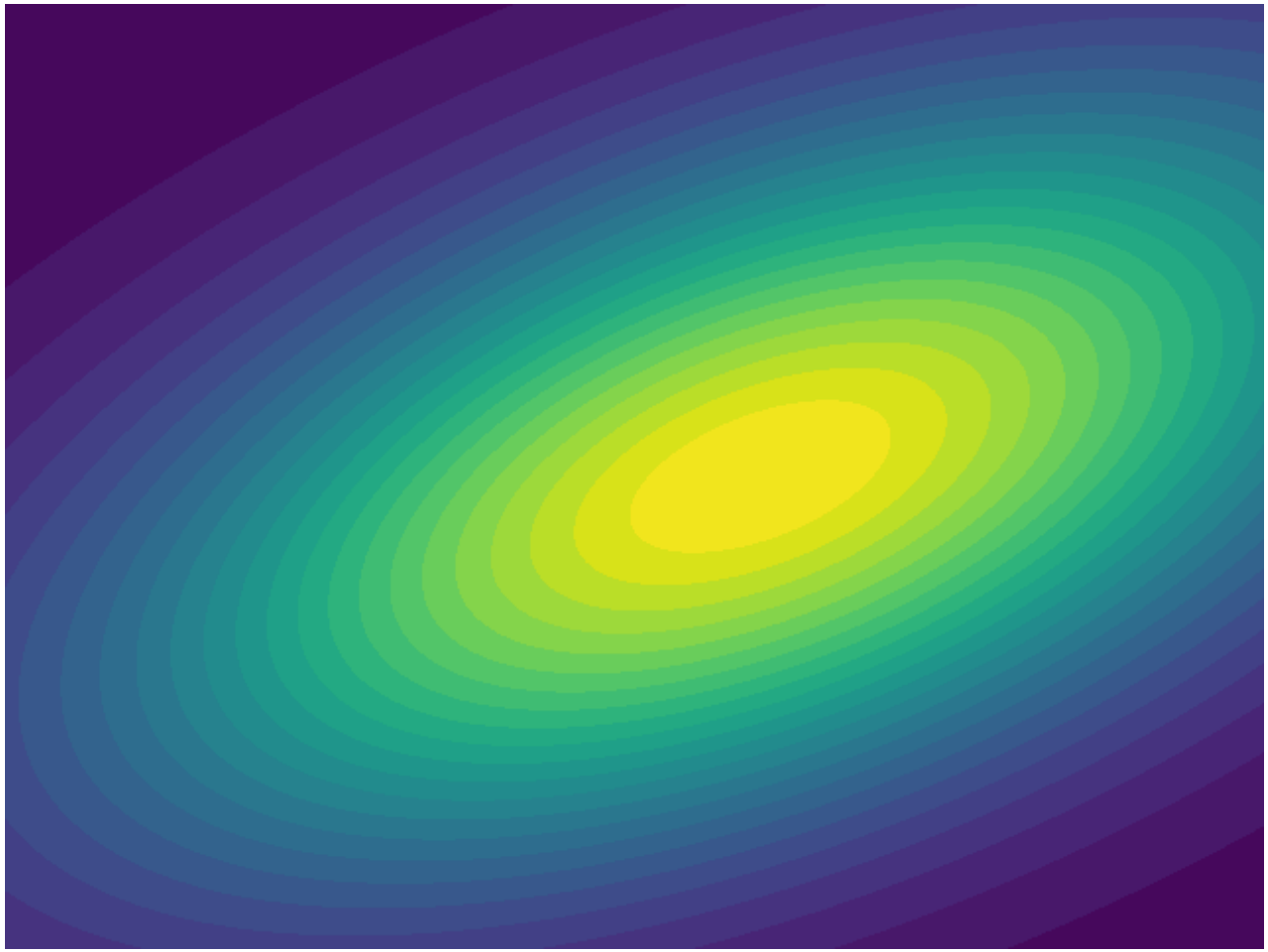
- Estimate using sample mean and sample covariance
- Possible for any classifier function Φ
- Might require large number of samples

Neural Network Approach:

- Use compositional structure of Φ
- Propagate distribution through the layers
- Project to simple family of distributions at each layer

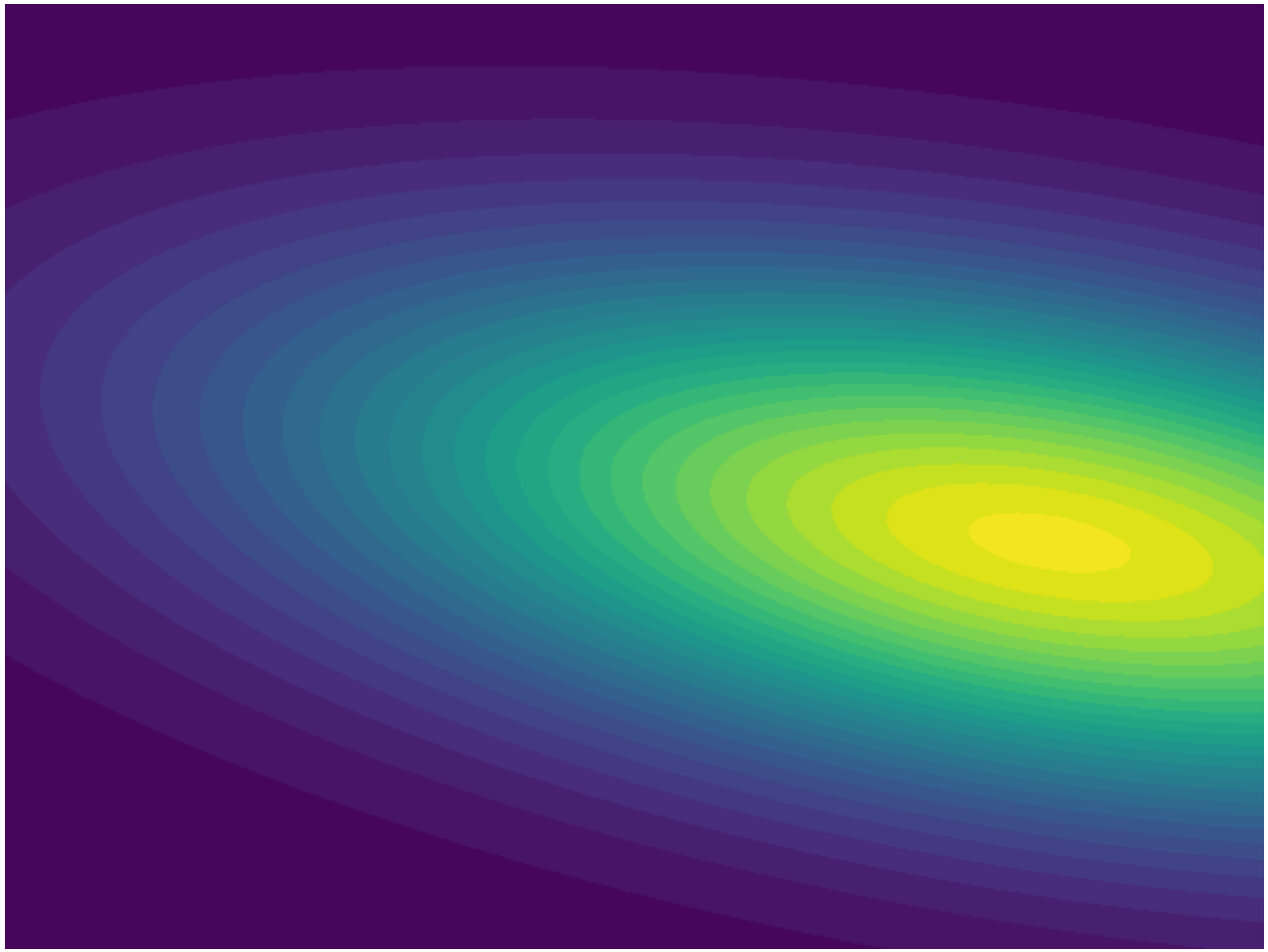
Assumed Density Filtering

Input distribution: $\mathcal{N}(\mu_{\text{in}}, \sigma_{\text{in}})$



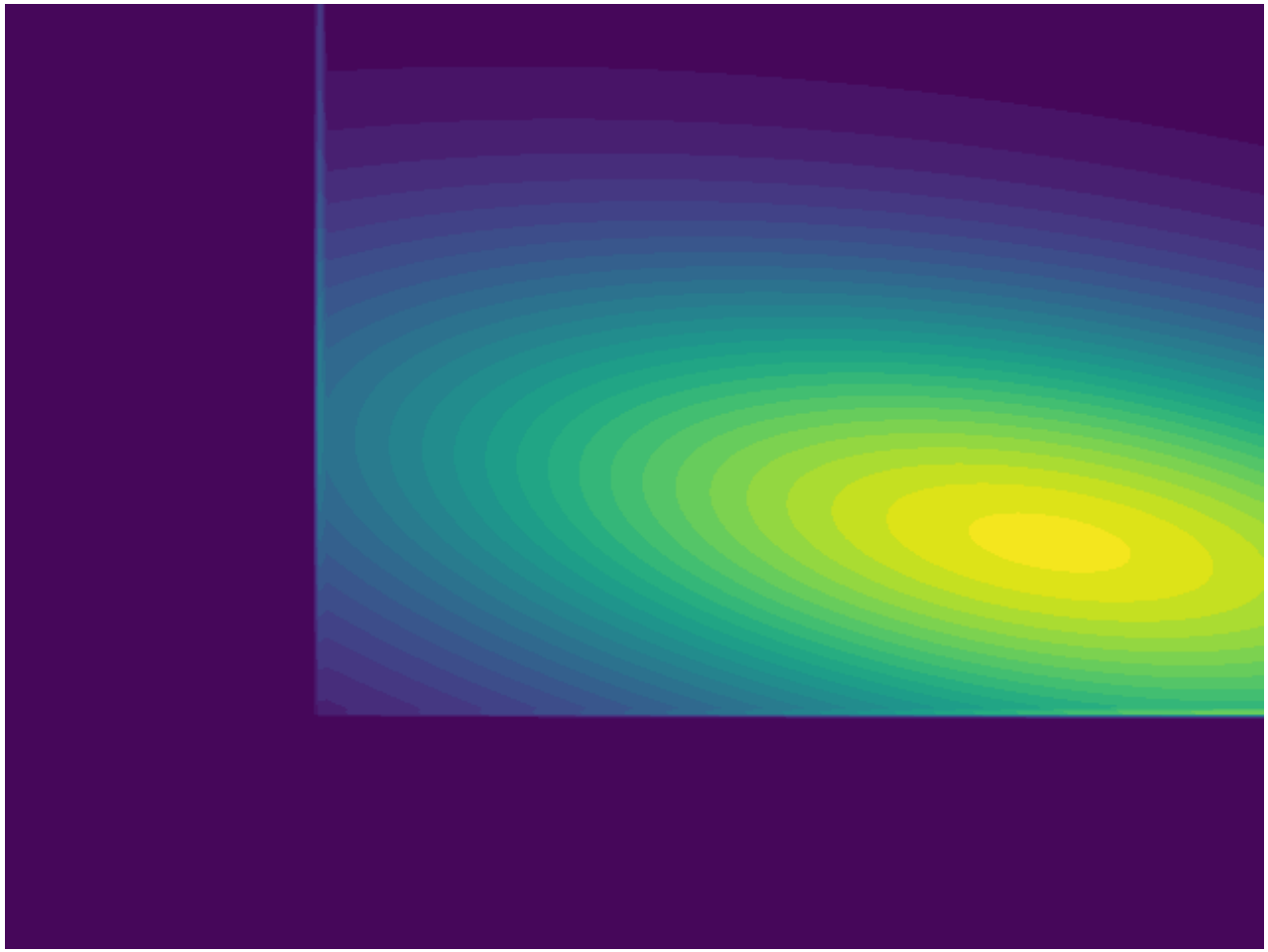
Assumed Density Filtering

Affine transform: $\mathcal{N}(W\mu_{\text{in}} + b, W\sigma_{\text{in}}W^\top)$



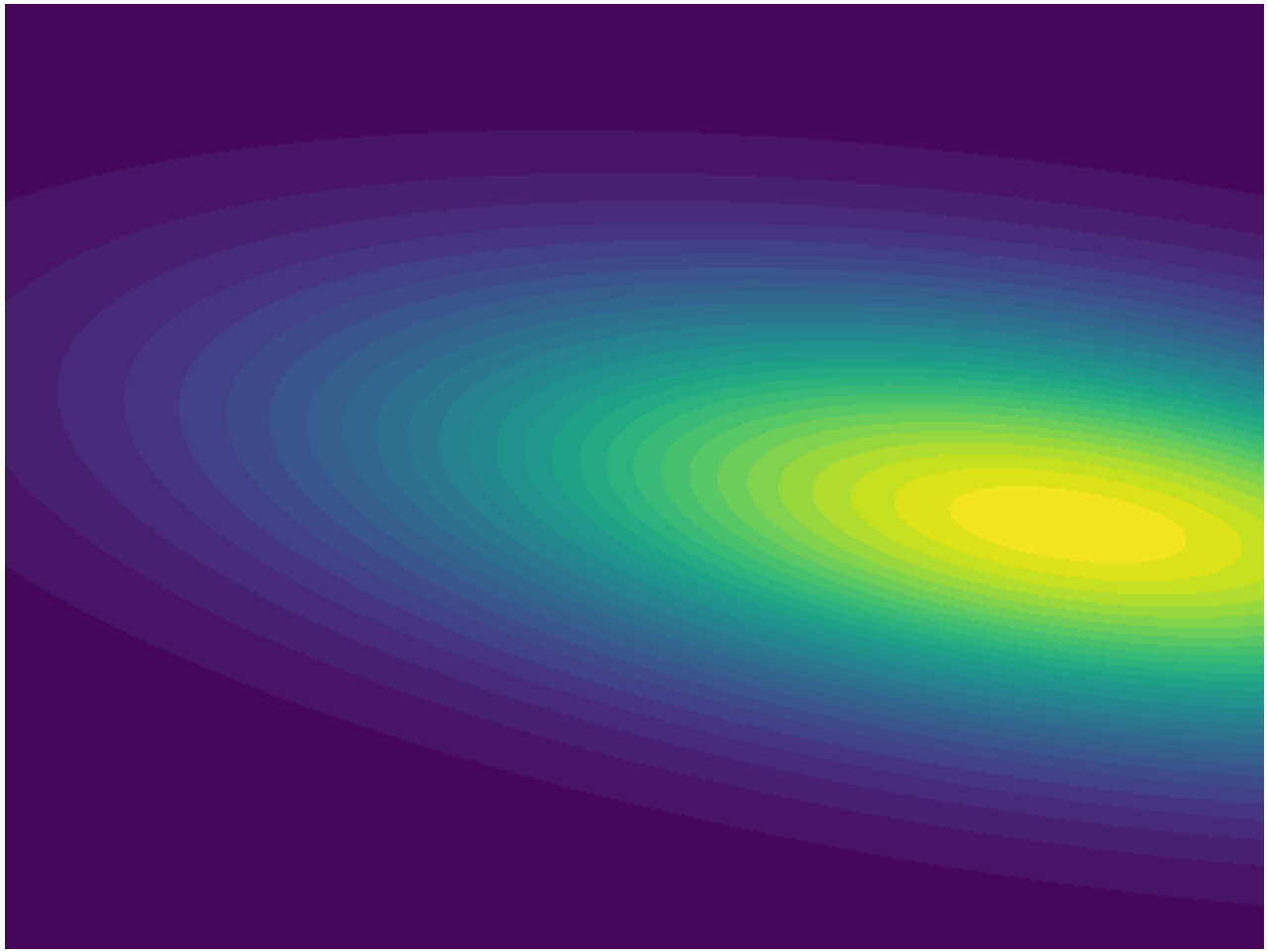
Assumed Density Filtering

ReLU activation: Not Gaussian anymore



Assumed Density Filtering

Moment matching output distribution: $\mathcal{N}(\mu_{\text{out}}, \sigma_{\text{out}})$

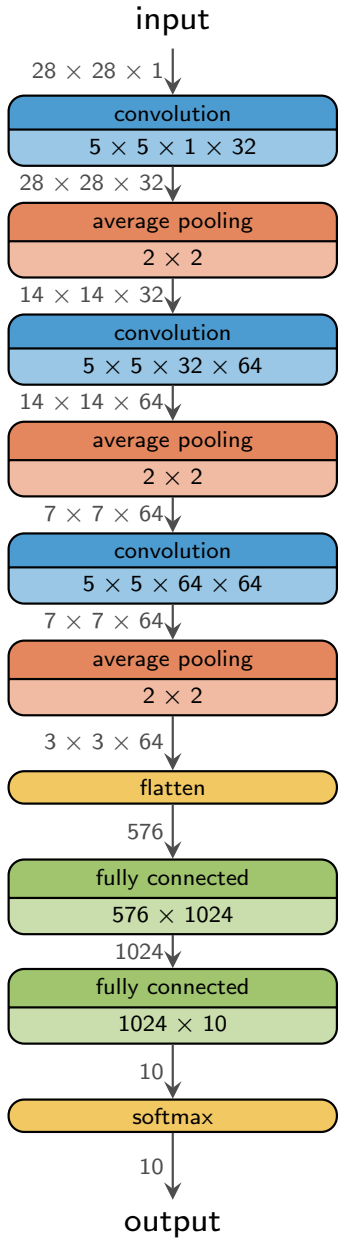


Numerical Experiments

MNIST Experiment

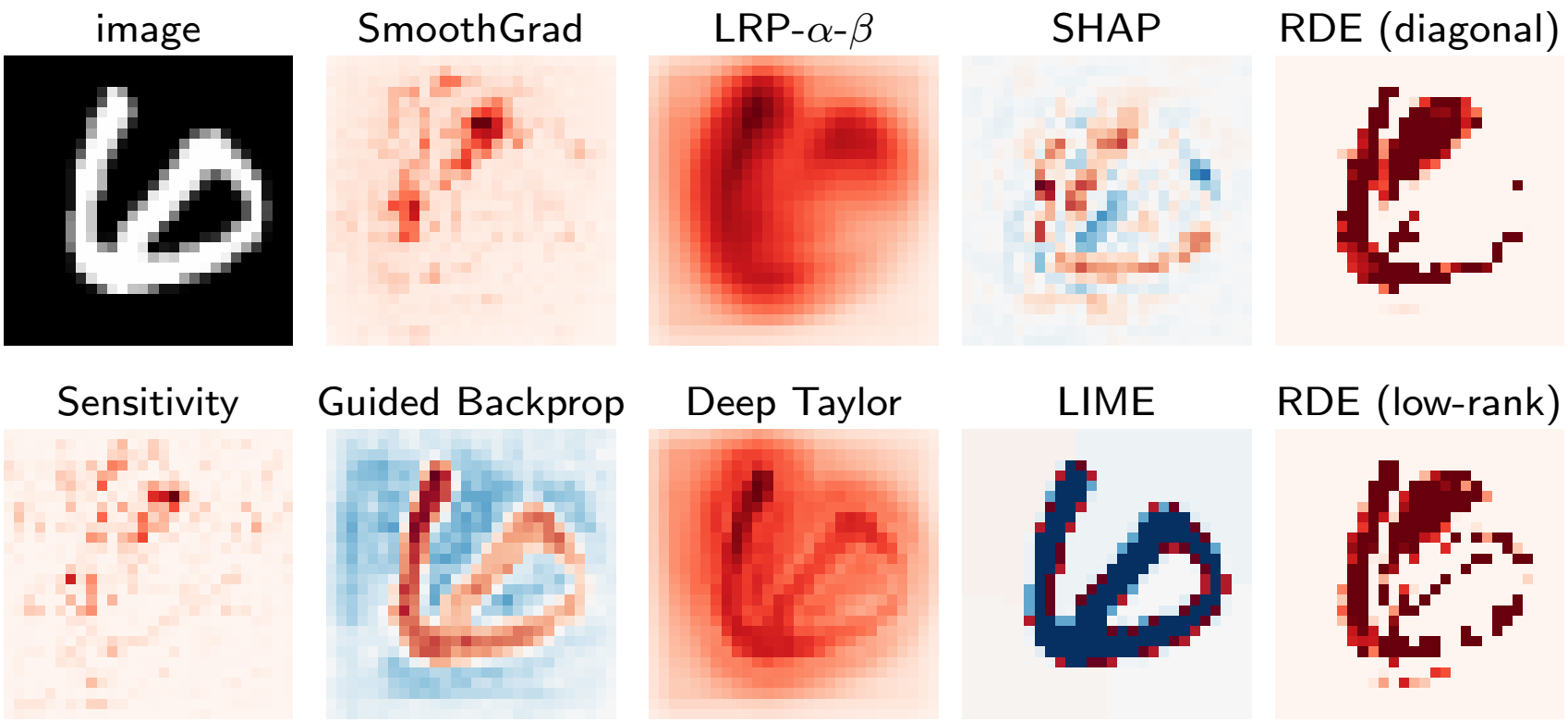
Data Set	
Image size	$28 \times 28 \times 1$
Number of classes	10
Training samples	50000

Test accuracy: 99.1%



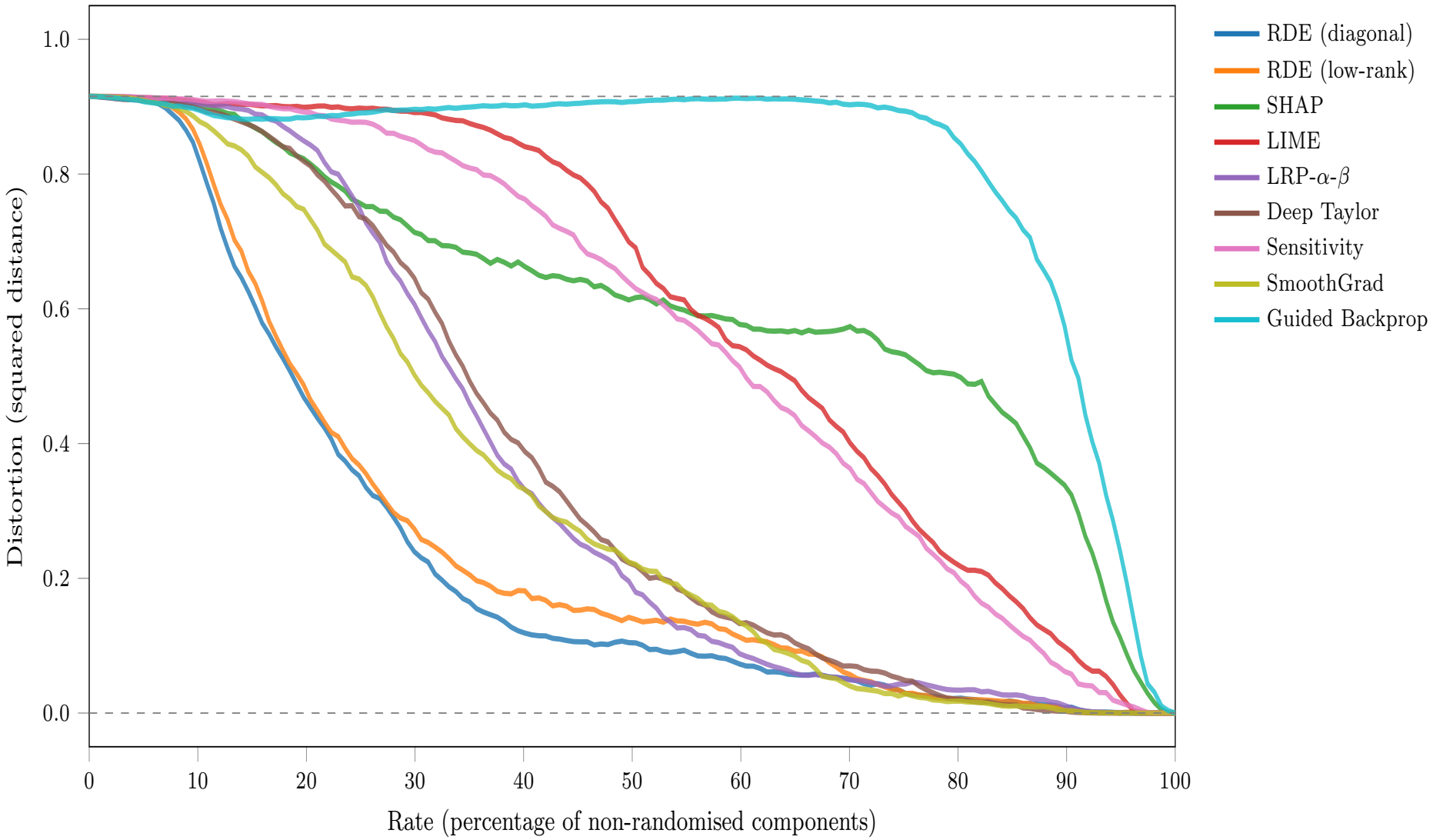
MNIST dataset of handwritten digits (LeCun, Cortes, 1998)

MNIST Experiment



SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017), Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015), SHAP (Lundberg, Lee, 2017), Sensitivity Analysis (Simonyan, Vedaldi, Zisserman, 2013), Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015), Deep Taylor Decompositions (Montavon, Samek, Müller, 2018), LIME (Ribeiro, Singh, Guestrin, 2016)

MNIST Experiment



SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017), Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015), SHAP (Lundberg, Lee, 2017), Sensitivity Analysis (Simonyan, Vedaldi, Zisserman, 2013), Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015), Deep Taylor Decompositions (Montavon, Samek, Müller, 2018), LIME (Ribeiro, Singh, Guestrin, 2016)

STL-10 Experiment

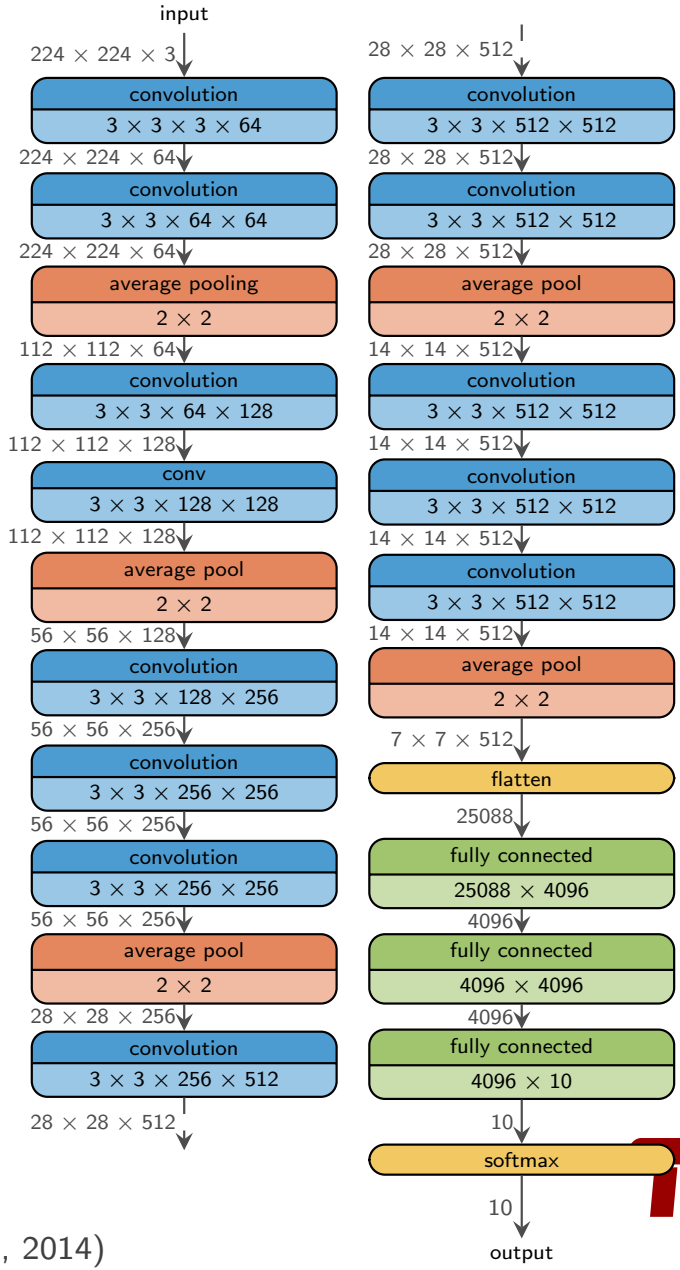



Data Set

Image size	$96 \times 96 \times 3$
	$(224 \times 224 \times 3)$
Number of classes	10
Training samples	4000

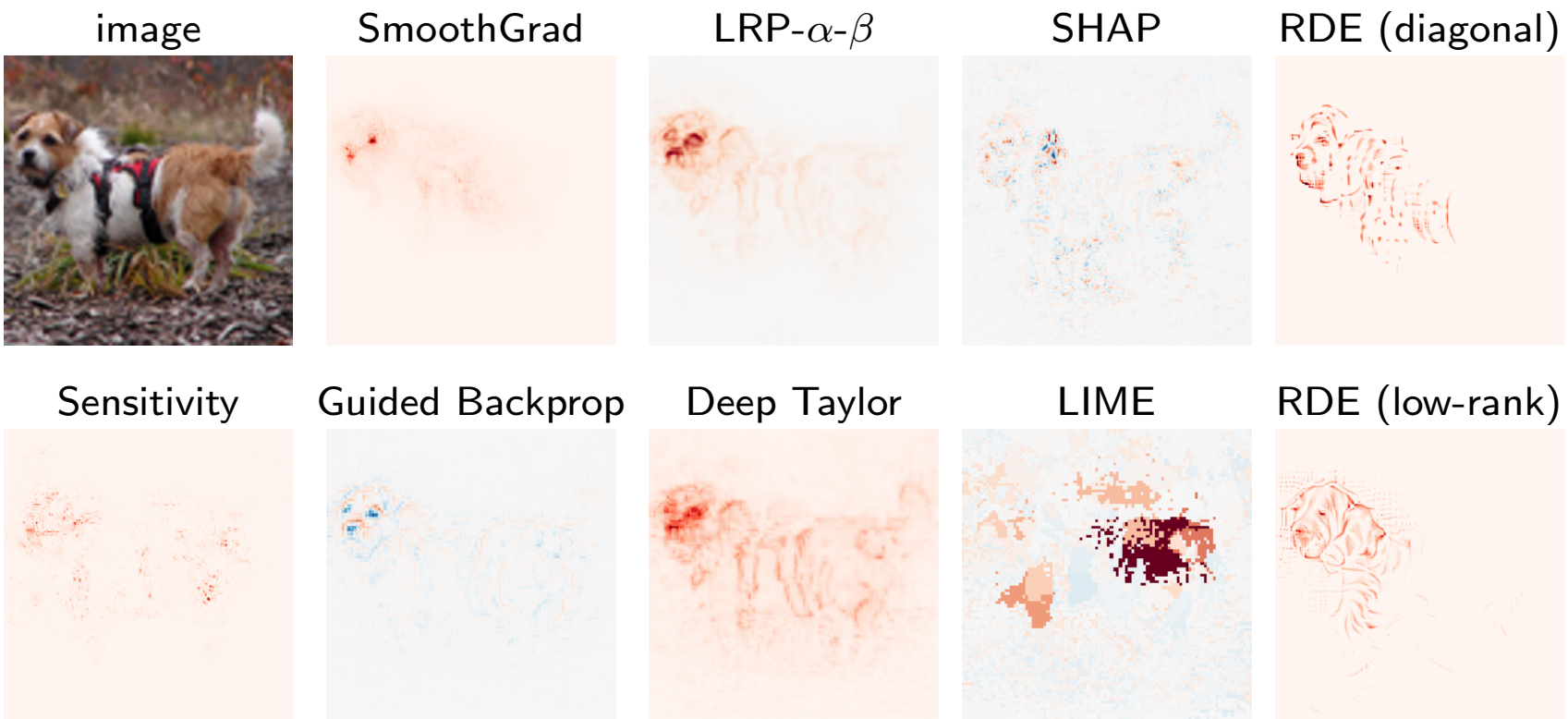
Test accuracy: 93.5%

(VGG-16 convolutions pretrained on Imagenet)



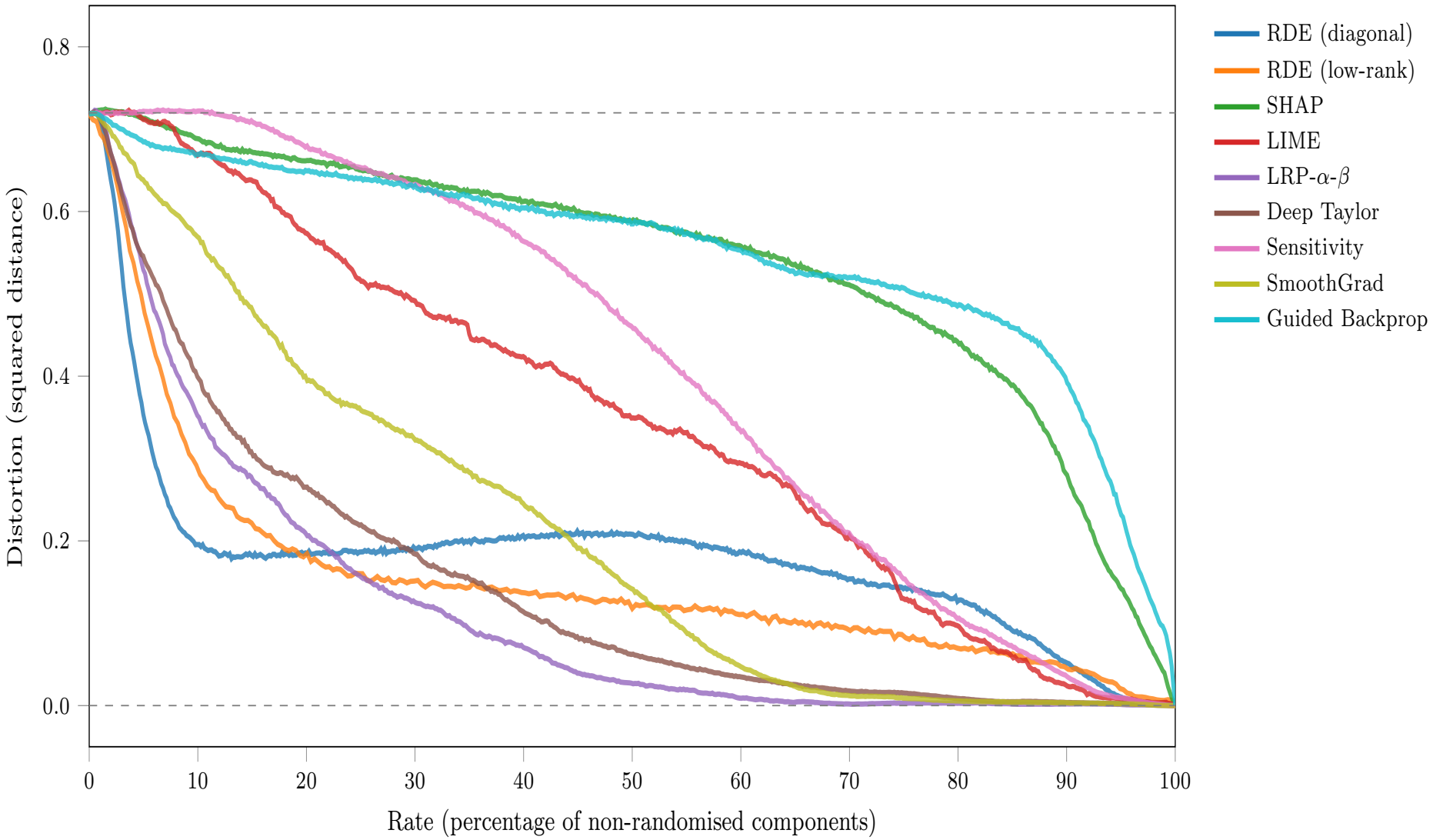
STL-10 dataset (Coates, Lee, Ng, 2011), VGG-16 network (Simonyan, Zisserman, 2014)

STL-10 Experiment



SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017), Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015), SHAP (Lundberg, Lee, 2017), Sensitivity Analysis (Simonyan, Vedaldi, Zisserman, 2013), Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015), Deep Taylor Decompositions (Montavon, Samek, Müller, 2018), LIME (Ribeiro, Singh, Guestrin, 2016)

STL-10 Experiment



SmoothGrad (Smilkov, Thorat, Kim, Viégas, Wattenberg, 2017), Layer-wise Relevance Propagation (Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015), SHAP (Lundberg, Lee, 2017), Sensitivity Analysis (Simonyan, Vedaldi, Zisserman, 2013), Guided Backprop (Springenberg, Dosovitskiy, Brox, Riedmiller, 2015), Deep Taylor Decompositions (Montavon, Samek, Müller, 2018), LIME (Ribeiro, Singh, Guestrin, 2016)

(Interpretability of) Deep Learning for Mathematical Problems

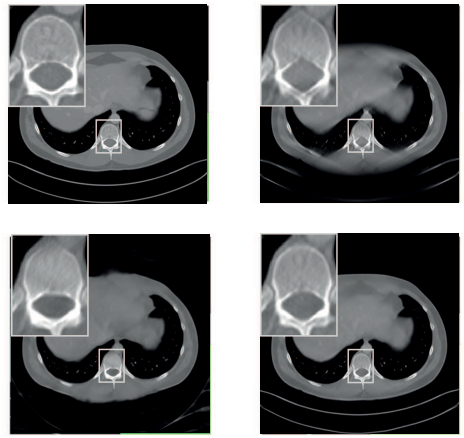
Data-Driven and (Physical) Model-Based Approaches

Optimal balancing of
data-driven and model-based approaches!

Impact of Deep Learning on Mathematical Problems

Some Examples:

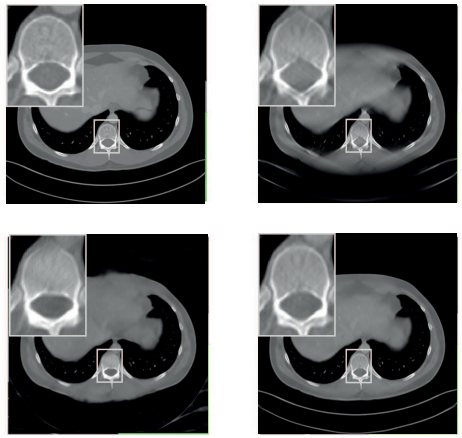
- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*

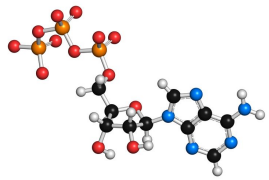


Impact of Deep Learning on Mathematical Problems

Some Examples:

- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*
- Numerical Analysis of Partial Differential Equations
  - ⇝ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012 –)*
  - ⇝ *Black-Scholes PDEs (Grohs, Hornung, Jentzen, von Wurstemberger; 2018)*
  - ⇝ *Parametric PDEs (Schwab, Zech; 2018) (K, Petersen, Raslan, Schneider; 2019)*

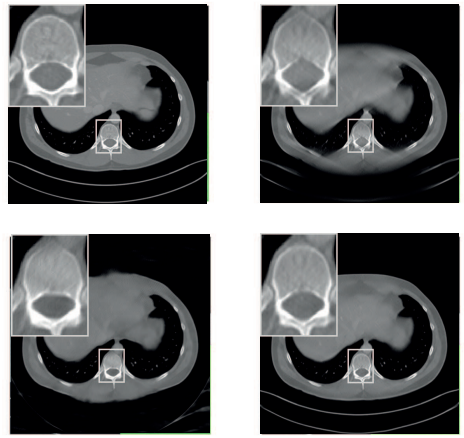




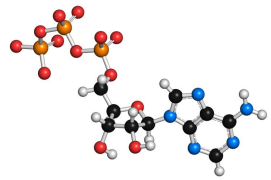
Impact of Deep Learning on Mathematical Problems

Some Examples:

- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*



- Numerical Analysis of Partial Differential Equations
  - ⇝ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012 –)*
  - ⇝ *Black-Scholes PDEs (Grohs, Hornung, Jentzen, von Wurstemberger; 2018)*
  - ⇝ *Parametric PDEs (Schwab, Zech; 2018) (K, Petersen, Raslan, Schneider; 2019)*

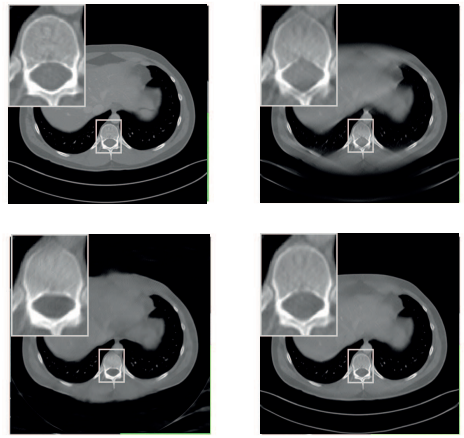


- Modelling
  - ⇝ *Learning equations from data (Sahoo, Lampert, Martius; 2018)*

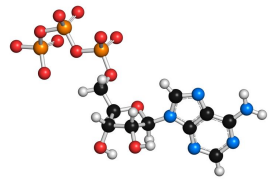
Impact of Deep Learning on Mathematical Problems

Some Examples:

- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*



- Numerical Analysis of Partial Differential Equations
  - ⇝ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012 –)*
  - ⇝ *Black-Scholes PDEs (Grohs, Hornung, Jentzen, von Wurstemberger; 2018)*
  - ⇝ *Parametric PDEs (Schwab, Zech; 2018) (K, Petersen, Raslan, Schneider; 2019)*



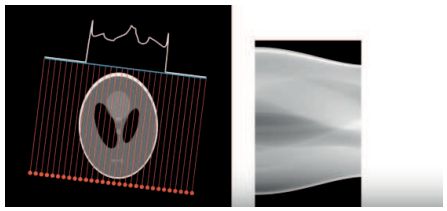
- Modelling
  - ⇝ *Learning equations from data (Sahoo, Lampert, Martius; 2018)*

Deep Learning and Inverse Problems

Example: Limited-Angle Computed Tomography

A CT scanner samples the *Radon transform*

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



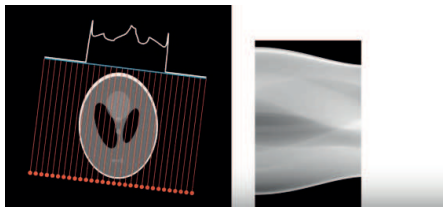
for $L(\phi, s) = \{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

Deep Learning and Inverse Problems

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

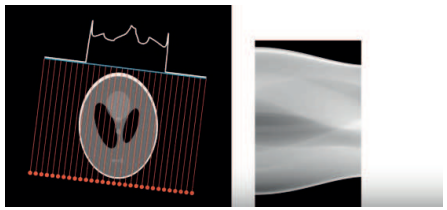
Challenging inverse problem if $\mathcal{R}f(\cdot, s)$ is only sampled on $[-\phi, \phi]$, $\phi < \pi/2$

Deep Learning and Inverse Problems

Example: 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 $[-\phi, \phi]$, $\phi < \pi/2$

Learn the Invisible (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018):

Step 1: Use model-based methods as far as possible

- Solve with sparse regularization using shearlets.

Step 2: Use data-driven methods where it is necessary

- Use a deep neural network to recover the missing components.

Step 3: Carefully combine both worlds

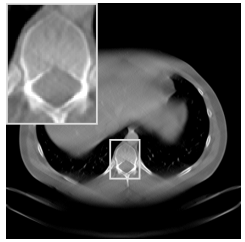
- Combine outcome of Step 1 and 2.

Our Approach "Learn the Invisible (LtI)"

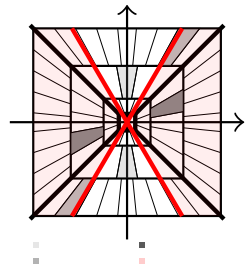
(Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)

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



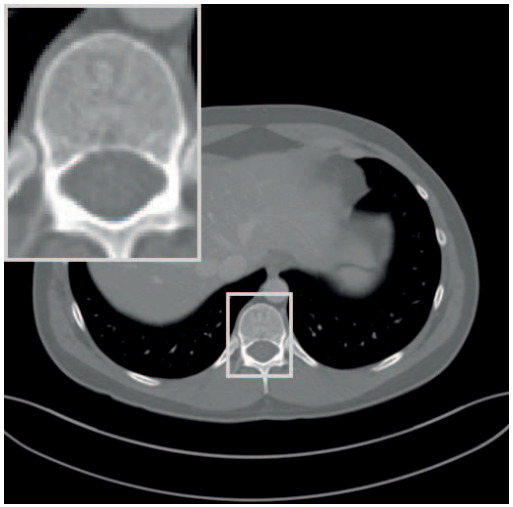
Step 2: *Learn the invisible*

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

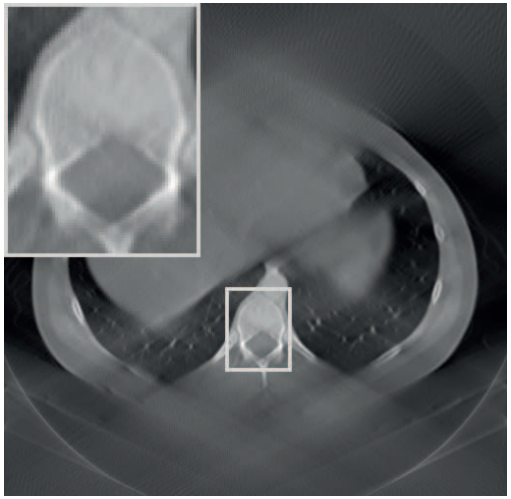
Step 3: *Combine*

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

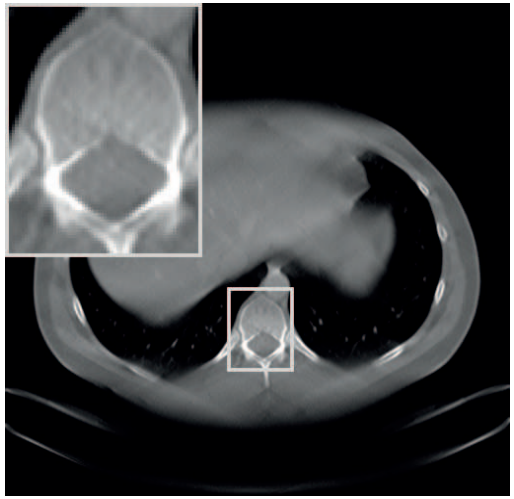
Numerical Results



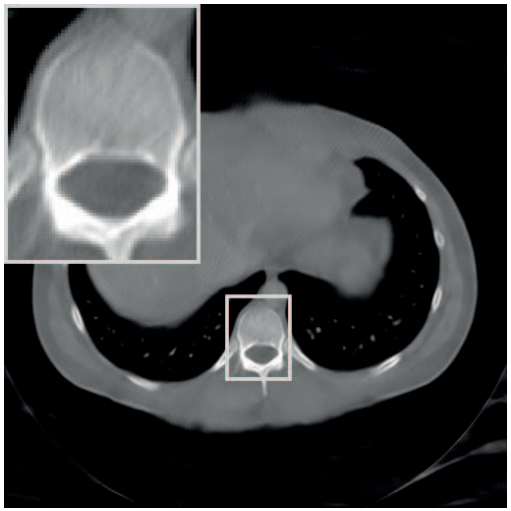
Original



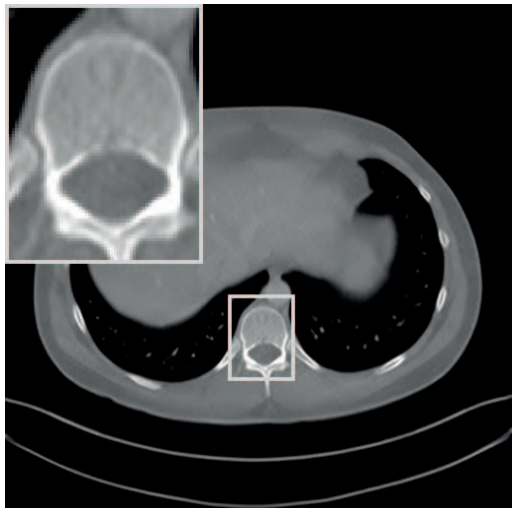
Filtered Backprojection



Sparse Regularization with Shearlets



[Gu & Ye, 2017]

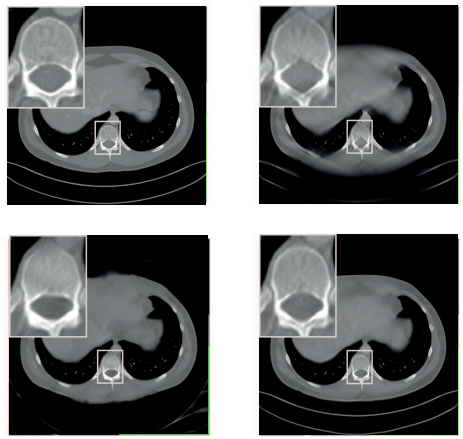


Learn the Invisible (LtI)

Meaning of Interpretability?

Some Thoughts on Interpretability:

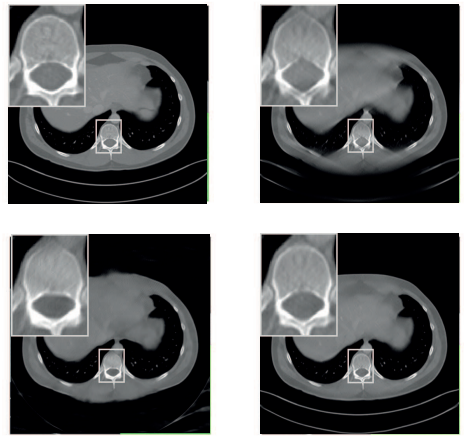
- Validation of result
  - Uncertainty quantification
  - Consistency
  - ...
- Explanation of result
  - Sensitively dependent on the observer
  - Understanding of the regularization of the (ill-posed) inverse problem
  - Meaningful with respect to a goal
  - Impact of input on output
  - ...



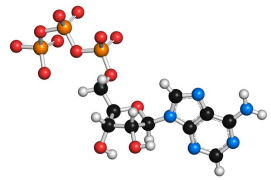
Impact of Deep Learning on Mathematical Problems

Some Examples:

- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*



- Numerical Analysis of Partial Differential Equations
  - ⇝ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012 –)*
  - ⇝ *Black-Scholes PDEs (Grohs, Hornung, Jentzen, von Wurstemberger; 2018)*
  - ⇝ *Parametric PDEs (Schwab, Zech; 2018) (K, Petersen, Raslan, Schneider; 2019)*



- Modelling
  - ⇝ *Learning equations from data (Sahoo, Lampert, Martius; 2018)*

Why Parametric PDEs?

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

Some Exemplary Problem Classes:

- Complex design 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, ...)

Parametric Map:

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

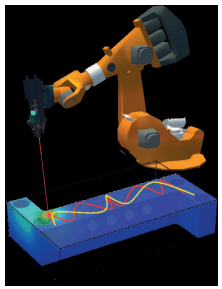
Multi-Query Situation

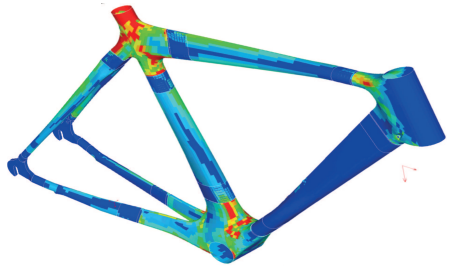
Many applications require solving the parametric PDE multiple times for different parameters:

$$\mathbb{R}^p \supset \mathcal{Y} \ni y = (y_1, \ldots, y_p) \quad \mapsto \quad u_y \in \mathcal{H}$$

Examples:

- Design optimization
- Optimal control
- Routine analysis
- Uncertainty quantification
- Inverse problems





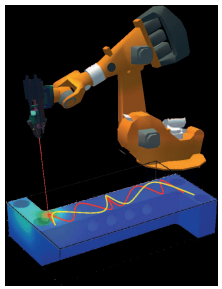
Multi-Query Situation

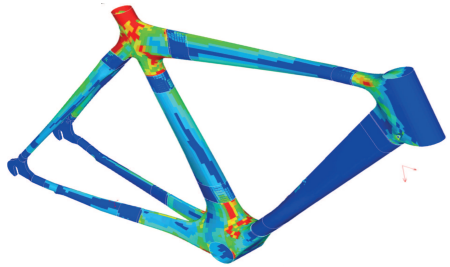
Many applications require solving the parametric PDE multiple times for different parameters:

$$\mathbb{R}^p \supset \mathcal{Y} \ni y = (y_1, \ldots, y_p) \quad \mapsto \quad u_y \in \mathcal{H}$$

Examples:

- Design optimization
- Optimal control
- Routine analysis
- Uncertainty quantification
- Inverse problems





Curse of Dimensionality:

Computational cost often much too high!

Our Contribution

Our Goal: Learn a general parametric map

$$\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}, \qquad \text{where } \mathcal{Y} \subset \mathbb{R}^p,$$

by a deep neural network with ReLU activation functions.

Theorem (K, Petersen, Raslan, Schneider; 2019): "In this setting neural networks beat the curse of dimensionality – they are able to sense the solution manifold – by explicably constructing such networks."

Our Contribution

Our Goal: Learn a general parametric map

$$\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}, \qquad \text{where } \mathcal{Y} \subset \mathbb{R}^p,$$

by a deep neural network with ReLU activation functions.

Theorem (K, Petersen, Raslan, Schneider; 2019): "In this setting neural networks beat the curse of dimensionality – they are able to sense the solution manifold – by explicably constructing such networks."

Idea of Proof:

(1) Construct a neural network, which inverts matrices!

(2) Approximate the matrix $V(B_y^{rb})^{-1} V^T f_y^h$ by a neural network and control its size!

Extremely fast computation of the parametric map, while beating the curse of dimensionality!

Meaning of Interpretability?

Some Thoughts on Interpretability:

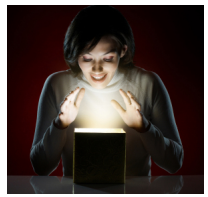
- Validation of result
  - Uncertainty quantification
  - Approximation of the parametric map
  - ...
- Explanation of result
  - Sensitively dependent on the user
  - Meaningful with respect to a goal
  - Impact of input on output
  - ...

Conclusions

What to take Home...?

Deep Learning:

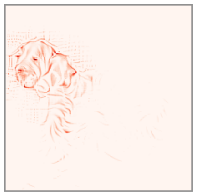
- Impressive performance *in combination with classical mathematical methods* (Inverse Problems, PDEs, ...).
- A theoretical foundation of neural networks is largely missing: *Expressivity, Learning, Generalization, and Interpretability*.



Interpretability:

- Opening the *black box* of a trained neural network.
- Determining which input features are *most relevant* for a decision.

Rate-Distortion Explanation (RDE):



- Determining the *minimal rate* is *hard*.
- Even the *approximation problem* of it is *hard*.
- *RDE considers a relaxed version* within the *rate-distortion framework*.
- It *outperforms current methods* for smaller rates.

THANK YOU!

References available at:

www.math.tu-berlin.de/∼kutyniok

Related Book:

- P. Grohs and G. Kutyniok
  Theory of Deep Learning
  Cambridge University Press (in preparation)