The Mathematics of Deep Learning
Lecture 7: Limitations of Deep Learning

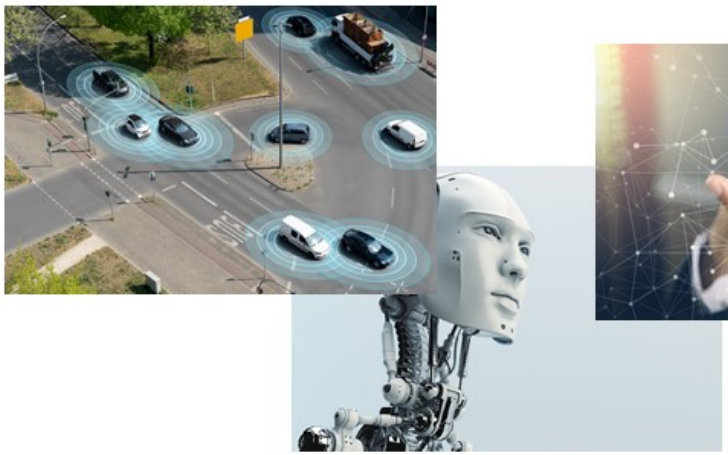
Gitta Kutyniok

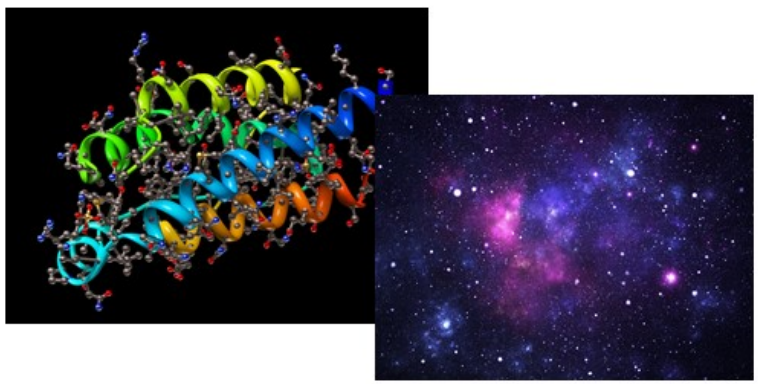
(Ludwig-Maximilians-Universität München)

LMS Invited Lecture Series

University of Cambridge, February 28 – March 4, 2022

Neural Network Successes





... and Failures

Deep neural networks have

- tremendous *success* for problems in scientific computing,
- but serious *downsides*.

... and Failures

Deep neural networks have

- tremendous *success* for problems in scientific computing,
- but serious *downsides*.

Problems/Limitations:

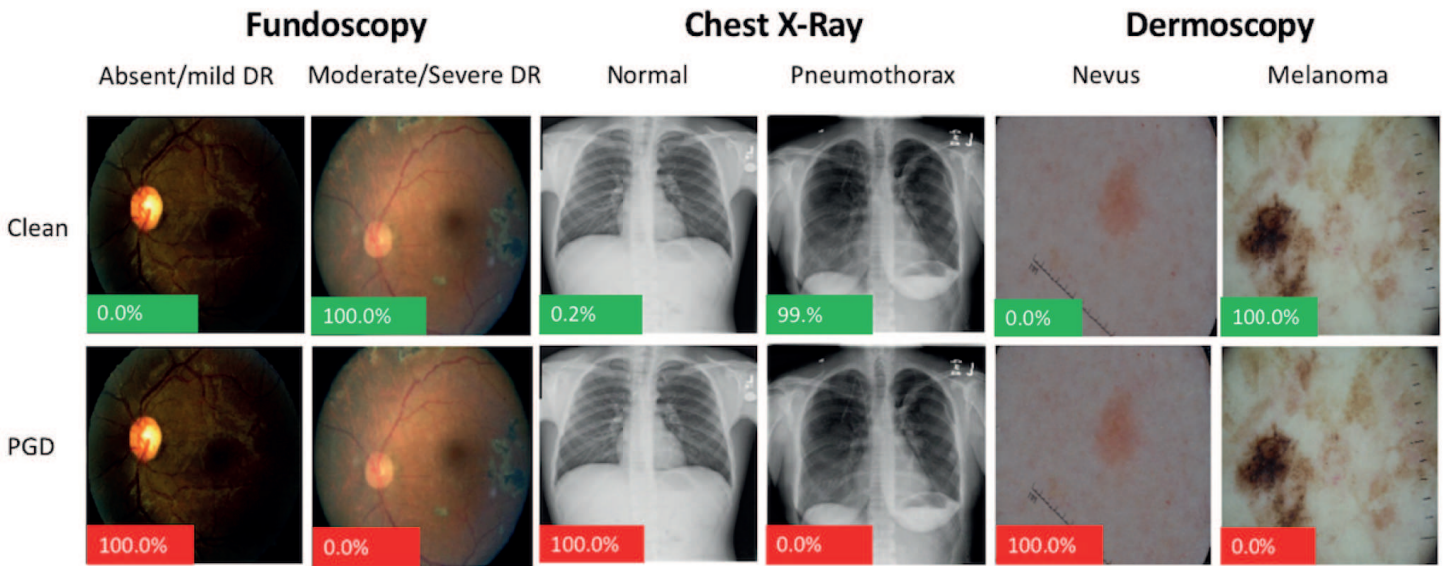
- Robustness
- Explainability
- Severe dependence on data
- Specific task
- Reasoning
- ....

Robustness

Requirements:

- Robustness problems should be immediately detectable or avoidable.
- Heuristic approaches do not satisfy certification standards.

Is complete robustness at all possible?



Source: Finlayson, Chung, Kohane, Beam, Adversarial Attacks Against Medical Deep Learning Systems, arXiv:1804.05296

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Explainability

Requirements:

- It should be possible to ask any question about a decision.
- The answer should reason as a human.

Is this achievable by connecting deep learning to natural language processing?

Severe Dependence on Data

Amount of Data:

- Many applications do not have large amounts of training data.
- Methods such as data augmentation do not compensate this fully.

Severe Dependence on Data

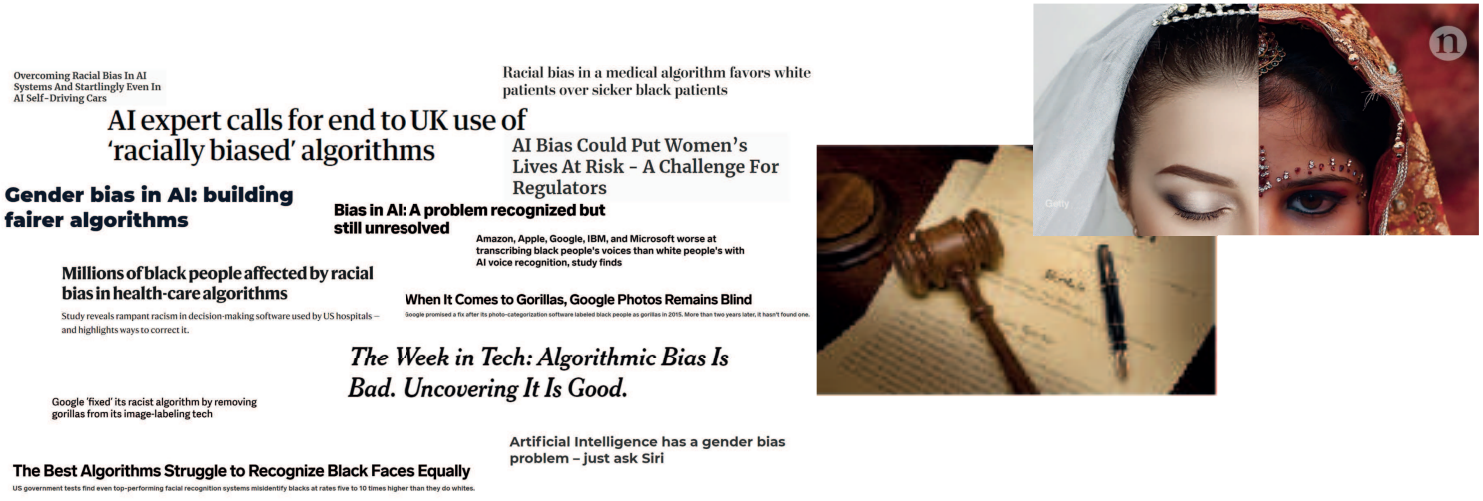
Amount of Data:

- Many applications do not have large amounts of training data.
- Methods such as data augmentation do not compensate this fully.

Problematic Data:

- Training data can unknowingly be biased.
- Uncertainty in the data can occur.

Can these problems be tackled at all?



Specific Task and Reasoning

Requirements:

- It should be possible to train for multiple tasks.
- The neural network should also be able to reason.
- Ideally, lifelong learning should be possible.

Are there fundamental limitations that constrain us?

An Even More Serious Problem

Computability

Theory asserts

- the expressibility of the class of deep neural networks
- convergence of training algorithms
- generalization abilities
- ...

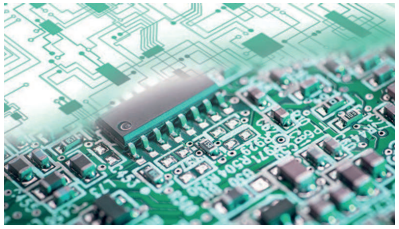
Computability

Theory asserts

- the expressibility of the class of deep neural networks
- convergence of training algorithms
- generalization abilities
- ...

Theory does not sufficiently consider

- practical performance when trained by modern approaches
- required sample complexity
- limits of **computability** on today's hardware



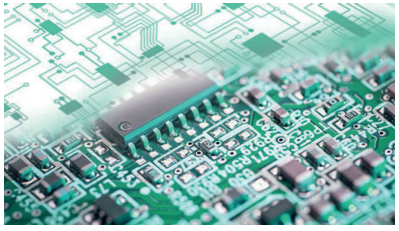
Computability

Theory asserts

- the expressibility of the class of deep neural networks
- convergence of training algorithms
- generalization abilities
- ...

Theory does not sufficiently consider

- practical performance when trained by modern approaches
- required sample complexity
- limits of **computability** on today's hardware



Theory-to-Practice Gap!

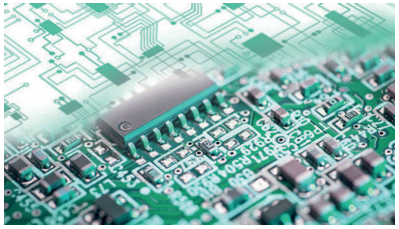
Computability

Theory asserts

- the expressibility of the class of deep neural networks
- convergence of training algorithms
- generalization abilities
- ...

Theory does not sufficiently consider

- practical performance when trained by modern approaches
- required sample complexity
- limits of **computability** on today's hardware



Theory-to-Practice Gap!

Goal: Examine the boundaries imposed by digital computations!

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

What can actually be computed?

Computability on Digital Machines (informal):

A *computable problem (function)* is one for which the input-output relation can be computed on a digital machine for any given accuracy.

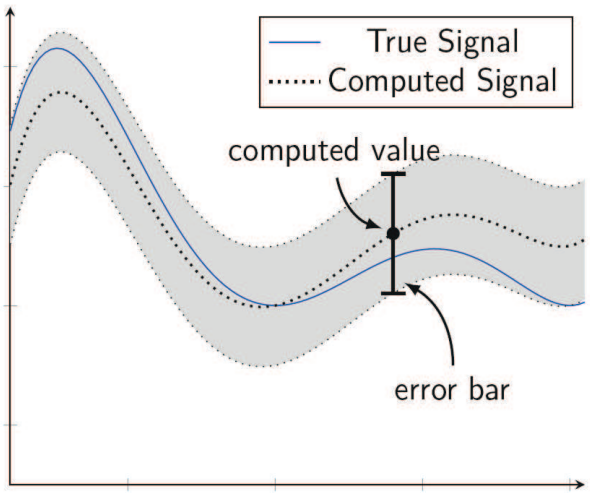
What can actually be computed?

Computability on Digital Machines (informal):

A *computable problem (function)* is one for which the input-output relation can be computed on a digital machine for any given accuracy.

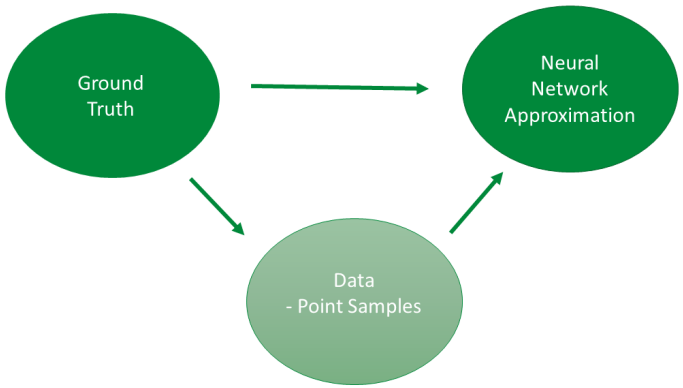
Questions:

- Is the underlying problem feasible?
  $\rightarrow$ *Computability of the ground truth*
- Are the neural networks computable?
  $\rightarrow$ *Computability of the network*
- Can the neural networks be found with the minimization problem?
  $\rightarrow$ *Computability of the mapping from data to approximation*



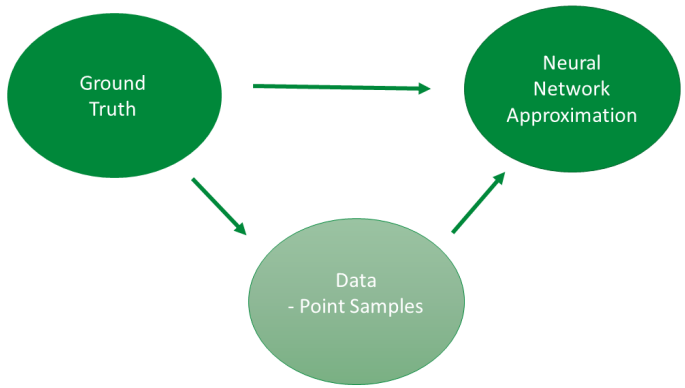
Why does Computability Matter?

- What is the *best* we can hope for?
  - Non-computability of the *ground truth*
    $\rightarrow$ *No approximation scheme*
  - Non-computability of the *network*
    $\rightarrow$ *Despite existence, network may not be computable*
  - Non-computability of the *mapping* from data to approximation
    $\rightarrow$ *Learning not feasible*



Why does Computability Matter?

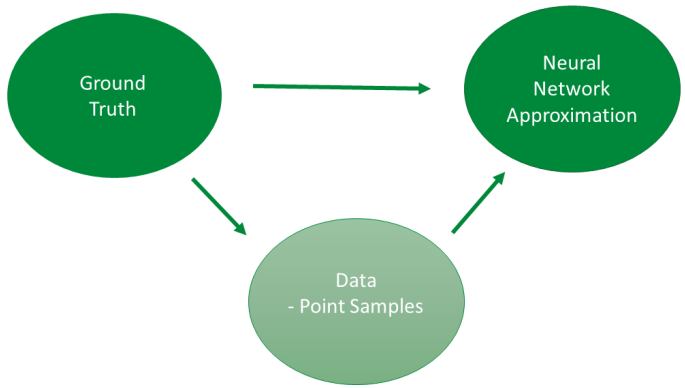
- What is the *best* we can hope for?
  - Non-computability of the *ground truth*
    $\rightarrow$ *No approximation scheme*
  - Non-computability of the *network*
    $\rightarrow$ *Despite existence, network may not be computable*
  - Non-computability of the *mapping* from data to approximation
    $\rightarrow$ *Learning not feasible*



- Can we *trust* the output of a computation?
  - Computability guarantees prescribed error bounds
    $\rightarrow$ Reliable output

Why does Computability Matter?

- What is the *best* we can hope for?
  - Non-computability of the *ground truth*
    $\rightarrow$ *No approximation scheme*
  - Non-computability of the *network*
    $\rightarrow$ *Despite existence, network may not be computable*
  - Non-computability of the *mapping* from data to approximation
    $\rightarrow$ *Learning not feasible*

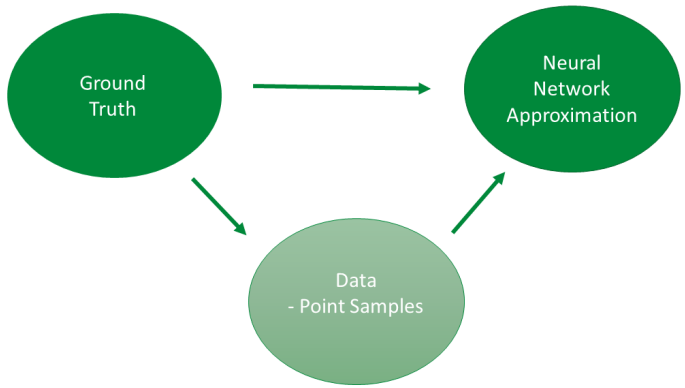


- Can we *trust* the output of a computation?
  - Computability guarantees prescribed error bounds
    $\rightarrow$ Reliable output

Non-computable problems can be tackled successfully in practice, if limited precision suffices!

Why does Computability Matter?

- What is the *best* we can hope for?
  - Non-computability of the *ground truth*
    $\rightarrow$ *No approximation scheme*
  - Non-computability of the *network*
    $\rightarrow$ *Despite existence, network may not be computable*
  - Non-computability of the *mapping* from data to approximation
    $\rightarrow$ *Learning not feasible*



- Can we *trust* the output of a computation?
  - Computability guarantees prescribed error bounds
    $\rightarrow$ Reliable output

Non-computable problems can be tackled successfully in practice, if limited precision suffices!

But we have no guarantees of correctness!

Theory of Computation

Turing machine $\longleftrightarrow$ Abstract idealization of *digital computer*

$\downarrow$ $\downarrow$

Discrete problems Scientific computing: *Continuous problems*

Theory of Computation

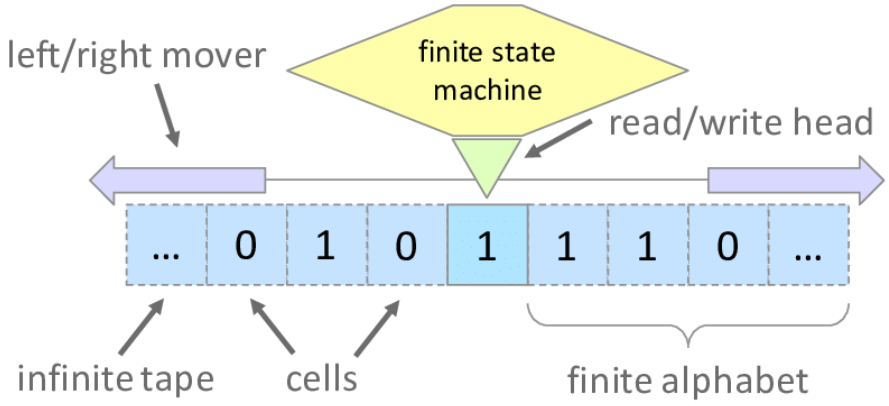
Turing machine $\longleftrightarrow$ Abstract idealization of *digital computer*

$\downarrow$ $\downarrow$

Discrete problems Scientific computing: *Continuous problems*

Definition:

"A Turing machine is a mathematical model of computation that defines an abstract machine that manipulates symbols on a strip of tape according to a table of rules."



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Computable Numbers

Definition:

A *computable real number* r is one for which there is a Turing machine with the following property: Given $n \in \mathbb{N}$ on its initial tape, it terminates with a rational number q such that $|r - q| \leq 2^{-n}$.

Computable Numbers

Definition:

A *computable real number* r is one for which there is a Turing machine with the following property: Given $n \in \mathbb{N}$ on its initial tape, it terminates with a rational number q such that $|r - q| \leq 2^{-n}$.

Properties:

- The set of computable real numbers $\mathbb{R}_c$ is countable and dense in $\mathbb{R}$.
- Irrational numbers can be computable, like e and π.
- There is no algorithm that decides whether two computable real numbers are equal.

Computable Functions

Definition:

A function $f : \mathbb{R} \to \mathbb{R}$ is *computable*, if there exists an algorithm (Turing machine) Γ_f, which gives for all computable $x \in \mathbb{R}_c$ and all $n \in \mathbb{N}$ an approximation to $f(x)$ with

$$|\Gamma_f(x, n) - f(x)| \leq 2^{-n}.$$

Computable Functions

Definition:

A function $f : \mathbb{R} \to \mathbb{R}$ is *computable*, if there exists an algorithm (Turing machine) Γ_f, which gives for all computable $x \in \mathbb{R}_c$ and all $n \in \mathbb{N}$ an approximation to $f(x)$ with

$$|\Gamma_f(x, n) - f(x)| \leq 2^{-n}.$$

Remarks:

- Minimal requirement for an algorithmic computation of a problem:
  - Algorithm takes the input(s)
  - computes the output(s)
  - with certain precision depending on the input precision.
- Straightforward extension to multi-dimensional and complex domain

Computable Functions

Definition:

A function $f : \mathbb{R} \to \mathbb{R}$ is *computable*, if there exists an algorithm (Turing machine) Γ_f, which gives for all computable $x \in \mathbb{R}_c$ and all $n \in \mathbb{N}$ an approximation to $f(x)$ with

$$|\Gamma_f(x, n) - f(x)| \leq 2^{-n}.$$

Remarks:

- Minimal requirement for an algorithmic computation of a problem:
  - Algorithm takes the input(s)
  - computes the output(s)
  - with certain precision depending on the input precision.
- Straightforward extension to multi-dimensional and complex domain

Example of Non-Computable Functions:

- Constant functions equal to non-computable numbers
- Discontinuous functions

Specific Example

Consider the modeling of a physical system S on a digital computer.

- Assume a mathematical model S_{mod} for S describes the physical process and,allows to predict the output of S for any given input.

How well does S_{mod} describe the real physical process S?

Specific Example

Consider the modeling of a physical system S on a digital computer.

- Assume a mathematical model S_{mod} for S describes the physical process and,allows to predict the output of S for any given input.

How well does S_{mod} describe the real physical process S?

- Compute the corresponding output $y = Sx$ for several input signals x.
- Compare these measurements with the theoretical prediction $y_{\text{pred}} = S_{\text{mod}}x$.

However, usually no closed-form solution for the output y_{pred} exists!

Specific Example

Consider the modeling of a physical system S on a digital computer.

- Assume a mathematical model S_{mod} for S describes the physical process and,allows to predict the output of S for any given input.

How well does S_{mod} describe the real physical process S?

- Compute the corresponding output $y = Sx$ for several input signals x.
- Compare these measurements with the theoretical prediction $y_{\text{pred}} = S_{\text{mod}}x$.

However, usually no closed-form solution for the output y_{pred} exists!

- Use a computer to determine y_{pred} of the model S_{mod} for an input x.
- A digital computer can only compute an approximation $\tilde{y}_{\text{pred}}$ of y_{pred}.

Control $\|\tilde{y}_{pred} - y_{pred}\|$ and y_{pred} of S_{mod} algorithmically on the computer!

Specific Example

Consider the modeling of a physical system S on a digital computer.

- Assume a mathematical model S_{mod} for S describes the physical process and,allows to predict the output of S for any given input.

How well does S_{mod} describe the real physical process S?

- Compute the corresponding output $y = Sx$ for several input signals x.
- Compare these measurements with the theoretical prediction $y_{\text{pred}} = S_{\text{mod}}x$.

However, usually no closed-form solution for the output y_{pred} exists!

- Use a computer to determine y_{pred} of the model S_{mod} for an input x.
- A digital computer can only compute an approximation $\tilde{y}_{\text{pred}}$ of y_{pred}.

Control $\|\tilde{y}_{pred} - y_{pred}\|$ and y_{pred} of S_{mod} algorithmically on the computer!

Otherwise...

- ...the calculated solution $\tilde{y}_{\text{pred}}$ might be far from y_{pred} and comparing measurements of S with $\tilde{y}_{\text{pred}}$ becomes meaningless!
- ...no information about the quality of the mathematical model!

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

A Large Problem Class

Computable Ground Truth?

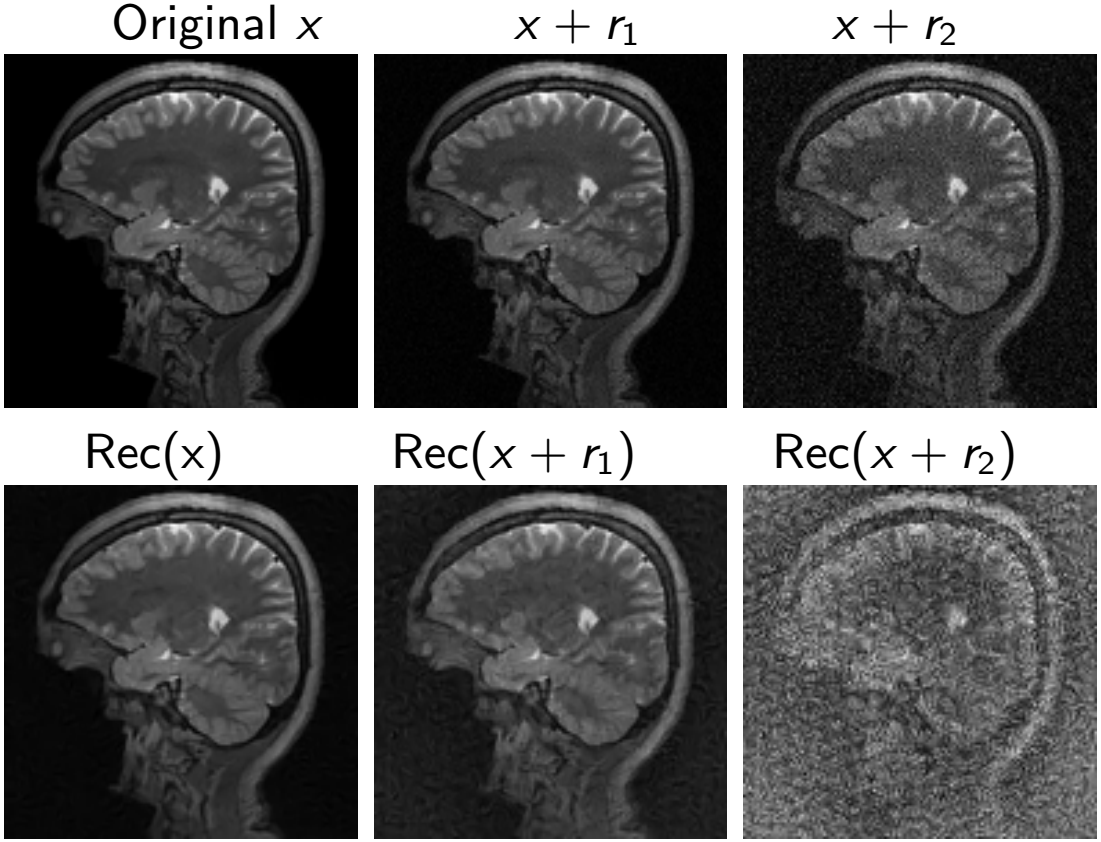
Question: Is the problem (function) we try to approximate computable?

$\rightarrow$ *No good approximation if the function is non-computable.*

Computable Ground Truth?

Question: Is the problem (function) we try to approximate computable?
$\rightarrow$ *No good approximation if the function is non-computable.*

Example (MRI): Let $A \in \mathbb{C}^{m \times N}$ and $y = Ax + e \in \mathbb{C}^m$, recover $x \in \mathbb{C}^N$.



Source: Gottschling, Antun, Adcock, Hansen. The troublesome kernel: why deep learning for inverse problems is typically unstable. (arXiv:2001.01258)

Inverse Problem in Imaging

Recall:

Given $A \in \mathbb{C}^{m \times N}$ and $y = Ax + e \in \mathbb{C}^m$ of $x \in \mathbb{C}^N$, recover x.

Properties:

- $A \in \mathbb{C}^{m \times N}$ sampling operator, typically $m < N$ or even $m \ll N$
- successful approaches:
  - Sparse regularization techniques
  - Deep learning techniques or hybrid approaches

Inverse Problem in Imaging

Recall:
Given $A \in \mathbb{C}^{m \times N}$ and $y = Ax + e \in \mathbb{C}^m$ of $x \in \mathbb{C}^N$, recover x.

Properties:

- $A \in \mathbb{C}^{m \times N}$ sampling operator, typically $m < N$ or even $m \ll N$
- successful approaches:
  - Sparse regularization techniques
  - Deep learning techniques or hybrid approaches

Optimization Problem:
Given $A \in \mathbb{C}^{m \times N}$ and measurements $y \in \mathbb{C}^m$, solve

$$\arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon, \quad \varepsilon > 0.$$

Solution Set

Solution Set:

For $A \in \mathbb{C}^{m \times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Solution Set

Solution Set:

For $A \in \mathbb{C}^{m \times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Fundamental Questions:

What can actually be computed on digital hardware?

Solution Set

Solution Set:

For $A \in \mathbb{C}^{m \times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Fundamental Questions:

What can actually be computed on digital hardware?

What are inherent restrictions of deep learning (performed on digital hardware)?

Solution Set

Solution Set:

For $A \in \mathbb{C}^{m \times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

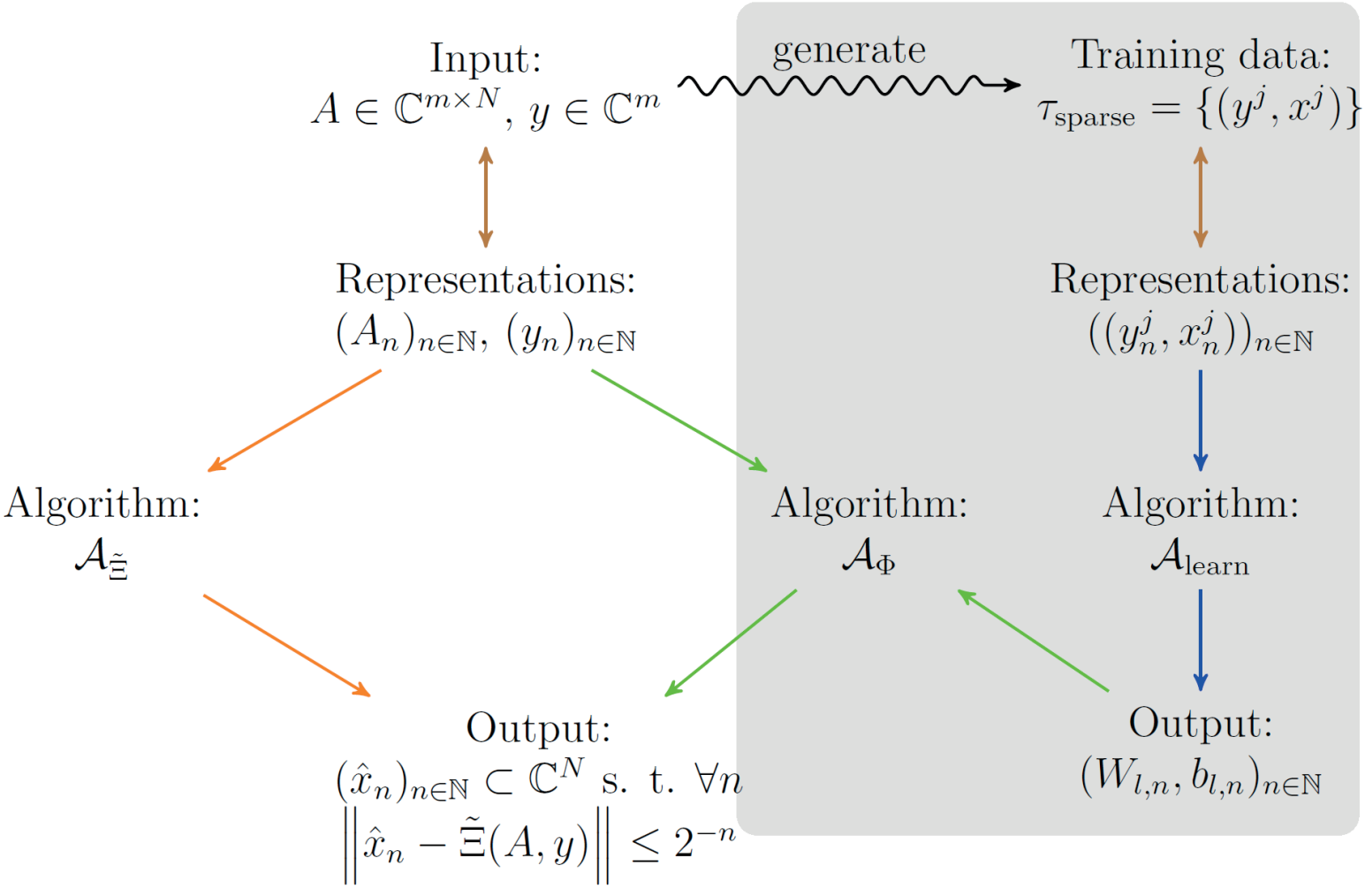
Fundamental Questions:

What can actually be computed on digital hardware?

What are inherent restrictions of deep learning (performed on digital hardware)?

Are we *missing the correct tools and algorithms* to train neural networks adequately on digital machines or do *such algorithms not exist at all*?

Connecting the Dots



deep learning approach

A Bit Disappointing News

Non-Computability of Finite Dimensional Inverse Problems

Solution Set:

For $A \in \mathbb{C}^{m \times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg \min_{x \in \mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Non-Computability of Finite Dimensional Inverse Problems

Solution Set:

For $A \in \mathbb{C}^{m\times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg\min_{x\in\mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Theorem (Boche, Fono, K; 2022):

The function $\Psi : \mathbb{C}^{m\times N} \times \mathbb{C}^m \to \mathbb{C}^N$ for fixed parameters $\epsilon \in (0, 1)$, $N \geq 2$, and $m < N$, is *not computable*.

Non-Computability of Finite Dimensional Inverse Problems

Solution Set:

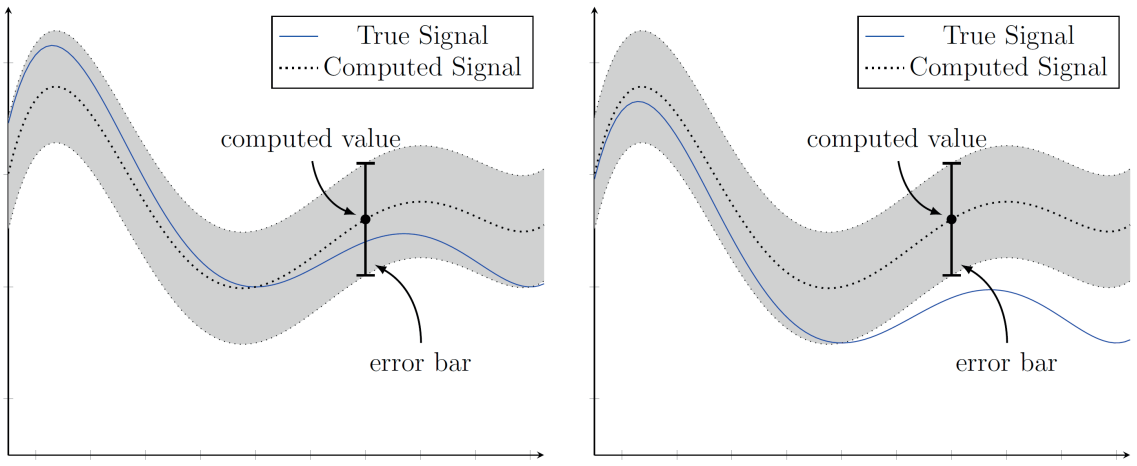
For $A \in \mathbb{C}^{m\times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A, y) := \arg\min_{x\in\mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax - y\|_{\ell^2} \leq \varepsilon.$$

Theorem (Boche, Fono, K; 2022):

The function $\Psi : \mathbb{C}^{m\times N} \times \mathbb{C}^m \to \mathbb{C}^N$ for fixed parameters $\epsilon \in (0, 1)$, $N \geq 2$, and $m < N$, is *not computable*.

Illustration of the Problem:



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Banach-Mazur Computability

Definition:

A function $f : I \to \mathbb{R}^n_c$, $I \subset \mathbb{R}^m_c$, is said to be *Banach-Mazur computable*, if f maps computable sequences $(t_n)_{n\in\mathbb{N}} \subset I$ onto computable sequences $(f(t_n))_{n\in\mathbb{N}} \subset \mathbb{R}^n_c$.

Banach-Mazur Computability

Definition:

A function $f : I \to \mathbb{R}^n_c$, $I \subset \mathbb{R}^m_c$, is said to be *Banach-Mazur computable*, if f maps computable sequences $(t_n)_{n\in\mathbb{N}} \subset I$ onto computable sequences $(f(t_n))_{n\in\mathbb{N}} \subset \mathbb{R}^n_c$.

Lemma:

If a function is not Banach–Mazur computable, then it is not computable with respect to any other reasonable notion of computability including (Borel-)Turing computability.

$\rightsquigarrow$ *Banach-Mazur computability is the weakest form of computability!*

Banach-Mazur Computability

Definition:
A function $f : I \to \mathbb{R}^n_c$, $I \subset \mathbb{R}^m_c$, is said to be *Banach-Mazur computable*, if f maps computable sequences $(t_n)_{n\in\mathbb{N}} \subset I$ onto computable sequences $(f(t_n))_{n\in\mathbb{N}} \subset \mathbb{R}^n_c$.

Lemma:
If a function is not Banach–Mazur computable, then it is not computable with respect to any other reasonable notion of computability including (Borel-)Turing computability.
$\rightsquigarrow$ *Banach-Mazur computability is the weakest form of computability!*

Remark:
The theorem holds even for Banach-Mazur computability.

Towards the Proof of...

Theorem (Boche, Fono, K; 2022):
The function $\Psi : \mathbb{C}^{m\times N} \times \mathbb{C}^m \to \mathbb{C}^N$ for fixed parameters $\epsilon \in (0,1)$, $N \geq 2$, and $m < N$, is *not (Banach-Mazur) computable*.

Recall the Solution Set:
For $A \in \mathbb{C}^{m\times N}$ and $y \in \mathbb{C}^m$ let

$$\Psi(A,y) := \arg\min_{x\in\mathbb{C}^N} \|x\|_{\ell^1} \text{ such that } \|Ax-y\|_{\ell^2} \leq \varepsilon.$$

Non-Computability Lemma

Lemma:

Suppose there are computable sequences $(\omega_n^1)_{n\in\mathbb{N}}, (\omega_n^2)_{n\in\mathbb{N}} \subset \mathbb{C}^{m\times N} \times \mathbb{C}^m$ satisfying the following conditions:

(a) There are sets $S^1, S^2 \subset \mathbb{C}^N$ and $\kappa > 0$ such that $\Psi(\omega_n^j) \subset S^j$ and

$$\inf_{x_1\in S^1, x_2\in S^2} \|x_1 - x_2\|_{\ell^2} > \kappa$$

.

(b) There is an $\omega^* \in \mathbb{C}^{m\times N} \times \mathbb{C}^m$ such that $\Psi(\omega^*)$ is computable and

$$\left\|\omega_n^j - \omega^*\right\|_{\ell^2} \leq 2^{-n} \quad \text{for all } n \in \mathbb{N}, j = 1, 2.$$

Then the map $\Psi : \mathbb{C}^{m\times N} \times \mathbb{C}^m \to \mathbb{C}^N$ is *not computable*.

Previous Work

Oracle Turing Machines

Closest Work:

- Classification: (Bastounis, Hansen, Vlacic; 2021)
- Inverse Problems: (Colbrook, Antun, Hansen; 2021)

Those approaches employ a so-called Oracle Turing machine.

Oracle Turing Machines

Closest Work:

- Classification: (Bastounis, Hansen, Vlacic; 2021)
- Inverse Problems: (Colbrook, Antun, Hansen; 2021)

Those approaches employ a so-called Oracle Turing machine.

Definition:

"The *Oracle Turing machine* is an extension of the regular Turing machine with an additional oracle tape that, given a real or complex number, grants the Turing machine access to a rational approximation of arbitrary — or more exactly demanded — precision."

Oracle Turing Machines

Closest Work:

- Classification: (Bastounis, Hansen, Vlacic; 2021)
- Inverse Problems: (Colbrook, Antun, Hansen; 2021)

Those approaches employ a so-called Oracle Turing machine.

Definition:

"The *Oracle Turing machine* is an extension of the regular Turing machine with an additional oracle tape that, given a real or complex number, grants the Turing machine access to a rational approximation of arbitrary — or more exactly demanded — precision."

Relation to Turing Machines:

- Oracle Turing machines exceed the potential of regular Turing machines.
- Oracle Turing machines omit the problem that not all complex numbers can be approximated.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

A More Detailed Comparison

Question Considered in Previous Work:

- Given *arbitrarily accurate approximations to any complex number* such as, in particular, the training samples.

Under these conditions:

- Do there exist algorithms that yield accurate neural networks?
- Are there computational barriers in the training of neural networks?

$\rightsquigarrow$ *Is this realistic for autonomous digital computations?*

A More Detailed Comparison

Example:

Consider the computation of a function f through an Oracle Turing machine.

- Assume that
  - on the *non-computable numbers*, f exhibits a behaviour that prevents the computation by means of a Turing machine,
  - restricted to the *computable numbers*, the function f is well-behaved and can be computed by a Turing machine.

A More Detailed Comparison

Example:

Consider the computation of a function f through an Oracle Turing machine.

- Assume that
  - on the *non-computable numbers*, f exhibits a behaviour that prevents the computation by means of a Turing machine,
  - restricted to the *computable numbers*, the function f is well-behaved and can be computed by a Turing machine.

Then we have the following:

(1) f is not computable by an Oracle Turing machine.

(2) A regular Turing machine, which operates solely on the computable numbers, is able to compute f on its natural domain.

A More Detailed Comparison

Example:

Consider the computation of a function f through an Oracle Turing machine.

- Assume that
  - on the *non-computable numbers*, f exhibits a behaviour that prevents the computation by means of a Turing machine,
  - restricted to the *computable numbers*, the function f is well-behaved and can be computed by a Turing machine.

Then we have the following:

(1) f is not computable by an Oracle Turing machine.

(2) A regular Turing machine, which operates solely on the computable numbers, is able to compute f on its natural domain.

$\rightsquigarrow$ *Thus non-computability of a function on a Turing machine is a strictly stronger result than non-computability on an Oracle Turing machine!*

A More Detailed Comparison

Example:

Consider the computation of a function f through an Oracle Turing machine.

- Assume that
  - on the *non-computable numbers*, f exhibits a behaviour that prevents the computation by means of a Turing machine,
  - restricted to the *computable numbers*, the function f is well-behaved and can be computed by a Turing machine.

Then we have the following:

(1) f is not computable by an Oracle Turing machine.

(2) A regular Turing machine, which operates solely on the computable numbers, is able to compute f on its natural domain.

$\rightsquigarrow$ *Thus non-computability of a function on a Turing machine is a strictly stronger result than non-computability on an Oracle Turing machine!*

$\rightsquigarrow$ *This raises some doubts whether Oracle Turing machines are an adequate model for autonomous real-world computations!*

Consequences

Some Thoughts on the Result

Corollary:

- Ψ can *not be approximated* to arbitrary accuracy on digital computers.
- *No algorithm exists*, which on digital hardware derives neural networks Φ_A approximating $\Psi(A, \cdot)$ for any given accuracy and all $A \in \mathbb{C}^{m \times N}$.
- The output of trained neural networks *not reliable (no guarantees)*.

Some Thoughts on the Result

Corollary:

- Ψ can *not be approximated* to arbitrary accuracy on digital computers.
- *No algorithm exists*, which on digital hardware derives neural networks Φ_A approximating $\Psi(A, \cdot)$ for any given accuracy and all $A \in \mathbb{C}^{m \times N}$.
- The output of trained neural networks *not reliable (no guarantees)*.

General Barrier:

This barrier on the capabilities of neural networks for finite-dimensional inverse problems is caused by *a combination* of the following two separate aspects:

- The mathematical structure and properties of *finite-dimensional inverse problems*.
- The mathematical structure and properties of *Turing machines* and thereby also of *digital machines*.

Some Additional Remarks

Some Additional Remarks

- This result "only" concerns *finite-dimensional inverse problems*.

Some Additional Remarks

- This result "only" concerns *finite-dimensional inverse problems*.
- The problem is *inherently connected* to digital computations.

Some Additional Remarks

- This result "only" concerns *finite-dimensional inverse problems*.
- The problem is *inherently connected* to digital computations.
- Deep learning has still impressive results, but a *necessary caution* is on order.

Some Additional Remarks

- This result "only" concerns *finite-dimensional inverse problems*.
- The problem is *inherently connected* to digital computations.
- Deep learning has still impressive results, but a *necessary caution* is on order.
- This result could point towards why *instabilities* and *non-robustness* occurs for deep neural networks.

Some Additional Remarks

- This result "only" concerns *finite-dimensional inverse problems*.
- The problem is *inherently connected* to digital computations.
- Deep learning has still impressive results, but a *necessary caution* is on order.
- This result could point towards why *instabilities* and *non-robustness* occurs for deep neural networks.
- A *subtlety*: This does not necessarily mean that an algorithm yielding a network with a required, fixed precision can not be obtained. However, any prescribed precision may require different Turing machines, i.e., different algorithms.

What now?

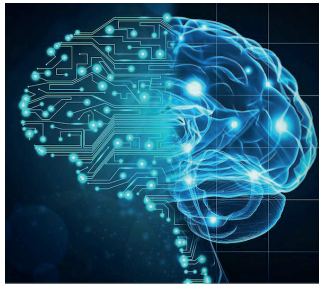
Today computations performed almost exclusively on digital hardware!

What now?

Today computations performed almost exclusively on digital hardware!

Other Models of Computations:

- New emerging hardware
  - *Neuromorphic computing*: Elements of computer modeled after systems in the human brain and nervous system.
  - *Biocomputing*: Living cells as the substrate for performing human-defined computations
- Different models of computation required



What now?

Today computations performed almost exclusively on digital hardware!

Other Models of Computations:

- New emerging hardware
  - *Neuromorphic computing*: Elements of computer modeled after systems in the human brain and nervous system.
  - *Biocomputing*: Living cells as the substrate for performing human-defined computations
- Different models of computation required

Key Future Question:

Does the non-computability result also hold for different computation models such as analog computers as well?

Some Final Thoughts...

Conclusions

Some General Limitations:

- Robustness (Adversarial attacks, etc.)
- Explainability
- Severe dependence on data (bias, requires huge amount, ...)
- Specific task
- Reasoning

Computability

- A *Turing machine* is a suitable model for digital computers.
- *Computability* needs to be studied.
- We focus on *finite-dimensional inverse problems*.

There does not exist an algorithm which performs the training of a neural network on digital hardware for any given accuracy!

THANK YOU!

References available at:

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

Survey Paper (arXiv:2105.04026):

Berner, Grohs, K, 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, to appear.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN